



深度学习简介

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>



本章内容

- 课程介绍
- 内容介绍



课程介绍

- 主讲教师：连德富
大数据学院 | 大数据分析与应用安徽省重点实验室
 - 办公室：西区科技楼东楼715室
 - 主要研究方向：深度学习、推荐系统、智慧城市
 - 手机（微信同号）：13739227137
 - 邮箱：liandefu@ustc.edu.cn
- 助教：
 - 贺颖，2021级博士研究生
 - 数学科学学院
 - yingheustc@mail.ustc.edu.cn

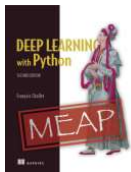
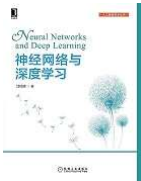
教材及参考资料

• 教材

- Deep Learning, Ian Goodfellow, Yoshua Bengio and Aaron Courville
- 神经网络与深度学习 邱锡鹏 机械工业出版社

• 参考资料

- Deep Learning with Python, François Chollet
- 动手学深度学习, 阿斯顿·张、李沐等





推荐课程

- 斯坦福大学CS224n: Deep Learning for Natural Language Processing
 - <https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/>
 - Chris Manning 主要讲解自然语言处理领域的各种深度学习模型
- 斯坦福大学CS231n: Convolutional Neural Networks for Visual Recognition
 - <http://cs231n.stanford.edu/>
 - Fei-Fei Li Andrej Karpathy 主要讲解CNN、RNN在图像领域的应用



推荐课程

- 林轩田“机器学习基石” “机器学习技法”
 - <https://www.csie.ntu.edu.tw/~htlin/mooc/>
- 李宏毅 “1天搞懂深度学习”
 - http://speech.ee.ntu.edu.tw/~tlkagk/slide/Tutorial_HYLee_Deep.pptx
- 李宏毅 “机器学习2020”
 - <https://www.bilibili.com/video/av94519857/>



课程安排及考核方式

- 课程作业

- 5个小实验，分别是CNN，RNN，GCN，Transformer，GAN
- 一个综合实验，参加KDD Cup、Kaggle竞赛的其中一个比赛
- 组队：前5个小实验个人完成，比赛两人组队完成，比赛得分根据排名确定

- 总成绩

- $14\text{分}+14\text{分}+14\text{分}+14\text{分}+14\text{分}+25\text{分}+5\text{分}=100\text{分}$



课程反馈

课程安排及考核方式

- 1st Place - \$50,000
- 2nd Place - \$20,000
- 3rd Place - \$10,000
- ...



8

- 赛题1: The goal of this competition is to classify isolated American Sign Language (ASL) signs
 - classify isolated American Sign Language (ASL) signs
 - 链接地址: <https://www.kaggle.com/competitions/asl-signs>

时间安排

- February 23, 2023 - Start Date.
- April 25, 2023 - Entry Deadline. You must accept the competition rules before this date in order to compete.
- April 25, 2023 - Team Merger Deadline. This is the last day participants may join or merge teams.
- May 2, 2023 - Final Submission Deadline.

课程安排及考核方式

1st Place - \$25,000
2nd Place - \$20,000
3rd Place - \$15,000



9

- 赛题2：AMP®-Parkinson's Disease Progression Prediction
 - Use protein and peptide data measurements from Parkinson's Disease patients to predict progression of the disease.
 - 链接地址： <https://www.kaggle.com/competitions/amp-parkinsons-disease-progression-prediction>
- February 16, 2023 - Start Date.
- May 11, 2023 - Entry Deadline. You must accept the competition rules before this date in order to compete.
- May 11, 2023 - Team Merger Deadline. This is the last day participants may join or merge teams.
- May 18, 2023 - Final Submission Deadline.

课程安排及考核方式

1st Place - \$12,000

2nd Place - \$8,000

3rd Place - \$5,000



10

- 赛题3: Predict Student Performance from Gamè Play
 - Trace student learning from Jo Wilder online educational game
 - 链接地址: <https://www.kaggle.com/competitions/predict-student-performance-from-game-play>
 - February 6, 2023 - Start Date.
 - May 1, 2023 - Entry Deadline. You must accept the competition rules before this date in order to compete.
 - May 1, 2023 - Team Merger Deadline. This is the last day participants may join or merge teams.
 - May 8, 2023 - Final Submission Deadline.

课程大纲

- 1. 深度学习简介
- 2. 前馈神经网络
- 3. 卷积神经网络
- 4. 循环神经网络
- 5. 注意机制与外部记忆
- 6. 图卷积网络
- 7. 神经网络优化
- 8. 神经网络正则化
- 9. 无监督表示学习
- 10. 深度生成模型
- 11. 深度强化学习
- 12. 深度学习前沿 (1)
- 13. 深度学习前沿 (2)

元学习、终身学习、多任务学习、神经结构搜索、极多类分类、自监督学习、小样本学习，以及部分应用

深度学习领域的顶级会议和期刊

- 会议 NeurIPS、ICLR、ICML
- 会议 CVPR、ICCV
- 会议 ACL、EMNLP
- 会议 AAAI、IJCAI
- 期刊 JMLR、MLJ、TPAMI
- 期刊 IJCV、TNNLS
- 期刊 TACL、TASLP
- 期刊 AIJ、JAIR

人工智能

- 人工智能 (artificial intelligence, AI) 就是让机器具有人类的智能。
 - “计算机控制” + “智能行为”

人工智能就是要让机器的行为看起来就像是人所表现出的智能行为一样。

John McCarthy (1927-2011)

- 人工智能这个学科的诞生有着明确的标志性事件，就是1956年的达特茅斯 (Dartmouth) 会议。在这次会议上，“人工智能”被提出并作为本研究领域的名称。

弱人工智能 VS 强人工智能

- **弱人工智能**：限制领域人工智能（Narrow AI）或应用型人工智能（Applied AI），指的是专注于且只能解决特定领域问题的人工智能
- **强人工智能**：又称通用人工智能（Artificial General Intelligence）或完全人工智能（Full AI），指的是可以胜任人类所有工作的人工智能。

强人工智能具备以下能力

1. 存在不确定性因素时进行推理，使用策略，解决问题，制定决策的能力
2. 知识表示的能力，包括常识性知识的表示能力
3. 规划能力
4. 学习能力
5. 使用自然语言进行交流沟通的能力
6. 将上述能力整合起来实现既定目标的能力

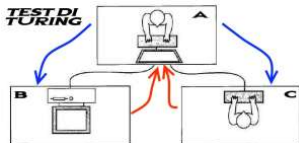
图灵测试

“一个人在不接触对方的情况下，通过一种特殊的方式，和对方进行一系列的问答。如果在相当长时间内，他无法根据这些问题判断对方是人还是计算机，那么就可以认为这个计算机是智能的”。

---Alan Turing [1950]
《Computing Machinery and Intelligence》



Alan Turing



人工智能的三个阶段

• 让机器具有人类的智能

感知智能

- 基于视觉、听觉及各种传感器的信息处理
- 表现为“能听会说、能看会认”
- 机器视觉、语音识别、文字识别

认知智能

- 更高层的语义处理、推理、规划、记忆、学习
- 表现为“能理解、会思考、有认知”
- 知识表示、模式识别、机器学习、自然语言处理

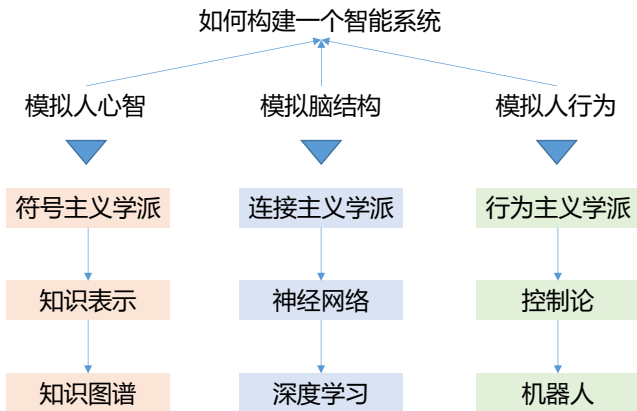
决策智能

- 复杂问题下，提升人机信任度，增强人类与智能系统交互协作智能
- 表现为“自主性”
- 规划、数据挖掘、强化学习

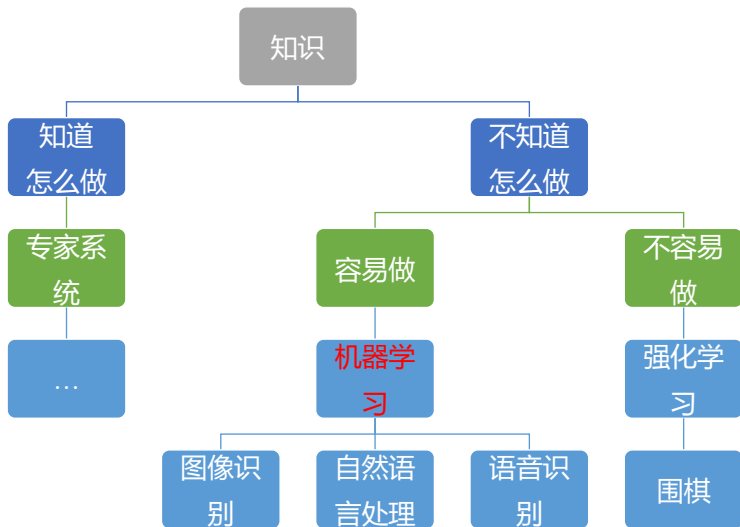
人工智能发展历史



人工智能流派



如何开发一个人工智能系统？



机器学习

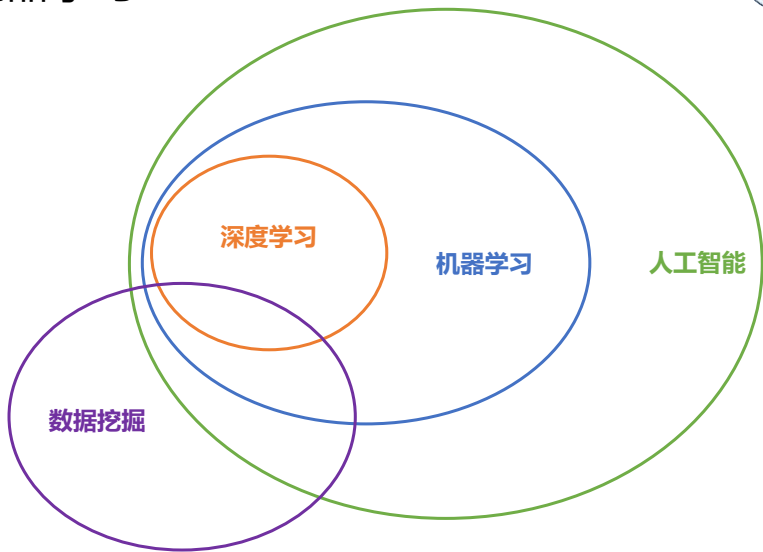
- 机器学习是从有限的观测数据中学习出一般性的规律，并利用规律对未知数据进行预测
- 当我们用机器学习来解决一些模式识别任务时，一般的流程包含以下几个步骤：



特征工程 (Feature Engineering)

- 这类方法也称为浅层学习 (Shallow Learning)，不涉及特征学习，其特征主要靠人工经验或特征转换方法来抽取

机器学习



语义鸿沟

- 语义鸿沟：输入数据的底层特征和高层语义信息之间的不一致性和差异性



相似的视觉特性(color, texture, shape), 不同的语义概念

不同视角



不同大小



形变



遮挡



不同光照



背景干扰



同类异形



语义鸿沟

- 人们对文本、图像的理解无法从字符串或者图像的底层特征直接获得，它需要使用人们日常生活中积累的大量经验和知识来进行推理和判断。



爆竹声中一岁除，
春风送暖入屠苏。
千门万户曈曈日，
总把新桃换旧符。

对于一些高层次的抽象概念，如一幅关于节日的图像或古诗所表达出的欢乐和喜庆的感觉等，更需要根据人的知识来判断。

表示学习

- 如何自动从数据中学习好的表示

表示学习：核心问题

- 什么是好的表示
- 如何学到好的表示

什么是好的数据表示？

- “好的表示” 是一个非常主观的概念，没有一个明确的标准
- 但一般而言，一个好的表示具有以下几个优点：

1

具有很强的表示能力

2

使后续的学习任务变得简单

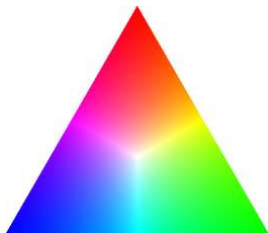
3

具有一般性，是任务或领域独立的

局部表示和分布式表示

- 以颜色为例说明：红、蓝、白、黑
- 局部表示（符号表示）：以不同的名字来命名不同的颜色，one-hot向量表示
- 分布式表示（分散式表示）：用RGB表示颜色，不同颜色对应到RGB三维空间中的一个点

	局部表示	分布式表示
红色	[1 0 0 0]	[1.00, 0.00, 0.00]
蓝色	[0 1 0 0]	[0.00, 0.00, 1.00]
白色	[0 0 1 0]	[1.00, 1.00, 1.00]
黑色	[0 0 0 1]	[0.00, 0.00, 0.00]



局部表示和分布式表示

局部表示

优点

- 解释性好、易归纳总结
- 稀疏向量，计算效率高

缺点

- 维数高，无法扩展
- 不同颜色相似度为0

分布式表示

优点

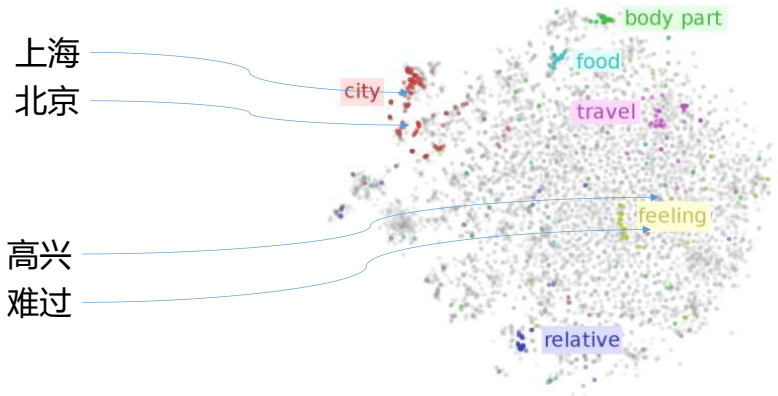
- 维度低、表示能力强
- 易计算相似度、易扩展

缺点

- 可解释性差
- 稠密、计算开销较大

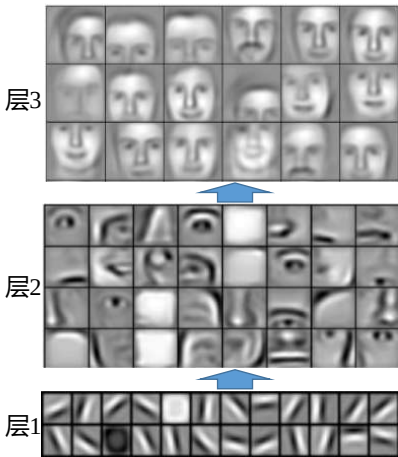
分布式表示 (嵌入)

- 分布式表示: 把高维局部表示空间映射到低维分布式表示空间
- 在机器学习中这个过程也称为嵌入
- 在自然语言中词的分布式表示叫作 **词嵌入**



如何学到好的表示

- 一个好的表示学习策略需要从底层特征开始，经过多步非线性变换才能得到
- 表示学习的关键：具有一定深度的多层次特征表示
 - 特征重用 指数级的表示能力
 - 抽象表示与不变性
抽象表示需要多步的构造



传统特征学习

- 线性投影（子空间）
 - PCA、LDA
- 非线性嵌入
 - LLE、Isomap、MDS
- 自编码器

传统特征学习

- 通过人为地设计准则，根据这些准则选取有效特征
- 特征学习和预测模型分开，学习特征不一定提升性能

表示学习

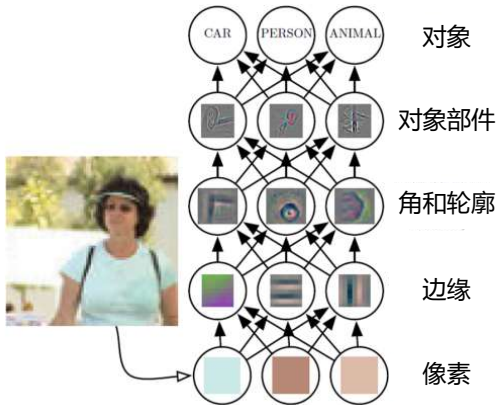
- 通过深度学习模型自动学习高层语义特征
- 特征学习和预测模型联合训练

深度学习

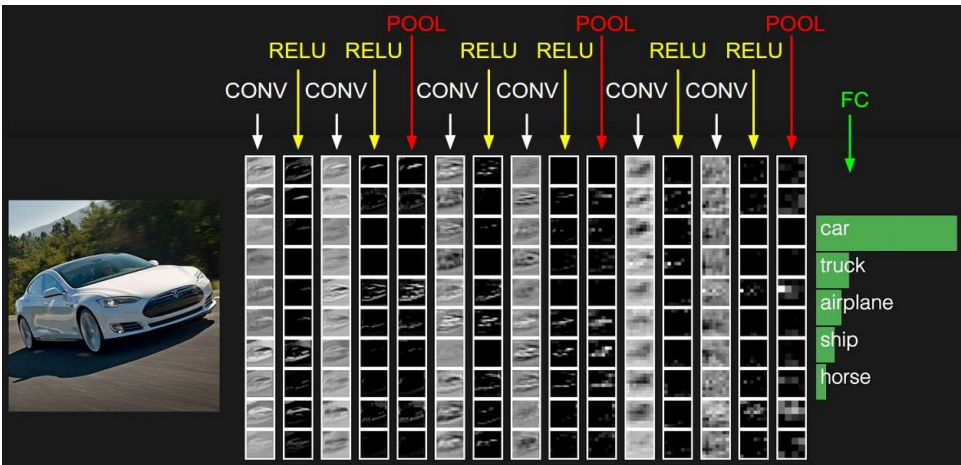
- 通过构建具有一定“深度”的模型，将大千世界表示为嵌套的层次概念体系，从而让模型来自动学习好的特征表示，最终提升预测或识别的准确性

通过较简单的表示来表达复杂表示

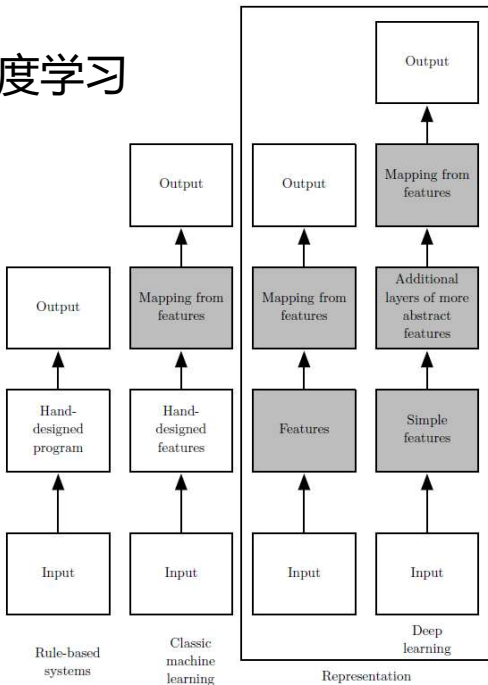
通过较简单概念构建复杂概念



表示学习与深度学习



什么是深度学习



深度学习的数学描述

浅层学习

$$y = f(x)$$

$$y = f^2(f^1(x))$$

$$y = f^3(f^2(f^1(x)))$$

⋮

$$y = f^n(f^{n-1}(\dots f^3(f^2(f^1(x))))))$$

深度学习

$f^l(x)$ 为非线性函数,
不一定连续

端到端学习

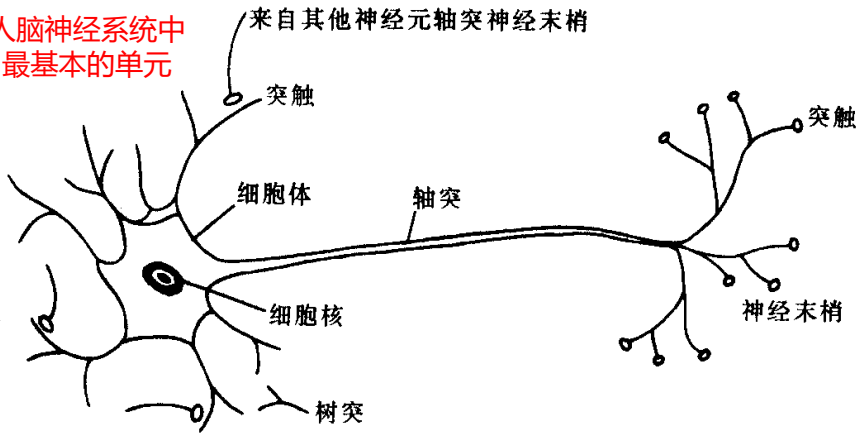
- 传统机器学习分模块单独学习
 - 例如自然语言理解任务，一般需要分词、词性标注、句法分析、语义分析、语义推理等不走

存在问题

- 每个模块单独优化，优化目标和任务目标可能不一致
 - 错误传播，前一步错误会对后续模型造成影响
-
- 端到端学习，学习过程中不进行分模块或分阶段训练，直接优化任务总体目标，中间过程不需要人为干预
 - 训练数据为“输入-输出”对的形式，无须提供额外信息

神经网络回顾—神经元

人脑神经系统中
最基本的单元



每个神经元与其他神经元相连, 当它“兴奋”时, 就会向相连的神经元发送化学物质, 从而改变这些神经元内的电位; 如果某神经元的电位超过一个“阈值”, 那么它就会被激活, 即“兴奋”起来, 向其它神经元发送化学物质

神经网络回顾—人脑神经网络学习规则

- 人的智力来源于知识吸取，即人脑神经网络具有学习能力
- 神经元并不重要，重要的是神经元之间的连接（突触）
- 赫布法则 Hebb's Rule （突触可塑性基本原理）

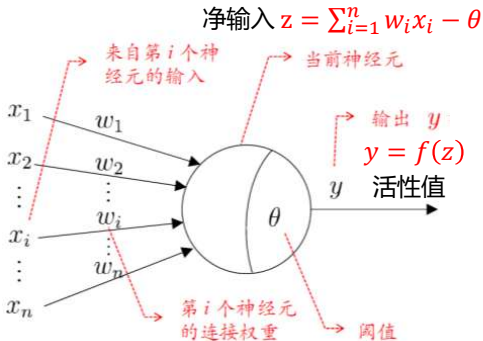
“当神经元 A 的一个轴突和神经元 B 很近，足以对它产生影响，并且持续地、重复地参与了对神经元 B 的兴奋，那么在这两个神经元或其中之一会发生某种生长过程或新陈代谢变化，以致于神经元 A 作为能使神经元 B 兴奋的细胞之一，它的效能加强了。”

----加拿大心理学家 Donald Hebb,
《行为的组织》，1949

神经网络回顾—人工神经元

M-P 神经元模型 [McCulloch and Pitts, 1943]

- 输入**：来自其他 n 个神经元传递过来的输入信号
- 处理**：输入信号通过带权重的连接进行传递, 神经元接受到总输入值将与神经元的阈值进行比较
- 输出**：通过激活函数的处理以得到输出



神经网络回顾—人工神经网络

- 人工神经网络从结构、实现机理、学习机制上模拟人脑神经系统

结构

- 有大量的人工神经元互相连接而成，不同神经元之间的连接被赋予了不同权重，每个权重代表了一个节点对另一个节点的影响大小，从而具有并行分布结构

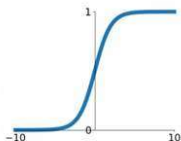
实现机理

- 每个节点代表一种特定函数，来自其他节点的信息经过相应权重计算，输入到激活函数，得到活性值，代表兴奋或者抑制

神经网络回顾—激活函数

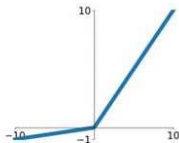
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



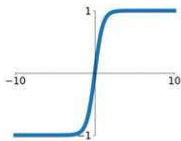
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

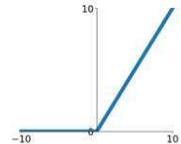


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

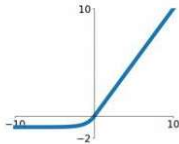
ReLU

$$\max(0, x)$$

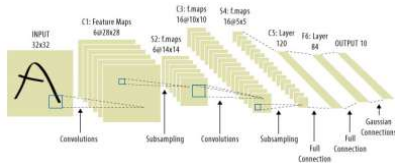
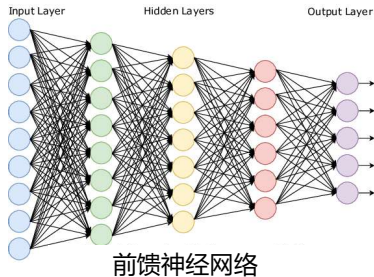


ELU

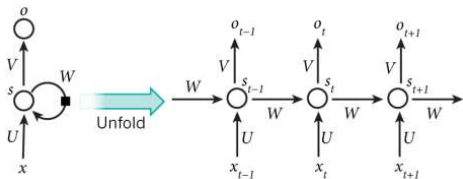
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



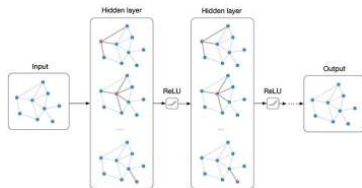
神经网络回顾—神经网络结构



深度卷积网络



循环神经网络



图卷积网络

神经网络回顾—人工神经网络学习机制

- 赫布网络学习规则 $w_{ij} \leftarrow w_{ij} + \eta x_i y_i$ 无监督学习方法

对应
更新
规则

若 x_i 与 y_j 同时被激活, 即 x_i 与 y_j 同时为正, 那么 w_{ij} 将增大
若 x_i 被激活, y_j 被抑制, 即 x_i 为正 y_j 为负, 那么 w_{ij} 将变小

- 感知机学习规则 $w_{ij} \leftarrow w_{ij} + \eta(y_j - \hat{y}_j)x_i$ 有监督学习方法

对应
更新
规则

若预测值更大, 降低激活输入的权重;
若预测值更小, 增加激活输入的权重

- 反向传播算法, 解决了多层神经网络的学习问题 有监督学习方法

神经网络发展史

• 第一阶段：模型提出

- 1943年, 心理学家 McCulloch和数学家 Pitts 提出第一个神经元数学模型, 即**M-P模型**, 从原理上证明了人工神经网络能够计算任何算数和逻辑函数
- 1948年, Alan Turing描述了一种“B型图灵机”, 可基于赫布法则学习
- 1949年, Hebb 发表《The Organization of Behavior》一书, 提出生物神经元学习的机理, 即**Hebb学习规则**
- 1951年, McCulloch和Pitts的学生Marvin Minsky建造了第一台神经网络机, 称为SNARC。
- 1958年, Rosenblatt最早提出可以模拟人类感知能力的神经网络模型, **感知机网络** (Perceptron) 模型, 提出了一种接近于人类学习过程 (迭代、试错) 的学习算法
- 1960年, Widrow和Hoff提出**自适应线性神经元** (Adaline) 模型和**最小均方学习算法**

神经网络发展史

• 第二阶段：冰河期

- 1969年, Marvin Minsky 和 Seymour Papert发表《Perceptrons》一书, 指出单层神经网络不能解决非线性问题, 多层网络的训练算法尚无希望, 导致神经网络十多年的“冰河期”
- 1974年, 哈佛大学的Paul Webos发明反向传播算法, 但当时未受到应有的重视。
- 1980年, Kuniyiko Fukushima (福岛邦彦) 提出了一种带卷积和子采样操作的多层神经网络: 新知机 (Neocognitron)

神经网络发展史

• 第三阶段：反向传播算法引起的复兴

- 1982年, 物理学家Hopfield对神经网络引入能量函数的概念, 并提出了用于联想记忆和优化计算的神经网络 (称为Hopfield网络), 在旅行商问题上获得当时最好结果, 引起轰动
- 1984年, Geoffrey Hinton提出一种随机化版本的Hopfield网络, 即玻尔兹曼机
- 1986年, David Rumelhart和James McClelland对于联结主义在计算机模拟神经活动中的应用提供了全面的论述, 并重新发明了反向传播算法
- 1986年, Geoffrey Hinton等人将引入反向传播算法到多层感知器
- 1987年, IEEE 在美国加州圣地亚哥召开第一届神经网络国际会议 (ICNN)
- 1989 年, LeCun等人将反向传播算法引入了卷积神经网络, 并在手写体数字识别上取得了很大的成功

神经网络发展史

- 第四阶段：流行度降低
 - 在20世纪90年代中期，统计学习理论和以支持向量机为代表的机器学习模型开始兴起。
 - 相比之下，神经网络的理论基础不清晰、可解释性差、试错性强、难以训练等缺点更加凸显，神经网络的研究又一次陷入低潮。

神经网络发展史

• 第五阶段：深度学习的崛起

- 2006年, Hinton等人提出了深度信念网络(DBN), 发现多层前馈神经网络可以先通过逐层预训练, 再用反向传播算法进行精调的方式进行有效学习
- 2012年, Hinton 组参加ImageNet 竞赛, 使用 CNN 模型以超过第二名10个百分点的成绩夺得当年竞赛的冠军
- 2013年, AlexNet: 第一个现代深度卷积网络模型, 是深度学习技术在图像分类上取得真正突破的开端。AlexNet不用预训练和逐层训练, 首次使用了很多现代深度网络的技术
- 伴随云计算、大数据时代的到来, 计算能力的大幅提升, 使得深度学习模型在计算机视觉、自然语言处理、语音识别等众多领域都取得了较大的成功

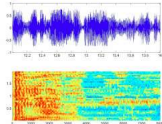
Images & Video



Text & Language



Speech & Audio



深度学习取得了巨大的进展

- **语音识别**：微软英语语音识别实现词错率5.9%的突破，第一次**超越人类**
- **人脸识别**：Facebook的人脸识别系统DeepFace达到97.53%的准确率，**达到人类水平**
- **图像识别**：微软在ImageNet图像数据库上，达到4.94%的错误率，**低于人类5.1%的错误率**
- **人机对弈**：AlphaGo以4：1的战绩击败李世石，Master在围棋快棋上击败柯杰，聂卫平等高手，取得**60胜0负**的战绩



深度学习优势（一）

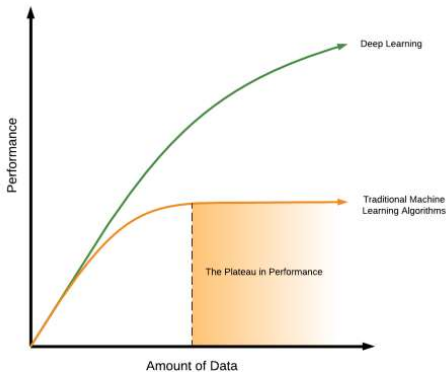
- 使用数据进行有效扩展：

传统机器学习

- 通常需要**更复杂的模型**来提高准确性

深度学习

- 可以通过使用**更多数据**来有效提高准确性



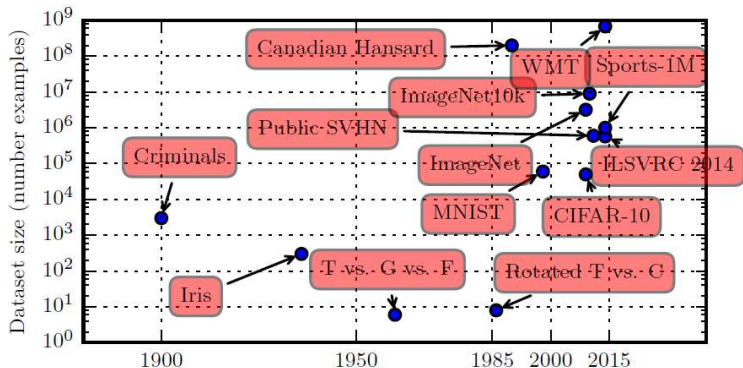
深度学习优势（二）

- 不需要特征工程
 - 只需将数据直接传递到网络，通常就可以实现良好的性能。这完全消除了传统机器学习的特征工程。
- 适应性强，易于转换
 - 深度学习技术更容易适应不同领域和应用
- 具有很强的通用性
 - 深度学习是一种大众化工具，大大延展了解决问题的领域

深度学习的发展趋势

• 与日俱增的数据量

- 尽管深度学习模型经历变革，但学习算法几乎未变。深度学习算法训练需要一定技巧，但随着训练数据增加，技巧变少



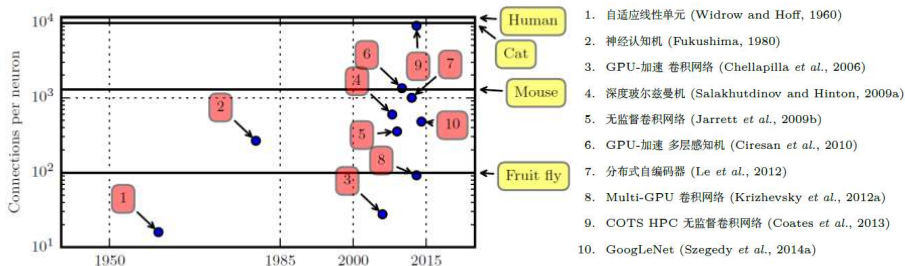
数据集规模从万级到10亿级

深度学习的发展趋势

- 数据量需求的粗略经验法则
 - 有监督深度学习算法在每类给定约5000 个标注样本情况下一般将达到可以接受的性能
 - 当至少有1000 万个标注样本的数据集用于训练时，它将达到或超过人类表现

深度学习的发展趋势

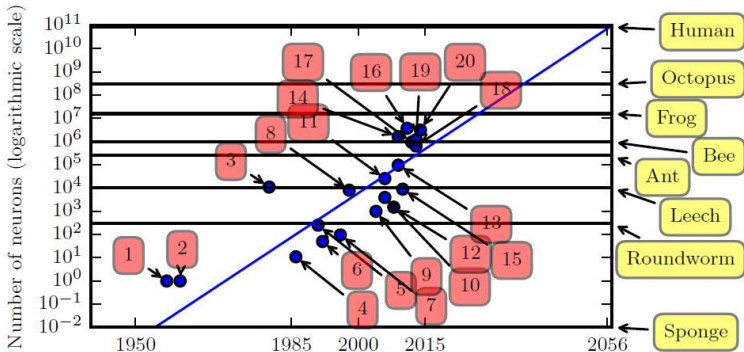
- 与日俱增的模型规模
 - 每个神经元的连接数量已经和哺乳动物大脑在同一数量级上



深度学习的发展趋势

• 与日俱增的模型规模

- 神经网络中神经元总数还很小。自从引入隐藏单元以来，人工神经网络的规模大约每 2.4 年扩大一倍，按照这个趋势，至少到21世纪50年代，才具备和人脑相同量级的神经元



深度学习的发展趋势

- 与日俱增的精度和复杂度以及对现实世界的冲击
 - 深度学习提供精确识别和预测的能力一直在提高，深度学习持续成功地被应用于越来越广泛的实际问题
 - 深度卷积网络在ImageNet的视觉识别挑战将top-5错误率从26.1%降到3.6%
 - 深度学习引入使得语音识别错误率陡然下降，甚至降低了一半
 - 深度网络在行人检测和图像分割中也取得了引人注目的成功，并且在交通标志分类上取得了超越人类的表现
 - 在深度网络的规模和精度提高同时，可以解决的任务也日益复杂
 - 生成图片、机器翻译、生成诗歌、生成曲谱
 - 深度学习的另一个最大的成就是其在 强化学习（reinforcement learning）领域的扩展—深度强化学习系统，能学会玩Atari 游戏，下围棋，打星际
 - 深度学习也为医学（疾病诊断）、化学（设计新药物）、物理（搜索亚原子粒子）等学科做出贡献

深度学习的广泛应用



深度学习应用—图片搜索



外观类似的图片



924f8...5a5f7.jpeg ×

蔡徐坤 打篮球

Q 全部 图片 地图 购物

找到约 576 条结果 (用时 0.83 秒)



图片尺寸:
440 × 262

查找该图片的其他尺寸
全部尺寸 - 小尺寸 -

可能相关的搜索查询: 蔡徐坤 打篮球



深度学习应用—图片搜索

Google
图片



深度学习应用—图片分类

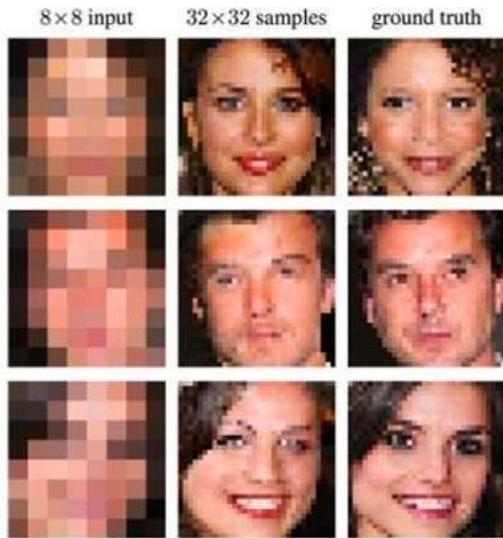


深度学习应用—图片上色



深度神经网络能学习自然存在于照片中的某些模式——比如，天通常是蓝的，云是白的或者灰的，草是绿的。从而不需要人类的介入就能对照片进行重新上色

深度学习应用—图片去模糊化



利用人工神经网络来填充低分辨率图像细节，能够把只有64个像素(8X8)的模糊图片，变成更清晰的图片，即去除图片中的“马赛克”

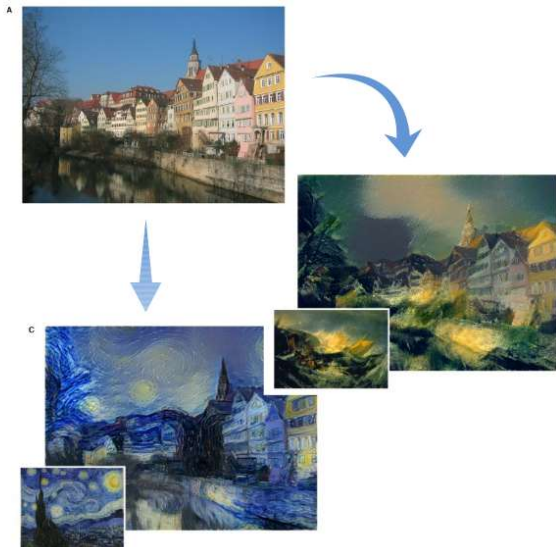
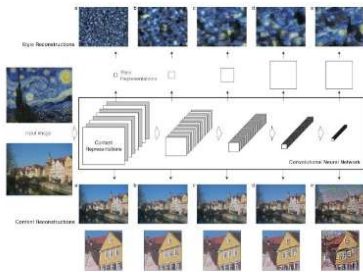
深度学习应用—换脸



学习算法利用人脸捕捉，让你在视频里实时扮演另一个人，简单来讲，就是可以把你的面部表情实时移植到视频里正在发表演讲的美国总统身上。



深度学习应用—风格迁移



深度学习算法让人工智能系统学习梵高，莫奈等世界著名画家的画风绘制新的“人工智能世界名画”

深度学习应用—翻译



翻译

关闭即时翻译

中文 英语 德语 检测语言

英语 中文(简体) 日语 翻译

Google神经机器翻译系统面世。该系统克服在大型数据集上工作的挑战，不再将句子分解为词和短语独立翻译，而是翻译完整的句子，使得误差降低了55%-85%以上。目前这项技术已经运用于Google Translate的汉英翻译。



112/5000

The Google Neuro Machine Translation System is available. The system overcomes the challenge of working on large data sets, no longer decomposing sentences into independent translations of words and phrases, but translating complete sentences, reducing errors by more than 55%-85%. This technology is currently used in Chinese-English translation of Google Translate.



提出修改建议

Google shénjīng jīqǐ fānyì xìtǒng miǎnshì. Gāi xìtǒng kěfú zài dàxíng shùjù jí shàng gōngzuò de tiáozhàn, bù zài jiāng jùzǐ fēnjiě wéi cí hé duǎnyǔ dúlì fānyì, ér shì fānyì wánzhěng de jùzǐ, shǐdé wùchā jiàngdīle 55%-85%yīshàng. Mùqián zhè xiàng jìshù yǐjīng yùnyòng yú Google Translate de hàn yīng fānyì.

文言文



中文

翻译



知之为知之，不知为不知，是知也。
扁鹊见蔡桓公，立有间。
长太息以掩涕兮，哀民生之多艰。



知道就知道，不知道就是知道，这才是聪明的。
扁鹊见蔡桓公，站了一会儿。
止不住的叹息擦不干的泪水啊，可怜人生道路多么艰难不顺利。

您输入的可能: 中文

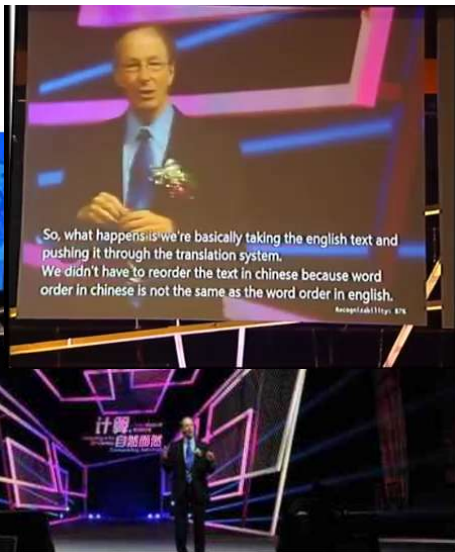


拼音 双语对照

深度学习应用—语音翻译



借助深度学习模型，语音的实时翻译成为可能



深度学习应用—文言文翻译

文言文 ▾



中文 ▾

翻译



知之为知之，不知为不知，是知也。
扁鹊见蔡桓公，立有间。
长太息以掩涕兮，哀民生之多艰。



知道就知道，不知道就是知道，这才是聪明的。
扁鹊见蔡桓公，站了一会儿。
止不住的叹息擦不干的泪水啊，可怜人生道路多么艰难不顺利。



您输入的可能是：中文

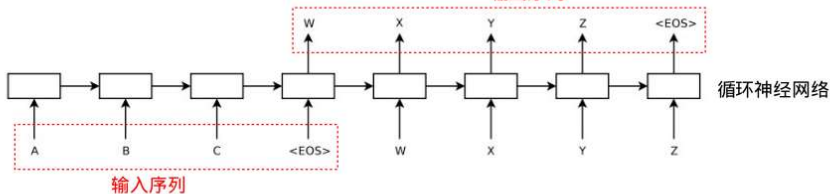


拼音



双语对照

输出序列



百度的文言文翻译，可以将现代汉语与文言文相互翻译，也是基于深度语言模型实现的

深度学习应用—聊天机器人



深度学习应用—写诗



语音作诗



藏头诗



看图作诗

深度学习应用——作曲编曲

念奴娇·赤壁怀古

【作者】苏轼【朝代】宋

大江东去，浪淘尽，千古风流人物。故垒西边，人道是，三国周郎赤壁。乱石穿空，惊涛拍岸，卷起千堆雪。江山如画，一时多少豪杰。遥想公瑾当年，小乔初嫁了，雄姿英发。羽扇纶巾，谈笑间，檣櫓灰飞烟灭。故国神游，多情应笑我，早生华发。人生如梦，一尊还酹江月。



深度学习应用—下棋



AlphaGo以3：0战胜柯洁

深度学习应用—德州扑克

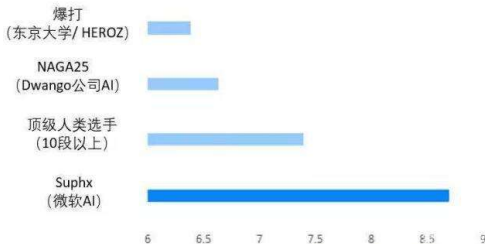


- 在无限限制德州扑克六人对决的比赛中，由 Facebook 与CMU共同开发的德扑 AI Pluribus 成功战胜了五名专家级人类玩家

深度学习应用—麻将



天凤平台“特上房”稳定段位对比



微软亚洲研究院研发的麻将AI Suphx，在国际知名麻将平台“天凤”上荣升十段，稳定段位显著超越人类顶级选手

深度学习应用—搜索

Google

Q machine learning course

Q machine learning course

Q machine learning course stanford

Q machine learning coursera github

Q machine learning course hk

Q machine learning coursera quiz

Q machine learning coursera andrew ng

Q machine learning coursera stanford

Q machine learning course recommendation

Q machine learning course python

Q machine learning course online

举报不当的联想查询

← 查询词建议

www.coursera.org > ... > Machine Learning ▾

Machine Learning by Stanford University | Coursera

About this **Course**. 9,957,819 recent views. **Machine learning** is the science of getting computers to act without being explicitly programmed. In the past decade, ...

[Machine learning and data ...](#) · [What Is Machine Learning?](#) · [Unsupervised Learning](#)

← 相关性计算
&PageRank

深度学习应用—推荐



机器学习

由AlphaGo的战胜柯洁 震撼人机大战的必须之选：人工智能大举进军神经网络领域，全面揭开机器学习的面纱

周志华 著

原价：¥68.40 (7.8折) [定价：¥88.00] [降价通知]

促销信息：购买1件可优惠购热销商品 立即抢购 >>

加价购：满10元另加2.88元， 或满12元另加16.90元， 或满15元另加9.90元，
即可在购物车换购热销商品 详情 >>

以上促销可在购物车任意选择一

增值业务：助力环保，传递知识，旧书换新

排名：商务 计算机与互联网销量榜 第4位

配送至：安徽省合肥市蜀山区麓湖山街道 <~>

由 京东 发货，并提供售后服务，23:10前下单，预计明天(09月14日)送达

重量：0.92kg

猜你喜欢

猜你喜欢

1/4



¥56.80



¥118.00



¥84.60



¥56.60



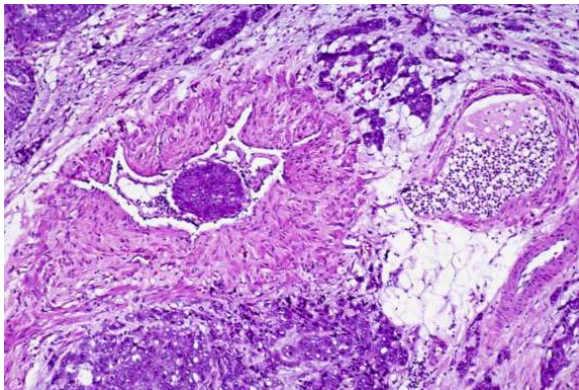
¥66.30



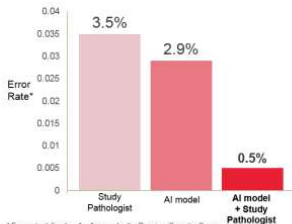
¥122.90

深度学习应用—疾病诊断

• 乳腺癌诊断



(AI + Pathologist) > Pathologist



* Error rate defined as 1 - Area under the Receiver Operator Curve

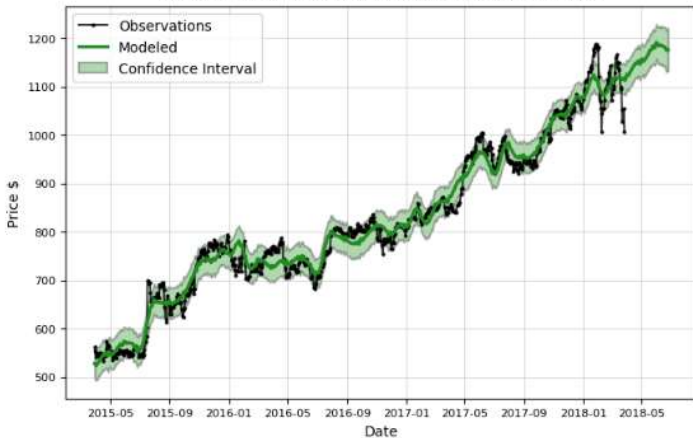
** A study pathologist, blinded to the ground truth diagnoses, independently scored all evaluation slides.

© 2016 PathAI

深度学习应用—股价预测

☐ Predicted Price on 2018-06-25 00:00:00 = \$1175.64

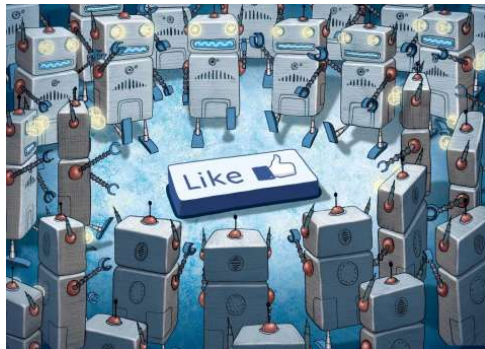
GOOGL Historical and Predicted Stock Price



深度学习应用—异常检测

• 社交网络中异常账号检测

☐ 货到付款 ☐ 海外商品 ☐ 二手 ☐ 天猫 ☐ 正品保障 ☐ 旺旺在线				更
	微博刷量 转发评论点赞/直发@人1元1万话题阅读量讨论大号推广	¥1.00	356人付款 227条评论	
	【皇冠信誉】 微博营销/话题直发/微博转发/评论点赞/活动推广参与	¥0.50	807人付款 152条评论	
	微博服务 微博营销/话题直发/微博转发/微博点赞/微博评论活动推广参与与运	¥0.50	738人付款 180条评论	
	微博转发包月 评论 点赞 话题 试听阅读量参与/1个QB1q币0.5元	¥0.50	598人付款 1条评论	



深度学习应用—虚假新闻检测



提问较真



疫情数据



北京疫情

新冠肺炎男性患者的死亡率更高 **确实如此**

2020-09-10



新冠死者平均只损失了几个月的寿命 **谣言**

2020-09-09



深度学习应用—自动驾驶



79

优酷

深度学习早期、长期学者

- Hinton 75 年 Edinburgh 大学博士

- Lecun (Now Facebook) 87 年
Hinton 博士后

- Bengio 91 年 M. Jordan 博士后



2018图灵奖获得者

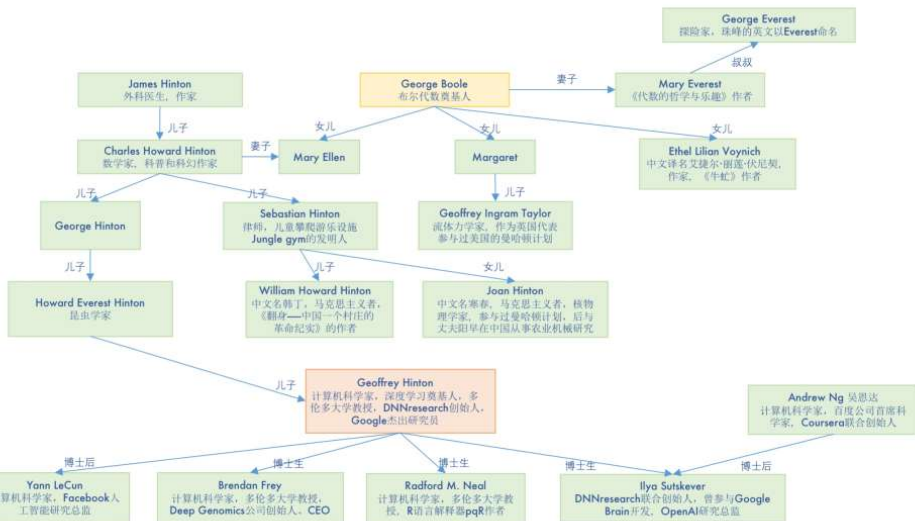
- Jürgen Schmidhuber, LSTM发明人

- 徐雷, RPCL学习、LMSER自组织学习

- 张钹, 发表国内第一篇PAMI



学术族谱



学术族谱



82

Geoffrey Hinton

Michael Jordan

PostDoc

PostDoc

PhD



Max Welling

Yann Lecun

Yoshua Bengio

Andrew Ng

PostDoc

PhD

PostDoc

PhD

PhD

PhD

PhD

B.S. and M.S.



Marc'Aurelio Ranzato Hugo Larochelle Ilya Sutskever Ruslan Salakhutdinov Ian Goodfellow

常用的深度学习框架

- 简易和快速的原型设计
- 自动梯度计算
- 无缝CPU和GPU切换

 PaddlePaddle

 TensorFlow

 Chainer

 Microsoft
CNTK

 K Keras

 Caffe2

 theano

 PYTORCH

 APACHE
mxnet

课后练习

- 熟悉Numpy，注意和matlab的差别。如果你对matlab熟悉，可以阅读如下文档
 - https://www.numpy.org.cn/user/numpy_for_matlab_users.html
 - Numpy核心数据结构是 multidimensional array，而matlab核心数据结构是 matrix
- 选择Pytorch或者Tensorflow一个学习
 - 建议使用Pytorch或者Tensorflow的高层接口Keras
 - 核心数据结构是Tensor，即Numpy中 multidimensional array 的包装
 - 试用Pytorch或者Keras写对数回归和SVM，比较和手写算法的差别



前馈神经网络

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>

本章内容

- 神经网络动机
- 神经网络
 - 神经元
 - 激活函数
 - 网络结构
- 前馈神经网络
 - 参数学习
 - 自动微分
 - 快速入门



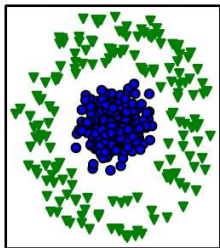
群名称：深度学习导论群

群 号：328068286

神经网络动机

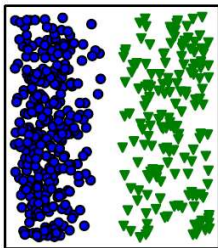
问题：线性模型难以处理线性不可分情形

Cartesian coordinates



$$\mathbf{x} = (x, y)$$

Polar coordinates



$$\phi(\mathbf{x}) = (r(x, y), \theta(x, y))^T$$

$$y = \mathbf{w}^T \mathbf{x} + b$$

缺点：无法理解输入变量非线性关系

$$y = \mathbf{w}^T \phi(\mathbf{x}) + b$$

线性模型用在一个非线性变换后的输入 $\phi(\mathbf{x})$

• ϕ 的选择：

- **核技巧**：隐式定义 ϕ 。如RBF核定义的无限维的 ϕ
- **特征工程**：根据经验手动设计 ϕ
- **神经网络学习** ϕ ，即 $y = f(\mathbf{x}; \theta, \mathbf{w}) = \phi(\mathbf{x}; \theta)^T \mathbf{w}$

神经网络动机

问题：线性模型泛化能力较弱



图片分类



青蛙

$$y = Wx$$

线性模型只能为每个类学习到一个模板



神经网络动机

问题：线性模型泛化能力较弱

图像特征

HoG:方向梯度直方图

SIFT:尺度不变特征变换

EoG:边缘方向直方图

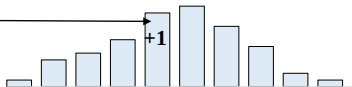
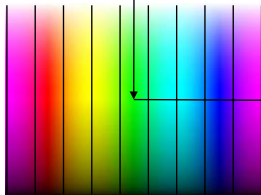
图片分类



青蛙



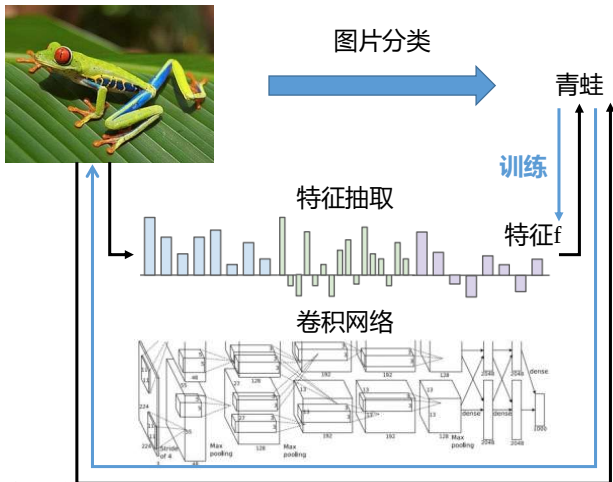
$$y = Wx$$



颜色直方图

神经网络动机

问题：线性模型泛化能力较弱



神经元模型

• 神经网络的定义

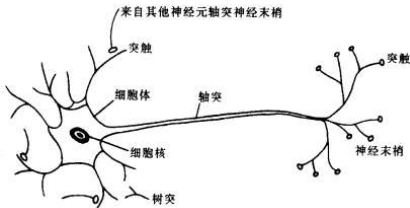
“神经网络是由具有适应性的简单单元组成的广泛并行互联的网络, 它的组织能够模拟生物神经系统对真实世界物体所作出的反应”

[Kohonen, 1988]

- 机器学习中的神经网络通常是指“神经网络学习”或者机器学习与神经网络两个学科的交叉部分
- 神经元模型即上述定义中的“简单单元”是神经网络的基本成分

生物神经网络

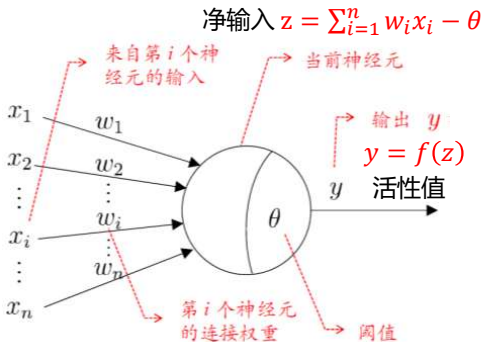
每个神经元与其他神经元相连, 当它“兴奋”时, 就会向相连的神经元发送化学物质, 从而改变这些神经元内的电位; 如果某神经元的电位超过一个“阈值”, 那么它就会被激活, 即“兴奋”起来, 向其它神经元发送化学物质



神经元模型

M-P 神经元模型 [McCulloch and Pitts, 1943]

- **输入**：来自其他 n 个神经元传递过来的输入信号
- **处理**：输入信号通过带权重的连接进行传递, 神经元接受到总输入值将与神经元的阈值进行比较
- **输出**：通过激活函数的处理以得到输出





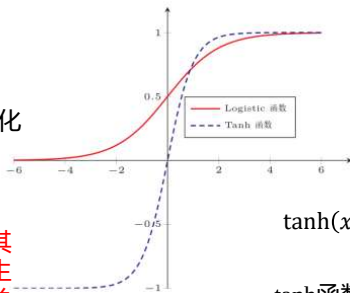
激活函数的性质

- 连续并可导（允许少数点上不可导）的非线性函数
 - 可导的激活函数可以直接利用数值优化的方法来学习网络参数
- 激活函数及其导函数要尽可能的简单
 - 有利于提高网络计算效率
- 激活函数的导函数的值域要在一个合适的区间内
 - 不能太大也不能太小，否则会影响训练的效率和稳定性

常用激活函数

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

sigmoid函数是**非**零中心化
输出恒大于0



$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

tanh函数是零中心化的

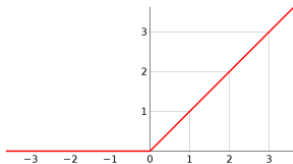
非零中心化的输出会使得其
后一层的神元输入发生
偏置偏移 (bias shift), 并
进一步使得梯度下降的收敛
速度变慢

当 $x \rightarrow -\infty$, 导数趋于0, 称为左饱和
饱和函数: 当 $x \rightarrow +\infty$, 导数趋于0, 称为右饱和
同时满足左、右饱和时, 称为两端饱和

常用激活函数

- $\text{ReLU}(x) = \max(0, x)$

大多数情况下默认选择



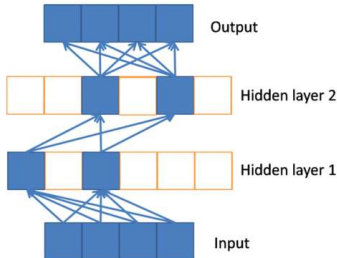
优点

- 计算上更加高效
- 生物学合理性
 - 单侧抑制、宽兴奋边界
 - 具有很好的稀疏性
- 在一定程度上缓解梯度消失问题

缺点

- 非零中心化，影响梯度下降效率
- 死亡ReLU问题

大约50%神经元处于激活状态



人脑在同一时刻只有1%-4%的神经元处于活跃状态

常用激活函数

$$\text{ReLU}(x) = \max(0, x)$$



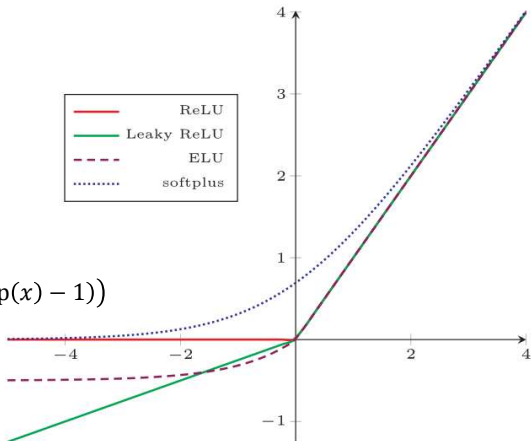
$$\text{LeakyReLU}(x) = \max(\gamma x, x)$$

$$\text{PReLU}_i(x) = \max(\gamma_i x, x)$$

$$\text{softplus}(x) = \log(1 + \exp(x))$$

$$\text{ELU}(x) = \max(0, x) + \min(0, \gamma(\exp(x) - 1))$$

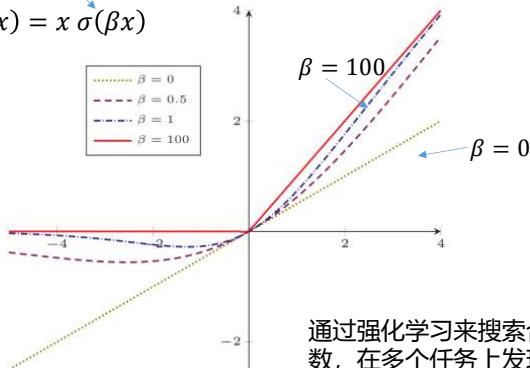
Glorot *et al.* (2011a) 比较了 softplus 和 ReLU, 发现后者的结果更好, 通常不鼓励使用 softplus 函数



常用激活函数

- Swish函数：自门控激活函数

$$\text{swish}(x) = x \sigma(\beta x)$$



通过强化学习来搜索合适的激活函数，在多个任务上发现Swish函数具有更好的性能

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

常见激活函数

- 高斯误差线性单元 (Gaussian Error Linear Unit, GELU)

$$\text{GELU}(x) = xP(X \leq x)$$

其中 $P(X \leq x)$ 是高斯分布 $N(\mu, \sigma^2)$ 的累积分布函数, 其中 μ, σ 为超参数, 一般设 $\mu = 0, \sigma = 1$ 即可

由于高斯分布的累积分布函数为S型函数, 因此GELU可以用Tanh函数或Logistic函数来近似

$$\text{GELU}(x) \approx 0.5x \left(1 + \tanh \left(\sqrt{2/\pi} (x + 0.044715 x^3) \right) \right)$$

$$\text{GELU}(x) \approx x \sigma(1.702x)$$

常见激活函数

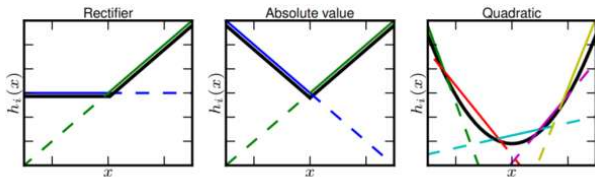
- Maxout 单元，也是一种分段线性函数
 - 假设将活性值 $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$ 分成组，每组 k 个值
 - Maxout单元输出每组的最大值，即

$$g(\mathbf{z})_i = \max_{j \in \mathbb{G}^{(i)}} z_j$$

$\mathbb{G}^{(i)}$ 是第 i 组的索引集 $\{(i-1)k+1, \dots, ki\}$

- 例子：输入为标量的 x
 - $z_j = w_j x + b_j$
 - $g(\mathbf{z})_1 = \max_{j=1}^k z_j$ 是分段线性凸函数

足够大的 k ，maxout 单元可以以任意的精确度来近似任何凸函数



$k=2$ 时，ReLU变种都可以通过maxout单元学习获得

常见激活函数及其导数

激活函数	函数	导数
Logistic 函数	$f(x) = \frac{1}{1+\exp(-x)}$	$f'(x) = f(x)(1 - f(x))$
Tanh 函数	$f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x) = 1 - f(x)^2$
ReLU 函数	$f(x) = \max(0, x)$	$f'(x) = I(x > 0)$
ELU 函数	$f(x) = \max(0, x) + \min(0, \gamma(\exp(x) - 1))$	$f'(x) = I(x > 0) + I(x \leq 0) \cdot \gamma \exp(x)$
SoftPlus 函数	$f(x) = \log(1 + \exp(x))$	$f'(x) = \frac{1}{1+\exp(-x)}$

神经网络

- 神经网络的定义

“神经网络是由具有适应性的简单单元组成的广泛并行互联的网络, 它的组织能够模拟生物神经系统对真实世界物体所作出的反应”

[Kohonen, 1988]

- 主要三个特性
 - 信息表示是分布式的
 - 记忆和知识是存储在单元之间的连接上的
 - 通过逐渐改变单元之间的连接强度来学习新知识

网络结构

- 神经网络设计的另一个关键点是确定它的结构
 - 具有多少单元，以及这些单元应该如何连接。
- 大多数神经网络被组织成层的单元组
- 大多数神经网络架构将这些层布置成链式结构，其中每一层都是前一层的函数
- 神经网络设计在于选择网络的深度和每一层的宽度

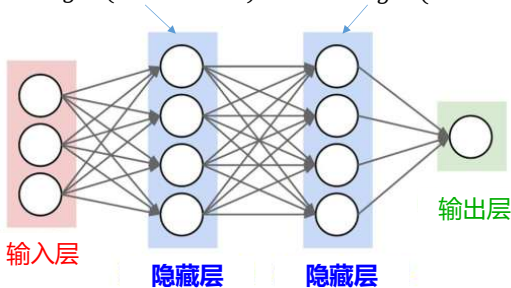
更深层网络通常能在每一层使用更少的单元数和更少的参数，并且有更强的泛化能力。但是通常也更难以优化

前馈神经网络（多层感知机）

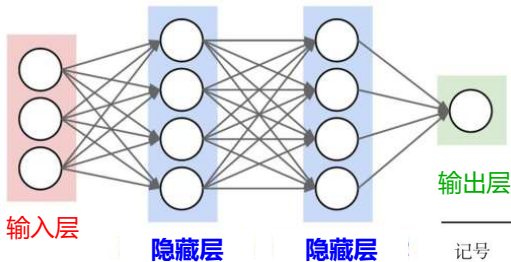
- 各神经元分别属于不同的层，层内无连接
- 相邻两层之间的神经元全部两两连接
- 整个网络中无反馈，信号从输入层向输出层单向传播

可用一个有向无环图表示

$$h^{(1)} = g^{(1)}(W^{(1)}x + b^{(1)}) \quad h^{(2)} = g^{(2)}(W^{(2)}x + b^{(1)})$$



前馈神经网络



给定一个前馈神经网络，用下面的记号来描述这样网络

一般层数 L 只考虑隐藏层和输出层

记号	含义
L	神经网络的层数
M_l	第 l 层神经元的个数
$f_l(\cdot)$	第 l 层神经元的激活函数
$\mathbf{W}^{(l)} \in \mathbb{R}^{M_l \times M_{l-1}}$	第 $l-1$ 层到第 l 层的权重矩阵
$\mathbf{b}^{(l)} \in \mathbb{R}^{M_l}$	第 $l-1$ 层到第 l 层的偏置
$\mathbf{z}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的净输入（净活性值）
$\mathbf{a}^{(l)} \in \mathbb{R}^{M_l}$	第 l 层神经元的输出（活性值）

前馈神经网络

- 前馈神经网络通过下面公式进行信息传播

$$\begin{array}{ll}
 \text{仿射变换} & \mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \\
 & \downarrow \\
 & \mathbf{a}^{(l)} = f_l(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}) \\
 \end{array}
 \qquad
 \begin{array}{ll}
 & \mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}) \quad \text{非线性变换} \\
 & \downarrow \\
 & \mathbf{z}^{(l)} = \mathbf{W}^{(l)} f_l(\mathbf{z}^{(l-1)}) + \mathbf{b}^{(l)}
 \end{array}$$

- 前馈计算：

$$\mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \mathbf{a}^{(2)} \rightarrow \dots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)}$$

前馈神经网络

- 若没有激活函数，那前馈神经网络是什么呢？
- 简单起见，假设没有偏置项

$$\mathbf{a}^{(l)} = \mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)}$$

$$\Rightarrow \mathbf{a}^{(l)} = \underbrace{\mathbf{W}^{(l)} \mathbf{W}^{(l-1)} \mathbf{W}^{(l-2)} \dots \mathbf{W}^{(1)}}_{\mathbf{W}} \mathbf{x}$$

前馈神经网络是线性分类器

万能近似定理 Universal Approximation Theorem

一个前馈神经网络如果具有线性输出层和至少一层具有任何一种“挤压”性质的激活函数（例如logistic sigmoid激活函数）的隐藏层，只要给予网络足够数量的隐藏单元，它可以以任意的精度来近似任何从一个有限维空间到另一个有限维空间的Borel 可测函数

- 在 $\mathbb{R}^n$ 的有界闭集上的任意连续函数是Borel 可测的
- 有界 指集合被包含在有限大小的球内
- 闭集 指包含所有极限点（边界）的集合

万能近似定理也已经被证明对于更广泛类别的激活函数，比如ReLU也是适用的



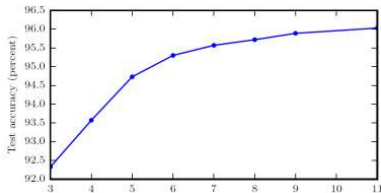
万能近似定理 Universal Approximation Theorem

- 可能需要**指数级**大的隐藏层，可能无法正确地学习和泛化
- 在很多情况下，使用**更深的模型**能够减少单元数量，并且可以减少泛化误差
- 存在一些函数族能够在网络的**深度大于某个值 d** 时被高效地近似，而当深度被限制到**小于或等于 d** 时需要一个远远大于之前的模型
- 浅层模型所需的隐单元数量是维度的指数级

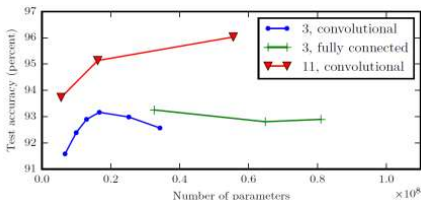
万能近似定理 Universal Approximation Theorem

25

- 更深层的网络能够更好地泛化



地址号码转录测试集上的性能随着深度的增加而不断增加



模型尺寸增加并不能产生相同的效果



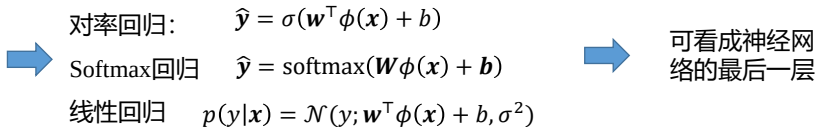
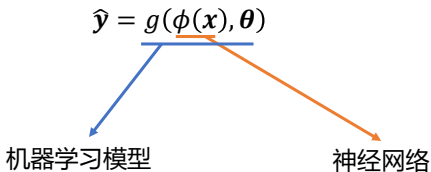
万能近似定理 Universal Approximation Theorem

- 万能近似定理只说明神经网络表达能力强大到可以近似任意一个的连续函数，却并没有给出如何找到这样的神经网络，以及是否是最优的

- 神经网络非线性，其学习是非凸优化问题，优化算法可能找不到近似函数的参数值
- 过于强大的表达能力，训练算法可能由于过拟合而选择了错误的函数

与机器学习结合

- 神经网络可以作为一个“万能”函数来使用，可以用来进行复杂的特征转换，或逼近一个复杂的条件分布。



多分类问题

- 若使用softmax回归分类器：相当于神经网络最后一层设置C个神经元，净激活值经过softmax函数变换，可以当做每个类的条件概率

$$\hat{y} = \text{softmax}(\mathbf{z}^{(L)}) \quad \mathbf{z}^{(L)} = \mathbf{W}\phi(\mathbf{x}) + \mathbf{b}$$

• softmax特点

- 过度参数化，因为概率总和为1，有一个参数可以通过其他的参数确定
- 更加简单，在实际中更加常用
- 输入值之间的差异变得极端饱和

z_i 不最大，且最大值非常大的时候，

$$\text{softmax}(\mathbf{z})_i \approx 0$$

z_i 最大，而且远大于其他时

$$\text{softmax}(\mathbf{z})_i \approx 1$$

• softmax的神经科学解释

- 参与其中的神经单元的相互竞争
- 一个单元的值增加导致其他单元值得减少
- 极端情况：赢者通吃

多分类问题

- 若使用softmax回归分类器：相当于神经网络最后一层设置C个神经元，净激活值经过softmax函数变换，可以当做每个类的条件概率

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(L)}) \quad \mathbf{z}^{(L)} = \mathbf{W}\phi(\mathbf{x}) + \mathbf{b}$$

- 采用交叉熵损失函数，对于样本 $(\mathbf{x}, \mathbf{y})$ ，其损失函数为

方便起见，这里用one-hot 向量 $\mathbf{y}$ 表示类别

$$\begin{aligned} \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) &= -\mathbf{y}^T \log \hat{\mathbf{y}} = -\log \frac{\exp z_c^{(L)}}{\sum_k \exp z_k^{(L)}} \\ &= -z_c^{(L)} + \log \sum_k \exp z_k^{(L)} \end{aligned}$$

如果正确分类，loss趋近0

$$\approx -z_c^{(L)} + \max_k z_k^{(L)}$$

深度学习库都提供了logsoftmax的函数和logsumexp的函数，以防止数值溢出

二分类问题

- 若使用对率回归分类器：相当于神经网络最后一层设置1个神经元，净激活值经过sigmoid函数变换，可以当做正类的条件概率

$$\hat{y} = \sigma(z^{(L)}) \quad z^{(L)} = \mathbf{w}^T \boldsymbol{\phi}(x) + b$$

- 采用交叉熵损失函数，对于样本 (x, y) ，其损失函数为

$$\mathcal{L}(y, \hat{y}) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

$$= -y \cdot z^{(L)} + \log(1 + \exp(z^{(L)}))$$

$$= \log(1 + \exp((1 - 2y)z^{(L)}))$$

深度学习库都提供了logsigmoid的函数和softplus的函数，以防止数值溢出

只有在 $z^{(L)}$ 取绝对值非常大的负值时才饱和
基于 $z^{(L)}$ 的二分类已经得到正确答案

回归

- 若使用回归器：神经网络表示函数 $g(\phi(x), \theta)$ ，预测 y 的条件分布的参数（高斯分布）

$$p(y|x) = \mathcal{N}(y; \mu, \sigma^2)$$

$$\mu = \mathbf{w}^\top \phi(x) + b$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \sigma^2 \mathbf{I})$$

$$\boldsymbol{\mu} = \mathbf{W}^\top \phi(x) + \mathbf{b}$$

- 采用负对数似然损失函数 $-\log p(y|x)$ 或 $-\log p(\mathbf{y}|\mathbf{x})$
- 学习方差：不同 x 预测出 y 不同的方差（异方差模型）
 - 把方差指定为 $g(\phi(x), \theta)$ 的其中一个输出
 - 用精度(precision)可以避免除法或者矩阵逆
 - 正定

标量（可以考虑用softplus激活函数）

矩阵（ $\Sigma(x) = B(x)B(x)^T$ ）

多峰回归(multimodal regression)

- 若使用回归器：神经网络表示函数 $g(\phi(\mathbf{x}), \boldsymbol{\theta})$ ，预测 y 条件分布的参数（高斯混合分布）

$$p(y|\mathbf{x}) = \sum_{i=1}^K p(c = i|\mathbf{x}) \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}^{(i)}(\mathbf{x}), \boldsymbol{\Sigma}^{(i)}(\mathbf{x}))$$

- 神经网络三个输出
 - $p(c = i|\mathbf{x})$ ，通过softmax获得
 - $\boldsymbol{\mu}^{(i)}(\mathbf{x})$
 - $\boldsymbol{\Sigma}^{(i)}(\mathbf{x})$
- 高斯混合输出在语音生成模型(Schuster, 1999) 和物理运动(Graves, 2013) 中特别有效

参数学习

- 给定训练集为 $D = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ ，将每个样本 $\mathbf{x}^{(n)}$ 输入给前馈神经网络，得到网络输出为 $\hat{\mathbf{y}}^{(n)}$ ，其在数据集 D 上的结构化风险函数为：

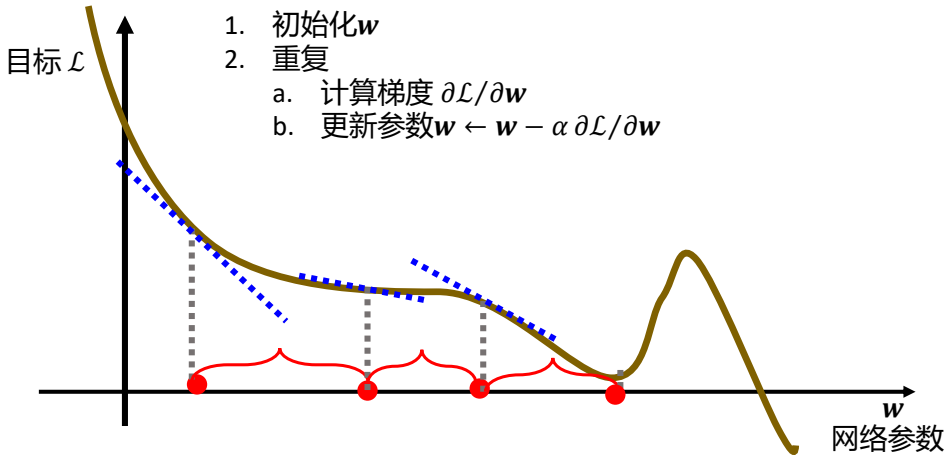
$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{1}{2} \lambda \|\mathbf{W}\|_F^2$$

- 梯度下降

$$\begin{aligned} \mathbf{W}^{(l)} &\leftarrow \mathbf{W}^{(l)} - \alpha \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}^{(l)}} \\ \mathbf{b}^{(l)} &\leftarrow \mathbf{b}^{(l)} - \alpha \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{b}^{(l)}} \end{aligned}$$

↑
学习率

梯度下降



随机梯度下降

- 由于数据集可能很大，无法全部放入内存计算梯度
- 一般采用小批量随机梯度下降法，每次从数据集中采样一部分样本（称为batch），计算batch上的梯度，并进行参数更新。
- 给定训练集为 $D = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$ ，每次采样B个样本

梯度下降

$$\mathcal{L}_D(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)}) + \frac{1}{2} \lambda \|\mathbf{W}\|_F^2$$

小批量随机梯度下降

$$\mathcal{L}_B(\mathbf{W}, \mathbf{b}) = \frac{1}{B} \sum_{i=1}^B \mathcal{L}(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}) + \frac{1}{2} \lambda \|\mathbf{W}\|_F^2$$

$$\frac{\partial \mathcal{L}_D(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}^{(l)}} = \frac{1}{N} \sum_{n=1}^N \frac{\partial \mathcal{L}(\mathbf{y}^{(n)}, \hat{\mathbf{y}}^{(n)})}{\partial \mathbf{W}^{(l)}}$$

$$\frac{\partial \mathcal{L}_B(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}^{(l)}} = \frac{1}{B} \sum_{i=1}^B \frac{\partial \mathcal{L}(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})}{\partial \mathbf{W}^{(l)}}$$

如何计算梯度?

- 神经网络为一个复杂的复合函数

$$y = f^5(f^4(f^3(f^2(f^1(x, \theta^{(1)}), \theta^{(2)}), \theta^{(3)}), \theta^{(4)}), \theta^{(5)})$$

链式法则



$$\frac{\partial y}{\partial \theta^{(1)}} = \frac{\partial f^1}{\partial \theta^{(1)}} \frac{\partial f^2}{\partial f^1} \frac{\partial f^3}{\partial f^2} \frac{\partial f^4}{\partial f^3} \frac{\partial f^5}{\partial f^4}$$

- 反向传播算法

根据前馈网络的特点而设计的高效方法

- 一个更加通用的计算方法

自动微分 (Automatic Differentiation, AD)

链式法则

- 标量: $z = f(y) = f(g(x))$

- $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

- 向量: $\mathbf{y} = g(\mathbf{x})$ 且 $z = f(\mathbf{y})$

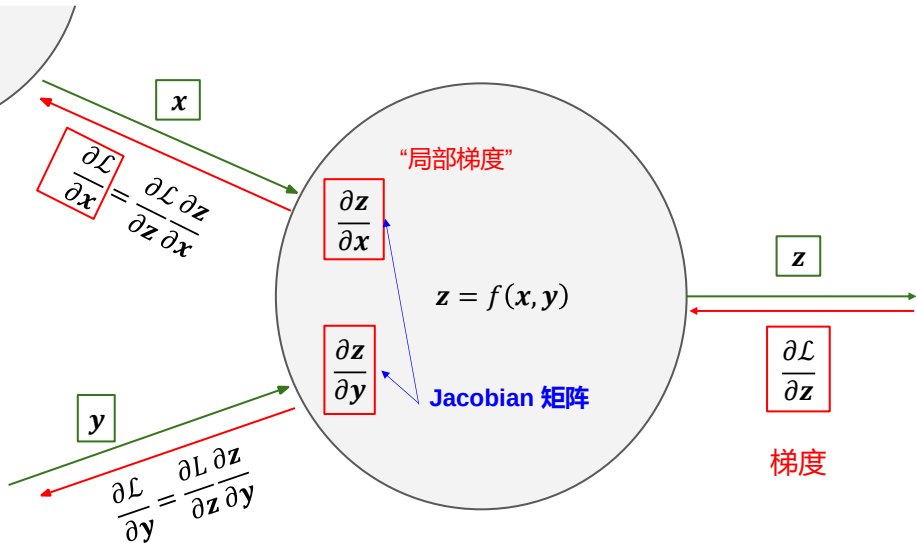
- $\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

- 张量: $\mathbf{Y} = g(\mathbf{X})$ 且 $z = f(\mathbf{Y})$

- $\nabla_{\mathbf{x}} z = \sum_j (\nabla_{\mathbf{x}} Y_j) \frac{\partial z}{\partial Y_j}$

链式法则



后向传播算法

$$\mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \mathbf{a}^{(2)} \rightarrow \dots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)}$$

$$\text{仿射变换} \quad \mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \quad \mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}) \quad \text{非线性变换}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{w}_{ij}^{(l)}} = \left\langle \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}, \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{w}_{ij}^{(l)}} \right\rangle$$

误差项 $\mathbf{e}^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$

$\mathbf{W}^{(l)}$ 的第 k 行

$$z_k^{(l)} = \langle \mathbf{w}_k^{(l)}, \mathbf{a}^{(l-1)} \rangle + b_k^{(l)}$$

$$\frac{\partial z_k^{(l)}}{\partial \mathbf{w}_i^{(l)}} = \delta(k = i) \mathbf{a}^{(l-1)}$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \left\langle \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}, \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \right\rangle$$

$$\frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} = \mathbf{1}_{m^{(l)}} \quad \text{单位阵}$$

后向传播算法

$$x = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \mathbf{a}^{(2)} \rightarrow \dots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)}$$

仿射变换 $\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ $\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}) \Rightarrow a_k = f_l(z_k^{(l)})$

误差项 $\mathbf{e}^{(l)} = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l)}}$

$$\begin{aligned} & \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} = \mathbf{W}^{(l+1)} \\ & = \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{z}^{(l+1)}} \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \\ & = \mathbf{e}^{(l+1)} \mathbf{W}^{(l+1)} \text{diag}(\nabla f_l(\mathbf{z}^{(l)})) \\ & \qquad \qquad \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} = \frac{\partial f_l(\mathbf{z}^{(l)})}{\partial \mathbf{z}^{(l)}} \\ & \qquad \qquad \qquad = \text{diag}(\nabla f_l(\mathbf{z}^{(l)})) \\ & = \nabla f_l(\mathbf{z}^{(l)}) \odot (\mathbf{e}^{(l+1)} \mathbf{W}^{(l+1)}) \end{aligned}$$

后向传播算法

$$\frac{\partial z_k^{(l)}}{\partial w_i^{(l)}} = \delta(k=i) \mathbf{a}^{(l-1)}$$



$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial w_{ij}^{(l)}} = \langle \mathbf{e}^{(l)}, \frac{\partial \mathbf{z}^{(l)}}{\partial w_{ij}^{(l)}} \rangle = e_i^{(l)} a_j^{(l-1)} \quad \rightarrow \quad \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{(l)}} = \mathbf{e}^{(l)} (\mathbf{a}^{(l-1)})^\top$$

$$\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{(l)}} = \langle \mathbf{e}^{(l)}, \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{b}^{(l)}} \rangle = \mathbf{e}^{(l)}$$

$$\mathbf{e}^{(l)} = \nabla f_l(\mathbf{z}^{(l)}) \odot (\mathbf{e}^{(l+1)} \mathbf{W}^{(l+1)})$$

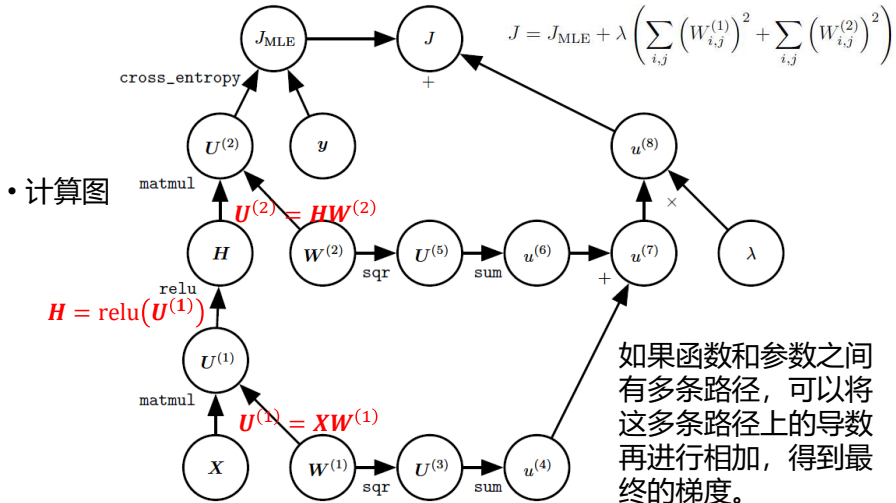
计算图与自动微分

- 自动微分是利用链式法则来自动计算一个复合函数的梯度

自动微分将符号微分应用于最基本的运算（如常数，幂函数，指数函数，对数函数，三角函数等基本函数），代入自变量的值得到其导数值，作为中间结果进行保留。然后再根据这些基本运算单元的求导结果计算出整个函数的导数值。

计算图与自动微分

- 自动微分是利用链式法则来自动计算一个复合函数的梯度



计算图与自动微分

- $u^{(8)} = \lambda u^{(7)}$
- $u^{(7)} = u^{(6)} + u^{(4)}$
- $u^{(6)} = \langle U^{(5)}, \mathbf{1} \rangle$
- $u^{(4)} = \langle U^{(3)}, \mathbf{1} \rangle$
- $U^{(5)} = W^{(2)} \odot W^{(2)}$
- $U^{(3)} = W^{(1)} \odot W^{(1)}$

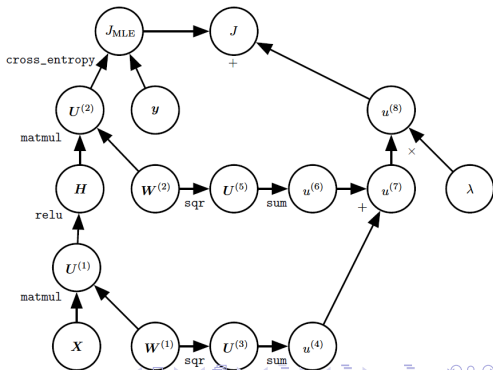
$$\frac{\partial U_{ij}^{(5)}}{\partial W_{ij}^{(2)}} = 2W_{ij}^{(2)} \Rightarrow \frac{\partial u^{(6)}}{\partial W^{(2)}} = 2W^{(2)}$$

$$\frac{\partial U_{ij}^{(3)}}{\partial W_{ij}^{(1)}} = 2W_{ij}^{(1)} \Rightarrow \frac{\partial u^{(4)}}{\partial W^{(1)}} = 2W^{(1)}$$

$$\frac{\partial u^{(8)}}{\partial u^{(7)}} = \lambda$$

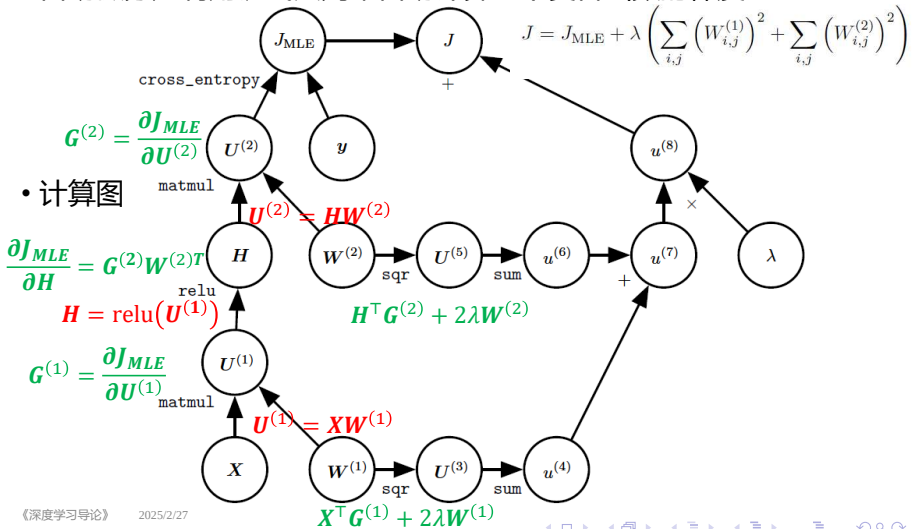
$$\frac{\partial u^{(7)}}{\partial u^{(6)}} = 1 \quad \frac{\partial u^{(7)}}{\partial u^{(4)}} = 1$$

$$\frac{\partial u^{(6)}}{\partial U^{(5)}} = 1 \quad \frac{\partial u^{(4)}}{\partial U^{(3)}} = 1$$



计算图与自动微分

- 自动微分是利用链式法则来自动计算一个复合函数的梯度。



静态计算图和动态计算图

- 静态计算图是在编译时构建计算图，计算图构建好之后在程序运行时不能改变。
 - Theano和Tensorflow

采用计算图以及添加一些额外的节点到计算图中，这些额外的节点提供了我们所需导数的符号描述

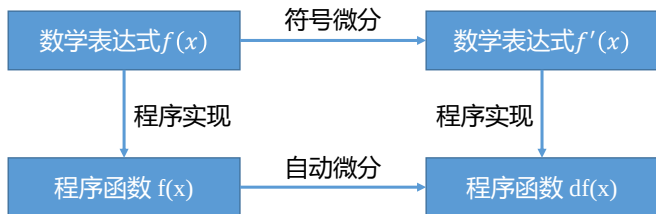
优点：构建时可以进行优化，并行能力强
缺点：灵活性比较低。

- 动态计算图是在程序运行时动态构建。
 - DyNet, Chainer和PyTorch

采用计算图和一组输入数值，然后返回在这些输入值处梯度的一组数值

优点：不容易优化，难以并行计算
缺点：灵活性比较高

符号微分与自动微分



反向传播算法

- 前馈神经网络的训练过程可以分为以下三步

前向计算：每一层的状态和激活值，直到最后一层

反向计算：每一层的参数的偏导数

更新参数

深度学习的三个步骤



深度学习就是这么简单



Keras快速入门

<https://keras.io/examples/>

```
from keras.models import Sequential  
from keras.layers import Dense, Activation  
from keras.optimizers import SGD
```

```
model = Sequential()  
model.add(Dense(output_dim=64, input_dim=100))  
model.add(Activation("relu"))  
model.add(Dense(output_dim=10))  
model.add(Activation("softmax"))
```

```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

```
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)
```

```
loss = model.evaluate(X_test, Y_test, batch_size=32)
```

Pytorch快速入门

```
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
```

```
model = NeuralNet(input_size, hidden_size, num_classes).to(device)
```

Loss and optimizer

```
criterion = nn.CrossEntropyLoss()
```

```
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

Train the model

```
total_step = len(train_loader)
```

```
for epoch in range(num_epochs):
```

```
    for i, (images, labels) in enumerate(train_loader):
```

Move tensors to the configured device

```
        images = images.reshape(-1, 28*28).to(device)
```

```
        labels = labels.to(device)
```

Forward pass

```
        outputs = model(images)
```

```
        loss = criterion(outputs, labels)
```

Backward and optimize

```
        optimizer.zero_grad()
```

```
        loss.backward()
```

```
        optimizer.step()
```

编程作业

- 使用pytorch或者tensorflow手写一个前馈神经网络，用于近似函数 $y = \sin(x) + \cos(x) + \sin(x)\cos(x)$, $x \in [0, 2\pi)$ ，研究网络深度、学习率、网络宽度、激活函数对模型性能的影响



卷积神经网络

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>



本章内容

- 卷积与相关
- 卷积的动机
- 池化
- 卷积神经网络及其典型网络结构

一维卷积

- 卷积经常用在信号处理中，用于计算信号的延迟累积。
- 假设一个**信号发生器**每个时刻 t 产生一个信号 x_t ，其信息的衰减率为 w_k ，即在 $k-1$ 个时间步长后，信息为原来的 w_k 倍
 - 假设 $w_1 = 1, w_2 = 1/2, w_3 = 1/4$
- 时刻 t 收到的信号 y_t 为当前时刻产生的信息和以前时刻延迟信息的叠加

$$y_t = 1 \times x_t + 1/2 \times x_{t-1} + 1/4 \times x_{t-2}$$

$$= w_1 \times x_t + w_2 \times x_{t-1} + w_3 \times x_{t-2}$$

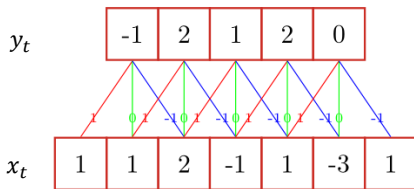
$$= \sum_{k=1}^3 w_k \times x_{t-k+1}$$


滤波器 (filter) 或卷积核 (convolution kernel)

一维卷积

- 给定一个输入信号序列 x 和滤波器 w ,卷积的输出为:

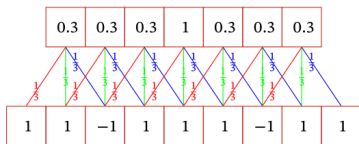
$$y_t = \sum_{k=1}^K w_k \times x_{t-k+1}$$



Filter: [-1,0,1]

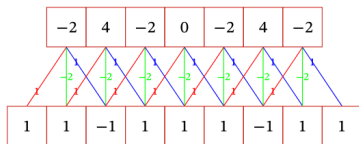
一维卷积

- 不同的滤波器来提取信号序列中的不同特征



(a) 滤波器 $[1/3, 1/3, 1/3]$

低频信息



(b) 滤波器 $[1, -2, 1]$

高频信息

$$y''(u) = y(u + 1) + y(u - 1) - 2y(u)$$

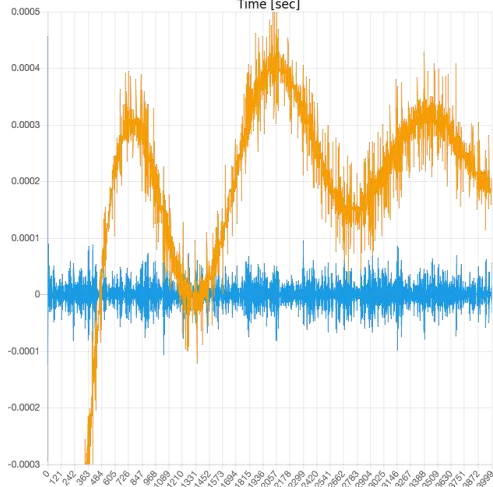
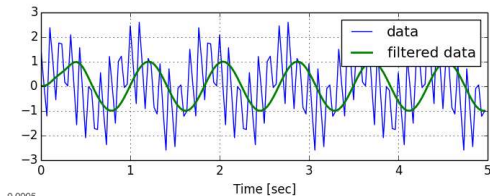
二阶微分

一维卷积



6

低通滤波



高通滤波

图像是一种波



一张 400×400 的图片，一共包含了 16 万个像素点

二维卷积

- 在图像处理中，图像是以二维矩阵的形式输入到神经网络中，因此我们需要二维卷积。
- 一个输入信息 X 和滤波器 W 的二维卷积定义如下

$$Y = W * X$$

$$y_{ij} = \sum_{u=1}^U \sum_{v=1}^V w_{uv} x_{i-u+1, j-v+1}$$

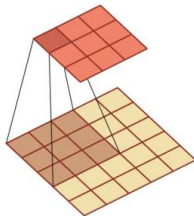
1	1	1	1	1
-1	0	-3	0	1
2	1	1	-1	0
0	-1	1	2	1
1	2	1	1	1

*

1	0	0
0	0	0
0	0	-1

=

0	-2	-1
2	2	4
-1	0	0

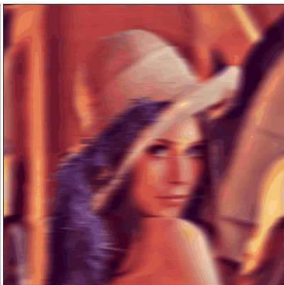


二维卷积

原始图像



低通滤波



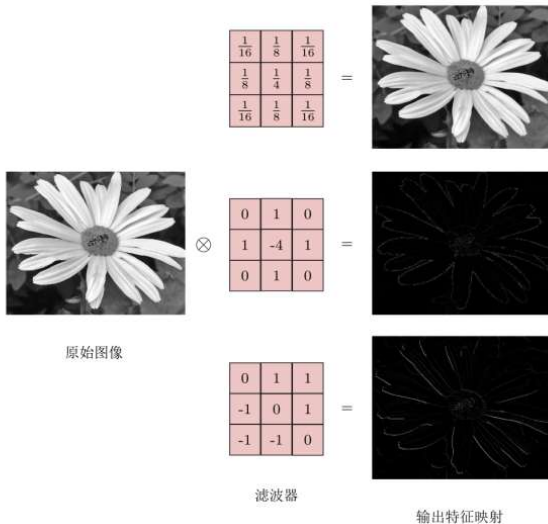
高通滤波



图像的高频区域变成低频，
即色彩变化剧烈的区域变得
平滑，也就是出现模糊效果

只保留那些变化最快速最
剧烈的区域，也就是图像
里面的物体边缘

二维卷积



互相关

- 计算卷积需要进行卷积核**翻转**
- 卷积操作的目标：**提取特征**

翻转是不必要的！

$$y_{ij} = \sum_{u=1}^U \sum_{v=1}^V w_{uv} x_{i-u+1, j-v+1}$$

卷积

1	1	1	1	1
-1	0	-3	0	1
2	1	1	-1	0
0	-1	1	2	1
1	2	1	1	1

*

1	0	0
0	0	0
0	0	-1



$$y_{ij} = \sum_{u=1}^U \sum_{v=1}^V w_{uv} x_{i+u+1, j+v+1}$$

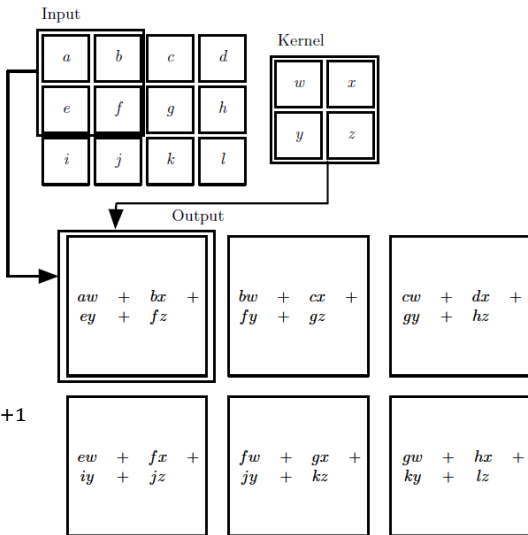
互相关

1	1	1	1	1
-1	0	-3	0	1
2	1	1	-1	0
0	-1	1	2	1
1	2	1	1	1

*

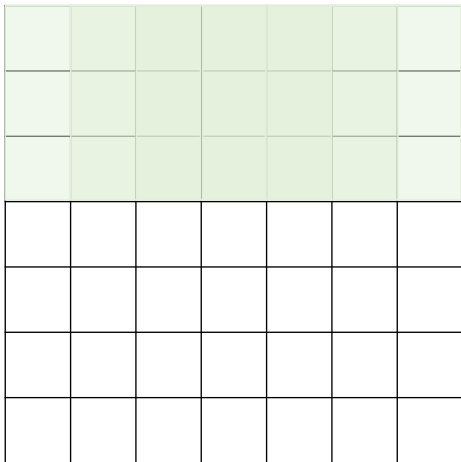
1	0	0
0	0	0
0	0	-1

互相关



$$y_{ij} = \sum_{u=1}^U \sum_{v=1}^V w_{uv} x_{i+u+1, j+v+1}$$

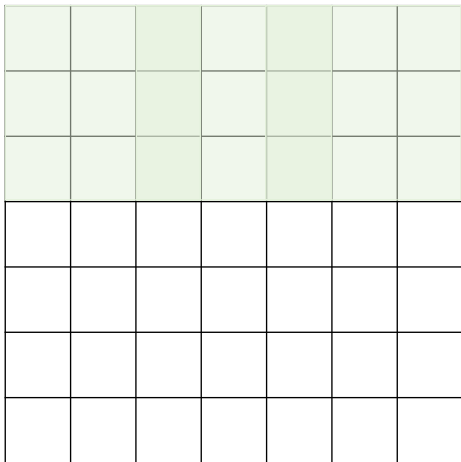
卷积



7x7的输入
3x3的核

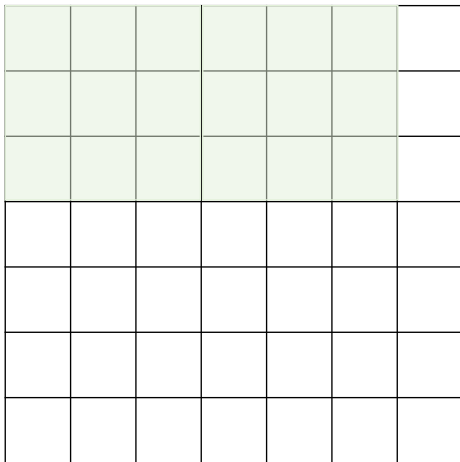
5x5的输出

卷积



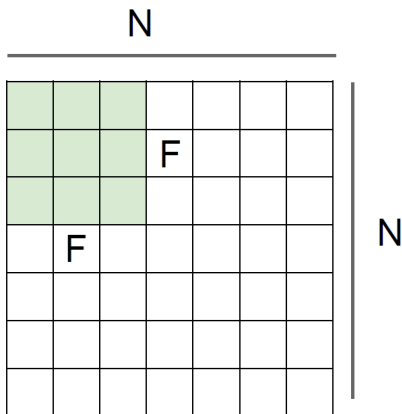
7x7的输入
3x3的核
步幅为2
3x3的输出

卷积



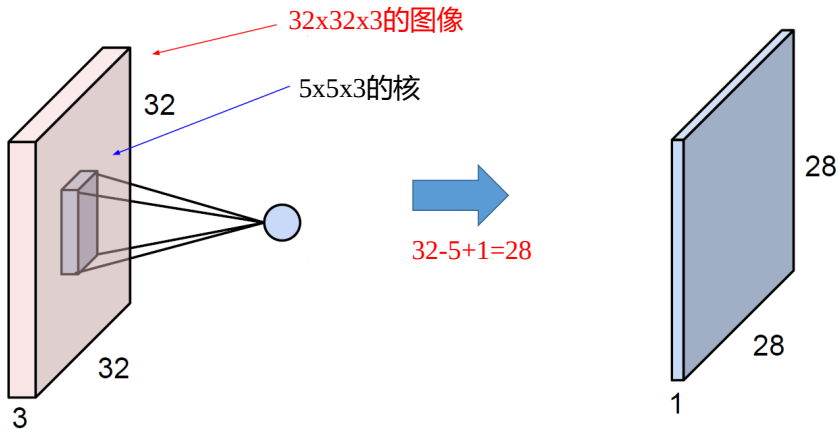
7x7的输入
3x3的核
步幅为3
不匹配

卷积

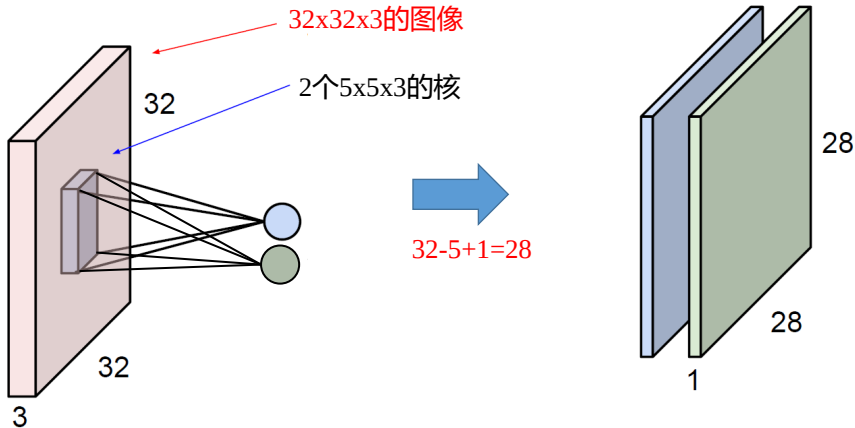


- 输出大小: $(N-F)/S+1$
 - S 为步幅大小
- 比如 $N=7, F=3$
 - 步幅为1, $(7-3)/1+1=5$
 - 步幅为2, $(7-3)/2+1=3$
 - 步幅为3, $(7-3)/3+1=2.3$

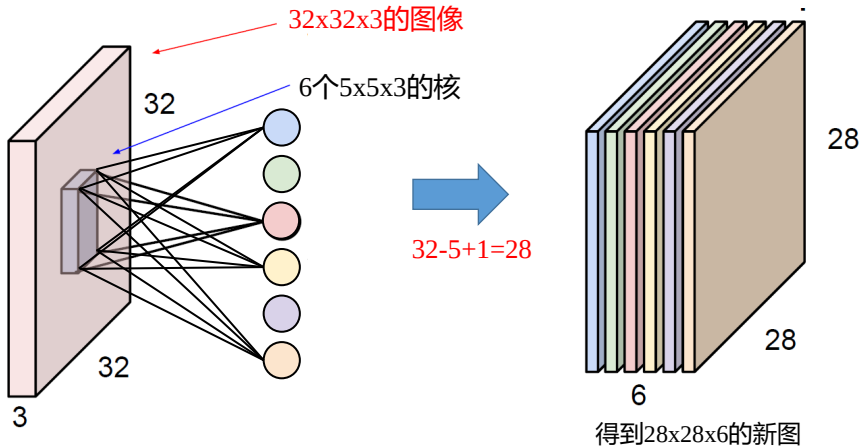
卷积



卷积

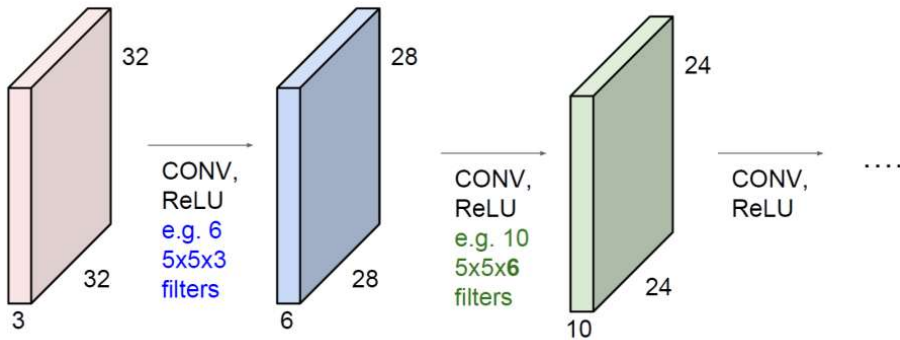


卷积

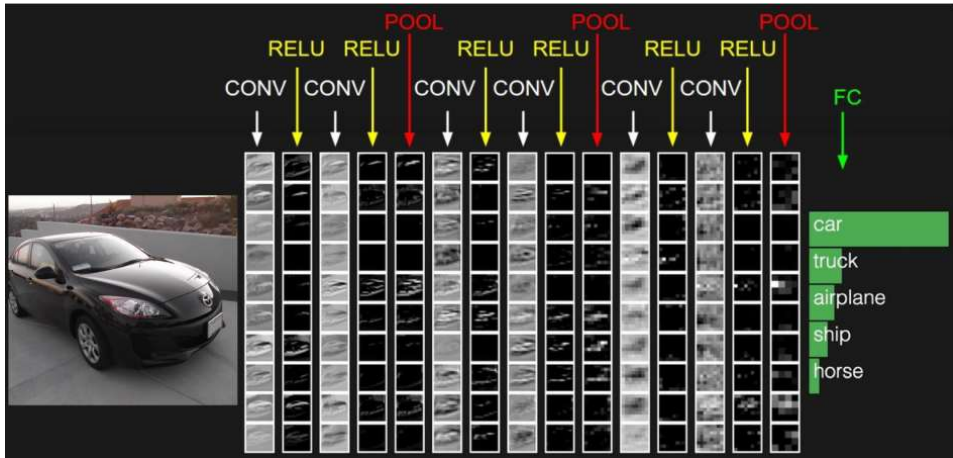


多层卷积

卷积神经网络是用卷积层代替全连接层

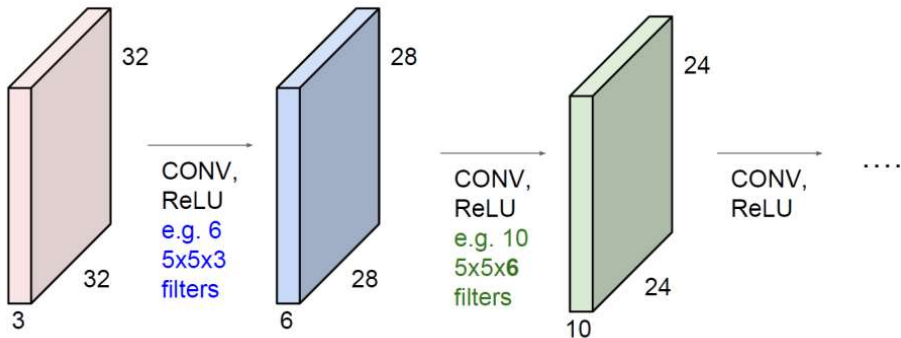


多层卷积



多层卷积

32x32 的输入，每次通过 5x5 的核卷积，大小都会发生改变!
(32 -> 28 -> 24 ...)。大小改变太快是不好的，实际中的性能很差



填充 (PADDING)

0	0	0	0	0	0			
0								
0								
0								
0								

- 在输入周边填0，以解决特征图大小改变太快问题

例子1

- 7x7的输入，3x3的核，步幅为1
- 填一个像素的边界，输入实际上是9x9
- 输出大小为： $(9-3)/1+1 = 7$
- 输出7x7，与输入相同大小

例子2

- 7x7的输入，3x3的核，步幅为3
- 填一个像素边界
- 输出大小为： $(9-3)/3+1=3$

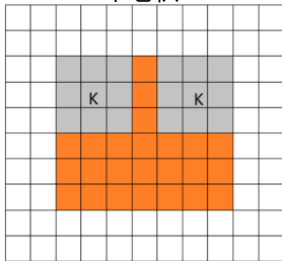
填充 (PADDING)

0	0	0	0	0	0			
0								
0								
0								
0								

- 填0时, 经常要求保持输出大小与输入大小一样
- 步幅为1, $F \times F$ 的核, 填 $(F-1)/2$ 的0
- 步幅为 S , $F \times F$ 的核, 填
 - $\frac{(S-1)N+F-S}{2}$ 的0

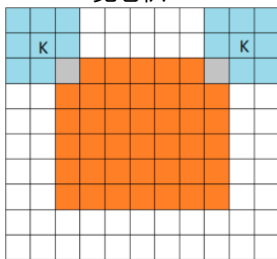
卷积模式

有效(valid)卷积
窄卷积



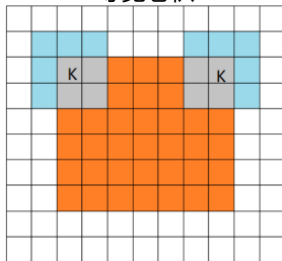
卷积核只在特征图内卷积

全(full)卷积
宽卷积



从卷积核和特征图刚相交
开始卷积

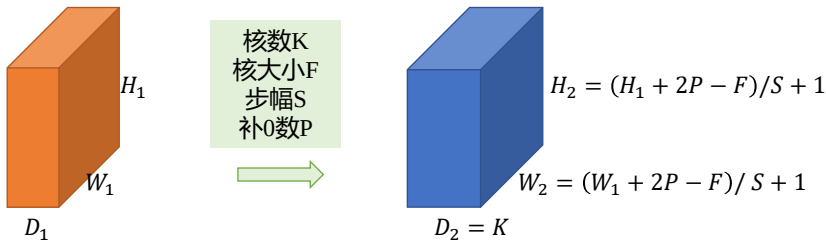
相同(same)卷积
等宽卷积



特征图的大小保持不变

通常零填充的最优数量（对于测试集的分类正确率）处于“有效卷积”和“相同卷积”之间的某个位置

卷积层的信息概括



常用设置：K为2的幂，32,64,128,512

- F=3, S=1, P=1
- F=5, S=1, P=2
- F=1, S=1, P=0

参数个数？

$F \times F \times D_1 \times K$ 个权重
K个偏置项

Pytorch中的卷积层

CLASS `torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')`

[SOURCE]

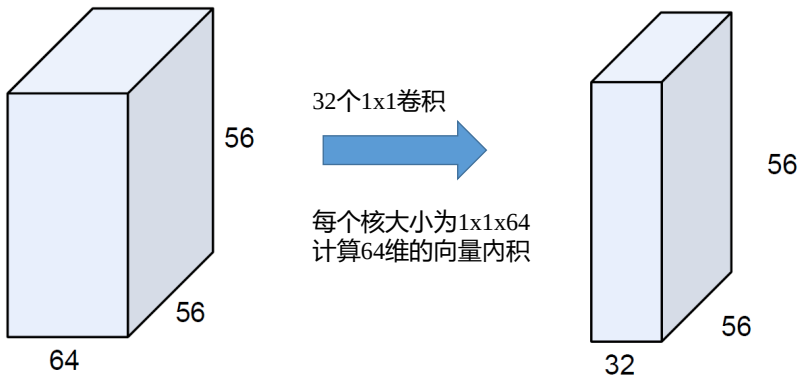
Parameters

- **in_channels** (*int*) – Number of channels in the input image
- **out_channels** (*int*) – Number of channels produced by the convolution
- **kernel_size** (*int* or *tuple*) – Size of the convolving kernel
- **stride** (*int* or *tuple*, optional) – Stride of the convolution. Default: 1
- **padding** (*int* or *tuple*, optional) – Zero-padding added to both sides of the input. Default: 0
- **padding_mode** (*string*, optional) – 'zeros', 'reflect', 'replicate' or 'circular'. Default: 'zeros'
- **dilation** (*int* or *tuple*, optional) – Spacing between kernel elements. Default: 1
- **groups** (*int*, optional) – Number of blocked connections from input channels to output channels. Default: 1
- **bias** (*bool*, optional) – If `True`, adds a learnable bias to the output. Default: `True`

```
>>> # With square kernels and equal stride
>>> m = nn.Conv2d(16, 33, 3, stride=2)
>>> # non-square kernels and unequal stride and with padding
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2))
>>> # non-square kernels and unequal stride and with padding and dilation
>>> m = nn.Conv2d(16, 33, (3, 5), stride=(2, 1), padding=(4, 2), dilation=(3, 1))
>>> input = torch.randn(20, 16, 50, 100)
>>> output = m(input)
```

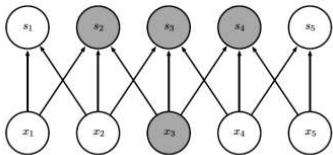
卷积

- 1×1 的卷积也有意义，可以方便一些运算

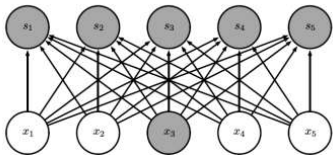


卷积的动机—稀疏链接

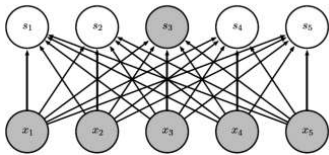
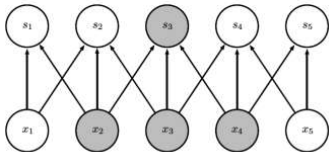
卷积
K=3



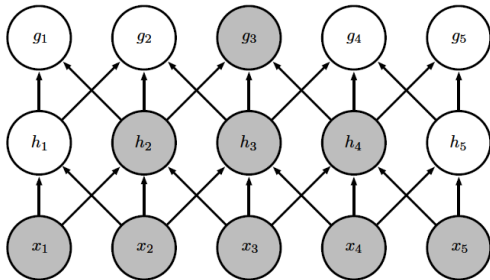
全连接



s_3 的接受域 (receptive field)



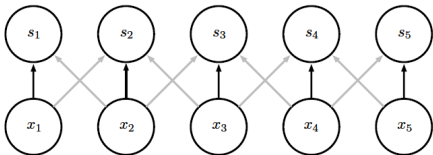
卷积的动机—稀疏链接



更深的层中的单元可以间接地连接到全部或者大部分输入图像

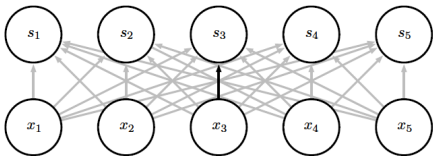
卷积的动机—参数共享

卷积
K=3



黑色箭头表示在卷积模型中对3元素核的中间元素的使用。因为参数共享，这个单独的参数被用于所有的输入位置

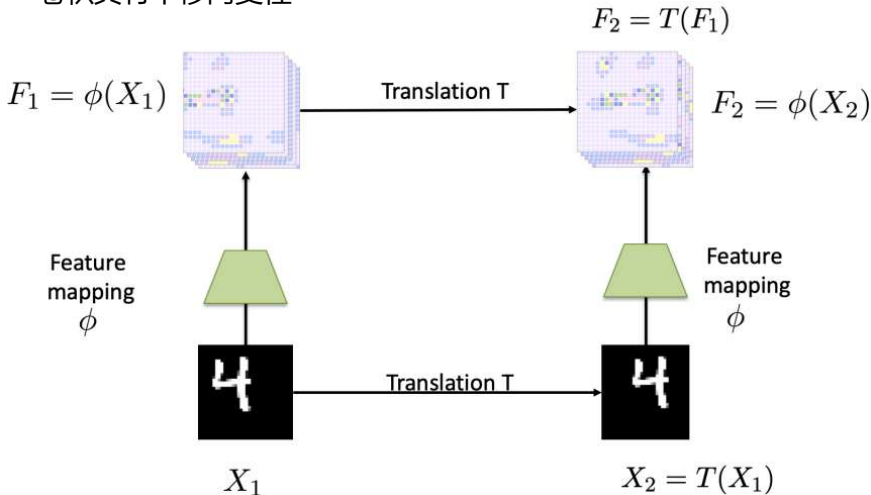
全连接



单独黑色箭头表示在全连接模型中对权重矩阵的中间元素的使用。这个模型没有使用参数共享，所以参数只使用了一次

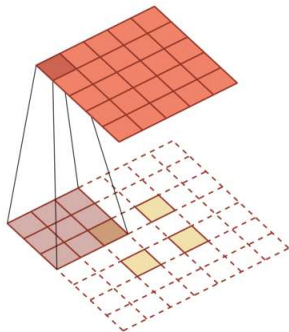
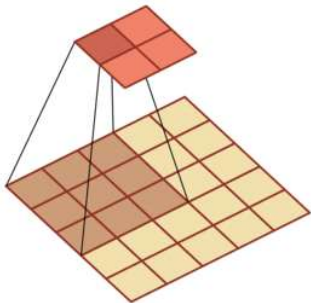
卷积的动机—平移同变性

- 卷积具有平移同变性



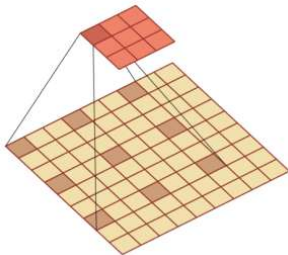
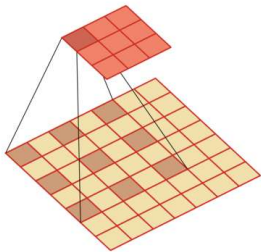
其它卷积种类—转置卷积/微步卷积

- 低维特征映射到高维特征



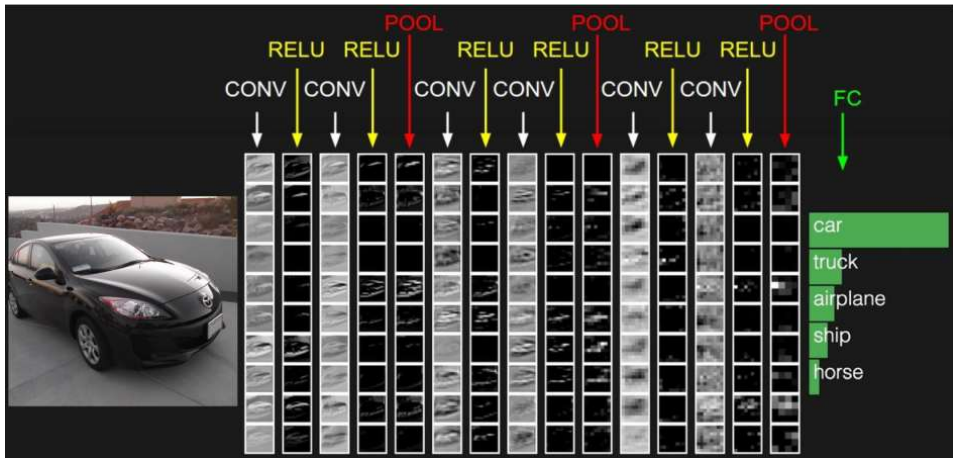
其它卷积种类—空洞卷积

- 如何增加输出单元的感受野
 - 增加卷积核的大小
 - 增加层数来实现
 - 在卷积之前进行汇聚操作
- 空洞卷积
 - 通过给卷积核插入“空洞”来变相地增加其大小。



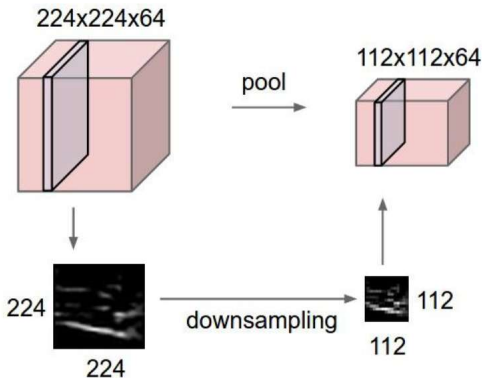
卷积神经网络

卷积网络是由卷积层、汇聚层、全连接层交叉堆叠而成



池化

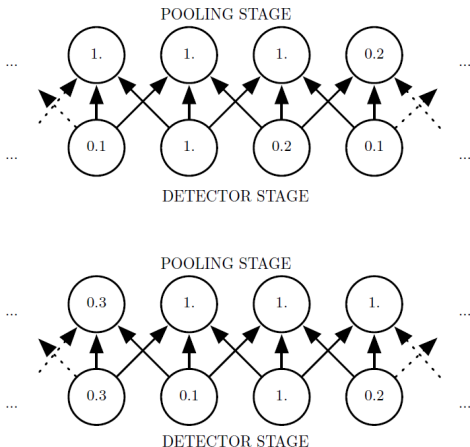
- 池化函数使用某一位置的相邻输出的总体统计特征来代替网络在该位置的输出，使得特征图尺寸更小且更易于管理
 - 最大池化 (max pooling) 函数给出相邻矩形区域内的最大值
 - 其他常用的池化函数包括相邻矩形区域内的平均值、 L_2 范数以及基于据中心像素距离的加权平均函数



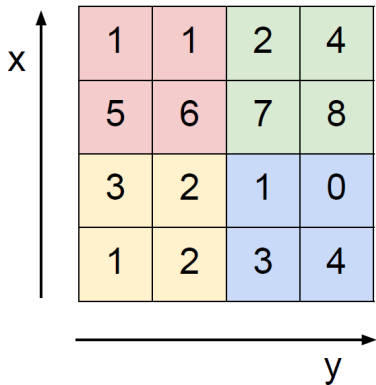
池化

- 数据表示进一步降低表示维度
- 对局部平移更加稳定

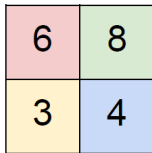
输入右移一位，输入全变，
但是输出只有一半的值发
生变化



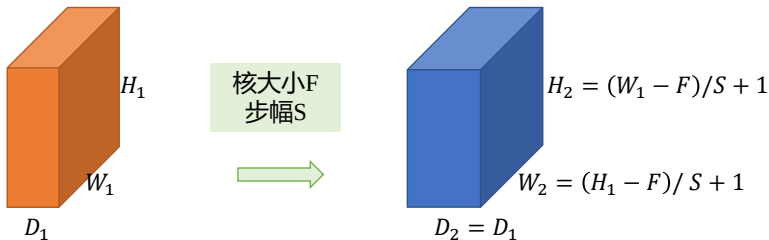
最大池化



2x2的核, 步幅为2



池化层的信息概括



常用设置:

- $F=2, S=2$
- $F=3, S=2$

Pytorch中的池化层

```
CLASS torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1,  
return_indices=False, ceil_mode=False)
```

[\[SOURCE\]](#)

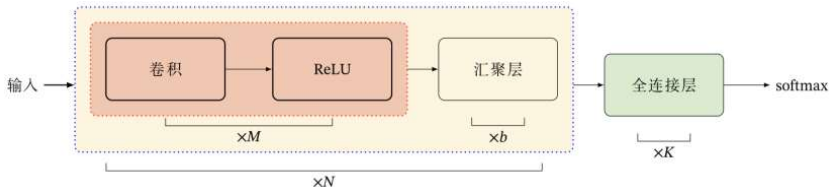
Parameters

- **kernel_size** – the size of the window to take a max over
- **stride** – the stride of the window. Default value is `kernel_size`
- **padding** – implicit zero padding to be added on both sides
- **dilation** – a parameter that controls the stride of elements in the window
- **return_indices** – if `True`, will return the max indices along with the outputs. Useful for `torch.nn.MaxUnpool2d` later
- **ceil_mode** – when `True`, will use *ceil* instead of *floor* to compute the output shape

```
>>> # pool of square window of size=3, stride=2  
>>> m = nn.MaxPool2d(3, stride=2)  
>>> # pool of non-square window  
>>> m = nn.MaxPool2d((3, 2), stride=(2, 1))  
>>> input = torch.randn(20, 16, 50, 32)  
>>> output = m(input)
```

卷积神经网络

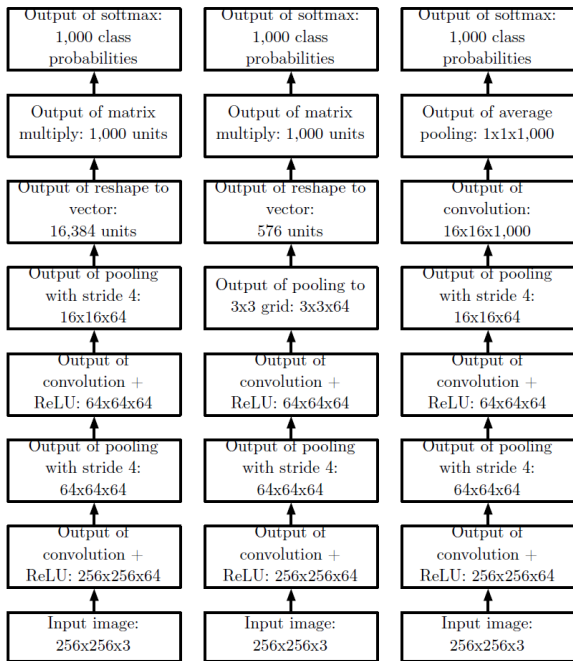
• 典型结构



一个卷积块为连续 M 个卷积层和 b 个汇聚层 (M 通常设置为 $2 \sim 5$, b 为0或1)。一个卷积网络中可以堆叠 N 个连续的卷积块, 然后在接着 K 个全连接层 (N 的取值区间比较大, 比如 $1 \sim 100$ 或者更大; K 一般为 $0 \sim 2$)。

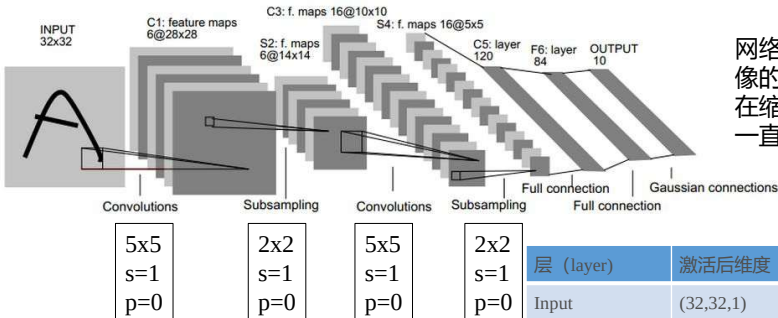
卷积神经网络

- 卷积网络用于分类的结构示例



典型的卷积网络—LeNet-5

网络越来越深，图像的宽度和高度都在缩小，信道数量一直在增加



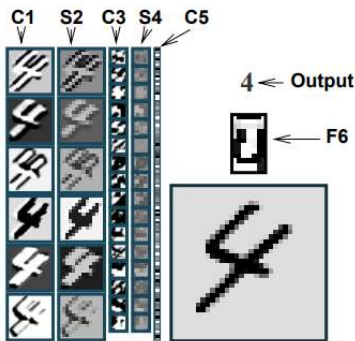
- 每个卷积层含：卷积、池化、非线性激活
- 使用卷积提取空间特征
- 降采样的平均池化层
- 双曲正切或S型的激活函数
- MLP作为最后的分类器

层 (layer)	激活后维度	参数个数
Input	(32,32,1)	0
CONV1(f=5,s=1)	(28,28,6)	$(5*5+1)*6=156$
POOL1	(14,14,6)	0
CONV2(f=5,s=1)	(10,10,16)	$(5*5*6+1)*16=2416$
POOL2	(5,5,16)	0
FC3	(120,1)	$120*(400+1)=48120$
FC4	(84,1)	$84*(120+1)=10164$
Softmax	(10,1)	$10*(84+1)=850$

典型的卷积网络—LeNet-5

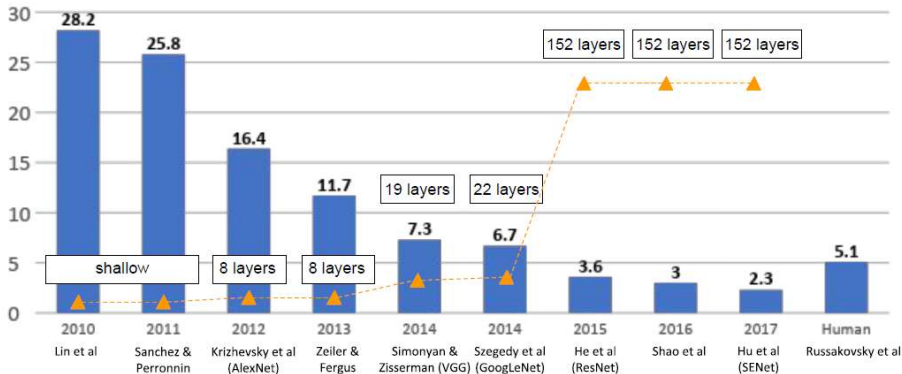
- MNIST (handwritten digits) 数据集

6万训练样本和1万测试样本

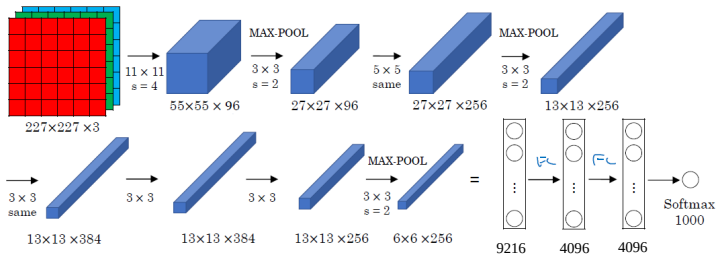


错误案例

Large Scale Visual Recognition Challenge



典型的卷积网络—AlexNet



[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

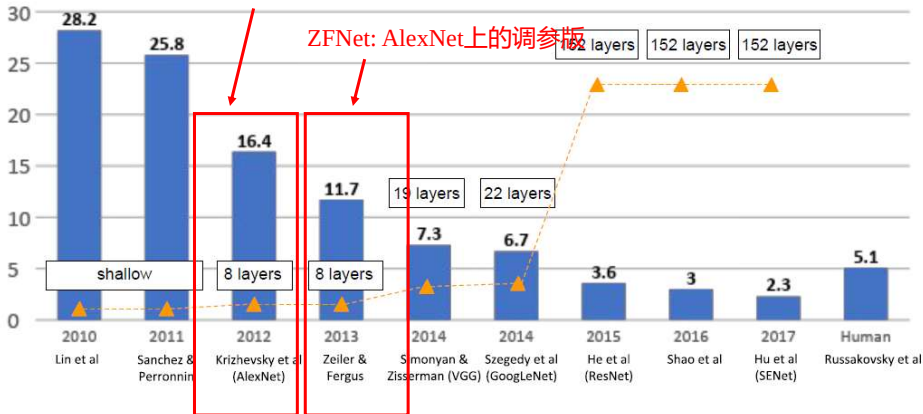
[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

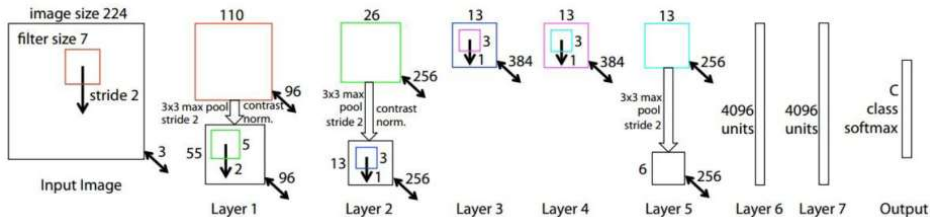
Large Scale Visual Recognition Challenge

第一个基于CNN的冠军

ZFNet: AlexNet上的调参版



典型的卷积网络—ZFNet



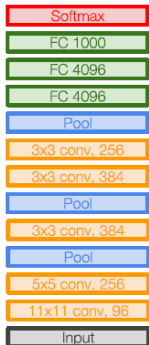
- 第一层卷积: 11x11 步幅 4 → 7x7 步幅 2
- 卷积层3, 4, 5: 核数从384, 384, 256 → 512, 1024, 512
- 通过可视化发现第一层的卷积核影响大, 对第一层的卷积核做了规范化

ImageNet top 5 error: 16.4% -> 11.7%

Large Scale Visual Recognition Challenge



典型的卷积网络—VGG



AlexNet

11.7% top 5 error in ILSVRC'13 (ZFNet)

- 更小的核，更小的步幅，更深的网络
- 8层 → 16-19层
- 只有3x3的卷积核，步幅为1，补1个0
- 2x2的池化核，步幅为2

→ 7.3% top 5 error in ILSVRC'14

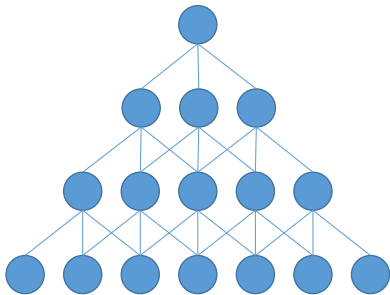


VGG16

VGG19

典型的卷积网络—VGG

- 为什么更小的核 (3x3)
 - 3个3x3的卷积层的堆放的有效感受野和一个7x7的相同



VGG16

VGG19

典型的卷积网络—VGG

- 为什么更小的核 (3x3)
 - 3个3x3的卷积层的堆放的有效感受野和一个7x7的相同
 - 更深的卷积层，非线性更强
 - 参数更少

假设每层C通道，参数个数

$$3(3^2C^2) \text{ vs. } 7^2C^2$$



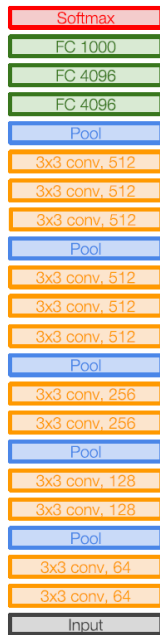
VGG16

VGG19

- INPUT: [224x224x3] **memory: 224*224*3=150K** params: 0
- CONV3-64: [224x224x64] **memory: 224*224*64=3.2M** params: $(3*3*3)*64 = 1,728$
- CONV3-64: [224x224x64] **memory: 224*224*64=3.2M** params: $(3*3*64)*64 = 36,864$
- POOL2: [112x112x64] **memory: 112*112*64=800K** params: 0
- CONV3-128: [112x112x128] **memory: 112*112*128=1.6M** params: $(3*3*64)*128 = 73,728$
- CONV3-128: [112x112x128] **memory: 112*112*128=1.6M** params: $(3*3*128)*128 = 147,456$
- POOL2: [56x56x128] **memory: 56*56*128=400K** params: 0
- CONV3-256: [56x56x256] **memory: 56*56*256=800K** params: $(3*3*128)*256 = 294,912$
- CONV3-256: [56x56x256] **memory: 56*56*256=800K** params: $(3*3*256)*256 = 589,824$
- CONV3-256: [56x56x256] **memory: 56*56*256=800K** params: $(3*3*256)*256 = 589,824$
- POOL2: [28x28x256] **memory: 28*28*256=200K** params: 0
- CONV3-512: [28x28x512] **memory: 28*28*512=400K** params: $(3*3*256)*512 = 1,179,648$
- CONV3-512: [28x28x512] **memory: 28*28*512=400K** params: $(3*3*512)*512 = 2,359,296$
- CONV3-512: [28x28x512] **memory: 28*28*512=400K** params: $(3*3*512)*512 = 2,359,296$
- POOL2: [14x14x512] **memory: 14*14*512=100K** params: 0
- CONV3-512: [14x14x512] **memory: 14*14*512=100K** params: $(3*3*512)*512 = 2,359,296$
- CONV3-512: [14x14x512] **memory: 14*14*512=100K** params: $(3*3*512)*512 = 2,359,296$
- CONV3-512: [14x14x512] **memory: 14*14*512=100K** params: $(3*3*512)*512 = 2,359,296$
- POOL2: [7x7x512] **memory: 7*7*512=25K** params: 0
- FC: [1x1x4096] **memory: 4096** params: $7*7*512*4096 = 102,760,448$
- FC: [1x1x4096] **memory: 4096** params: $4096*4096 = 16,777,216$
- FC: [1x1x1000] **memory: 1000** params: $4096*1000 = 4,096,000$

TOTAL memory: 24M * 4 bytes ~= 96MB / image

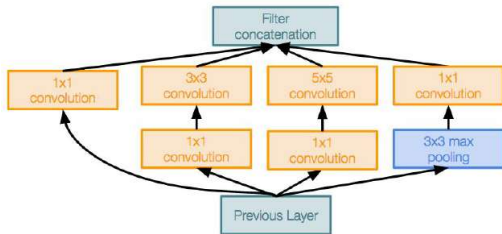
TOTAL params: 138M parameters



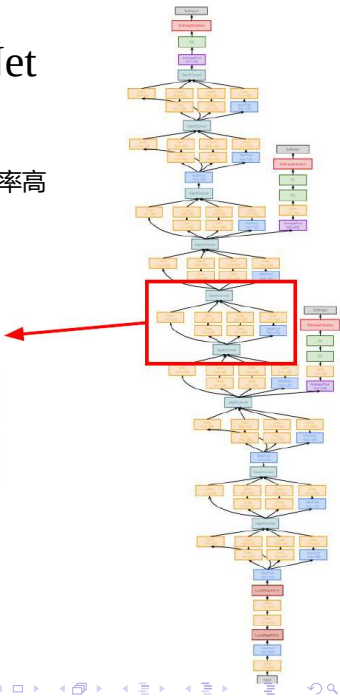
VGG16

典型的卷积网络—GoogLeNet

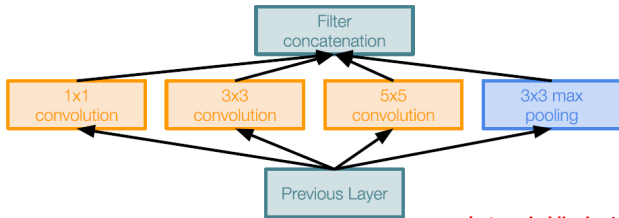
- 2014 ILSVRC winner (6.7% top 5 error)
 - 更深的网络（22层），没有FC层，且计算效率高
 - 只有5M 参数，比AlexNet的60M参数的1/12
 - 由多个inception模块和少量的汇聚层堆叠而成



Inception module



典型的卷积网络—GoogLeNet



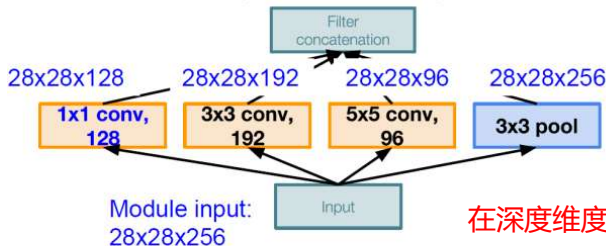
在深度维度上拼接滤波器的输出

Naive Inception module

- 在卷积网络中，如何设置卷积层的卷积核大小是一个十分关键的问题
- Inception同时使用 1×1 、 3×3 、 5×5 等不同大小的卷积核（对应不同感受野），将得到的特征映射在深度上拼接起来作为输出特征映射

典型的卷积网络—GoogLeNet

$$28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$$



在深度维度上拼接滤波器的输出

Naive Inception module

计算复杂度大

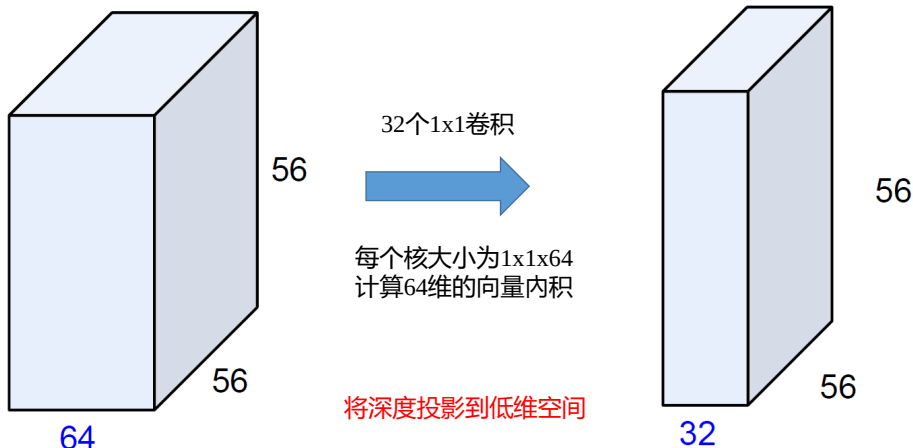
• Conv Ops:

- $[1 \times 1 \text{ conv}, 128]$ $28 \times 28 \times 128 \times 1 \times 1 \times 256$
- $[3 \times 3 \text{ conv}, 192]$ $28 \times 28 \times 192 \times 3 \times 3 \times 256$
- $[5 \times 5 \text{ conv}, 96]$ $28 \times 28 \times 96 \times 5 \times 5 \times 256$

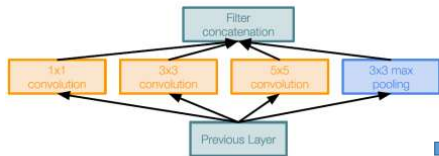
• Total: 854M ops

典型的卷积网络—GoogLeNet

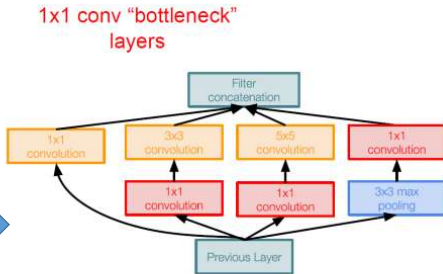
- 使用“bottleneck”层 (1×1 卷积) 来减少特征深度



典型的卷积网络—GoogLeNet

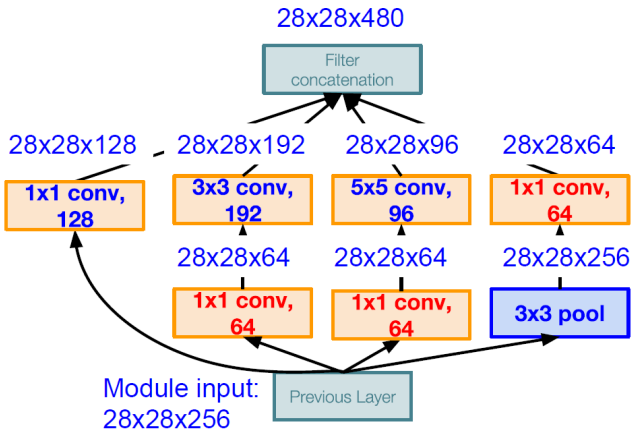


Naive Inception module



Inception module with dimension reduction

典型的卷积网络—GoogLeNet



• Conv Ops:

- [1x1 conv, 64]
28x28x64x1x1x256
- [1x1 conv, 64]
28x28x64x1x1x256
- [1x1 conv, 128]
28x28x128x1x1x256
- [3x3 conv, 192]
28x28x192x3x3x64
- [5x5 conv, 96]
28x28x96x5x5x64
- [1x1 conv, 64]
28x28x64x1x1x256

从Total: 854M ops降到了Total: 358M ops

Inception模块 v3

- 用多层的小卷积核来替换大的卷积核，以减少计算量和参数量。
 - 使用两层 3×3 的卷积来替换v1中的 5×5 的卷积
 - 使用连续的 $n \times 1$ 和 $1 \times n$ 来替换 $n \times n$ 的卷积。

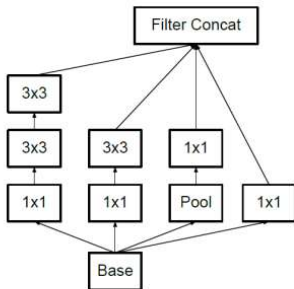


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle [3] of Section [2].

<http://blog.csdn.net/xbinworld>

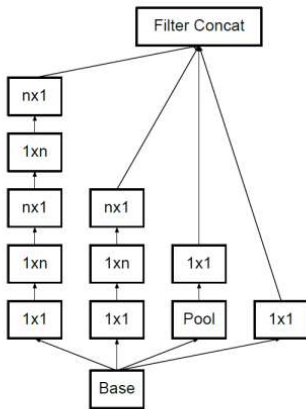
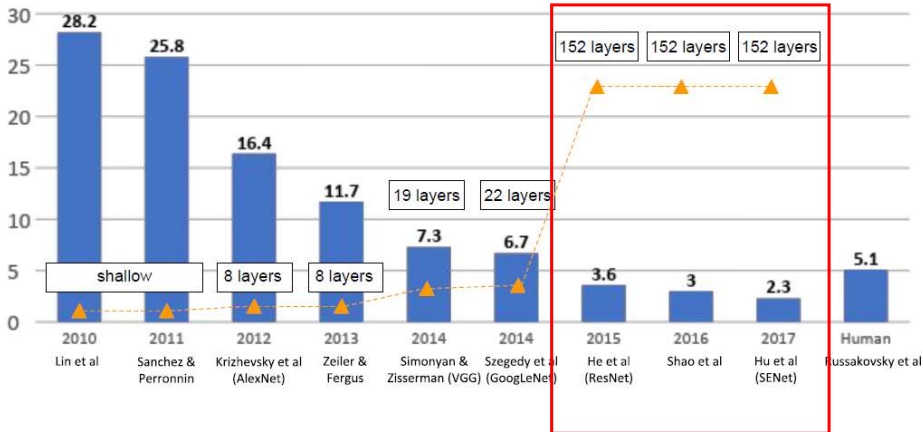


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle [3])

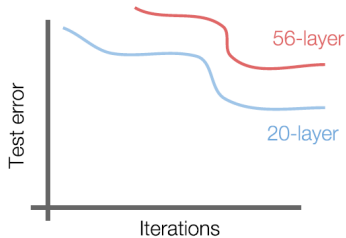
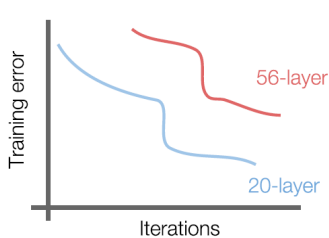
Large Scale Visual Recognition Challenge

“深度” 革命



典型的卷积网络—ResNet

- 在卷积神经网络上不断加深网络，训练集的错误率会上升



不是过拟合导致

典型的卷积网络—ResNet

- 这种现象背后的问题是优化问题，越深的模型越难训练
- 但是直观上深层模型不会比浅层模型更差，因为可以通过如下方式构造出这样的深层模型
 - 将浅层模型的所有层拷贝到深层模型的低层，其余的高层都设置为恒等映射

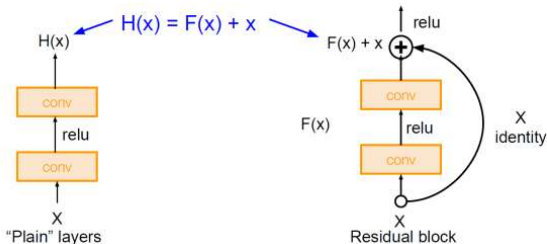
典型的卷积网络—ResNet

- 残差网络 (Residual Network, ResNet) 是通过给非线性的卷积层增加**直连边**的方式来提高信息的传播效率。
 - 假设在一个深度网络中, 我们期望一个非线性单元 (可以为一层或多层的卷积层) $F(x, \theta)$ 去逼近一个目标函数为 $h(x)$ 。
 - 将目标函数拆分成两部分: **恒等函数**和**残差函数**

$$h(x) = \underbrace{x}_{\text{恒等函数}} + \underbrace{(h(x) - x)}_{\text{残差函数}}$$

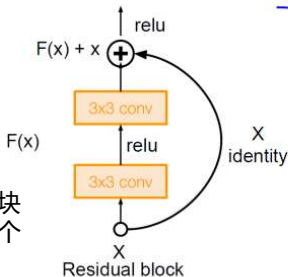
恒等函数 残差函数

$$F(x, \theta)$$

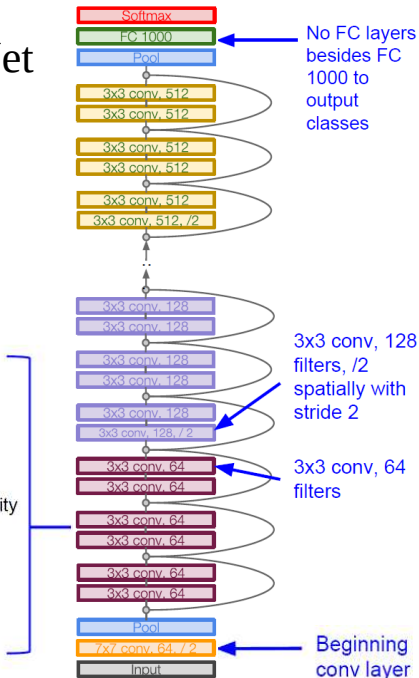


典型的卷积网络—ResNet

- 2015 ILSVRC winner (152层, 3.57% top 5 error)
- 用残差连接的极深网络
- 在ILSVRC'15和COCO'15的所有分类和检测比赛中横扫所有对手

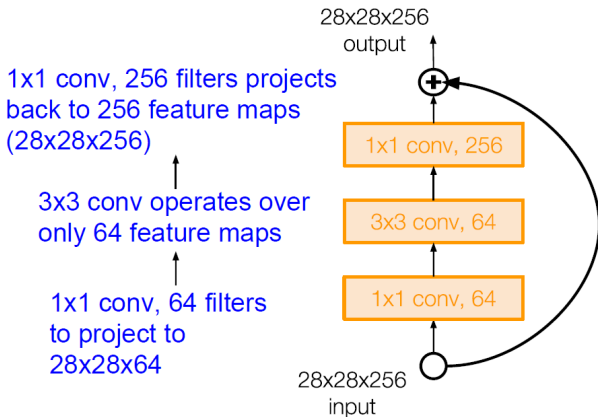


- 残差网络堆砌残差块
- 每个残差块里有两个 3×3 的卷积层

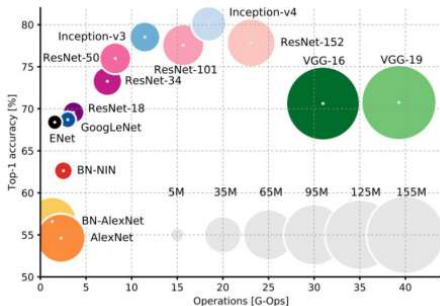
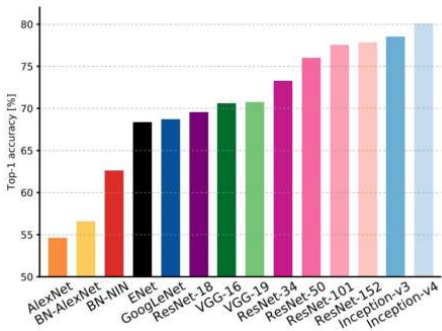


典型的卷积网络—ResNet

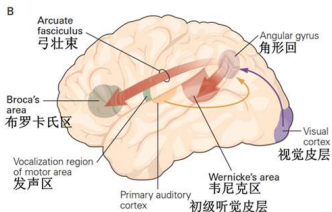
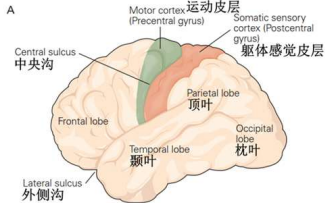
- 利用“bottleneck”层改进效率



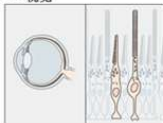
典型的卷积网络—复杂度对比



卷积神经网络的神经科学基础



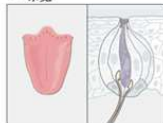
Vision 视觉



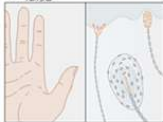
Smell 嗅觉



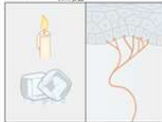
Taste 味觉



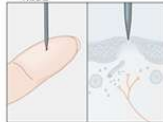
Touch 触觉



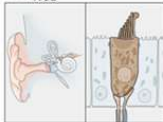
Thermal senses 热觉



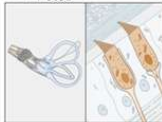
Pain 痛觉



Hearing 听觉



Balance 平衡觉

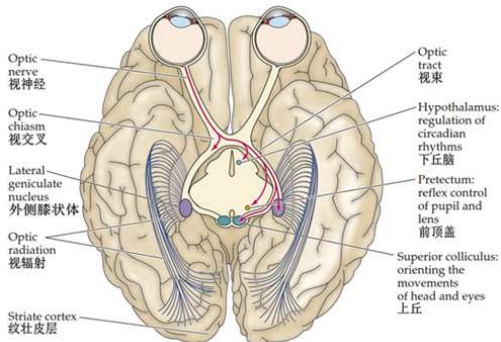


Proprioception 本体感觉



卷积神经网络的神经科学基础

- 图像信息沿着视神经由眼传递到大脑
- 来自不同眼球的视神经在视交叉发生交汇
- 右侧视野的信息在左视束中进行传递，终止于丘脑的外侧膝状体核（LGN）
- 外侧膝状体核（LGN）是丘脑的一个感觉中继核团
- 视辐射将信息由外侧膝状体核传递到视皮层

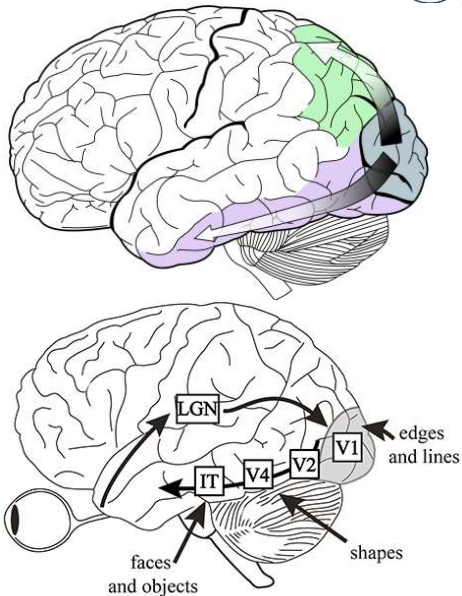


卷积神经网络的神经科学基础

- 卷积神经网络被设计为描述V1的三个性质
 - V1 可以进行空间映射
 - V1 包含许多简单细胞，在一个小的空间位置感受野内的图像的线性函数
 - 卷积网络的检测器单元被设计为模拟简单细胞的这些性质
 - 还包括许多复杂细胞，特征的位置微小偏移、照明变化具有不变性
 - 启发了池化操作

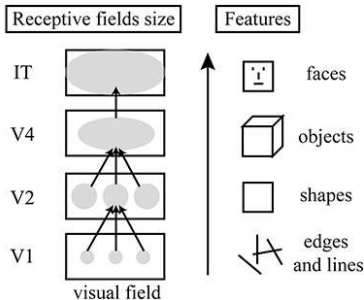
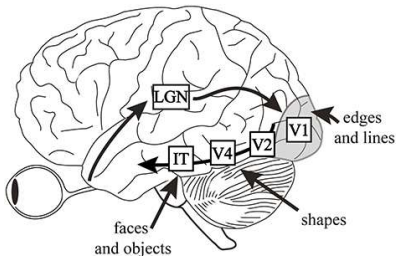
卷积神经网络的神经科学基础

- 初级视皮层 (V1) 的输出信息出送到两个渠道，分别成为背侧流和腹侧流
 - 背侧流起始于V1，通过V2，进入背内侧区和中颞区 (MT，亦称V5)，然后抵达顶下小叶。背侧流常被称为“空间通路”，参与处理物体的空间位置信息以及相关的运动控制，例如眼跳和伸取。
 - 腹侧流起始于V1，依次通过V2，V4，进入下颞叶 (IT)。该通路常被称为“内容通路”，参与物体识别，例如面孔识别。该通路也于长期记忆有关



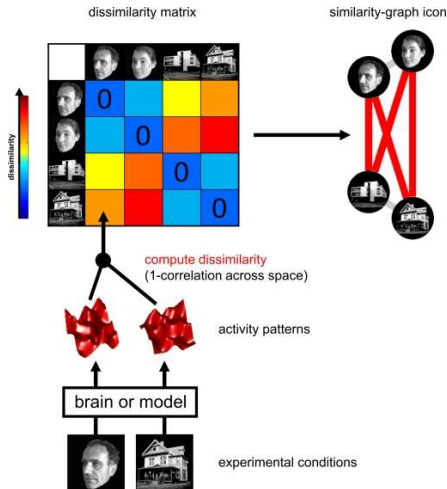
卷积神经网络的神经科学基础

- 信息从V1传播到IT，可视野越来越大
- 神经元所响应的图像特征也越来越复杂，从V1中的简单线条到IT中的目标部分
- 人工网络中更深的层能对应于腹侧流中更后的区域
- 在观看同样的图像时，卷积神经网络中的活动可与视觉系统的活动关联起来



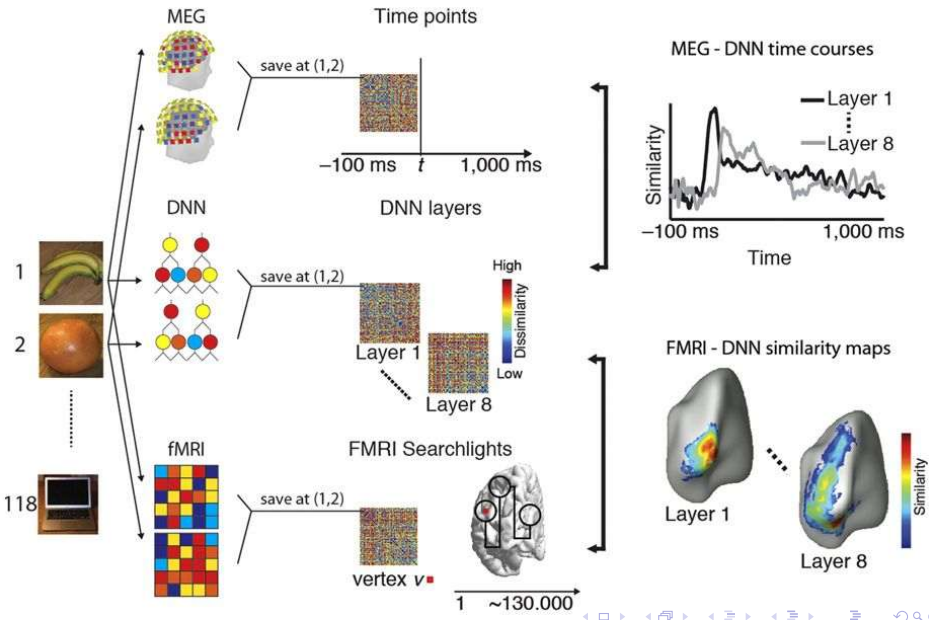
卷积神经网络的神经科学基础

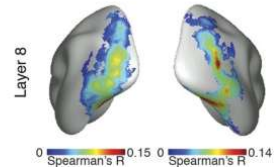
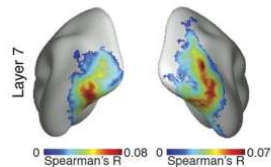
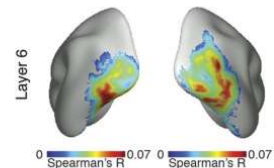
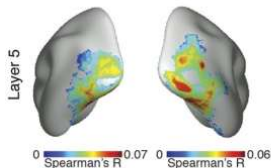
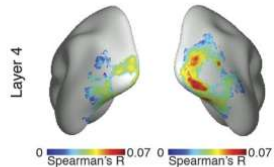
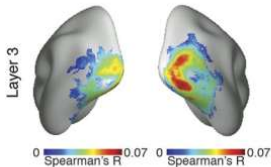
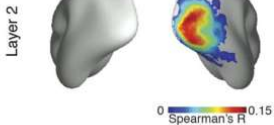
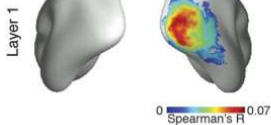
- 表征相似度分析 (representational similarity analysis), 分析两个系统是否能以相同的方式表征信息
- 为每个系统构建一个矩阵实现的, 其中的值代表对两个不同输入的响应的相似度



Kriegeskorte, N., Mur, M., & Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2, 4.

Stimuli	Signal space activation patterns	Pairwise Dissimilarity calculation	Representational dissimilarity matrices (RDMs)	RDM comparison (Spearman's R)	Comparison Results
---------	--	--	--	-------------------------------------	-----------------------





- 卷积神经网络的低层与大脑枕叶（即中低层视觉区域）具有显著相似性
- 较高的皮层与颞叶和顶叶的前部区域具有明显的相似性
- 第7和8层深入到颞下皮层和顶下皮层

编程作业

- 使用pytorch或者tensorflow实现卷积神经网络，在ImageNet数据集上进行图片分类。研究dropout、normalization、learning rate decay、residual connection、网络深度等超参数对分类性能的影响
- 由助教准备数据集，下周布置完后，两周内提交



神经网络的参数优化

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>



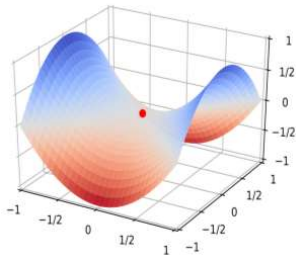
本章内容

- 网络优化基本介绍
- 优化算法
 - SGD
 - Adagrad
 - RMSProp
 - Adam等
- 初始化方法
 - 随机初始化
 - Xavier 初始化
 - He 初始化

神经网络优化的挑战

- 局部极小值非常多甚至不可数无限多的局部极小值
- 对于足够大的神经网络而言，大部分局部极小值具有很小的损失
- 很多高维非凸函数的局部极小值事实上都远少于鞍点
- 真实的神经网络也存在包含很多高代价鞍点的损失函数

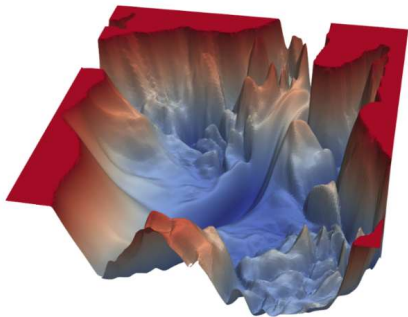
鞍点：梯度为 0 的点



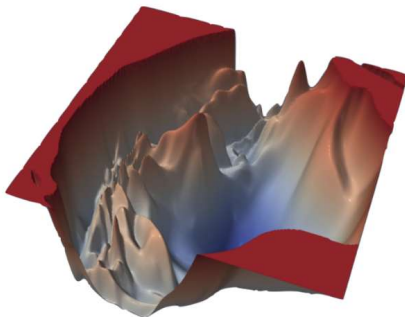
神经网络优化的挑战

优化地形（ Optimization Landscape ）：在高维空间中损失函数的曲面形状

VGG-56



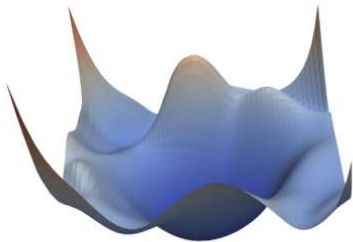
VGG-110



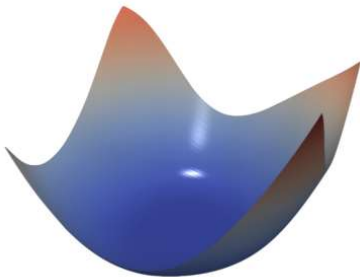
修改网络结构获得更好的优化地形

- 好的优化地形通常比较平滑，更容易优化
- 使用 ReLU 激活函数、残差连接、逐层归一化等

Renset-56



Densenet-121





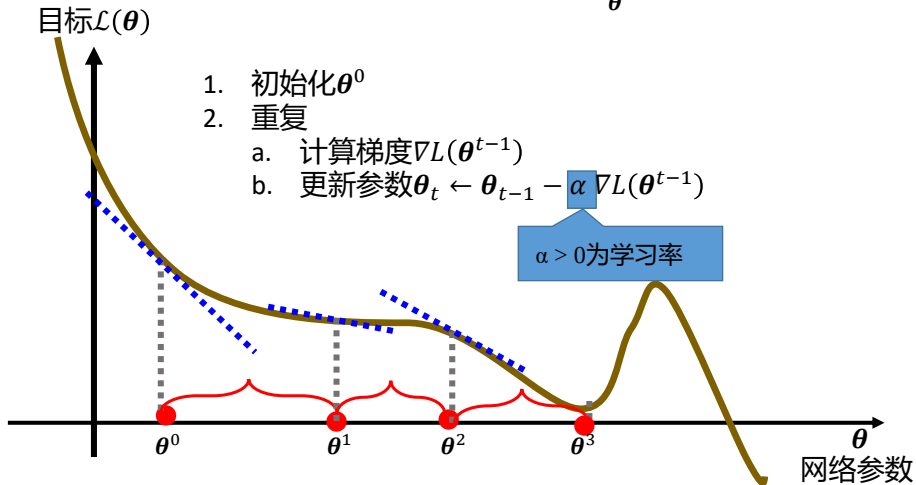
神经网络优化的其他改善方法

- 更有效的优化算法来提高优化方法的效率和稳定性
- 更好的参数初始化方法
- 更好的数据预处理方法
- 使用更好的超参数优化方法

} 提高优化效率

梯度下降

优化算法解决 $\theta^* = \arg \min_{\theta} \mathcal{L}(\theta)$

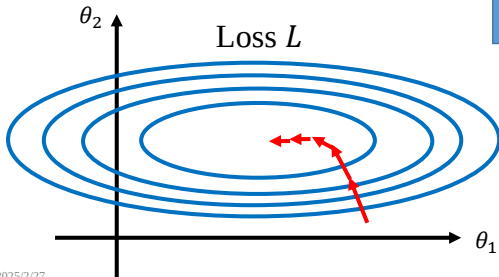


梯度下降 (二维)

优化算法解决 $\theta^* = \arg \min_{\theta} \mathcal{L}(\theta)$

1. 初始化 θ^0
2. 重复
 - a. 计算梯度 $\nabla L(\theta^{t-1})$
 - b. 更新参数 $\theta_t \leftarrow \theta_{t-1} - \alpha \nabla L(\theta^{t-1})$

$\alpha > 0$ 为学习率



随机梯度下降

- 给定训练集 $D = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$

梯度下降

优化目标是单个训练样本损失的平均

$$\mathcal{L}_D(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\mathbf{y}^{(n)}, f(\mathbf{x}^{(n)}; \boldsymbol{\theta}))$$

$$\nabla \mathcal{L}_D(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \nabla \mathcal{L}(\mathbf{y}^{(n)}, f(\mathbf{x}^{(n)}; \boldsymbol{\theta}))$$

由于数据集可能很大，无法全部放入内存计算梯度

小批量随机梯度下降

每次从数据集中采样一部分样本（称为batch），计算batch上的梯度，并进行参数更新。

$$\mathcal{L}_B(\boldsymbol{\theta}) = \frac{1}{B} \sum_{i=1}^B \mathcal{L}(\mathbf{y}^{(i)}, f(\mathbf{x}^{(i)}; \boldsymbol{\theta}))$$

$$\nabla \mathcal{L}_B(\boldsymbol{\theta}) = \frac{1}{B} \sum_{i=1}^B \nabla \mathcal{L}(\mathbf{y}^{(i)}, f(\mathbf{x}^{(i)}; \boldsymbol{\theta}))$$

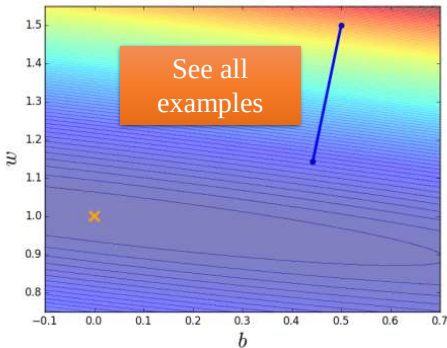
关键因素

- 小批量样本数量
- 梯度
- 学习率

随机梯度下降 vs. 梯度下降

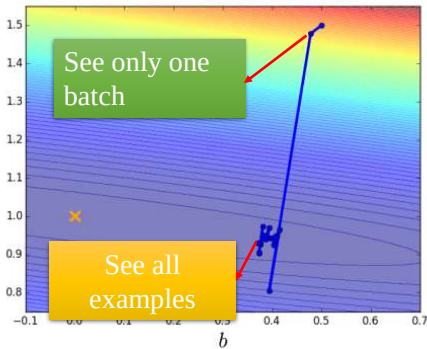
梯度下降

用所有样本对参数更新



随机梯度下降

每个批都对参数更新
更新次数更多



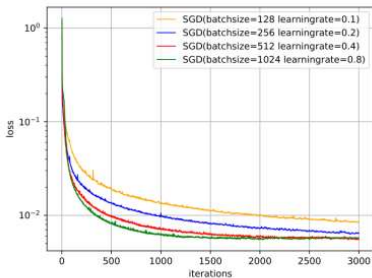


样本随机性的影响

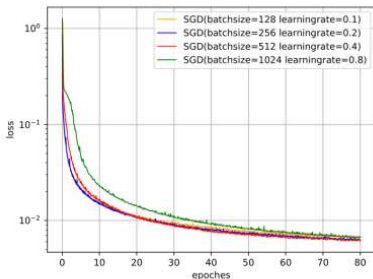
- 在每次迭代时，随机选择 B 个样本，这里的随机性非常重要
- 但随机性在大规模数据情况下很难满足
- 实践中通常将样本顺序打乱一次，然后按照这个顺序存储起来
- 虽然偏离真实随机采样，但不会有严重的有害影响

批量大小的影响

- 批量大小不影响梯度期望，但会影响梯度方差，一般为2的幂数
 - 批量越大，随机梯度的方差越小，引入的噪声也越小，训练也越稳定，因此可以设置较大的学习率
 - 批量较小时，需要设置较小的学习率，否则模型会不收敛



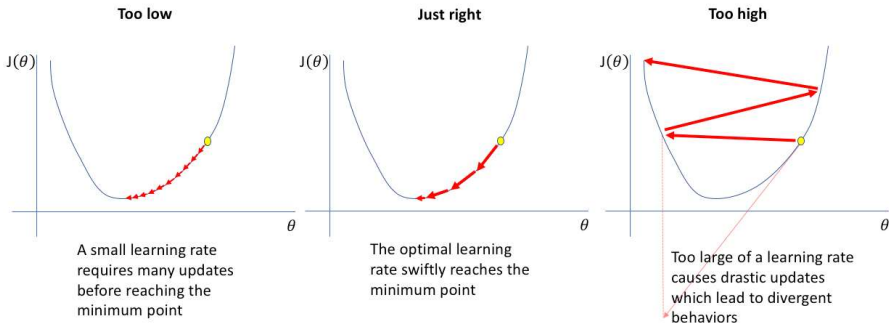
(a) 按 Iteration 的损失变化



(b) 按 Epoch 的损失变化

小批量梯度下降中，每次选取样本数量对损失下降的影响

学习率的影响

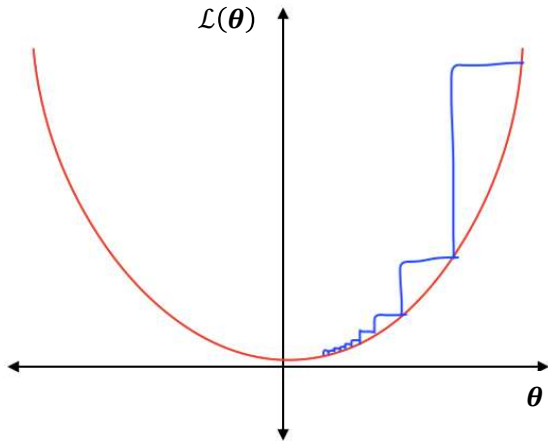




学习率调整方法

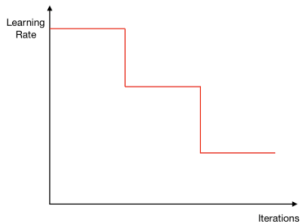
- 学习率衰减
- 学习率预热
- 周期性学习率调整
- 自适应调整学习率

学习率衰减

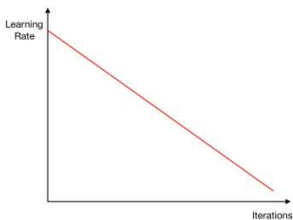


- 学习率一开始要保持大些保证收敛速度
- 在收敛到最优点附近时要小些以避免来回振荡

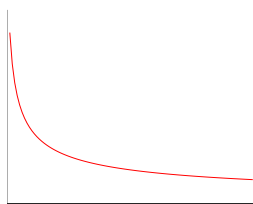
学习率衰减



梯级衰减 (step decay)

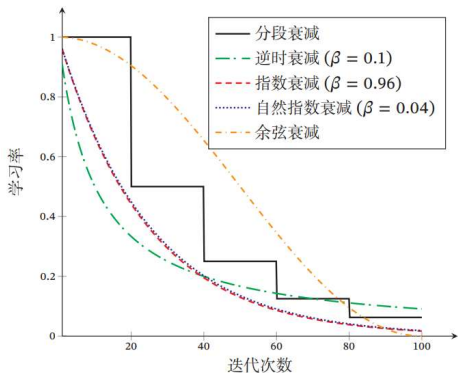


线性衰减 (Linear Decay)



$1/t$ 衰减 ($1/t$ decay)

学习率衰减



- 假设初始学习率为 α_0 ，在第 t 次迭代时的学习率为 α_t

β 为衰减率

- 逆时衰减

$$\alpha_t = \alpha_0 \frac{1}{1 + \beta \times t}$$

- 指数衰减

$$\alpha_t = \alpha_0 \beta^t \quad \beta < 1$$

- 自然指数衰减

$$\alpha_t = \alpha_0 \exp(-\beta t)$$

- 余弦衰减

$$\alpha_t = \frac{1}{2} \alpha_0 \left(1 + \cos\left(\frac{t\pi}{T}\right) \right)$$

学习率预热

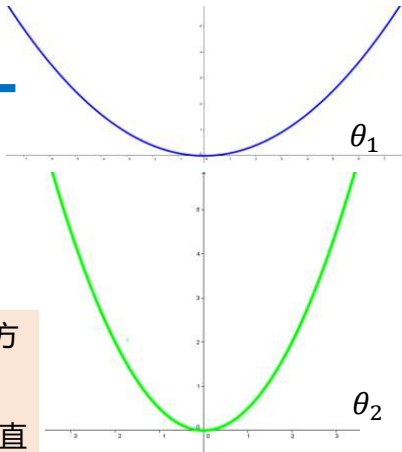
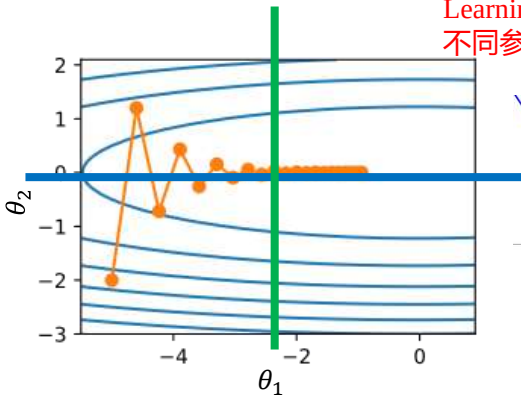
- 刚开始训练时，参数是随机初始化的，梯度往往也比较大，比较大的初始学习率使得训练不稳定
- 为了提高训练稳定性，可以在最初几轮迭代时，采用比较小的学习率，等梯度下降到一定程度后在恢复到初始学习率
- 采用的方法是：在最初的 T' 迭代中，初始学习率为 α_0 ，每次更新的学习率为

$$\alpha'_t = \frac{t}{T'} \alpha_0, 1 \leq t \leq T'$$

- 在这个预热阶段结束后，在选择另外一种学习率衰减方法来逐渐降低学习率

随机梯度下降的问题

Learning rate cannot be one-size-fits-all
不同参数不同学习率



- 同一位置，损失在 θ_2 轴方向比在 θ_1 轴方向的斜率的绝对值更大
- 给定学习率，梯度下降迭代参数时在竖直方向会比在水平方向移动幅度更大

自适应学习率—Adagrad

- 将学习率除以每个参数历史梯度的平方根
- $$g^t = \frac{1}{B} \sum_{i=1}^B \frac{\partial \mathcal{L}(y^{(i)}, f(x^{(i)}; \theta))}{\partial w}$$

Gradient Descent

$$w^{t+1} \leftarrow w^t - \eta^t g^t$$

w 是其中一个参数

Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t$$

σ^t 是参数 w 的历史梯度平方根



自适应学习率—Adagrad

$$g^t = \frac{1}{B} \sum_{i=1}^B \frac{\partial \mathcal{L}(\mathbf{y}^{(i)}, f(\mathbf{x}^{(i)}; \boldsymbol{\theta}))}{\partial \mathbf{w}}$$

$$\mathbf{w}^1 \leftarrow \mathbf{w}^0 - \frac{\eta}{\sigma^0} g^0$$

$$\sigma^0 = \sqrt{(g^0)^2}$$

$$\mathbf{w}^2 \leftarrow \mathbf{w}^1 - \frac{\eta}{\sigma^1} g^1$$

$$\sigma^1 = \sqrt{(\sigma^0)^2 + (g^1)^2}$$

$$\mathbf{w}^3 \leftarrow \mathbf{w}^2 - \frac{\eta}{\sigma^2} g^2$$

$$\sigma^2 = \sqrt{(\sigma^1)^2 + (g^2)^2}$$

$$\vdots$$
$$\vdots$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{(\sigma^{t-1})^2 + (g^t)^2}$$

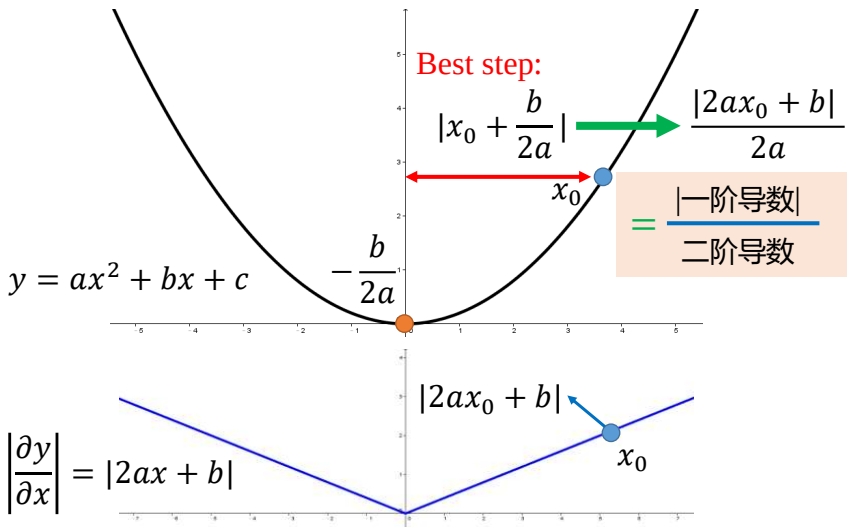
σ^t 是参数 $\mathbf{w}$ 的历史梯度平方根



22

自适应学习率—Adagrad

$$\frac{\partial^2 y}{\partial x^2} = 2a$$

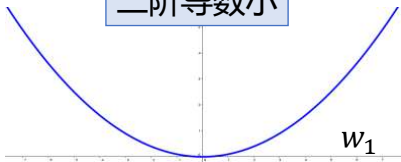


自适应学习率—Adagrad

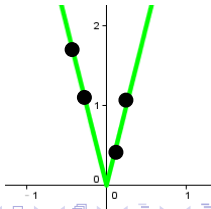
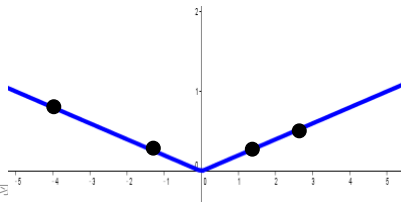
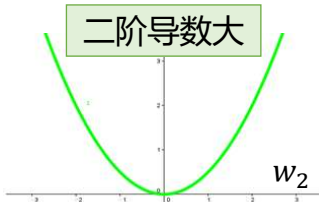
$$w^{t+1} \leftarrow w^t - \eta \frac{g^t}{\sqrt{\sum_{i=0}^t (g^i)^2}}$$

→ $| \text{一阶导数} |$
= 二阶导数
?

二阶导数小



二阶导数大



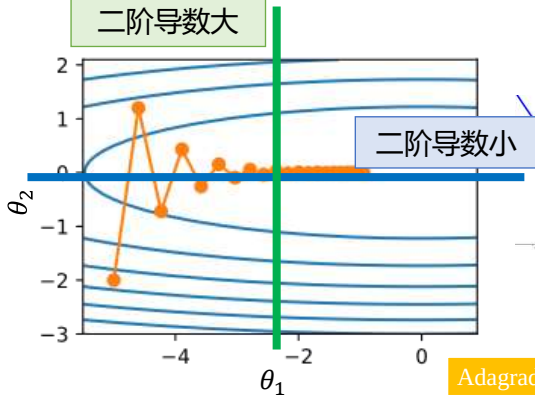
Learning rate can be one-size-fits-all
不同参数可以相同学习率



24

随机梯度下降的问题

二阶导数大



二阶导数小

最优步

$$= \frac{|\text{一阶导数}|}{\text{二阶导数}}$$

二阶导数小

Adagrad影响

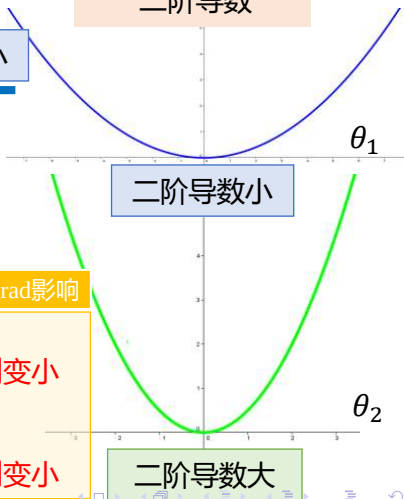
- 同一位置，损失在 θ_2 轴方向比在 θ_1 轴方向的斜率的绝对值更大

差别变小

- 给定学习率迭代参数时在竖直方向会比在水平方向移动幅度更大

差别变小

二阶导数大



自适应学习率—Adagrad

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$

累积平方梯度: $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}$

计算更新: $\Delta \boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$ 逐元素地应用除和求平方根操作

应用更新: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$

经验上发现, 对于训练深度神经网络模型而言, 从训练开始时积累梯度平方会导致**有效学习率过早和过量的减小**

自适应学习率—RMSProp

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0$$

$$\sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1$$

$$\sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1 - \alpha)(g^1)^2}$$

$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2$$

$$\sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1 - \alpha)(g^2)^2}$$

$$\vdots$$

$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1 - \alpha)(g^t)^2}$$

指数加权的
移动平均

$$(\sigma^t)^2 = (\alpha)^t g_0^2 + (1 - \alpha) \sum_{i=1}^t (\alpha)^{t-i} (g^i)^2$$

自适应学习率—RMSProp

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$

累积平方梯度: $\mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}$

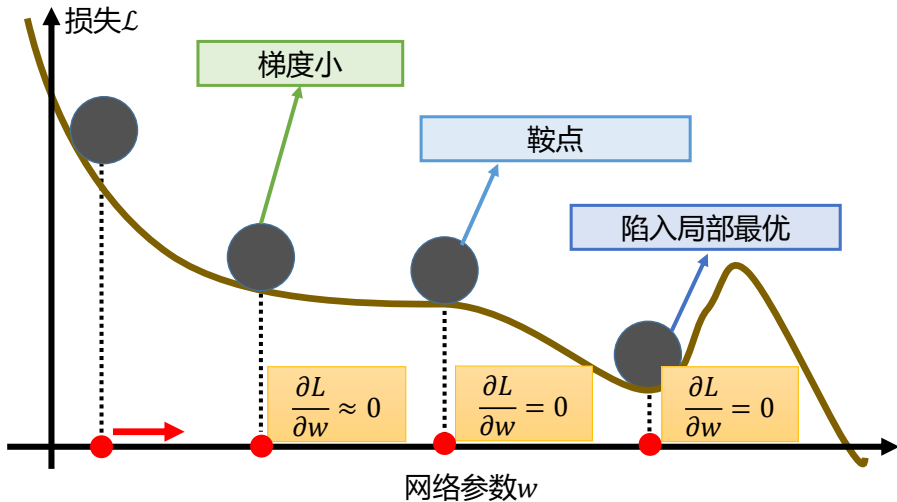
计算更新: $\Delta \boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$

应用更新: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$

Adagrad

累积平方梯度: $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{g} \odot \mathbf{g}$

神经网络优化的挑战

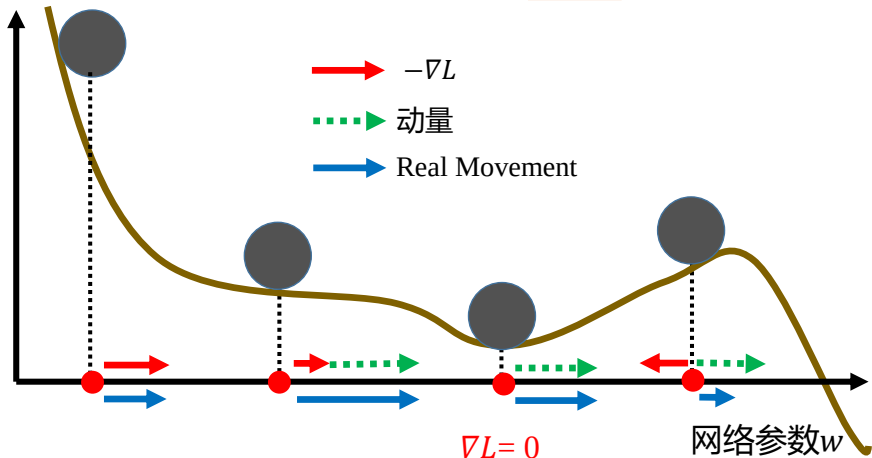


动量法—动机

一个物体的动量指的是该物体在它运动方向上保持运动的趋势，是该物体的质量和速度的乘积

$$\text{Movement} = -\nabla L + \text{动量}$$

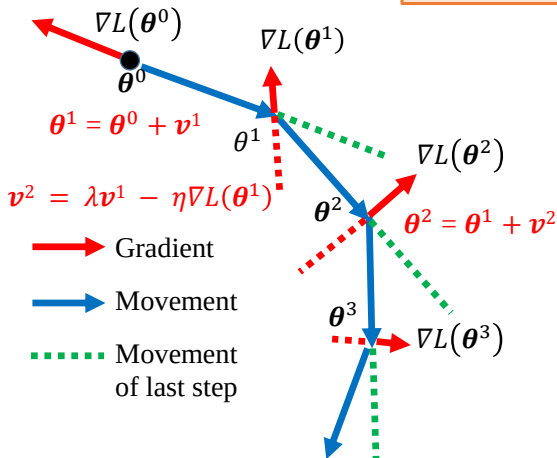
损失 $\mathcal{L}$



动量法

$$v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$$

Movement = $-\nabla L$ + 动量



动量法

• v^i 是历史负梯度的加权和

$$v^0 = 0$$

$$v^1 = \lambda v^0 - \eta \nabla L(\theta^0)$$

$$v^2 = \lambda v^1 - \eta \nabla L(\theta^1)$$

$$v^3 = \lambda v^2 - \eta \nabla L(\theta^2)$$

$$\vdots$$

$$v^i = \lambda v^{i-1} - \eta \nabla L(\theta^{i-1})$$

$$v^0 = 0$$

$$v^1 = -\eta \nabla L(\theta^0)$$

$$v^2 = -\lambda \eta \nabla L(\theta^0) - \eta \nabla L(\theta^1)$$

$$v^3 = -\lambda^2 \eta \nabla L(\theta^0) - \lambda \eta \nabla L(\theta^1) - \eta \nabla L(\theta^2)$$

$$\vdots$$

$$v^i = -\eta \sum_{k=1}^i \lambda^{i-k} \nabla L(\theta^{k-1})$$

$$\theta^t \leftarrow \theta^{t-1} + v^t$$

在第 t 次迭代时, 计算负梯度的“加权移动平均”
作为参数的更新方向

动量法的物理意义

- 考虑参数关于迭代步的函数 $\theta(t)$, 类比于单位质量粒子 t 时刻位置

- 根据牛顿第二定律 $f(t) = \frac{\partial^2}{\partial t^2} \theta(t)$

势能	力
重力势能 mgh	重力 mg
弹性势能 $\frac{1}{2}kx^2$	弹力 kx

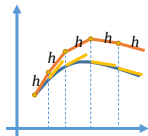
- 引入速度 $v(t) = \theta'(t)$

- 目标函数 $J(\theta)$ 类比于能量函数, 那么 $f(t) \propto -\nabla_{\theta} J(\theta(t))$

$$\begin{pmatrix} \theta(t) \\ v(t) \end{pmatrix}' = \begin{pmatrix} v(t) \\ -\nabla_{\theta} J(\theta(t)) \end{pmatrix}$$

- 利用欧拉折线法求解微分方程 $\frac{dy}{dx} = f(x, y)$

- $y_{n+1} - y_n = hf(x_n, y_n)$



- 求解上述的微分方程组

$$\begin{pmatrix} \theta_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} \theta_n \\ v_n \end{pmatrix} + \begin{pmatrix} h_1 \\ \epsilon \end{pmatrix} \odot \begin{pmatrix} v_n \\ -\nabla_{\theta} J(\theta_n) \end{pmatrix}$$

动量法的物理意义

- 如果代价函数的梯度是唯一的力，那么粒子可能永远不会停下来
 - 假设冰面没有摩擦，一个冰球从山谷的一端下滑，上升到另一端，永远来回振荡
- 添加一个正比于 $-v(t)$ 的阻力，物理术语 粘性阻力
 - 弹簧振子的阻尼力
 - 空气阻力
 - 摩擦力

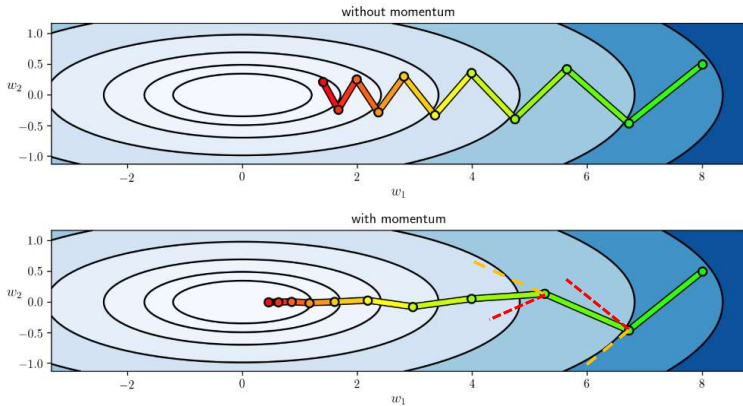
$$\begin{pmatrix} \theta_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} \theta_n \\ v_n \end{pmatrix} + \begin{pmatrix} h \\ \epsilon \end{pmatrix} \odot \begin{pmatrix} v_n \\ -\nabla_{\theta} J(\theta_n) - \textcolor{red}{c}v_n \end{pmatrix}$$



$$v_{n+1} = (1 - c\epsilon)v_n - \epsilon \nabla_{\theta} J(\theta_n)$$

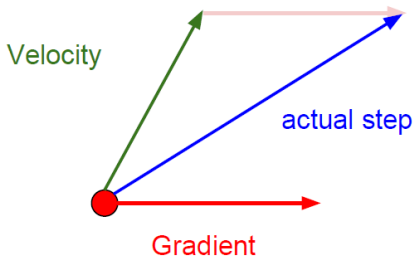
$$\theta_{n+1} = \theta_n + hv_n$$

动量法的作用



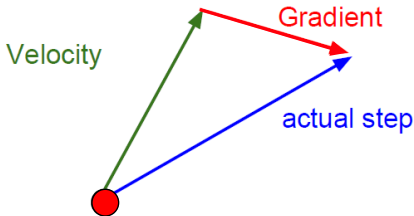
Nesterov 动量

动量更新



结合梯度和速度来更新权重

Nesterov动量更新



沿着速度方向试走一步再计算梯度，再结合速度来更新权重

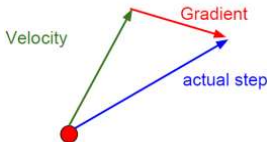
Nesterov 动量法

动量法

计算梯度估计: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$

计算动量更新: $\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \mathbf{g}$

应用更新: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}$



Nesterov
动量法

应用临时更新: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \mathbf{v}$

计算梯度(临时点): $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \tilde{\boldsymbol{\theta}}), y^{(i)})$

计算动量更新: $\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon \mathbf{g}$

应用更新: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}$

RMSProp+ Nesterov 动量

RMSProp

$$\text{计算梯度: } \mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \theta), y^{(i)})$$

$$\text{累积平方梯度: } \mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}$$

$$\text{计算更新: } \Delta \theta \leftarrow -\frac{\epsilon}{\delta + \sqrt{\mathbf{r}}} \odot \mathbf{g}$$

$$\text{应用更新: } \theta \leftarrow \theta + \Delta \theta$$

RMSProp
+Nesterov 动量

$$\text{计算临时更新: } \tilde{\theta} \leftarrow \theta + \alpha v$$

$$\text{计算梯度: } \mathbf{g} \leftarrow \frac{1}{m} \nabla_{\theta} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \tilde{\theta}), y^{(i)})$$

$$\text{累积平方梯度: } \mathbf{r} \leftarrow \rho \mathbf{r} + (1 - \rho) \mathbf{g} \odot \mathbf{g}$$

$$\text{计算动量更新: } v \leftarrow \alpha v - \frac{\epsilon}{\sqrt{\mathbf{r}}} \odot \mathbf{g}$$

$$\text{应用更新: } \theta \leftarrow \theta + \Delta \theta$$

动量作用于缩放后的梯度

Adam

计算梯度: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i \mathcal{L}(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$

类似于动量

$t \leftarrow t + 1$

更新有偏一阶矩估计: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1) \mathbf{g}$

更新有偏二阶矩估计: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2) \mathbf{g} \odot \mathbf{g}$

修正一阶矩的偏差: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \rho_1^t}$

修正二阶矩的偏差: $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \rho_2^t}$

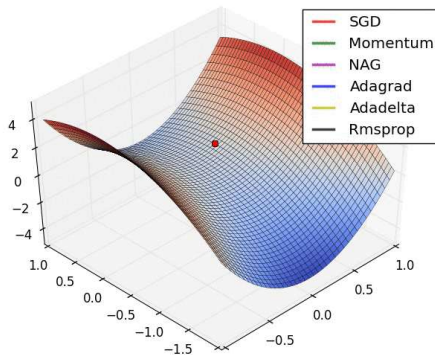
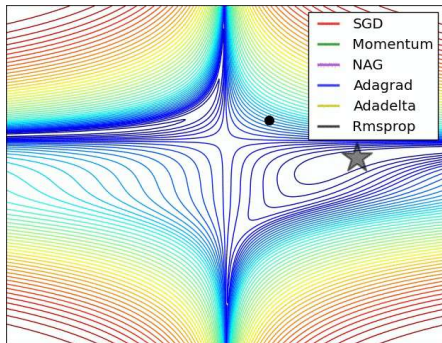
偏置修正

计算更新: $\Delta \boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\hat{\mathbf{r}}}} \odot \hat{\mathbf{s}}$

应用更新: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$

$\rho_1 = 0.9$
 $\rho_2 = 0.999$

优化算法的对比



优化算法改进小结

- 大部分优化算法可以使用下面公式来统一描述概括：

$$\Delta\theta_t = -\frac{\alpha_t}{\sqrt{G_t + \epsilon}} M_t,$$

$$G_t = \psi(\mathbf{g}_1, \dots, \mathbf{g}_t),$$

$$M_t = \phi(\mathbf{g}_1, \dots, \mathbf{g}_t),$$

$\mathbf{g}_t$ 为第 t 步的梯度
 α_t 为第 t 步的学习率

类别		优化算法
学习率设置	固定衰减学习率	分段常数衰减、逆时衰减、(自然)指数衰减、余弦衰减
	周期性学习率	循环学习率、SGDR
	自适应学习率	AdaGrad、RMSprop、AdaDelta
梯度估计修正		动量法、Nesterov 加速梯度、梯度截断
综合方法		Adam $\approx$ 动量法 + RMSprop

参数初始化

- 参数不能初始化为0! 为什么?
 - 对称权重问题!
- 初始化方法
 - 预训练初始化
 - 随机初始化
 - 固定值初始化 (偏置 (Bias) 通常用 0 来初始化)

随机初始化

- Gaussian分布初始化

- 参数从一个固定均值（比如0）和固定方差（比如0.01）的Gaussian分布进行随机初始化。

$$W = \sigma * \text{np.random.randn}(\text{fan_in}, \text{fan_out})$$

- 均匀分布初始化

- 参数可以在区间 $[-r, r]$ 内采用均匀分布进行初始化

若方差为 σ^2 ,
那么 $r = \sqrt{3\sigma^2}$

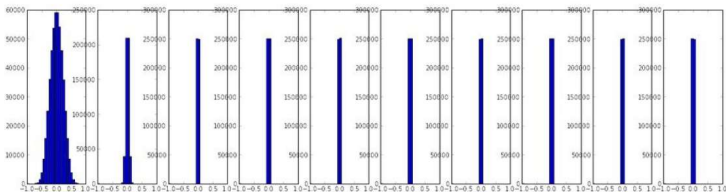
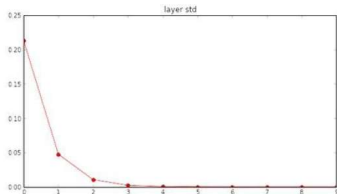
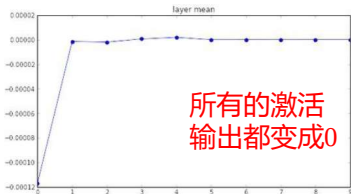
$$W = r * \text{np.random.rand}(\text{fan_in}, \text{fan_out})$$

在浅层网络时效果不错，但是深层网络就有较大问题

随机初始化的问题

```
input layer had mean 0.000927 and std 0.998388
hidden layer 1 had mean -0.000117 and std 0.213081
hidden layer 2 had mean -0.000001 and std 0.047551
hidden layer 3 had mean -0.000002 and std 0.010630
hidden layer 4 had mean 0.000001 and std 0.002378
hidden layer 5 had mean 0.000002 and std 0.000532
hidden layer 6 had mean -0.000000 and std 0.000119
hidden layer 7 had mean 0.000000 and std 0.000026
hidden layer 8 had mean -0.000000 and std 0.000006
hidden layer 9 had mean 0.000000 and std 0.000001
hidden layer 10 had mean -0.000000 and std 0.000000
```

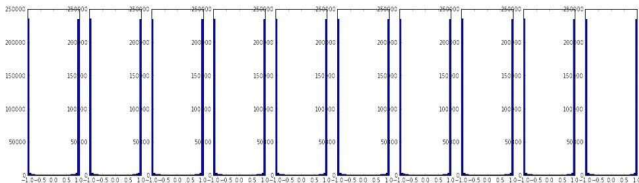
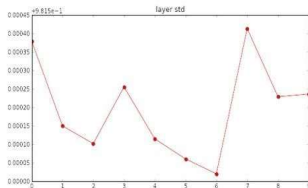
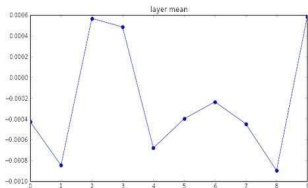
- 看看激活函数的输出统计
- 10层，每层500个神经元，用tanh非线性函数，用小随机数初始化



随机初始化的问题

```
input layer had mean 0.001800 and std 1.001311
hidden layer 1 had mean -0.000430 and std 0.981879
hidden layer 2 had mean -0.000849 and std 0.981649
hidden layer 3 had mean 0.000566 and std 0.981601
hidden layer 4 had mean 0.000483 and std 0.981755
hidden layer 5 had mean -0.000682 and std 0.981614
hidden layer 6 had mean -0.000401 and std 0.981560
hidden layer 7 had mean -0.000237 and std 0.981520
hidden layer 8 had mean -0.000448 and std 0.981913
hidden layer 9 had mean -0.000899 and std 0.981728
hidden layer 10 had mean 0.000584 and std 0.981736
```

激活函数输出饱和，输出
1或者-1， 梯度为0

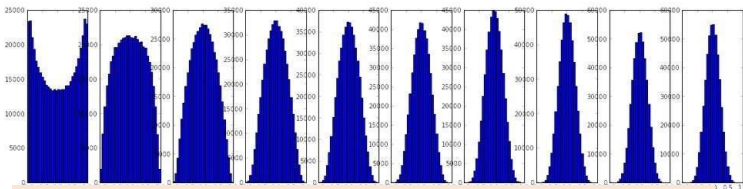
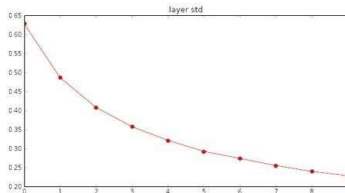
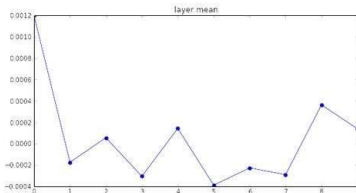


Xavier 初始化

也可以为均匀分布

```
input layer had mean 0.001800 and std 1.001311
hidden layer 1 had mean 0.001198 and std 0.627953
hidden layer 2 had mean -0.000175 and std 0.486051
hidden layer 3 had mean 0.000055 and std 0.407723
hidden layer 4 had mean -0.000306 and std 0.357108
hidden layer 5 had mean 0.000142 and std 0.320917
hidden layer 6 had mean -0.000389 and std 0.292116
hidden layer 7 had mean -0.000228 and std 0.273387
hidden layer 8 had mean -0.000291 and std 0.254935
hidden layer 9 had mean 0.000361 and std 0.239266
hidden layer 10 had mean 0.000139 and std 0.228008
```

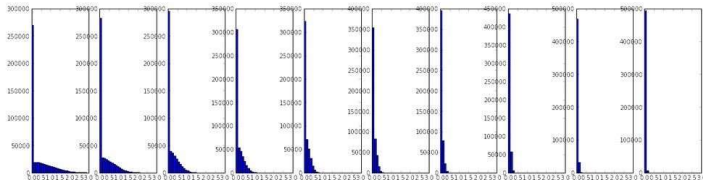
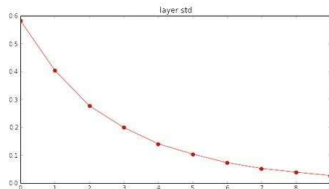
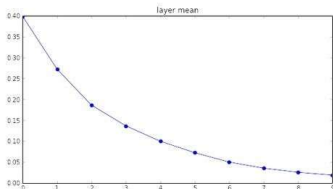
$$U\left(-\frac{\sqrt{3}}{\sqrt{\text{fan_in}}}, \frac{\sqrt{3}}{\sqrt{\text{fan_in}}}\right)$$



Xavier 初始化的问题

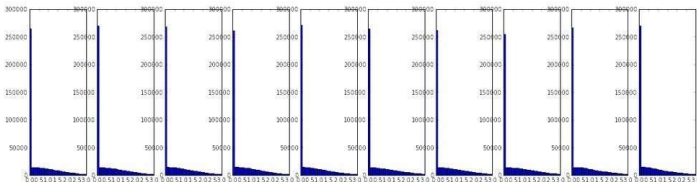
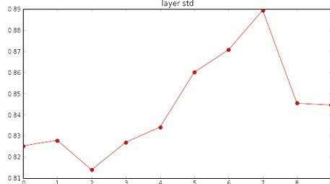
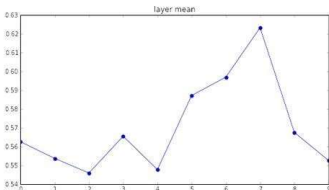
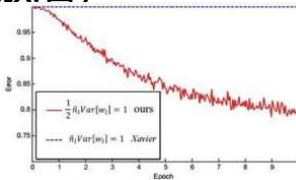
```
input layer had mean 0.000501 and std 0.999444
hidden layer 1 had mean 0.398623 and std 0.582273
hidden layer 2 had mean 0.272352 and std 0.403795
hidden layer 3 had mean 0.186076 and std 0.276912
hidden layer 4 had mean 0.136442 and std 0.198685
hidden layer 5 had mean 0.099568 and std 0.140299
hidden layer 6 had mean 0.072234 and std 0.103280
hidden layer 7 had mean 0.049775 and std 0.072748
hidden layer 8 had mean 0.035138 and std 0.051572
hidden layer 9 had mean 0.025404 and std 0.038583
hidden layer 10 had mean 0.018408 and std 0.026076
```

当使用ReLU时,
方差又接近0



He 初始化 (针对ReLU激活)

input layer had mean 0.000501 and std 0.999444
 hidden layer 1 had mean 0.562488 and std 0.825232
 hidden layer 2 had mean 0.553614 and std 0.827835
 hidden layer 3 had mean 0.545867 and std 0.813855
 hidden layer 4 had mean 0.565396 and std 0.826902
 hidden layer 5 had mean 0.547678 and std 0.834092
 hidden layer 6 had mean 0.587103 and std 0.860035
 hidden layer 7 had mean 0.596867 and std 0.870610
 hidden layer 8 had mean 0.623214 and std 0.889348
 hidden layer 9 had mean 0.567498 and std 0.845357
 hidden layer 10 had mean 0.552531 and std 0.844523



权重初始化背后的数学

- 第 l 层的仿射变换 $y_l = W_l x_l + b_l$ 非线性变换 $x_l = \text{relu}(y_{l-1})$
- 假设 W_l 每个元素是独立同分布的, x_l 每个元素也是独立同分布
- 假设 W_l 与 x_l 互相独立

$$\begin{aligned} y_l[i] &= W_l[i, :]^T x_l + b_l[i] \\ &= \sum_d W_l[i, d] x_l[d] + b_l[i] \end{aligned}$$

$$\text{var}[y_l] = n_l \text{var}[w_l x_l]$$

y_l 的元素

x_l 的维度

W_l 的元素

x_l 的元素

$$\text{var}[x_1 + \dots + x_n] = \text{var}[x_1] + \dots + \text{var}[x_n]$$

权重初始化背后的数学

- 设 $\mathbb{E}[w_l] = 0$, 那么 $\mathbb{E}[w_l x_l] = \mathbb{E}[w_l] \mathbb{E}[x_l] = 0$

$$\text{var}[y_l] = n_l \text{var}[w_l x_l] \quad \Rightarrow \quad \text{var}[y_l] = n_l \text{var}[w_l] \mathbb{E}[x_l^2]$$

$$\begin{aligned} \text{var}[w_l x_l] &= \mathbb{E}[w_l^2 x_l^2] \\ &= \mathbb{E}[w_l^2] \mathbb{E}[x_l^2] \\ &= \text{var}[w_l] \mathbb{E}[x_l^2] \end{aligned}$$

权重初始化背后的数学

- 假设 $b_{l-1} = 0$ 且 $\mathbb{E}[w_{l-1}] = 0$, 那么 $\mathbb{E}[y_{l-1}] = \mathbb{E}[w_{l-1}x_{l-1}] = 0$
- 由于 $x_l = \text{relu}(y_{l-1})$, $\mathbb{E}[x_l^2] = \frac{1}{2}\mathbb{E}[y_{l-1}^2] = \frac{1}{2}\text{var}[y_{l-1}]$

$$\text{var}[y_l] = n_l \text{var}[w_l] \mathbb{E}[x_l^2] \quad \Rightarrow \quad \text{var}[y_l] = \frac{n_l}{2} \text{var}[w_l] \text{var}[y_{l-1}]$$

$$\text{var}[y_l] = \text{var}[y_{l-1}] \quad \Rightarrow \quad \frac{n_l}{2} \text{var}[w_l] = 1$$

研究反向传播时可以得出类似的结论 $\frac{n_{l+1}}{2} \text{var}[w_l] = 1$

权重初始化背后的数学

- 如果使用PReLU, 则 $x_l = \text{prelu}(y_{l-1})$, $\mathbb{E}[x_l^2] = \frac{1}{2}(1 + a^2)\text{var}[y_{l-1}]$

$$\text{var}[y_l] = n_l \text{var}[w_l] \mathbb{E}[x_l^2] \quad \Rightarrow \quad \text{var}[y_l] = \frac{1 + a^2}{2} n_l \text{var}[w_l] \text{var}[y_{l-1}]$$

$$\text{var}[y_l] = \text{var}[y_{l-1}] \quad \Rightarrow \quad \frac{1 + a^2}{2} n_l \text{var}[w_l] = 1$$

$$\text{var}[w_l] = \frac{2}{1 + a^2} \frac{1}{n_l}$$

权重初始化的总结

初始化方法	激活函数	均匀分布 $[-r, r]$	高斯分布 $\mathcal{N}(0, \sigma^2)$
Xavier 初始化	Logistic	$r = 4\sqrt{\frac{6}{M_{l-1}+M_l}}$	$\sigma^2 = 16 \times \frac{2}{M_{l-1}+M_l}$
Xavier 初始化	Tanh	$r = \sqrt{\frac{6}{M_{l-1}+M_l}}$	$\sigma^2 = \frac{2}{M_{l-1}+M_l}$
He 初始化	ReLU	$r = \sqrt{\frac{6}{M_{l-1}}}$	$\sigma^2 = \frac{2}{M_{l-1}}$

偏置设置的一些方案

- 通常情况设偏置为0
- 输出单元的偏置可以用 $\text{softmax}(\mathbf{b})=\mathbf{c}$ 来初始化, $\mathbf{c}$ 为类别分布
- ReLU的偏置为0.1
- 门控结构中, 门单元的偏置设为门输出接近1
 - 单元输出 u , 门单元 $h \in [0,1]$, 输出为 uh
 - $h \approx 1$

权重初始化的参考文献

1. ***Understanding the difficulty of training deep feedforward neural networks*** by Glorot and Bengio, 2010
2. ***Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*** by Saxe et al, 2013
3. ***Random walk initialization for training very deep feedforward networks*** by Sussillo and Abbott, 2014
4. ***Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification*** by He et al., 2015
5. ***Data-dependent Initializations of Convolutional Neural Networks*** by Krähenbühl et al., 2015
6. ***All you need is a good init***, by Mishkin and Matas, 2015



神经网络的训练技巧

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>

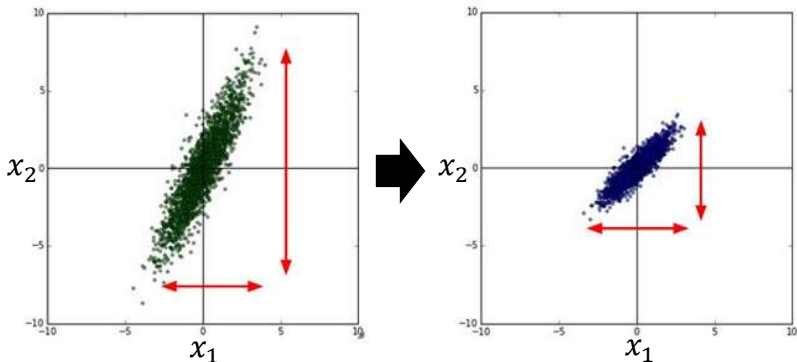


本章内容

- 归一化 (normalization)
 - 数据归一化
 - 逐层归一化
- 训练目标和方式
 - 多任务、迁移学习、课程学习
- 数据增强
- 正则化
 - Early stop
 - Dropout

数据归一化

$$\hat{y} = b + w_1x_1 + w_2x_2$$

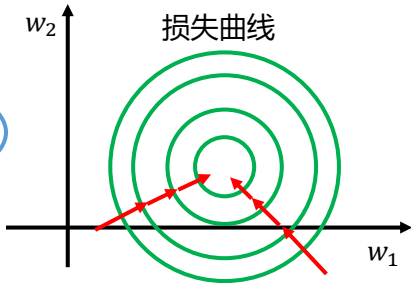
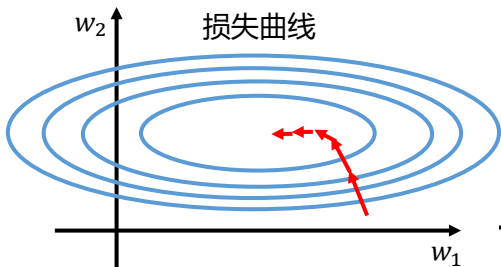
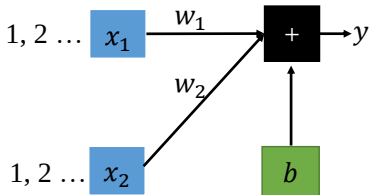
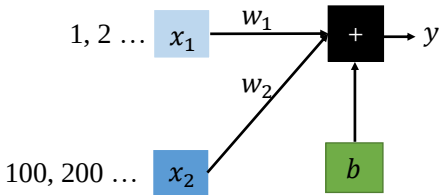


数据归一化：使得不同的特征有相同的尺度

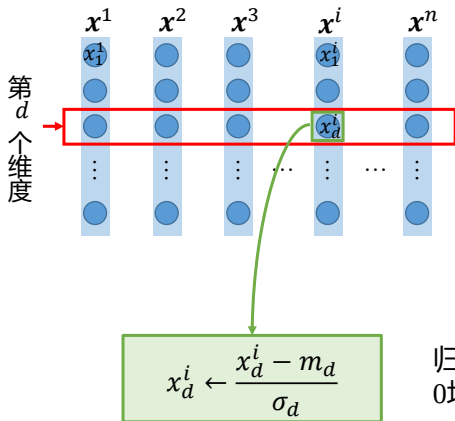
数据归一化

$$\hat{y} = b + w_1x_1 + w_2x_2$$

$$\mathcal{L} = (y - \hat{y})^2$$



数据归一化—标准化

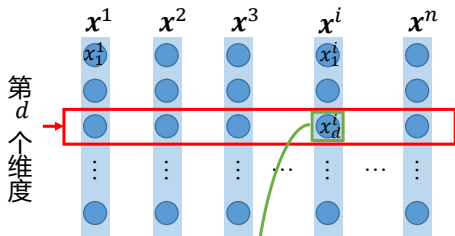


计算均值 $m_d = \frac{1}{n} \sum_{i=1}^n x_d^i$

计算方差 $\sigma_d^2 = \frac{1}{n} \sum_{i=1}^n (x_d^i - m_d)^2$

归一化后数据的所有维度均为
0均值 1方差

数据归一化—最大最小值归一化



计算最大值 $\max_d$

计算最小值 $\min_d$

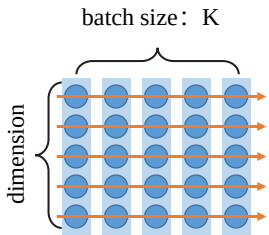
$$x_d^i \leftarrow \frac{x_d^i - \min_d}{\max_d - \min_d}$$

归一化后数据的所有维度均在 $[0,1]$

逐层归一化—Batch Normalization (BN)

神经网络中第 l 层的净输入为 $\mathbf{z}^l$ ，神经元的输出为 $\mathbf{a}^l$ ，

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}) = f_l(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})$$



1. 独立计算该层每维度的均值和方差

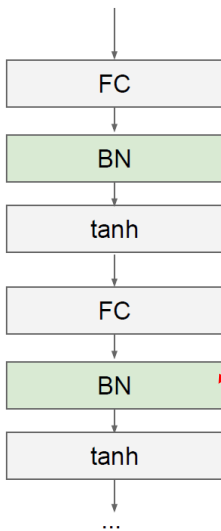
$$\mu_B = \frac{1}{K} \sum_{k=1}^K \mathbf{z}_k$$

$$\sigma_B^2 = \frac{1}{K} \sum_{k=1}^K (\mathbf{z}_k - \mu_B) \odot (\mathbf{z}_k - \mu_B)$$

2. 归一化

$$\mathbf{z}_k = \frac{\mathbf{z}_k - \mu_B}{\sigma_B}$$

逐层归一化—Batch Normalization (BN)



BN层一般在非线性激活函数和仿射变换之后

- 每层的输入通常是前一层的非线性激活函数的输出
- 由于非线性激活函数，每层输入通常是非高斯的，不适合进行标准化



逐层归一化—Batch Normalization (BN)

- 标准化单元的均值和标准差会降低该单元的表达能力
- 为了使得归一化不对网络的表示能力造成负面影响，引入 γ 和 β 两个可学习参数使得各单元的净输出有任意均值和方差

$$\mathbf{z}_k = \frac{\mathbf{z}_k - \mu_B}{\sigma_B} \quad \rightarrow \quad \mathbf{z}_k = \frac{\mathbf{z}_k - \mu_B}{\sigma_B} \odot \gamma + \beta$$

- 网络可能学习出这两个参数，即得到一个恒等映射

$$\gamma = \sigma_B \quad \beta = \mu_B$$

旧参数

- μ_B 和 σ_B 取决于低层神经网络的复杂关联

新参数

- γ 和 β 解除了与下层计算的密切耦合，直接通过梯度下降来学习

逐层归一化—Batch Normalization (BN)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

BN的作用

- 改善梯度的传播
- 允许比较大的学习率
- 减小初始化的影响

batch size不能太小，否则会导致算法性能下降



逐层归一化—Batch Normalization (BN)

- 测试过程和训练过程有所不同
- 均值和方差不是在测试集的batch上计算，而是整个数据集上的均值和方差。
- 实践中，可以通过移动平均来计算
- 可以对单一样本评估

逐层归一化—Batch Normalization (BN)

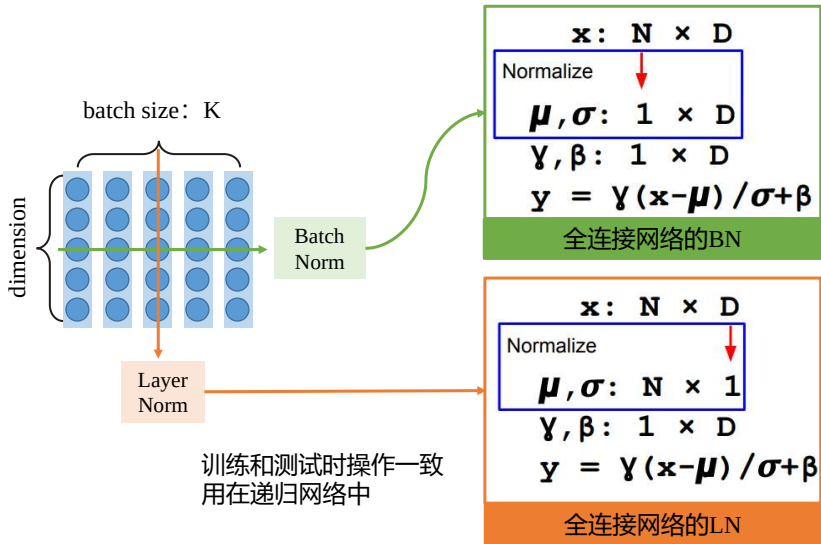
全连接网络的BN

$$\begin{aligned}
 &\mathbf{x}: \mathbf{N} \times \mathbf{D} \\
 &\text{Normalize} \quad \downarrow \\
 &\boldsymbol{\mu}, \boldsymbol{\sigma}: \mathbf{1} \times \mathbf{D} \\
 &\boldsymbol{\gamma}, \boldsymbol{\beta}: \mathbf{1} \times \mathbf{D} \\
 &\mathbf{y} = \boldsymbol{\gamma}(\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma} + \boldsymbol{\beta}
 \end{aligned}$$

卷积网络的BN

$$\begin{aligned}
 &\mathbf{x}: \mathbf{N} \times \mathbf{C} \times \mathbf{H} \times \mathbf{W} \\
 &\text{Normalize} \quad \downarrow \quad \downarrow \quad \downarrow \\
 &\boldsymbol{\mu}, \boldsymbol{\sigma}: \mathbf{1} \times \mathbf{C} \times \mathbf{1} \times \mathbf{1} \\
 &\boldsymbol{\gamma}, \boldsymbol{\beta}: \mathbf{1} \times \mathbf{C} \times \mathbf{1} \times \mathbf{1} \\
 &\mathbf{y} = \boldsymbol{\gamma}(\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma} + \boldsymbol{\beta}
 \end{aligned}$$

逐层归一化—Layer Normalization (LN)



逐层归一化—Instance Normalization (IN)



14

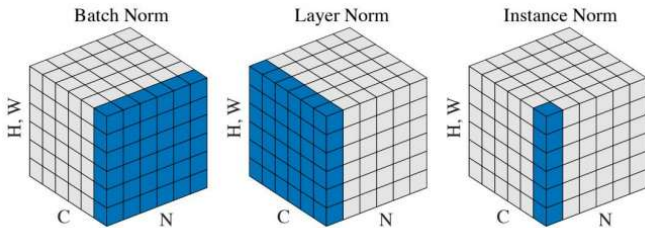
卷积网络的BN

$$\begin{array}{l} \mathbf{x}: N \times C \times H \times W \\ \text{Normalize} \quad \downarrow \quad \downarrow \quad \downarrow \\ \boldsymbol{\mu}, \boldsymbol{\sigma}: 1 \times C \times 1 \times 1 \\ \gamma, \beta: 1 \times C \times 1 \times 1 \\ \mathbf{y} = \gamma(\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma} + \beta \end{array}$$

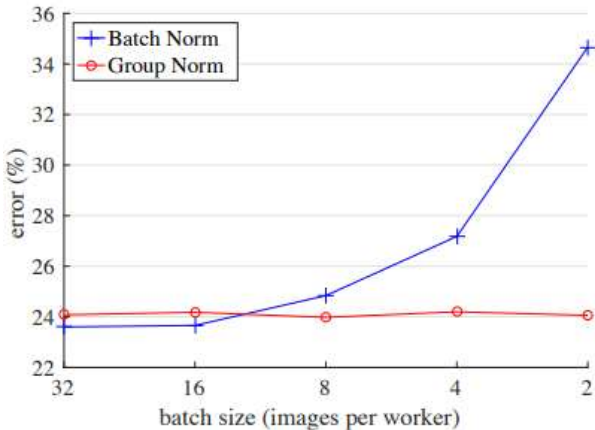
卷积网络的IN

$$\begin{array}{l} \mathbf{x}: N \times C \times H \times W \\ \text{Normalize} \quad \downarrow \quad \downarrow \\ \boldsymbol{\mu}, \boldsymbol{\sigma}: N \times C \times 1 \times 1 \\ \gamma, \beta: 1 \times C \times 1 \times 1 \\ \mathbf{y} = \gamma(\mathbf{x} - \boldsymbol{\mu}) / \boldsymbol{\sigma} + \beta \end{array}$$

归一化层的比较

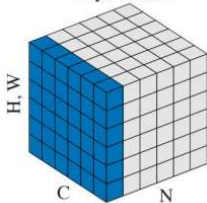


逐层归一化—Group Normalization (GN)

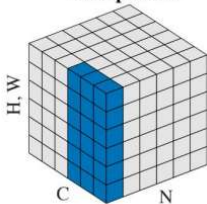


归一化层的比较

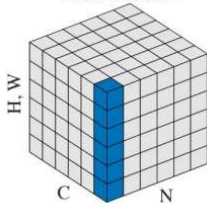
Layer Norm



Group Norm

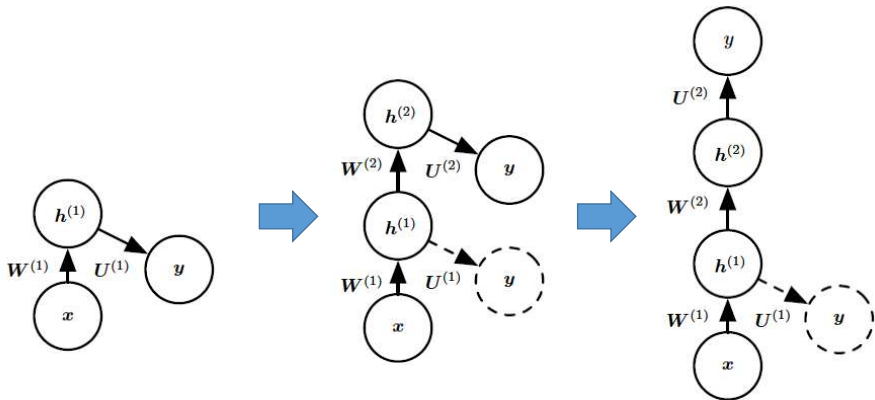


Instance Norm



监督预训练

- 模型太复杂，或任务非常困难

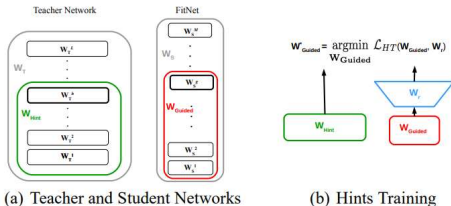


监督预训练-迁移学习

- 在一组任务上预训练了8层权重的深度卷积网络
 - 1000个ImageNet对象类的子集
- 用该网络的前 k 层初始化同样规模的网络，其余层随机初始化
 - 联合训练以执行不同的任务（1000个ImageNet对象类的另一个子集）
 - 训练样本少于第一个任务

监督预训练-FitNets

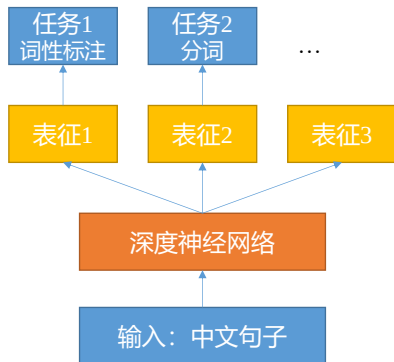
- 教师网络：深度浅（5层）且足够宽的网络，较容易训练
- 学生网络：深度深（11-19层）且窄的网络，很难用SGD训练
- 学生网络预测**原任务的输出**而且预测教师网络**中间层的值**
 - 学生网络中间层去回归教师网络的中间层



$$\mathcal{L}_{HT}(\mathbf{W}_{\text{Guided}}, \mathbf{W}_{\text{r}}) = \frac{1}{2} \|u_h(\mathbf{x}; \mathbf{W}_{\text{Hint}}) - r(v_g(\mathbf{x}; \mathbf{W}_{\text{Guided}}); \mathbf{W}_{\text{r}})\|^2$$

多任务学习

- 通过合并几个任务中的样例来提高泛化的一种方式



课程学习

- 基于规划学习过程的想法，首先学习简单的概念，然后逐步学习依赖于这些简化概念的复杂概念
 - 在大规模的神经语言模型任务上使用课程学习，可以获得更好的结果
- 课程学习被证实为与人类教学方式一致
 - 教师刚开始会展示更容易、更典型的示例，然后帮助学习者在不太显然的情况下提炼决策面
 - 在人类教学上，基于课程学习的策略比基于样本均匀采样的策略更有效

课程学习

- 首先学习简单的、普适性的知识结构，然后逐渐增加难度，过渡到学习更复杂、更专业化的知识。



Samples of "Dog" to learn earlier.



Samples of "Dog" to learn later.

课程学习

- 首先学习简单的、普适性的知识结构，然后逐渐增加难度，过渡到学习更复杂、更专业化的知识。



“bus” samples to learn earlier



Age



“bus” samples to learn later



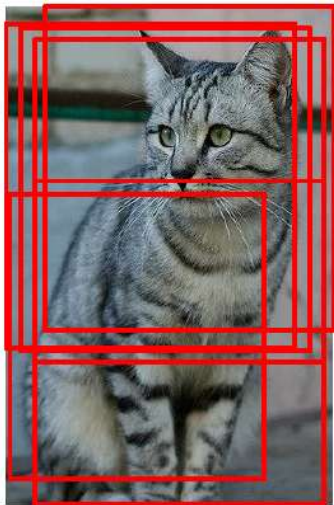
数据集增强—水平翻转



- 需要保证水平翻转不会改变类别
 - OCR中的字符b和d, 6和9

数据集增强——随机裁剪和缩放

- 训练
 - 从 $[256, 480]$ 的范围内随机取 L
 - 按比例缩放，其中短边长度为 L
 - 从缩放的图上采样 224×224 的图
- 测试
 - 五个尺度上的缩放 $\{224, 256, 384, 480, 640\}$
 - 每个大小的图片，用10个 224×224 的裁剪
 - $2(\text{水平翻转}) \times (4 \text{ 角} + 1 \text{ 中心})$



数据集增强—变色

- 调整色调，饱和度和亮度（例如 $[0.5, 1.5]$ ）



亮度

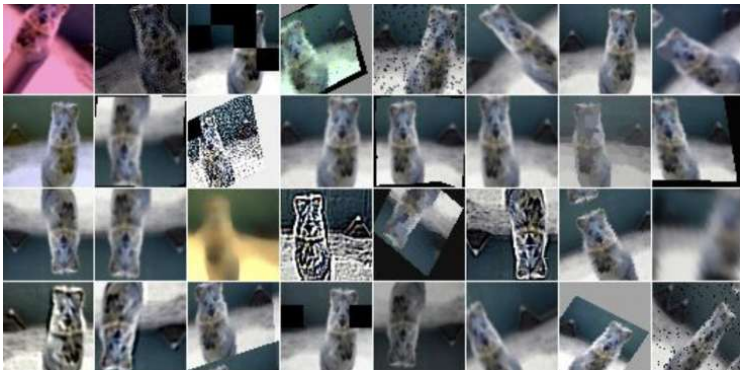


色调



数据集增强—更多方法

- 平移、旋转、伸缩、裁剪、扭曲的任意组合



数据集增强

- 向神经网络的输入层注入噪声也是一种数据集增强的方法

原图



加入 $\mathcal{N}(0, 0.1^2)$ 的噪声



也可以向隐藏单元施加噪声

数据集增强—标签平滑

- 在输出标签中添加噪声来避免模型过拟合
- 大多数数据集的 y 标签都有一定错误，错误的 y 不利于最大化 $\log p(y|x)$
- 一个样本 x 的标签一般用onehot向量表示

$$y = [0, \dots, 0, \textcolor{red}{1}, 0, \dots, 0]$$

- 引入一个噪声对标签进行平滑，即假设样本以 ϵ 的概率为其它类。平滑后的标签为

$$y = \left[\frac{\epsilon}{k-1}, \dots, \frac{\epsilon}{k-1}, \textcolor{red}{1-\epsilon}, \frac{\epsilon}{k-1}, \dots, \frac{\epsilon}{k-1} \right]$$

对抗学习

- 对抗样本: x 与 x' 非常相似, 人类难以察觉差异, 但神经网络会作出不同的预测


 x
 $y =$ 熊猫

w/ 57.7%

confidence

 $+ .007 \times$

 $\text{sign}(\nabla_x J(\theta, x, y))$

线虫

w/ 8.2%

confidence

 $=$

 $x +$
 $\epsilon \text{sign}(\nabla_x J(\theta, x, y))$

长臂猿

w/ 99.3 %

confidence

对抗学习

在对抗扰动的训练集样本上训练网络, 以减少原有独立同分布的测试集的错误率

正则化

- 新优化目标：不仅最小化损失，而且要让权重尽可能小

$$\mathcal{L}'(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2} \lambda \|\boldsymbol{\theta}\|_2^2 \quad \rightarrow \text{正则化项目}$$

损失，比如平方损失，
交叉熵损失等等

$$\boldsymbol{\theta} = \{w_1, w_2, \dots\}$$

L2 正则:

$$\|\boldsymbol{\theta}\|_2^2 = (w_1)^2 + (w_2)^2 + \dots$$

一般不考虑bias

L1 正则:

$$\|\boldsymbol{\theta}\|_1 = |w_1| + |w_2| + \dots$$

正则化

L2 正则:

$$\|\boldsymbol{\theta}\|_2^2 = (w_1)^2 + (w_2)^2 + \dots$$



33

$$\mathcal{L}'(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2}\lambda\|\boldsymbol{\theta}\|_2^2$$

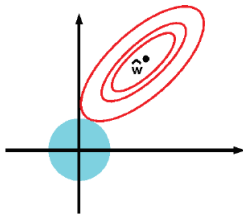
$$\text{梯度: } \nabla \mathcal{L}' = \nabla \mathcal{L} + \lambda \boldsymbol{\theta}$$

梯度下降更新:

$$\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^{t-1} - \eta \nabla \mathcal{L}' = \boldsymbol{\theta}^{t-1} - \eta \nabla \mathcal{L} - \eta \lambda \boldsymbol{\theta}^{t-1}$$

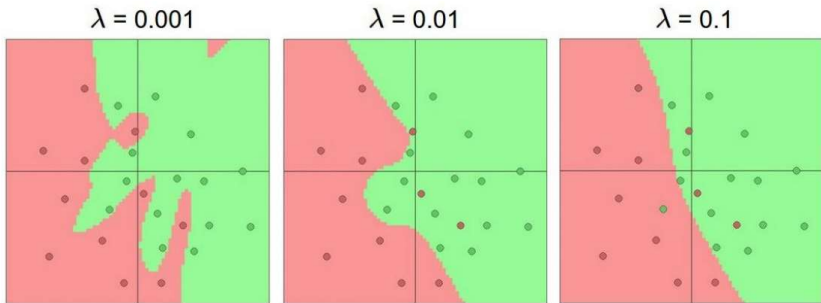
$$\eta < 1, \lambda < 1 \quad = \underbrace{(1 - \eta \lambda)}_{\text{越来越小}} \boldsymbol{\theta}^{t-1} - \eta \nabla \mathcal{L}$$

越来越小，但由于 $-\eta \nabla \mathcal{L}$ 项，使得参数不会变为0



L2正则也称权重衰减 (weight decay)

神经网络示例



<http://playground.tensorflow.org/>

正则化

L1 正则:

$$\|\boldsymbol{\theta}\|_1 = |w_1| + |w_2| + \dots$$

$$\mathcal{L}'(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{1}{2}\lambda\|\boldsymbol{\theta}\|_1$$

$$\text{梯度: } \nabla \mathcal{L}' = \nabla \mathcal{L} + \lambda \text{sign}(\boldsymbol{\theta})$$

梯度下降更新:

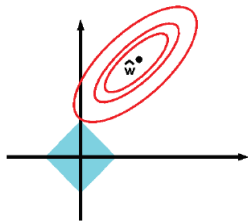
$$\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^{t-1} - \eta \nabla \mathcal{L}' = \boldsymbol{\theta}^{t-1} - \eta \nabla \mathcal{L} - \eta \lambda \text{sign}(\boldsymbol{\theta}^{t-1})$$

$$= \underbrace{\boldsymbol{\theta}^{t-1} - \eta \lambda \text{sign}(\boldsymbol{\theta}^{t-1})}_{\text{L1 正则项}} - \eta \nabla \mathcal{L}$$

- 如果 $w^{t-1} > 0$, w^{t-1} 变小
- 如果 $w^{t-1} < 0$, w^{t-1} 变大

$$\Rightarrow w^{t-1} \rightarrow 0$$

具有稀疏的特性



添加噪声是一种正则化

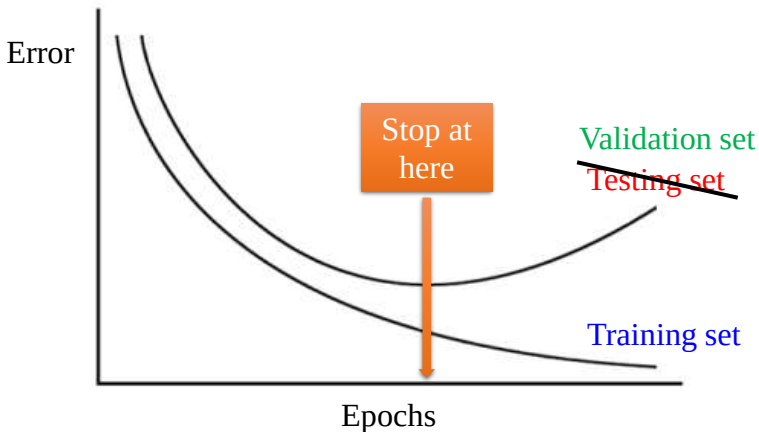
- 假设网络权重添加随机扰动 $\epsilon_w \sim \mathcal{N}(0, \eta I)$, 扰动模型为: $\hat{y}_{\epsilon_w}(x)$
- 将平方损失作为目标函数

$$\begin{aligned}
 \tilde{J}_W &= \mathbb{E}_{p(x,y,\epsilon_w)} \left[(\hat{y}_{\epsilon_w}(x) - y)^2 \right] && \text{一阶泰勒展开近似} \\
 &= \mathbb{E}_{p(x,y,\epsilon_w)} \left[\hat{y}_{\epsilon_w}^2(x) - 2y\hat{y}_{\epsilon_w}(x) + y^2 \right] && \hat{y}_{\epsilon_w}(x) \approx y_w(x) + \nabla_w y_w(x)^T \epsilon_w \\
 &\approx \mathbb{E}_{p(x,y,\epsilon_w)} \left[y_w^2(x) + 2y_w(x) \nabla_w y_w(x)^T \epsilon_w + \epsilon_w \nabla_w y_w(x) \nabla_w y_w(x)^T \epsilon_w \right. \\
 &\quad \left. + \mathbb{E}_{p(x,y,\epsilon_w)} [-2y y_w(x) - 2y \nabla_w y_w(x)^T \epsilon_w + y^2] \right] \\
 &= \mathbb{E}_{p(x,y)} [(y_w(x) - y)^2] + \mathbb{E}_{p(x,y)} \left[\text{tr}(\nabla_w y_w(x) \nabla_w y_w(x)^T \mathbb{E}_{p(\epsilon_w)} [\epsilon_w \epsilon_w^T]) \right] \\
 &= J_W + \eta \mathbb{E}_{p(x,y)} [\|\nabla_w y_w(x)\|^2]
 \end{aligned}$$

$\mathbb{E}_{p(\epsilon_w)} [\epsilon_w \epsilon_w^T] = \eta I$

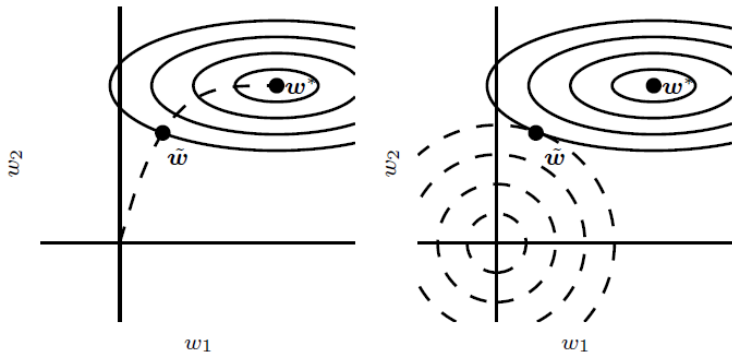
鼓励参数进入 **权重小扰动对输出相对影响较小**的参数空间区域

提前终止



- 验证集上准确率下降（损失上升）的时候停止训练
- 训练很长时间，保存在验证集上最优的模型

提前终止也是一种正则化



提前终止也是一种正则化

- 考虑损失函数 J 在经验最佳值 $\mathbf{w}^*$ 附近的二次近似

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\top \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

$\mathbf{H}$ 是 J 关于 $\mathbf{w}$ 在 $\mathbf{w}^*$ 点的Hessian矩阵

梯度为 $\nabla \hat{J}(\mathbf{w}) = \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$

- 梯度下降 τ 次之后

$$\begin{aligned}\mathbf{w}^{(\tau)} &= \mathbf{w}^{(\tau-1)} - \epsilon \nabla \hat{J}(\mathbf{w}^{(\tau-1)}) \\ &= \mathbf{w}^{(\tau-1)} - \epsilon \mathbf{H}(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*)\end{aligned}$$

$$\begin{aligned}\Rightarrow \mathbf{w}^{(\tau)} - \mathbf{w}^* &= (\mathbf{I} - \epsilon \mathbf{H})(\mathbf{w}^{(\tau-1)} - \mathbf{w}^*) \\ &= (\mathbf{I} - \epsilon \mathbf{H})^\tau (\mathbf{w}^{(0)} - \mathbf{w}^*)\end{aligned}$$

$$\mathbf{w}^{(0)} = \mathbf{0}$$



$$\mathbf{w}^{(\tau)} = (\mathbf{I} - (\mathbf{I} - \epsilon \mathbf{H})^\tau) \mathbf{w}^*$$

提前终止也是一种正则化

- 考虑损失函数 J 在经验最佳值 $\mathbf{w}^*$ 附近的二次近似

$$\hat{J}(\mathbf{w}) = J(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^\top \mathbf{H}(\mathbf{w} - \mathbf{w}^*)$$

$\mathbf{H}$ 是 J 关于 $\mathbf{w}$ 在 $\mathbf{w}^*$ 点的Hessian矩阵

- 考虑L2 正则的新目标

$$\hat{J}'(\mathbf{w}) = \hat{J}(\mathbf{w}) + \frac{1}{2}\alpha\|\mathbf{w}\|_2^2$$

- 对 $\hat{J}'(\mathbf{w})$ 最小化有最优解

$$\mathbf{w} = (\mathbf{H} + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{w}^*$$

提前终止也是一种正则化

假设 H 的特征值分解为 $H = Q\Lambda Q^T$

L2正则

$$\begin{aligned} \mathbf{w} &= (H + \alpha I)^{-1} H \mathbf{w}^* \\ &= Q(\Lambda + \alpha I)^{-1} \Lambda Q^T \mathbf{w}^* \\ &= Q(I - (\Lambda + \alpha I)^{-1} \alpha) Q^T \mathbf{w}^* \end{aligned}$$

提前终止

$$\begin{aligned} \mathbf{w}^{(\tau)} &= (I - (I - \epsilon H)^\tau) \mathbf{w}^* \\ &= Q[I - (I - \epsilon \Lambda)^\tau] Q^T \mathbf{w}^* \end{aligned}$$

$$\text{令 } (\Lambda + \alpha I)^{-1} \alpha = (I - \epsilon \Lambda)^\tau \quad \text{即 } \frac{\alpha}{\lambda_i + \alpha} = (1 - \epsilon \lambda_i)^\tau$$

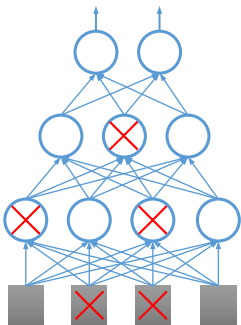
$$\text{取对数 } -\log\left(1 + \frac{\lambda_i}{\alpha}\right) = \tau \log(1 - \epsilon \lambda_i)$$

$$\begin{aligned} \text{当 } \frac{\lambda_i}{\alpha} \ll 1 \text{ 且 } \epsilon \lambda_i \ll 1, \\ \text{有 } \log(1 + x) \approx x \end{aligned} \quad \Rightarrow \quad -\frac{\lambda_i}{\alpha} \approx -\epsilon \tau \lambda_i \quad \Rightarrow \quad \tau \approx \frac{1}{\alpha \epsilon}$$

训练迭代次数起着与 L^2 参数成反比的作用

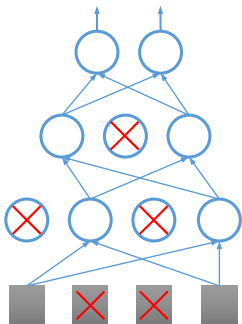
Dropout

- 在训练时, 取得一个batch后, 以概率 p 设置一些神经元为0



Dropout

- 在训练时，取得一个batch后，以概率 p 设置一些神经元为0



- 网络结构发生了变化
- 用新网络在batch上计算梯度
- 在新网络上进行参数更新

每次取新batch时，都需要重新随机对神经元置为0

Dropout—训练时的实现

```
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
    """ X contains the data """

    # forward pass for example 3-layer neural network
    H1 = np.maximum(0, np.dot(W1, X) + b1)
    U1 = np.random.rand(*H1.shape) < p # first dropout mask
    H1 *= U1 # drop!
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    U2 = np.random.rand(*H2.shape) < p # second dropout mask
    H2 *= U2 # drop!
    out = np.dot(W3, H2) + b3

    # backward pass: compute gradients... (not shown)
    # perform parameter update... (not shown)
```

输入单元被包括概率为0.8，隐藏单元为0.5

Dropout

- 测试时，没有dropout，即所有的神经员都处于激活状态
- 缩放激活函数的输出，使得每个神经元测试时输出等于训练时的期望输出

权重比例推断规则

如果训练时dropout 的概率为 p ，那么所有的权重要乘以 $1 - p$

Dropout—训练和测试时实现

该概率为保留概率

$p = 0.5$ # *probability of a unit active. higher = less dropout*

```
def train_step(X):
```

```
    """ X contains the data """
```

```
    # forward pass for example 3-layer neural network
```

```
    H1 = np.maximum(0, np.dot(W1, X) + b1)
```

```
    U1 = np.random.rand(*H1.shape) < p # first dropout mask
```

```
    H1 *= U1 # drop!
```

```
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
```

```
    U2 = np.random.rand(*H2.shape) < p # second dropout mask
```

```
    H2 *= U2 # drop!
```

```
    out = np.dot(W3, H2) + b3
```

```
def predict(X):
```

```
    # ensembled forward pass
```

```
    H1 = np.maximum(0, np.dot(W1, X) + b1) * p # NOTE: scale the activations
```

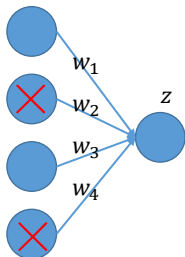
```
    H2 = np.maximum(0, np.dot(W2, H1) + b2) * p # NOTE: scale the activations
```

```
    out = np.dot(W3, H2) + b3
```

Dropout—直观解释

训练时dropout

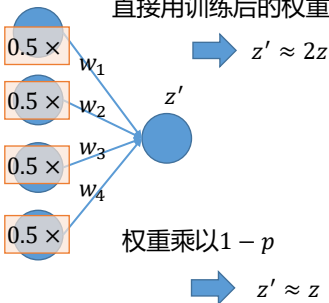
dropout 概率为0.5



测试时dropout

No dropout

直接用训练后的权重



权重乘以 $1 - p$

$z' \approx z$

Dropout动机

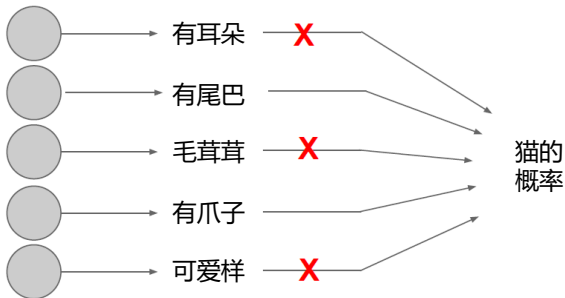
1

强制网络有冗余特征表示

2

防止特征的co-adaptation

co-adaptation: feature detectors只有在一些其它特定的feature detectors存在时才能发挥作用的情况



Dropout动机

- 人脸识别情形下的动机：

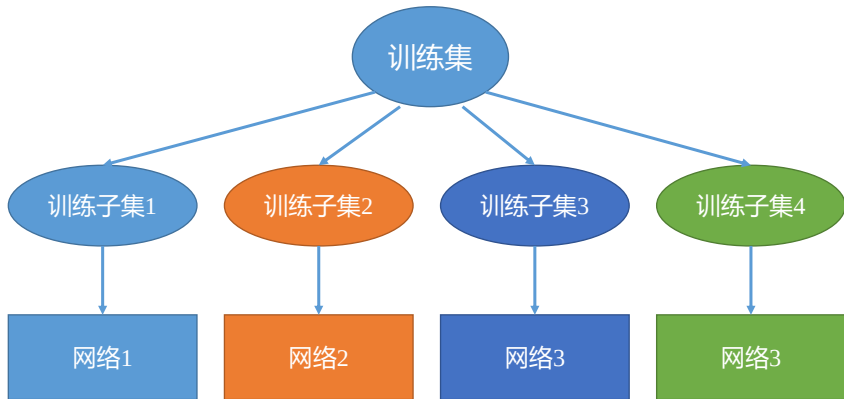
1

模型学得通过鼻检测脸的隐藏单元 h_i ，丢失 h_i 对应于擦除图像中有鼻子的信息

2

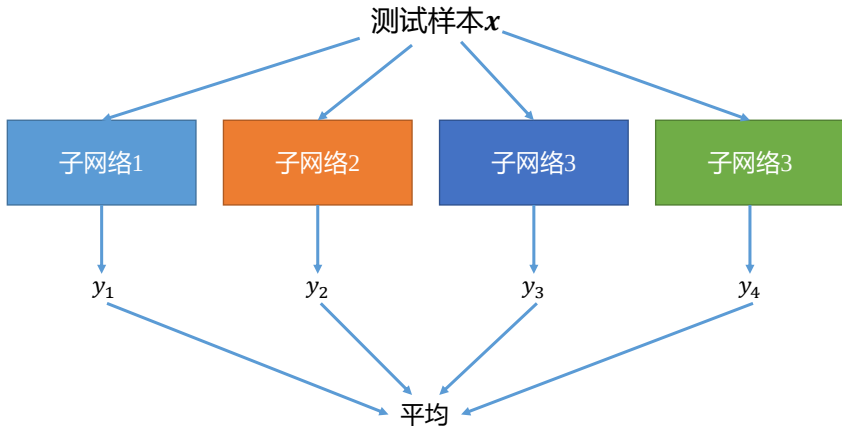
模型必须学习另一种 h_i ，要么是鼻子存在的冗余编码，要么是像嘴这样的脸部的另一特征

Dropout 是一种模型集成 (Bagging)



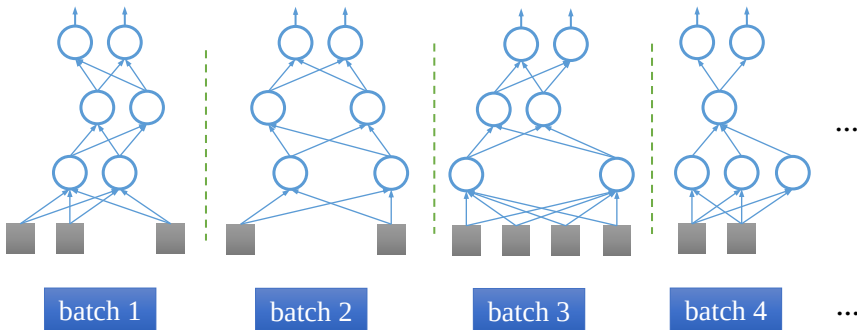
训练一系列不同结构的神经网络

Dropout 是一种模型集成 (Bagging)



Dropout 是一种模型集成 (Bagging)

训练时的dropout

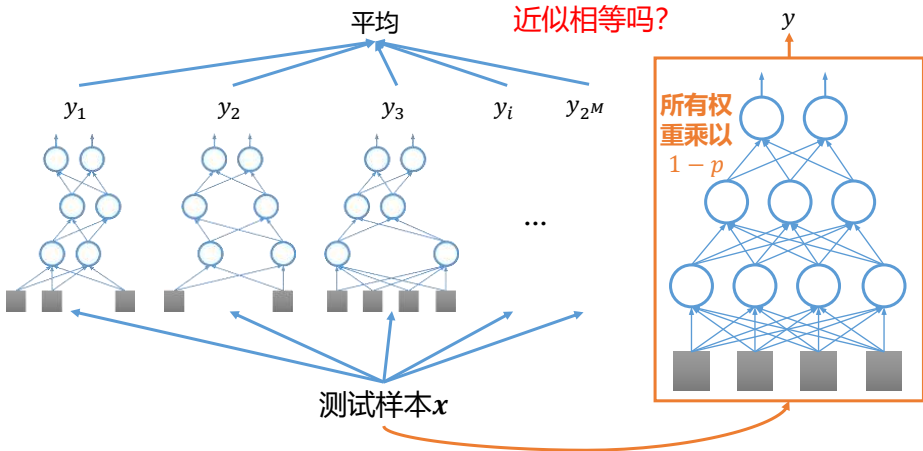


- 若神经网络有 M 个神经元，则共有 2^M 个不同结构的网络
- 一个batch数据训练一个结构的网络，但只更新参数一次
- 不同结构的网络共享部分参数

Dropout 是一种模型集成 (Bagging)

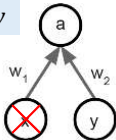
Dropout	Bagging
所有模型共享参数	所有模型相互独立
每一步训练一小部分的网络	每个模型在相应数据集上训练到收敛

Dropout 是一种模型集成 (Bagging)

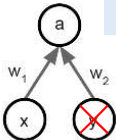


Dropout 是一种模型集成 (Bagging)

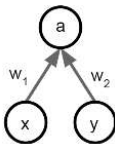
$$a = 0x + w_2y$$



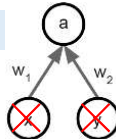
$$a = w_1x + 0y$$



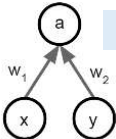
$$a = w_1x + w_2y$$



$$a = 0x + 0y$$



$$a = w_1x + w_2y$$



训练时:
$$\begin{aligned} \mathbb{E}[a] &= \frac{1}{4}(w_1x + w_2y) + \frac{1}{4}(w_1x + 0y) + \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2y) \\ &= \frac{1}{2}(w_1x + w_2y) \end{aligned}$$

测试时:
$$\mathbb{E}[a] = w_1x + w_2y$$
 所有权重都乘以 1/2

Dropout 是一种模型集成 (Bagging)

- 权重比例推断规则对许多非线性隐藏单元模型族是精确的
- 考虑 $P(y = y|\mathbf{v}, \boldsymbol{\mu}) = \text{softmax}(\mathbf{W}^T(\mathbf{v} \odot \boldsymbol{\mu}) + \mathbf{b})_y$

$$\begin{aligned}
 \tilde{P}_{\text{ensemble}}(y = y|\mathbf{v}) &= \sqrt[n]{\prod_{\boldsymbol{\mu} \in \{0,1\}^n} P(y = y|\mathbf{v}; \boldsymbol{\mu})} && \text{使用集成成员预测分布的几何平均而不是算术平均} \\
 &= \sqrt[n]{\prod_{\boldsymbol{\mu} \in \{0,1\}^n} \frac{\exp(\mathbf{W}_{y,:}^T(\mathbf{v} \odot \boldsymbol{\mu}) + b_y)}{\sum_{y'} \exp(\mathbf{W}_{y',:}^T(\mathbf{v} \odot \boldsymbol{\mu}) + b_{y'})}} \\
 &\propto \sqrt[n]{\prod_{\boldsymbol{\mu} \in \{0,1\}^n} \exp(\mathbf{W}_{y,:}^T(\mathbf{v} \odot \boldsymbol{\mu}) + b_y)} \\
 &= \exp\left(\frac{1}{n} \sum_{\boldsymbol{\mu} \in \{0,1\}^n} (\mathbf{W}_{y,:}^T(\mathbf{v} \odot \boldsymbol{\mu}) + b_y)\right) \\
 &= \exp\left(\frac{1}{2} \mathbf{W}_{y,:}^T \mathbf{v} + b_y\right)
 \end{aligned}$$



Dropout 是一种模型集成 (Bagging)

- 权重比例推断规则对许多**无非线性隐藏单元**的模型族是**精确的**
 - softmax函数回归分类
 - 条件正态输出的回归网络
 - 隐层不含非线性的深度网络

Dropout是一种正则化

- 考虑基于指数分布族的广义线性模型，囊括线性回归、对率回归

$$P_{\boldsymbol{\beta}}(y|\mathbf{x}) = h(y) \exp\{y \mathbf{x}^{\top} \boldsymbol{\beta} - A(\mathbf{x}^{\top} \boldsymbol{\beta})\}$$

$$A(\mathbf{x}^{\top} \boldsymbol{\beta}) = \log \int h(y) \exp\{y \mathbf{x}^{\top} \boldsymbol{\beta}\} dy \quad \text{为对数配分函数，用于归一化}$$

- 比如对率回归

$$P_{\boldsymbol{\beta}}(y|\mathbf{x}) = \frac{\exp(y\mathbf{x}^{\top} \boldsymbol{\beta})}{1 + \exp(\mathbf{x}^{\top} \boldsymbol{\beta})}$$

$$y \in \{0,1\} \quad = \exp\{y\mathbf{x}^{\top} \boldsymbol{\beta} - \log(1 + \exp(\mathbf{x}^{\top} \boldsymbol{\beta}))\}$$

- 比如线性回归

$$\begin{aligned} P_{\boldsymbol{\beta}}(y|\mathbf{x}) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(y - \mathbf{x}^{\top} \boldsymbol{\beta})^2\right) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) \exp\left(y\mathbf{x}^{\top} \boldsymbol{\beta} - \frac{1}{2}(\mathbf{x}^{\top} \boldsymbol{\beta})^2\right) \end{aligned}$$

Dropout是一种正则化

- 考虑基于指数分布族的广义线性模型，囊括线性回归、对率回归

$$P_{\boldsymbol{\beta}}(y|\mathbf{x}) = h(y) \exp\{y \mathbf{x}^{\top} \boldsymbol{\beta} - A(\mathbf{x}^{\top} \boldsymbol{\beta})\}$$

$A(\mathbf{x}^{\top} \boldsymbol{\beta}) = \log \int h(y) \exp\{y \mathbf{x}^{\top} \boldsymbol{\beta}\} dy$ 为对数配分函数，用于归一化

- 指数分布族的重要性质

$$A'(\mathbf{x}^{\top} \boldsymbol{\beta}) = \mathbb{E}_{P_{\boldsymbol{\beta}}(y|\mathbf{x})}[y]$$

$$A''(\mathbf{x}^{\top} \boldsymbol{\beta}) = \mathbb{V}_{P_{\boldsymbol{\beta}}(y|\mathbf{x})}[y]$$

Dropout是一种正则化

- 对于样本 (x, y) , 其损失定义为 $\ell_{x,y}(\boldsymbol{\beta}) = -\log P_{\boldsymbol{\beta}}(y|x)$

- 考虑Dropout 噪声 $\tilde{x} = x \odot \xi$

ξ_j	0	$1/(1-\delta)$
概率p	δ	$1-\delta$

$$\mathbb{E}[\tilde{x}] = x$$

$$\mathbb{V}[\tilde{x}] = \frac{\delta}{1-\delta} \text{diag}(xx^\top)$$



$$\mathbb{E}[\tilde{x}_j] = x_j \quad \mathbb{V}[\tilde{x}_j] = \frac{\delta}{1-\delta} x_j^2$$

- 含噪声的损失函数定义为

$$\begin{aligned}
 \mathcal{L}(\boldsymbol{\beta}) &= \sum_{i=1}^n \mathbb{E}_{\xi} [\ell_{\tilde{x}_i, y_i}(\boldsymbol{\beta})] \\
 &= -\sum_{i=1}^n (y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \mathbb{E}_{\xi} [A(\tilde{\mathbf{x}}_i^\top \boldsymbol{\beta})]) \\
 &= \sum_{i=1}^n \underbrace{-\left(y_i \mathbf{x}_i^\top \boldsymbol{\beta} - A(\mathbf{x}_i^\top \boldsymbol{\beta})\right)}_{\ell_{x_i, y}(\boldsymbol{\beta})} + \underbrace{\left(\mathbb{E}_{\xi} [A(\tilde{\mathbf{x}}_i^\top \boldsymbol{\beta})] - A(\mathbf{x}_i^\top \boldsymbol{\beta})\right)}_{R(\boldsymbol{\beta})} \\
 &= \sum_{i=1}^n \ell_{x_i, y}(\boldsymbol{\beta}) + R(\boldsymbol{\beta})
 \end{aligned}$$

正则项

Dropout是一种正则化

• 正则项
$$R(\boldsymbol{\beta}) = \sum_{i=1}^n \mathbb{E}_{\xi} [A(\tilde{\mathbf{x}}_i^{\top} \boldsymbol{\beta})] - A(\mathbf{x}_i^{\top} \boldsymbol{\beta})$$

考虑 $A(\tilde{\mathbf{x}}_i^{\top} \boldsymbol{\beta})$ 在 $\mathbf{x}_i^{\top} \boldsymbol{\beta}$ 附近的二阶泰勒展开

$$\begin{aligned} \mathbb{E}_{\xi} [A(\tilde{\mathbf{x}}_i^{\top} \boldsymbol{\beta})] &\approx A(\mathbf{x}_i^{\top} \boldsymbol{\beta}) + A'(\mathbf{x}_i^{\top} \boldsymbol{\beta}) \mathbb{E}_{\xi} [(\tilde{\mathbf{x}}_i^{\top} \boldsymbol{\beta} - \mathbf{x}_i^{\top} \boldsymbol{\beta})] + 1/2 A''(\mathbf{x}_i^{\top} \boldsymbol{\beta}) \mathbb{E}_{\xi} [(\tilde{\mathbf{x}}_i^{\top} \boldsymbol{\beta} - \mathbf{x}_i^{\top} \boldsymbol{\beta})^2] \\ &= A(\mathbf{x}_i^{\top} \boldsymbol{\beta}) + \frac{1}{2} A''(\mathbf{x}_i^{\top} \boldsymbol{\beta}) \boldsymbol{\beta}^{\top} \mathbb{V}_{\xi}(\tilde{\mathbf{x}}_i) \boldsymbol{\beta} \end{aligned}$$

➡
$$\begin{aligned} R(\boldsymbol{\beta}) &= \frac{1}{2} \frac{\delta}{1 - \delta} \boldsymbol{\beta}^{\top} \sum_{i=1}^n A''(\mathbf{x}_i^{\top} \boldsymbol{\beta}) \text{diag}(\mathbf{x}_i \mathbf{x}_i^{\top}) \boldsymbol{\beta} \\ &= \frac{1}{2} \frac{\delta}{1 - \delta} \boldsymbol{\beta}^{\top} \text{diag}(\mathbf{X}^{\top} V(\boldsymbol{\beta}) \mathbf{X}) \boldsymbol{\beta} \end{aligned}$$

记 $V(\boldsymbol{\beta})$ 为 $n \times n$ 对角阵, 对角线上元素为 $A''(\mathbf{x}_i^{\top} \boldsymbol{\beta})$

Dropout是一种正则化

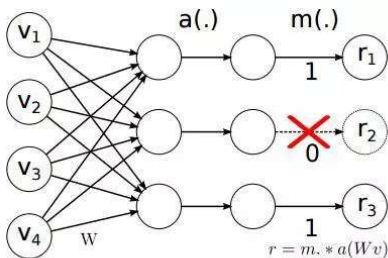
- 对率回归, 记 $P_{\beta}(y = 1|x_i) = p_i$,

$$\text{则 } A''(\mathbf{x}_i^{\top} \boldsymbol{\beta}) = \mathbb{V}_{P_{\beta}(y|x_i)}[y] = p_i(1 - p_i)$$

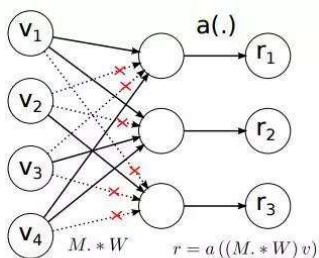
$$\begin{aligned} R(\boldsymbol{\beta}) &= \frac{1}{2} \frac{\delta}{1 - \delta} \boldsymbol{\beta}^{\top} \text{diag}(\mathbf{X}^{\top} V(\boldsymbol{\beta}) \mathbf{X}) \boldsymbol{\beta} \\ &= \frac{1}{2} \frac{\delta}{1 - \delta} \sum_{i,j} x_{ij}^2 \beta_j^2 p_i(1 - p_i) \end{aligned}$$

Dropout偏好置信度高的预测和小的 β

DropConnect



DropOut Network



DropConnect Network

DropConnect: 将节点中的每个与其相连的输入权值以 $1-p$ 的概率变成0

Dropout: 随机的将隐层节点的输出变成0



谢谢





循环神经网络

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>

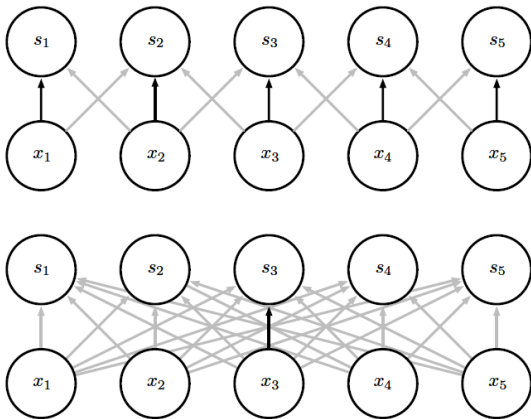


本章内容

- 循环神经网络的动机
- 循环神经网络的应用方式
- 前馈神经网络的参数学习
- 前馈神经网络的长程依赖
- 循环神经网络的应用

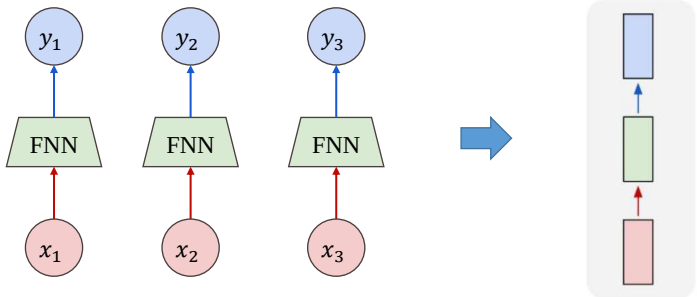
前馈神经网络

- 连接存在层与层之间，每层的节点之间是无连接的（无循环）



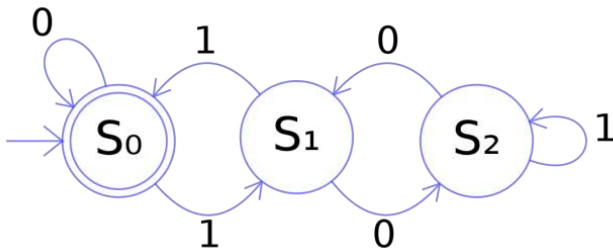
前馈神经网络

- 假设每次输入都是独立的，也就是说每次网络的输出只依赖于当前的输入



前馈神经网络存在问题

- 输入和输出维数固定，不能任意改变，无法处理变长的序列数据，比如视频、语音、文本等
- 很多现实任务，网络输出不仅和当前时刻的输入相关，也和其过去一段时间的输出相关
 - 比如，在有限状态自动机中，下一个时刻的状态不仅和当前输入相关，也和当前状态（上一个时刻的输出）相关



如何用FNN去模拟一个有限状态自动机？

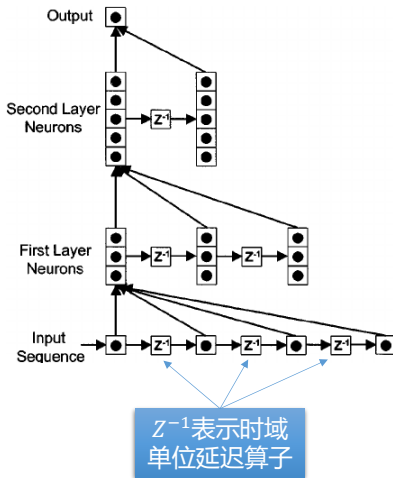
延时神经网络(Time Delay Neural Network)

- 在前馈神经网络的非输出层都添加一个延时器，记录神经元的最近几次活性值

$$h_t^{(l)} = f(h_t^{(l-1)}, h_{t-1}^{(l-1)}, \dots, h_{t-K}^{(l-1)})$$

$h_t^{(l)} \in \mathbb{R}^{M_l}$ 表示第 l 层神经元在时刻 t 的活性值

- 这样，前馈网络就具有了短期记忆的能力。



自回归模型 (Autoregressive Model, AR)



- 一类时间序列模型，用变量 y_t 的历史信息来预测自己

$$y_t = w_0 + \sum_{k=1}^K w_k y_{t-k} + \epsilon_t$$

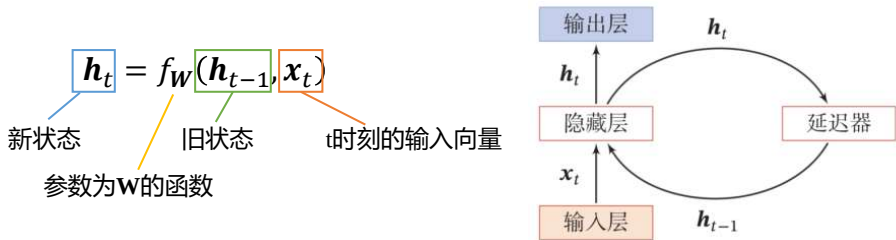
- K 为超参数， $w_0, \dots, w_K$ 为可学习参数， $\epsilon_t \sim N(0, \sigma^2)$ 为第 t 个时刻的噪声
- 有外部输入的非线性自回归模型
 - 每个时刻 t 都有一个输入 x_t ，产生一个输出 y_t
 - 通过一个延时器记录最近 K_x 次的外部输入和最近的 K_y 次输出

$$y_t = f(x_t, x_{t-1}, \dots, x_{t-K_x}, y_{t-1}, y_{t-2}, \dots, y_{t-K_y})$$

- $f(\cdot)$ 表示非线性函数，可以为前馈网络， K_x 和 K_y 为超参数

循环神经网络 (Recurrent Neural Network)

- 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。



状态 h_t “存储” 直到t时刻的历史数据 $\{x_1, \dots, x_t\}$

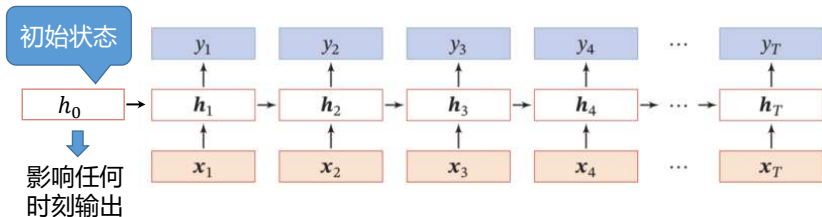
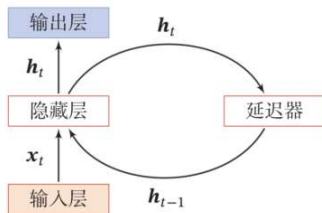
$$h_t = f_W(f_W(f_W(h_{t-3}, x_{t-2}), x_{t-1}), x_t)$$

按时间展开

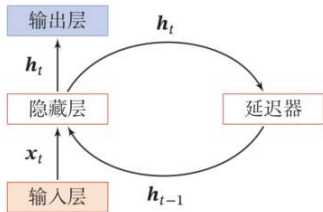
$$\boxed{h_t} = f_w(\boxed{h_{t-1}}, \boxed{x_t})$$

新状态 旧状态 t时刻的输入向量

参数为 w 的函数



Elman RNN



$$h_t = f_W(h_{t-1}, x_t)$$



$$h_t = f(Uh_{t-1} + Wx_t + b)$$

U 为状态-状态权重矩阵, W 为状态-输入权重矩阵, b 为偏置向量

循环神经网络的计算能力

- 一个完全连接的循环网络

$$\mathbf{h}_t = f(U\mathbf{h}_{t-1} + W\mathbf{x}_t + \mathbf{b})$$

$$\mathbf{y}_t = V\mathbf{h}_t$$

则该网络是任何非线性动力系统的近似器

定理 6.1 – 循环神经网络的通用近似定理 [Haykin, 2009]: 如果一个完全连接的循环神经网络有足够数量的 sigmoid 型隐藏神经元, 它可以以任意的准确率去近似任何一个非线性动力系统

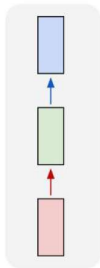
$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, \mathbf{x}_t), \quad (6.10)$$

$$\mathbf{y}_t = o(\mathbf{s}_t), \quad (6.11)$$

其中 $\mathbf{s}_t$ 为每个时刻的隐状态, $\mathbf{x}_t$ 是外部输入, $g(\cdot)$ 是可测的状态转换函数, $o(\cdot)$ 是连续输出函数, 并且对状态空间的紧致性没有限制。

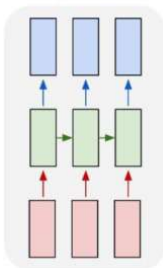
循环神经网络的使用方式

one to one



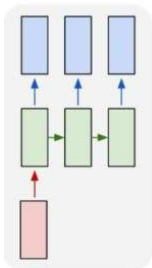
前向网络

many to many



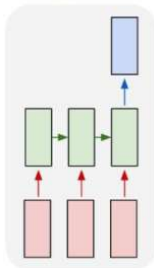
例：词性标注
句子 → 类别

one to many



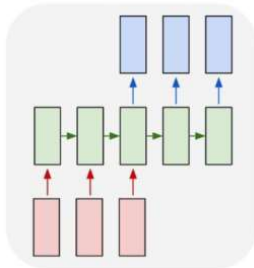
例：图像描述
图像 → 句子

many to one



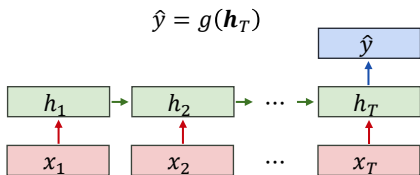
例：文本分类
文档 → 类别

many to many

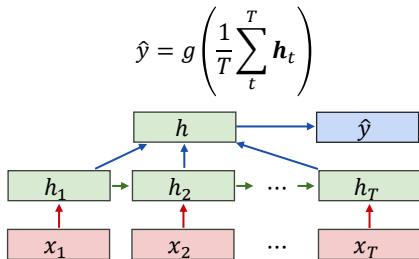


例：机器翻译
句子 → 句子

序列到类别



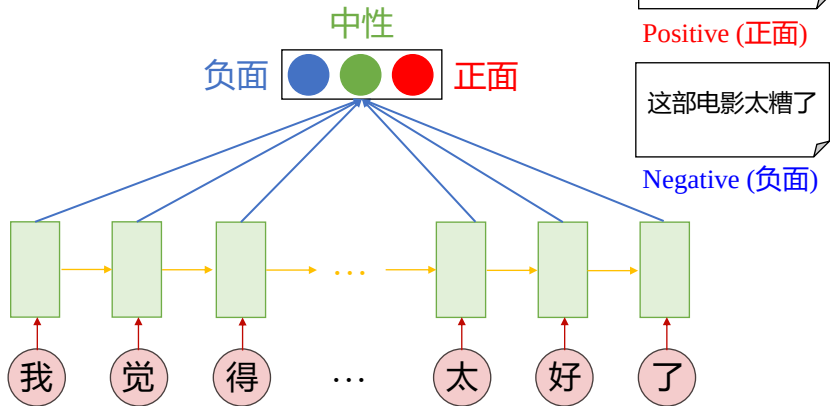
正常模式



按时间进行平均采样模式

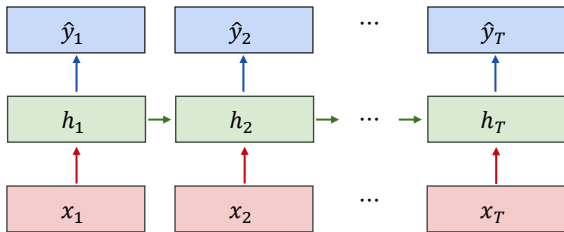
$g(\cdot)$ 可以是简单的线性分类器（对率回归）或者复杂分类器（多层前馈神经网络）

序列到类别—情感分类



同步的序列到序列模式

$$\hat{y}_t = g(\mathbf{h}_t), \forall t \in [1, T]$$



可以用于词性标注、分词、词义消歧、视频帧分类、信息抽取、语音识别等

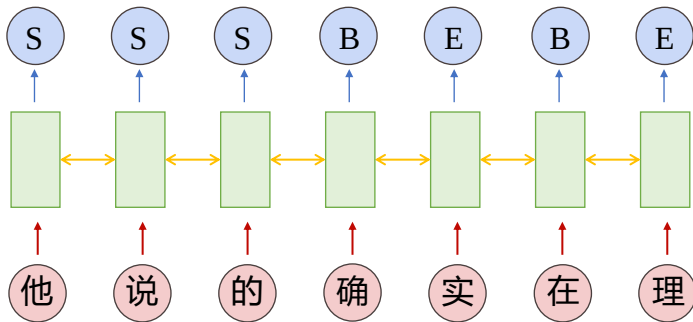
同步的序列到序列模式—中文分词

词首 B

词中 M

词尾 E

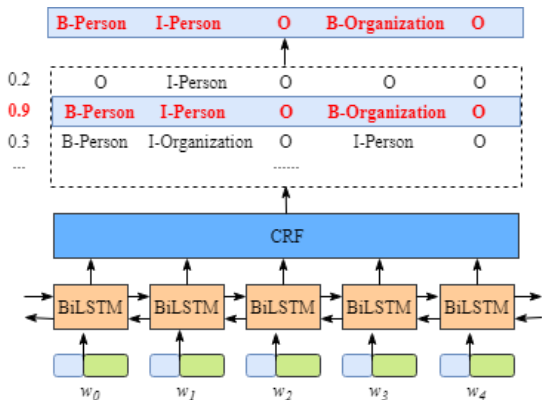
单字成词 S



同步的序列到序列模式—信息抽取

- 从无结构的文本中抽取结构化的信息，形成知识

小米创始人雷军表示，该公司2015年营收达到780亿元人民币，较2014年的743亿元人民币增长了5%。

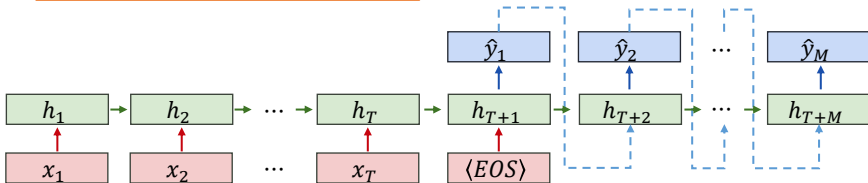


异步的序列到序列模式

编码器

$$\mathbf{h}_t = f_1(\mathbf{h}_{t-1}, \mathbf{x}_t), \forall t \in [1, T]$$

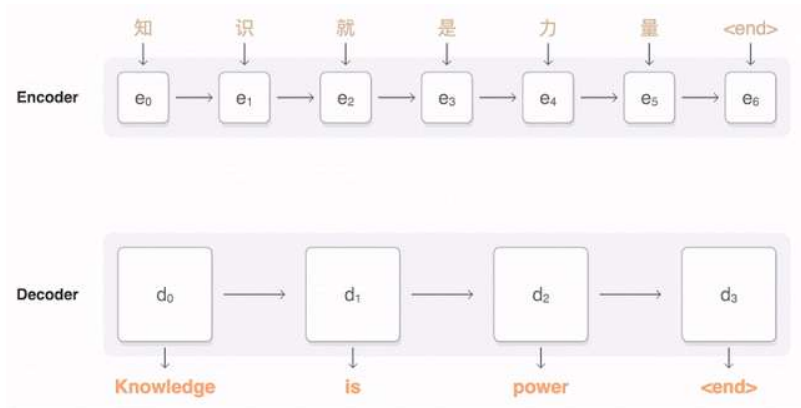
$$\mathbf{y}_t = g(\mathbf{h}_{T+t}), \forall t \in [1, M]$$



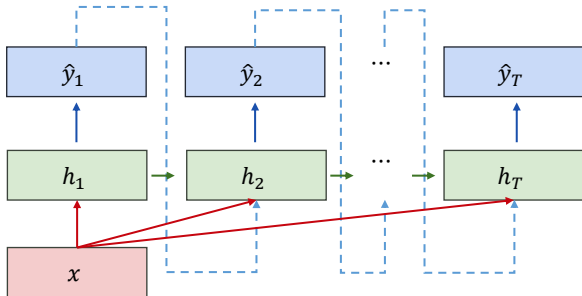
$$\mathbf{h}_{T+t} = f_2(\mathbf{h}_{T+t-1}, \hat{\mathbf{y}}_{t-1}), \forall t \in [1, M]$$

解码器

异步的序列到序列模式—机器翻译



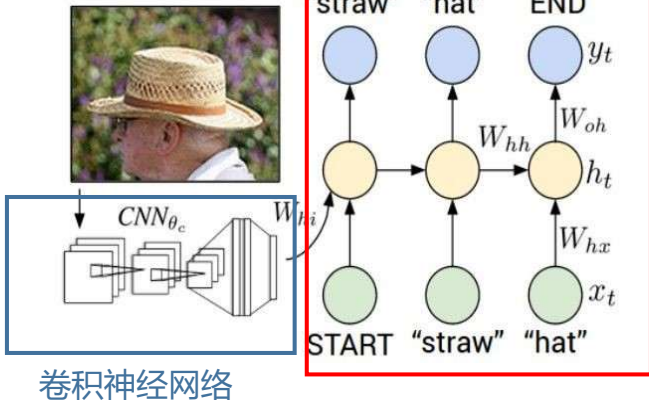
向量到序列



$$h_t = f(h_{t-1}, x, \hat{y}_{t-1}), \forall t \in [1, T]$$

向量到序列—图像描述

循环神经网络

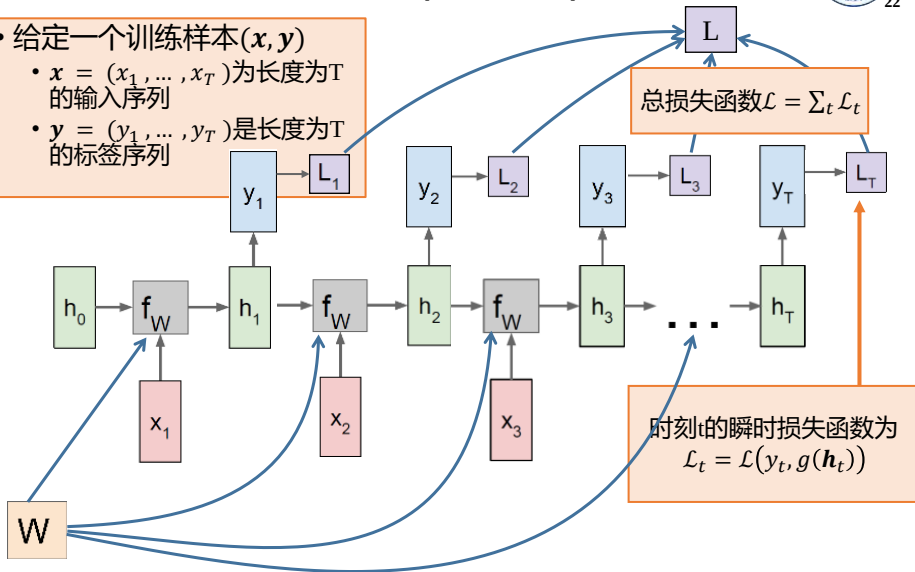


卷积神经网络

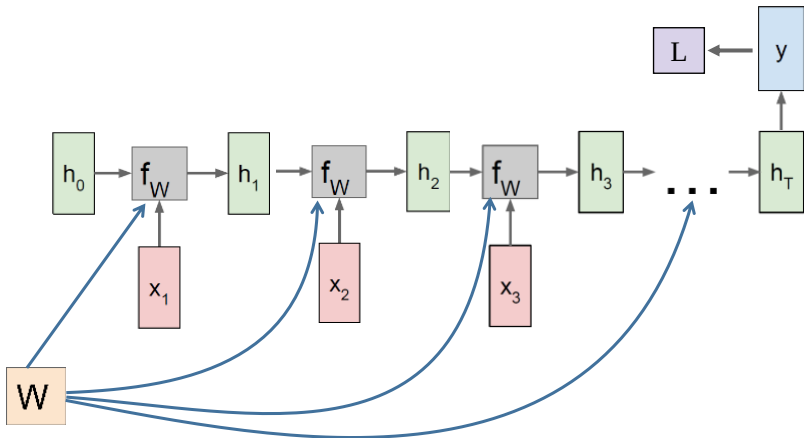
参数学习—计算图 (多对多)

• 给定一个训练样本 (x, y)

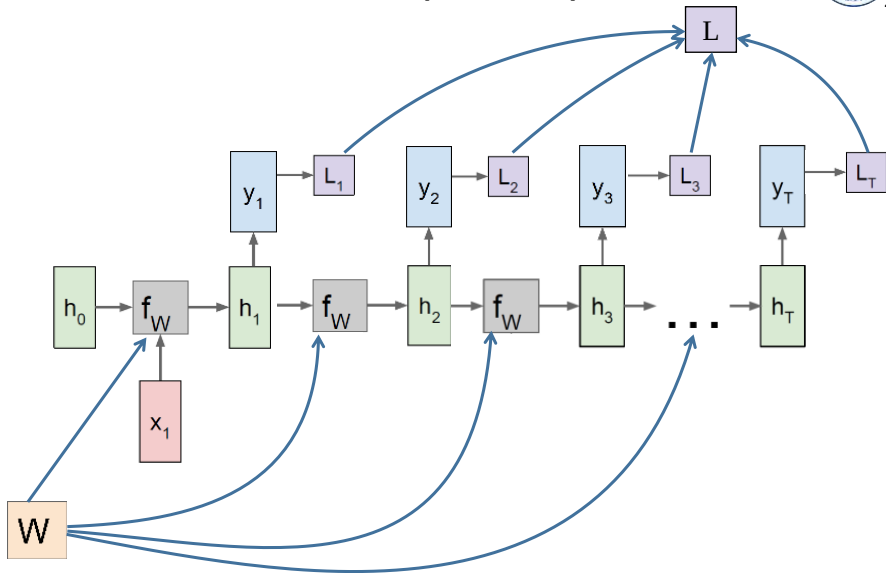
- $x = (x_1, \dots, x_T)$ 为长度为 T 的输入序列
- $y = (y_1, \dots, y_T)$ 是长度为 T 的标签序列



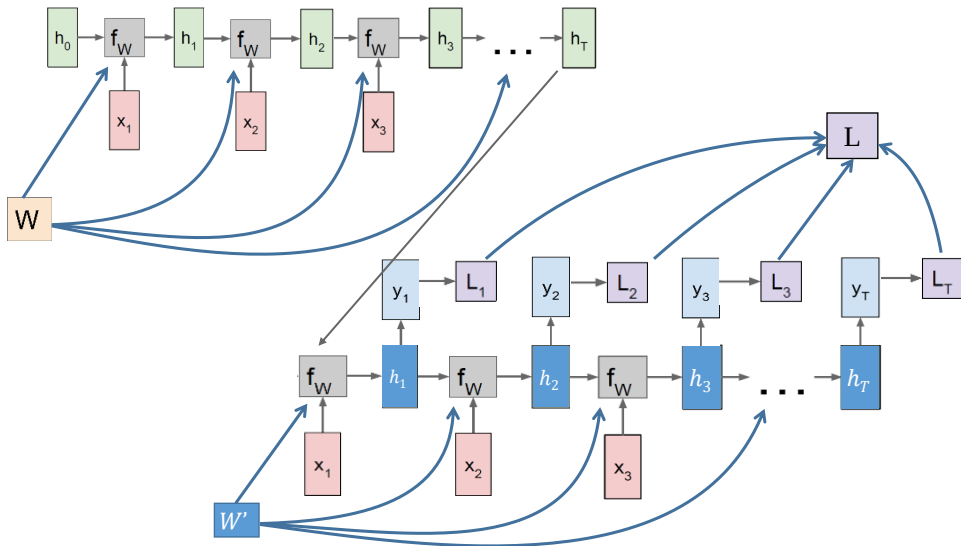
参数学习—计算图（多对一）



参数学习—计算图（一对多）



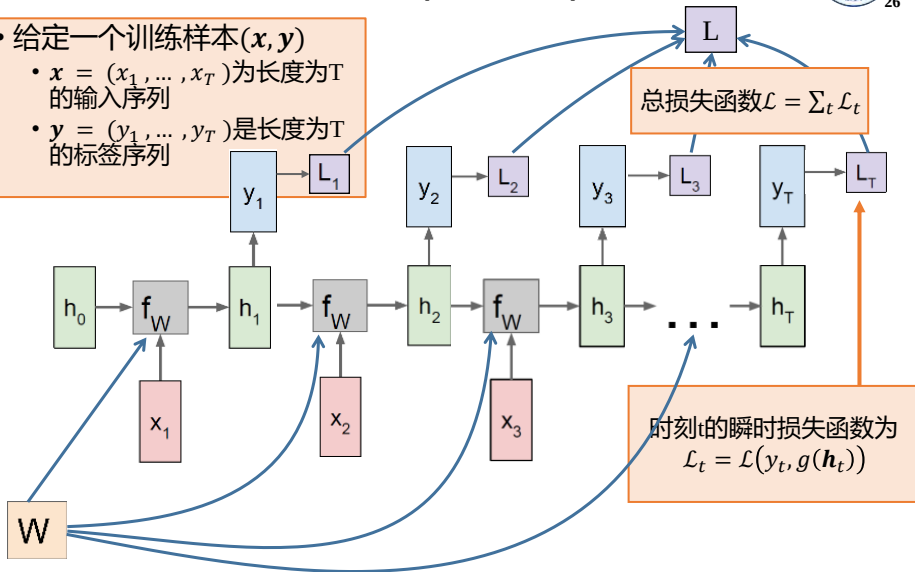
参数学习—计算图（多对一 + 一对多）



参数学习—计算图（多对多）

• 给定一个训练样本 (x, y)

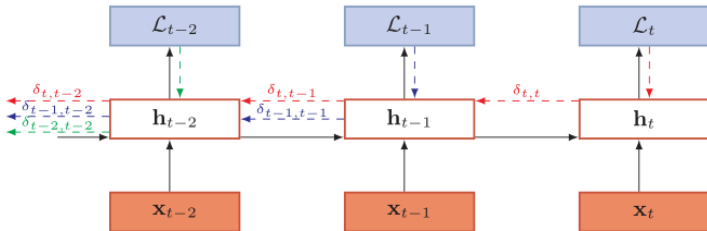
- $x = (x_1, \dots, x_T)$ 为长度为 T 的输入序列
- $y = (y_1, \dots, y_T)$ 是长度为 T 的标签序列



梯度计算

- 随时间反向传播算法 (BPTT)

$$\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_{t+1} + \mathbf{b})$$



$\delta_{t,k}$ 为第 t 时刻的损失
对第 k 步隐藏神经元的
净输入 $\mathbf{z}_k$ 的导数

$$\begin{aligned} \delta_{t,k} &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \frac{\partial \mathbf{h}_k}{\partial \mathbf{z}_k} \frac{\partial \mathbf{z}_{k+1}}{\partial \mathbf{h}_k} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_{k+1}} \\ &= \text{diag}(f'(\mathbf{z}_k)) \mathbf{U}^\top \boldsymbol{\delta}_{t,k+1} \end{aligned}$$

梯度计算

- 随时间反向传播算法 (BPTT) $\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_{t+1} + \mathbf{b})$

$$\boldsymbol{\delta}_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \text{diag}(f'(\mathbf{z}_k)) \mathbf{U}^\top \boldsymbol{\delta}_{t,k+1}$$

$$\begin{aligned} \frac{\partial \mathcal{L}_t}{\partial u_{ij}} &= \sum_k \frac{\partial \mathbf{z}_k}{\partial u_{ij}} \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \sum_{k=1}^t \sum_l \frac{\partial z_{k,l}}{\partial u_{ij}} \frac{\partial \mathcal{L}_t}{\partial z_{k,l}} \\ &= \sum_{k=1}^t \sum_l \mathbb{I}(l=i) [\mathbf{h}_{k-1}]_j [\boldsymbol{\delta}_{t,k}]_l \quad \Rightarrow \quad \frac{\partial \mathcal{L}_t}{\partial \mathbf{U}} = \sum_{k=1}^t \boldsymbol{\delta}_{t,k} \mathbf{h}_{k-1}^\top \\ &= \sum_{k=1}^t [\mathbf{h}_{k-1}]_j [\boldsymbol{\delta}_{t,k}]_i \end{aligned}$$

$\mathbf{z}_k = \mathbf{U}\mathbf{h}_{k-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}$

梯度计算

- 随时间反向传播算法 (BPTT) $\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_{t+1} + \mathbf{b})$

$$\boldsymbol{\delta}_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \text{diag}(f'(\mathbf{z}_k)) \mathbf{U}^\top \boldsymbol{\delta}_{t,k+1}$$

$$\mathbf{z}_k = \mathbf{U}\mathbf{h}_{k-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}$$

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{W}} = \sum_{k=1}^t \boldsymbol{\delta}_{t,k} \mathbf{x}_k^\top$$

同理可得

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{b}} = \sum_{k=1}^t \boldsymbol{\delta}_{t,k}$$

梯度计算

- 随时间反向传播算法 (BPTT) $\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_{t+1} + \mathbf{b})$

$$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \text{diag}(f'(\mathbf{z}_k)) \mathbf{U}^\top \delta_{t,k+1} = \prod_{\tau=k}^{t-1} (\text{diag}(f'(\mathbf{z}_\tau)) \mathbf{U}^\top) \delta_{t,t}$$

为了便于分析, 假设 $f'(\mathbf{z}_\tau) = \gamma$

$$= \gamma^{t-k} \mathbf{U}^{t-k\top} \delta_{t,t}$$

计算 $\mathbf{z}_k$ 梯度涉及 $\mathbf{U}$ 幂乘
还包括 tanh 的重复使用

最大特征值大于1, $t - k$ 比较大时, **梯度爆炸**, 造成系统不稳定;
最大特征值小于1, $t - k$ 比较小时, 梯度变得非常小, 出现**梯度消失**

由于梯度爆炸或消失问题, 实际上只能学习到短周期的依赖关系。这就是所谓的**长程依赖问题**。

梯度计算

- 随时间反向传播算法 (BPTT) $\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_{t+1} + \mathbf{b})$

$$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \text{diag}(f'(\mathbf{z}_k)) \mathbf{U}^\top \delta_{t,k+1} = \prod_{\tau=k}^{t-1} (\text{diag}(f'(\mathbf{z}_\tau)) \mathbf{U}^\top) \delta_{t,t}$$

$$\text{如果 } f'(\mathbf{z}_\tau) = 1 \quad = \mathbf{U}^{t-k} \delta_{t,t}$$

谱半径 $\rho(\mathbf{U})$ 大于 1, $t - k$ 比较大时, **梯度爆炸**, 造成系统不稳定;
谱半径 $\rho(\mathbf{U})$ 小于 1, $t - k$ 比较大时, 梯度变得非常小, 出现**梯度消失**

Theorem. Let $\mathbf{A} \in \mathbb{C}^{n \times n}$ with spectral radius $\rho(\mathbf{A})$.
Then $\rho(\mathbf{A}) < 1$ if and only if $\lim_{k \rightarrow \infty} \mathbf{A}^k = 0$

梯度计算

- 随时间反向传播算法 (BPTT) $\mathbf{h}_{t+1} = f(\mathbf{z}_{t+1}) = f(\mathbf{U}\mathbf{h}_t + \mathbf{W}\mathbf{x}_{t+1} + \mathbf{b})$

$$\delta_{t,k} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{z}_k} = \text{diag}(f'(\mathbf{z}_k)) \mathbf{U}^\top \delta_{t,k+1} = \prod_{\tau=k}^{t-1} (\text{diag}(f'(\mathbf{z}_\tau)) \mathbf{U}^\top) \delta_{t,t}$$

如果 $|f'(\mathbf{z}_\tau)| \leq \gamma$,

$\mathbf{U}$ 的最大奇异值 $\lambda_1 < \frac{1}{\gamma}$



$$\|\text{diag}(f'(\mathbf{z}_\tau)) \mathbf{U}^\top\| \leq \gamma \|\mathbf{U}\| < 1$$

$$|\tanh(x)| \leq 1$$

$$|\sigma(x)| \leq \frac{1}{4}$$

一定存在 $\eta < 1$, 使得 $\|\text{diag}(f'(\mathbf{z}_\tau)) \mathbf{U}^\top\| \leq \eta$

$$\Rightarrow \|\delta_{t,k}\| \leq \left\| \prod_{\tau=k}^{t-1} (\text{diag}(f'(\mathbf{z}_\tau)) \mathbf{U}^\top) \delta_{t,t} \right\| \leq \eta^{t-k} \|\delta_{t,t}\|$$

$$t - k \rightarrow \infty$$



$$\|\delta_{t,k}\| \rightarrow 0$$

长程依赖问题

- 由于梯度爆炸或消失问题，实际上只能学习到短周期的依赖关系。这就是所谓的长程依赖问题

- 循环神经网络中的梯度消失不是 $\frac{\partial \mathcal{L}_t}{\partial \mathbf{u}}$ 的梯度消失，而是 $\frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_k}$ 的梯度消失

$$\frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_k} = \mathbf{U}^\top \text{diag}(f'(\mathbf{z}_{k+1})) \frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_{k+1}}$$

- $\mathbf{U}$ 的更新主要靠当前时刻 t 的几个相邻状态 $\mathbf{h}_k$ 来更新，长距离的状态对参数几乎没有影响

长程依赖问题

- 循环神经网络在时间维度上非常深!
 - 梯度消失或梯度爆炸
- 如何改进?

梯度爆炸问题

- 权重衰减
- 梯度截断

梯度消失问题

- 改进模型, 产生新的RNN结构

Gradient clipping: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```

长程依赖问题—改进方法

- 可以通过调参，尽量使得 $U^T \text{diag}(f'(z_{k+1})) \approx 1$ ，但是需要经验
- 循环边改为线性依赖关系

$$h_t = f(Uh_{t-1} + Wx_t + b) \quad \rightarrow \quad h_t = h_{t-1} + g(Wx_t + b) \quad \frac{\partial \mathcal{L}_t}{\partial h_t} = \frac{\partial \mathcal{L}_t}{\partial h_{t-1}}$$

- 但上述模型只刻画线性依赖关系，降低了模型的表示能力
- 解决办法：增加非线性

$$h_t = h_{t-1} + g(Uh_{t-1} + Wx_t + b)$$

h_t 和 h_{t-1} 既有线性关系也有非线性关系

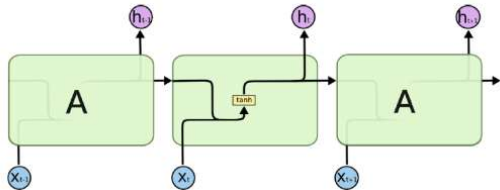
缺点

- 梯度爆炸仍存在
- 记忆容量问题，导致h饱和

长程依赖问题—门控机制

- 长短期记忆神经网络 (Long Short-Term Memory, LSTM)

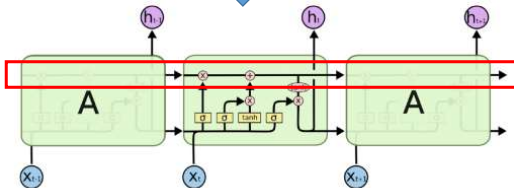
Vanilla RNN



$$h_t = f\left(w\left(\begin{matrix} h_{t-1} \\ x_t \end{matrix}\right)\right)$$



LSTM

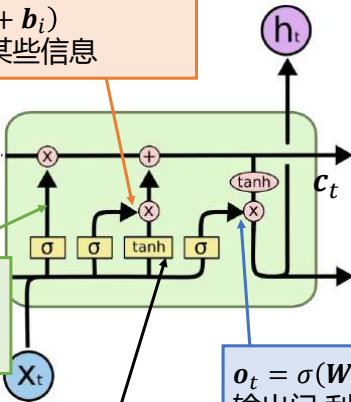


引入一个新的内部状态 c_t , 隐状态 h_t 由该内部状态导出

长短期记忆神经网络 (LSTM)

$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
输入门:记忆现在的某些信息

$$h_t = o_t \odot \tanh(c_t)$$



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
遗忘门:选择忘记过去某些信息

$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$:
输出门:利用多少存储信息输出

$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$
感知层, 确定关键和新的信息

长短期记忆神经网络 (LSTM)

- 可以更加简洁的描述为

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix} + \mathbf{b} \right)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

长短期记忆神经网络 (LSTM)

- 循环神经网络的隐状态 h 记录了历史信息，看作一种记忆
- 简单循环神经网络中， h 每个时刻都会被重写，看作是短期记忆
- 神经网络中的参数隐含了从数据中学到的经验，看作是长期记忆
- LSTM网络中，记忆单元 c 可以在某个时刻捕捉到关键信息，并有能力保存一定的时间间隔，因此生命周期要长期短期记忆，因此称为长短期记忆

LSTM的各种变体

- 没有遗忘门

$$\mathbf{c}_t = \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

- 耦合输入门和遗忘门

$$\mathbf{c}_t = (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

- peephole连接：三个门不但依赖于输入 $\mathbf{x}_t$ 和 $\mathbf{h}_{t-1}$ ，还依赖于 $\mathbf{c}_{t-1}$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_{t-1}] + \mathbf{b}_i)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_{t-1}] + \mathbf{b}_f)$$

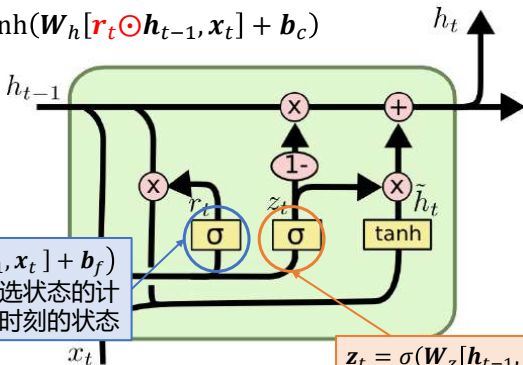
$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t] + \mathbf{b}_o)$$

门控循环单元 (GRU)

- 统一输出单元(h_t) 和记忆单元(c_t), 引入**更新门**控制当前状态需从历史状态中保留多少信息以及需要从候选状态中接收多少新信息

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

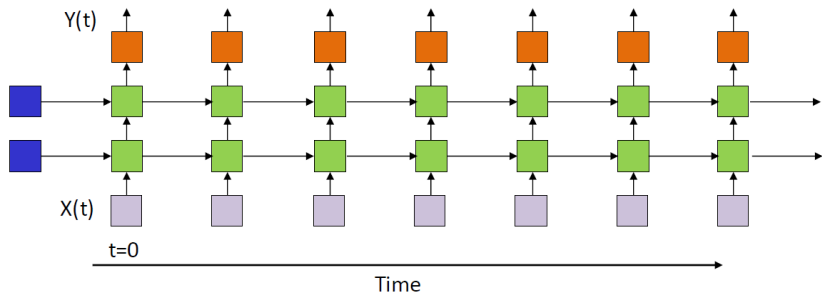
$$\tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_c)$$



$r_t = \sigma(W_r[h_{t-1}, x_t] + b_f)$
重置门：控制候选状态的计算是否依赖上一时刻的状态

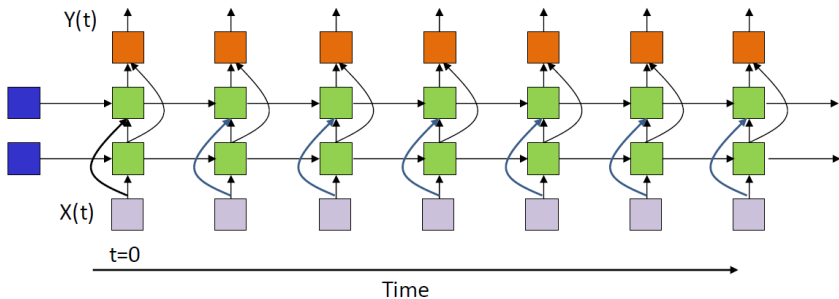
$z_t = \sigma(W_z[h_{t-1}, x_t] + b_i)$
更新门：

深层循环神经网络



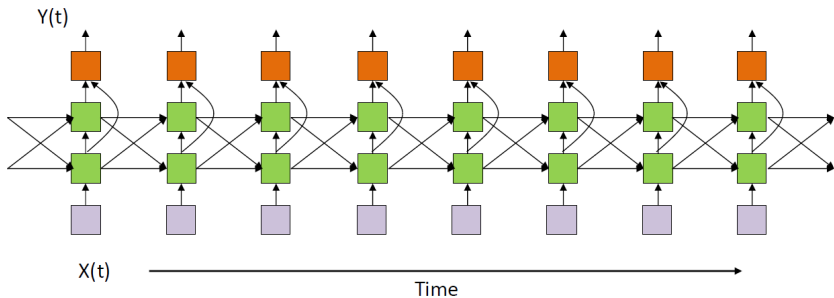
多隐藏层

深度学习



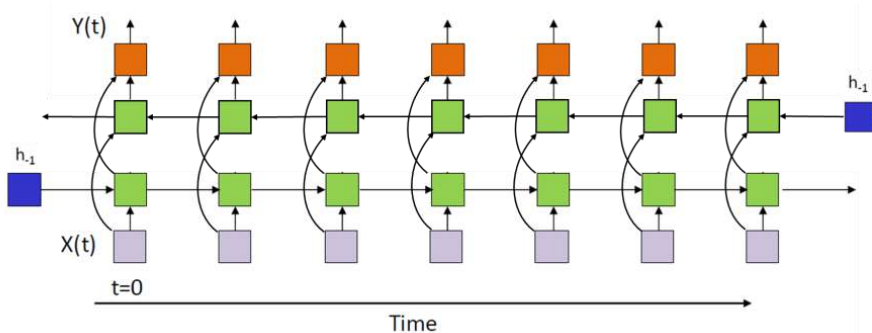
多隐藏层

深层循环神经网络



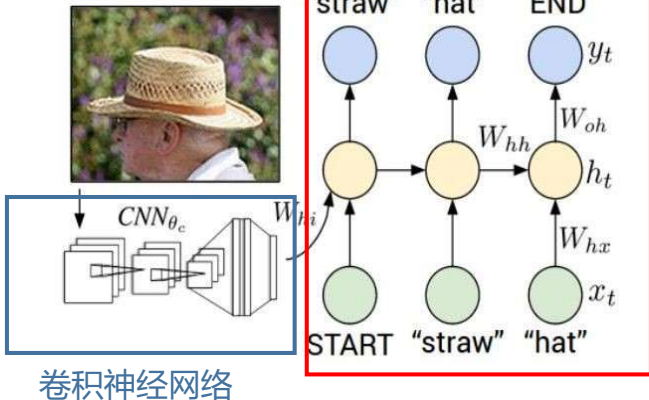
多隐藏层

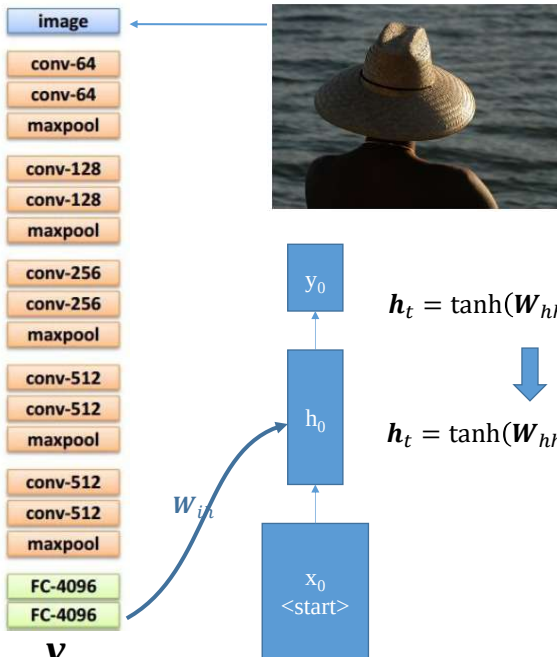
双向循环神经网络



循环神经网络的应用—图像描述

循环神经网络

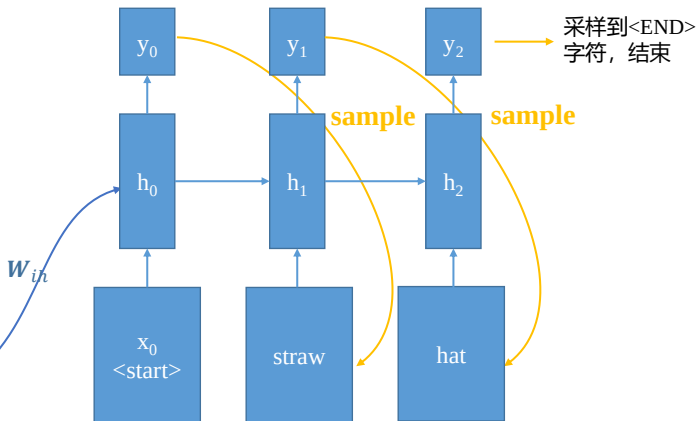




$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + W_{ih}v)$$



循环神经网络的应用—图像描述



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A dog is running in the grass with a frisbee



A white teddy bear sitting in the grass



Two people walking on the beach with surfboards



A tennis player in action on the court



Two giraffes standing in a grassy field



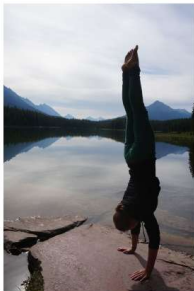
A man riding a dirt bike on a dirt track

循环神经网络的应用—图像描述

失败例子



A woman is holding a cat in her hand



A woman standing on a beach holding a surfboard



A bird is perched on a tree branch



A person holding a computer mouse on a desk



A man in a baseball uniform throwing a ball

循环神经网络的应用—语言模型

- 自然语言理解 → 一个句子的可能性/合理性

！ 在报那猫告做只
那只猫在作报告！
那个人在作报告！

$$P(x_1, x_2, \dots, x_T) = \prod_{i=1}^T P(x_i | x_1, \dots, x_{i-1})$$
$$\approx \prod_{i=1}^T P(x_i | x_{i-n+1}, \dots, x_{i-1})$$

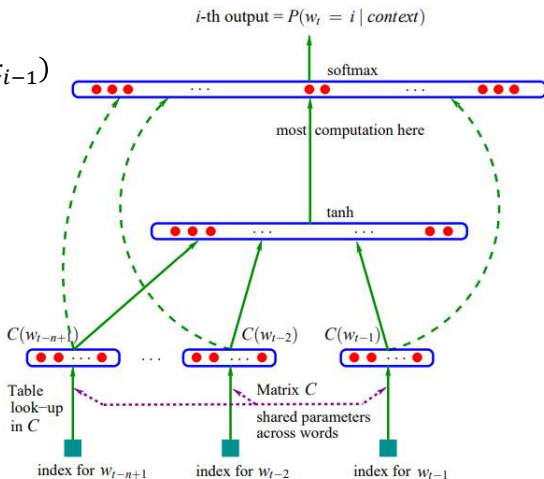
N元语言模型

循环神经网络的应用—语言模型

$$P(x_1, x_2, \dots, x_T)$$

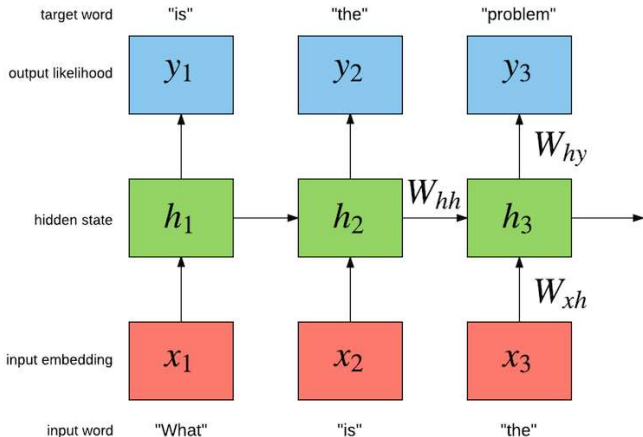
$$\approx \prod_{i=1}^T P(x_i | x_{i-n+1}, \dots, x_{i-1})$$

N元语言模型



循环神经网络的应用—语言模型

基于RNN的语言模型



循环神经网络的应用—生存Linux内核代码

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
    unsigned long flags;
    int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
    buf[0] = 0xFFFFFFFF & (bit << 4);
    min(inc, slist->bytes);
    printk(KERN_WARNING "Memory allocated %02x/%02x, "
        "original MLL instead\n"),
        min(min(multi_run - s->len, max) * num_data_in,
            frame_pos, sz + first_seg);
    div_u64_w(val, inb_p);
    spin_unlock(&disk->queue_lock);
    mutex_unlock(&s->sock->mutex);
    mutex_unlock(&func->mutex);
    return disassemble(info->pending_bh);
}

static void num_serial_settings(struct tty_struct *tty)
{
    if (tty == tty)
        disable_single_st_p(dev);
    pci_disable_spool(port);
}
```



作词

- RNN在“学习”过汪峰全部作品后自动生成的歌词
- <https://github.com/phunterlau/wangfeng-rnn>

我在这里中的夜里
就像一场是一种生命的意叶
就像我的生活变得在我一样
可我们这是一个知道
我只是一天你会怎吗
可我们这是我们是不要为你

我们想这有一种生活的时候

我在哭泣
我不能及你的时光
我是我们在这是一种无法少得可以没有一天
我们想这样
我们远在一场我在我一个相多地在此向
可我是个想已经把我的时候
我看到在心情一种痛定的时候

我看着我的感觉在飞歌
我不能在这多在我心中
就像在我一瞬间
我的生命中的感觉
我在我一次到孤独所
我们在这是一看可以是一场我们在这样
我在这里失命
我不能在这感觉在天里
夜里
夜里
没有一天 我想心的是不吗到了

循环神经网络的应用——作诗

白鹭窥鱼立，

Egrets stood, peeping fishes.

青山照水开。

Water was still, reflecting mountains.

夜来风不动，

The wind went down by nightfall,

明月见楼台。

as the moon came up by the tower.

满怀风月一枝春，

Budding branches are full of romance.

未见梅花亦可人。

Plum blossoms are invisible but adorable.

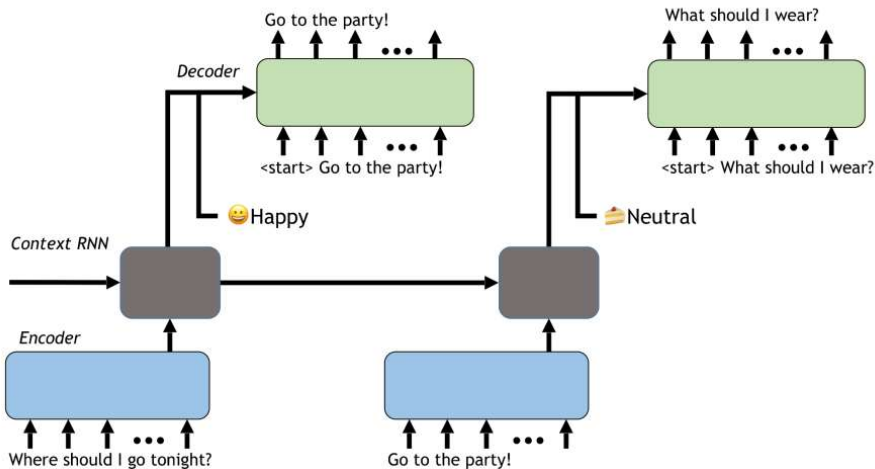
不为东风无此客，

With the east wind comes Spring.

世间何处是前身。

Where on earth do I come from?

循环神经网络的应用—对话系统



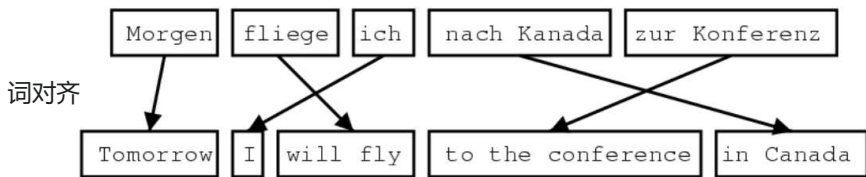
循环神经网络的应用—机器翻译

- 假设给定源语言句子: f 、目标语言句子: e
- 统计机器翻译模型

$$\hat{e} = \operatorname{argmax}_e P(e|f)$$

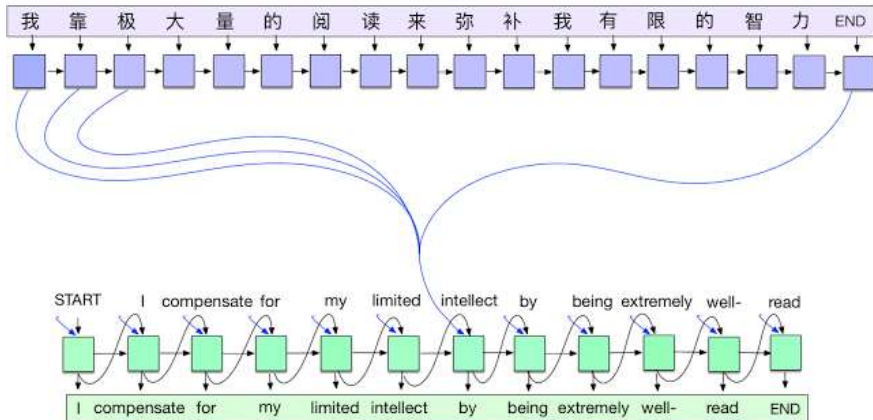
$$= \operatorname{argmax}_e P(f|e)P(e)$$

保证翻译的意义 ← 翻译模型 语言模型 → 保证翻译的流畅



循环神经网络的应用—机器翻译

用于编码的RNN ENCODER



DECODER 用于解码的RNN

非序列数据的序列化处理—图片分类

60

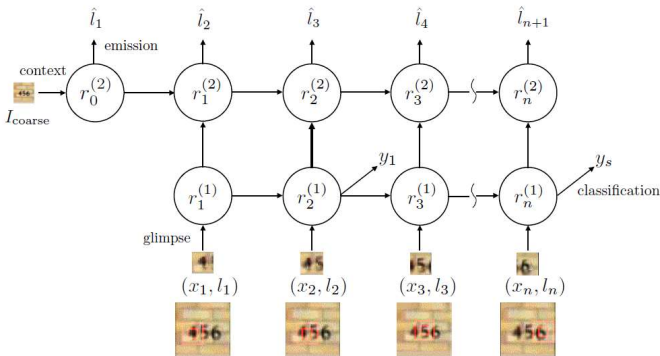
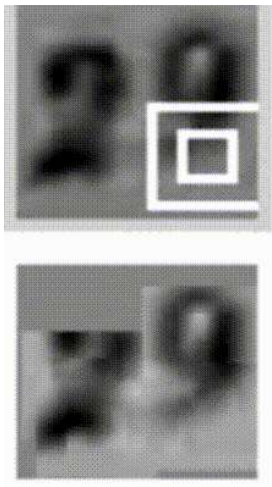
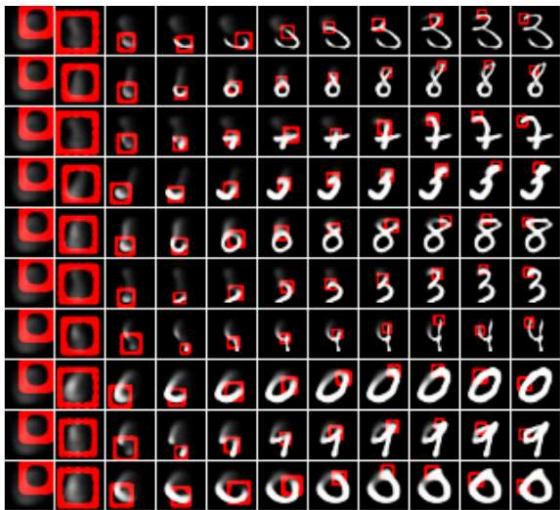


Figure 1: The deep recurrent attention model.

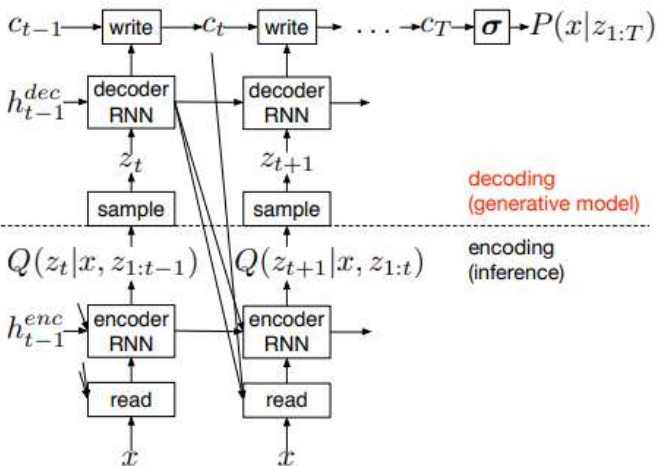
Classify images by taking a series of “glimpses”

非序列数据的序列化处理—图片生成



Time →

非序列数据的序列化处理—图片生成



非序列数据的序列化处理

Reading MNIST



注意力机制与外部记忆

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>

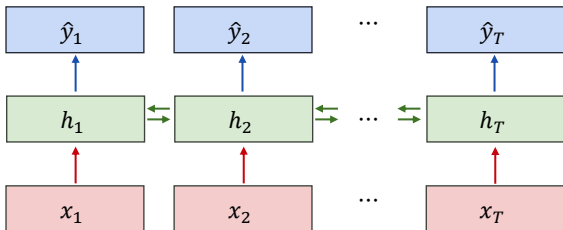


本章内容

- 注意力机制
 - 注意力及其应用
 - 自注意力模型
 - 指针网络
- 基于自注意力的深度序列模型
 - Transformer
 - BERT/GPT
- 记忆增强神经网络
 - 端到端记忆网络
 - 神经图灵机

循环神经网络的问题

- 循环神经网络难以并行





循环神经网络的问题

- 简单循环神经网络存在长程依赖问题，LSTM引入近似线性依赖的记忆单元来存储远距离信息，但都应对**阅读理解**等长序列任务

Facebook bAbi tasks

- 1 John moved to the bedroom.
- 2 Mary grabbed the football there.
- 3 Sandra journeyed to the bedroom.
- 4 Sandra went back to the hallway.
- 5 Mary moved to the garden.
- 6 Mary journeyed to the office.
- 7 Where is the football?

Answer: office

Stanford SQuAD

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well as the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge **through contact with Persian traders** since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

循环神经网络的问题

- 由于优化算法和计算能力的限制，不希望在过度增加模型复杂度的情况下来提升模型表达能力，循环神经网络在实践中很难达到通用近似的能力

如何提高
网络能力

- 局部连接
- 权重共享
- 汇聚操作

外部记忆

注意力



大脑中的注意力

- 人脑每个时刻接收的外界输入信息非常多，包括来源于视觉、听觉、触觉的各种各样的信息。
- 但就视觉来说，眼睛每秒钟都会发送千万比特的信息给视觉神经系统。
- 人脑通过**注意力**来解决**信息超载问题**。

注意力示例





注意力示例—鸡尾酒会效应

- 当一个人在吵闹的鸡尾酒会上和朋友聊天时，尽管周围噪音干扰很多，他还是可以听到朋友的谈话内容，而忽略其他人的声音。

——聚焦式注意力

- 同时，如果未注意到的背景声中有重要的词（比如他的名字），他会马上注意到。

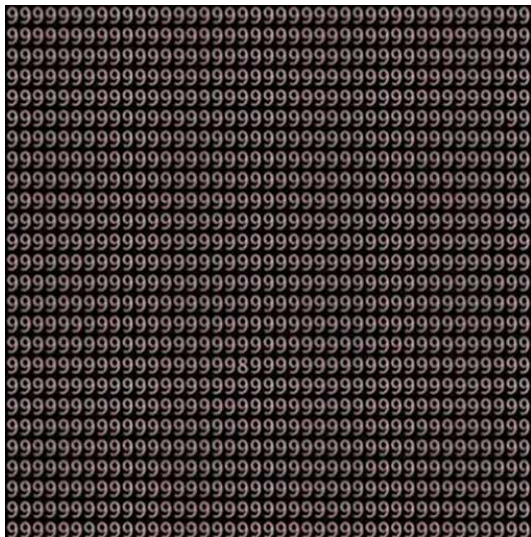
——显著性注意力

注意力实验

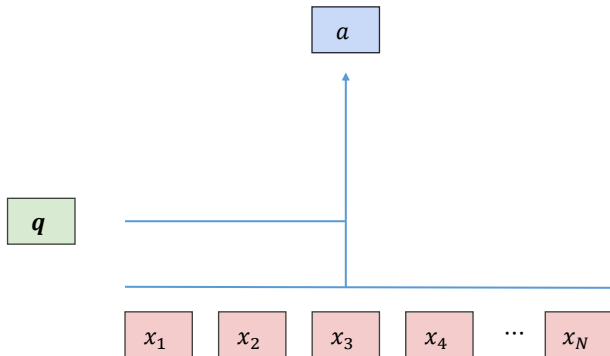


所有黑色牌的数字之和为多少？

注意力实验—找数字8



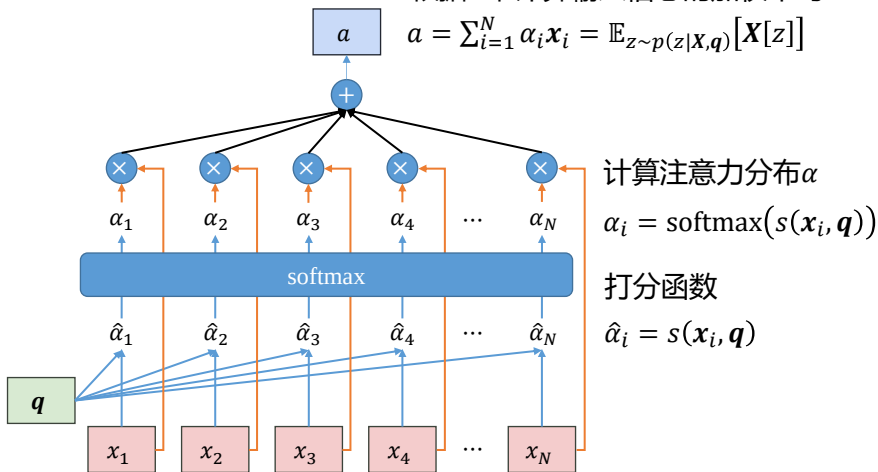
注意力背后的计算问题



注意力模型—软性注意力机制

根据 α 来计算输入信息的加权平均

$$a = \sum_{i=1}^N \alpha_i x_i = \mathbb{E}_{z \sim p(z|X, q)}[X[z]]$$



打分函数 $s(\mathbf{x}_i, \mathbf{q})$

加性模型

$$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{v}^T \tanh(W\mathbf{x}_i + U\mathbf{q}),$$

点积模型

$$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{x}_i^T \mathbf{q},$$

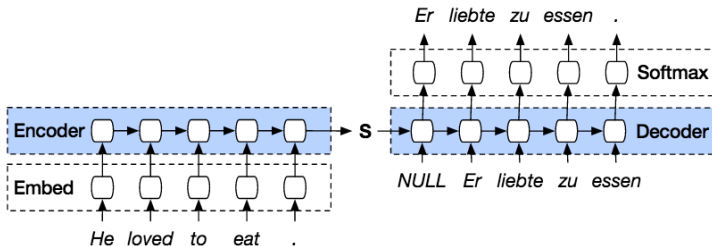
缩放点积模型

$$s(\mathbf{x}_i, \mathbf{q}) = \frac{\mathbf{x}_i^T \mathbf{q}}{\sqrt{d}},$$

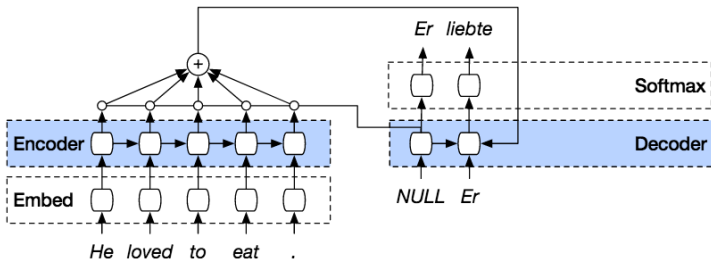
双线性模型

$$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{x}_i^T W \mathbf{q},$$

注意力机制的应用—机器翻译



注意力机制

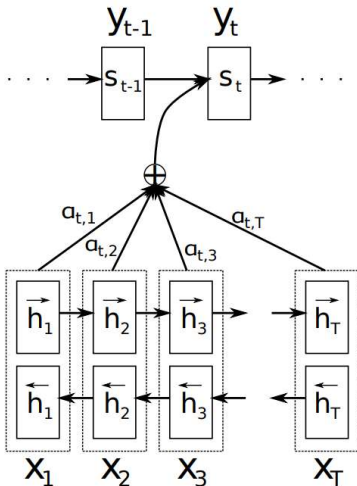


注意力机制的应用—机器翻译

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$c_i = \sum_j^T \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(\alpha(s_{i-1}, h_j))}{\sum_{k=1}^T \exp(\alpha(s_{i-1}, h_k))}$$

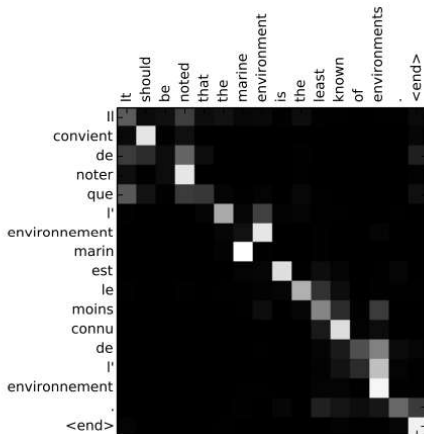
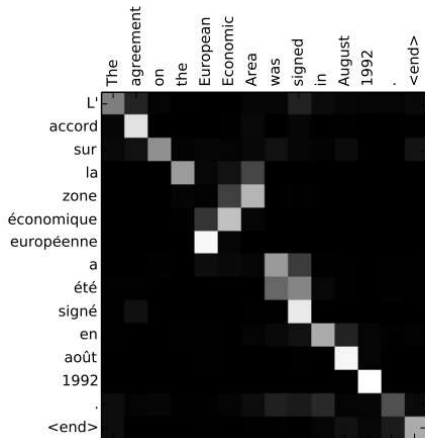


Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

注意力机制的应用—机器翻译

la maison de Léa <end>

注意力机制的应用——机器翻译



注意力机制的应用 —文本分类

加性模型，但没有query向量

$$\alpha_{it} = \text{softmax}(\mathbf{u}_w^\top \tanh(\mathbf{W}_w \mathbf{h}_{it} + \mathbf{b}_w))$$

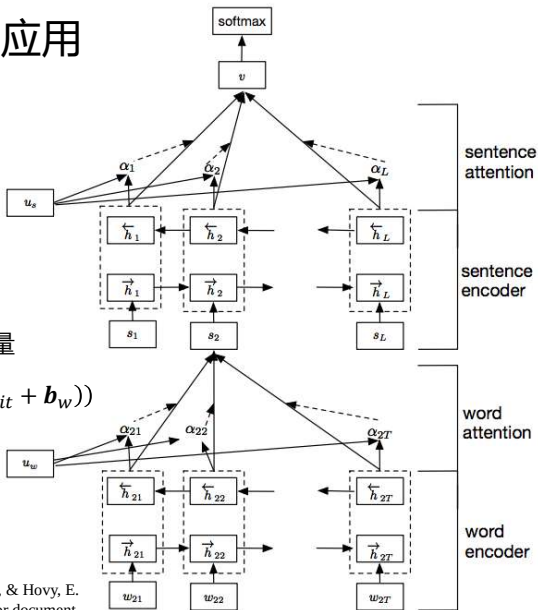


Figure 2: Hierarchical Attention Network.

注意力机制的应用—文本分类

GT: 4 Prediction: 4

pork belly = delicious .
 scallops ?
 i do n't .
 even .
 like .
 scallops , and these were a-m-a-z-i-n-g .
 fun and tasty cocktails .
 next time i 'm in phoenix , i will go
 back here .
 highly recommend .

GT: 0 Prediction: 0

terrible value .
 ordered pasta entree .
 .
 \$ 16.95 good taste but size was an
 appetizer size .
 .
 no salad , no bread no vegetable .
 this was .
 our and tasty cocktails .
 our second visit .
 i will not go back .

Yelp 2013评论数据集。Label 4表示打5分，Label 0 表示打1分

注意力机制的应用—文本分类

GT: 1 Prediction: 1

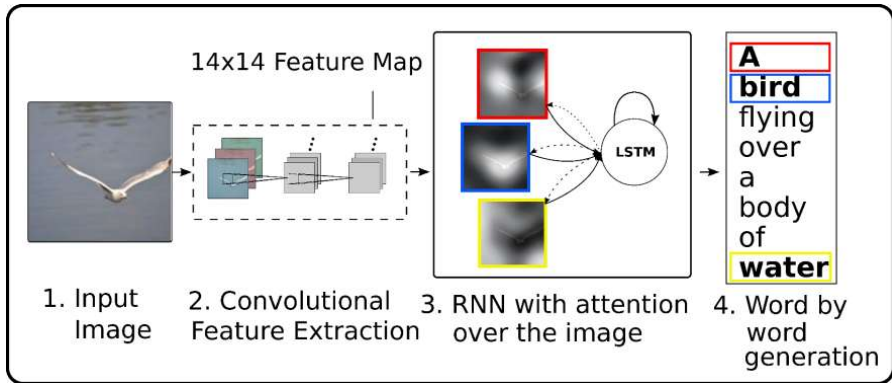
why does zebras have stripes ?
what is the purpose or those stripes ?
who do they serve the zebras in the
wild life ?
this provides camouflage - predator
vision is such that it is usually difficult
for them to see complex patterns

GT: 4 Prediction: 4

how do i get rid of all the old web
searches i have on my web browser ?
i want to clean up my web browser
go to tools > options .
then click “ delete history ” and “
clean up temporary internet files . ”

Yahoo Answers数据集。Label 1表示科学和数学，Label 4表示计算机和互联网

注意力机制的应用—图像描述



注意力机制的应用—图像描述

remote(0.23)



a(0.41)



sky(0.61)



on(0.27)



red(0.38)



woman(0.36)



of(0.19)



A(0.99)



racket(0.16)



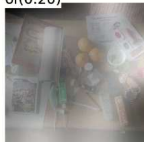
a(0.33)



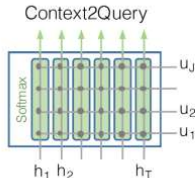
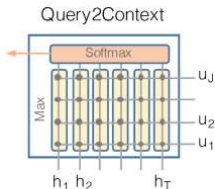
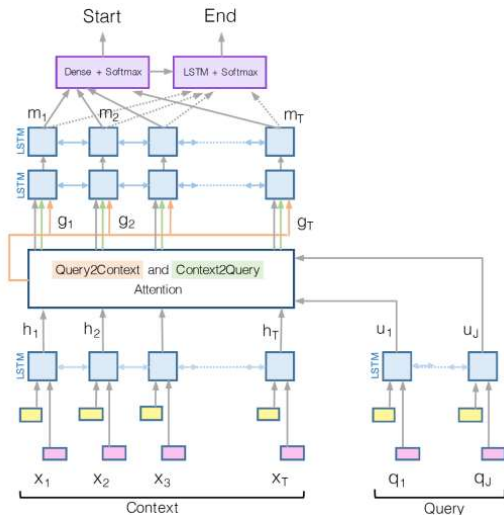
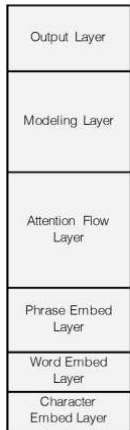
sink(0.48)



of(0.20)



注意力机制的应用—阅读理解



Word Embedding

Character Embedding

GLOVE

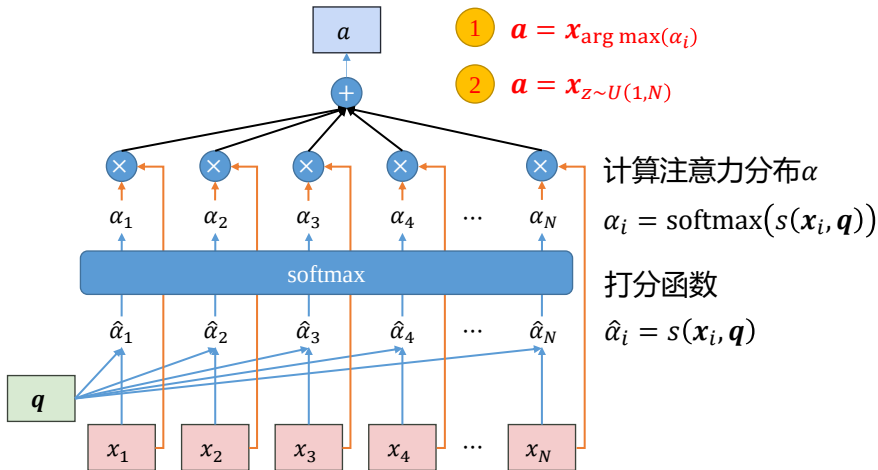
Char-CNN

注意力机制的变体—硬性注意力

根据 α 来计算输入的某个信息

1 $\mathbf{a} = \mathbf{x}_{\arg \max(\alpha_i)}$

2 $\mathbf{a} = \mathbf{x}_{z \sim U(1, N)}$

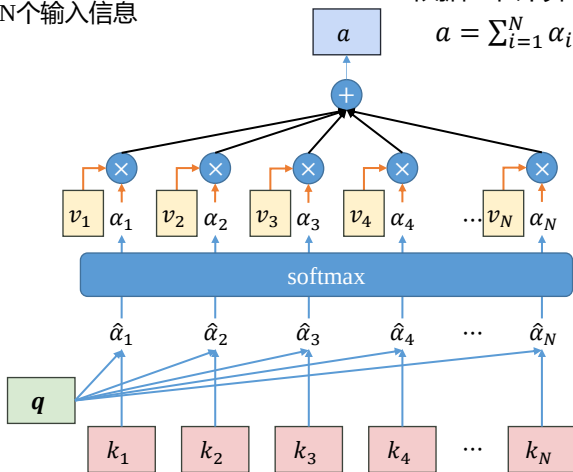


注意力机制的变体—键值对注意力

用 $(K, V) = [(k_1, v_1), \dots, (k_N, v_N)]$
表示 N 个输入信息

根据 α 来计算输入信息的加权平均

$$a = \sum_{i=1}^N \alpha_i v_i$$



计算注意力分布 α

$$\alpha_i = \text{softmax}(s(x_i, k_i))$$

打分函数

$$\hat{\alpha}_i = s(x_i, k_i)$$

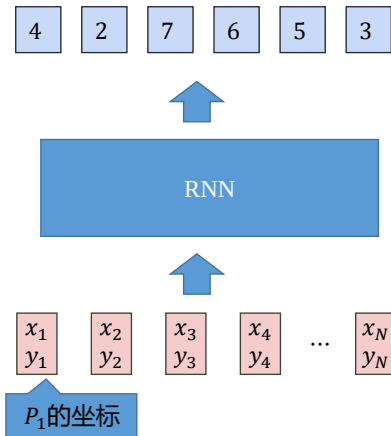
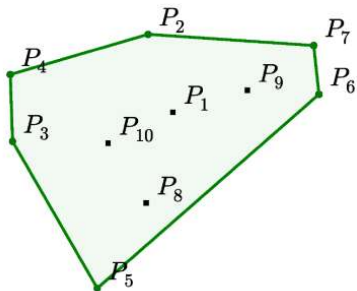
注意力机制的变体—多头注意力

- 利用多个查询 $Q = [q_1, \dots, q_M]$ ，来并行地从输入信息中选取多组信息，每个注意力关注输入信息的不同部分。

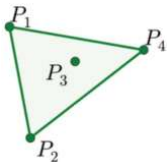
$$\text{att}((K, V), Q) = \text{att}((K, V), q_1) \oplus \dots \oplus \text{att}((K, V), q_1)$$

向量拼接

凸包问题

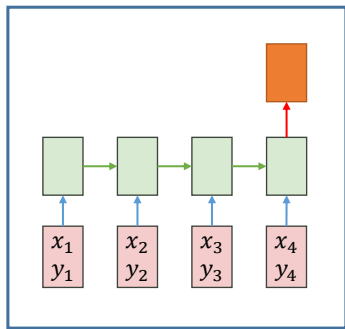


Seq2Seq?

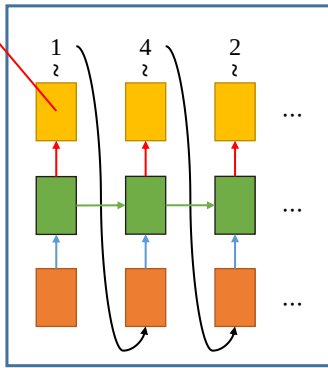


有问题吗?

{1,2,3,4,END}



Encoder



Decoder

指针网络

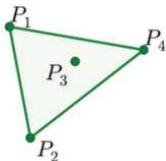
- 指针网络一种数列到序列模型：输入是长度为 N 的向量序列 $\mathbf{x} = \mathbf{x}_1, \dots, \mathbf{x}_N$ ，输出是长度为 M 的下标序列 $\mathbf{c}_{1:M}$ ，每个下标 $c_m \in [1, N]$
- 目标是根据条件概率 $P(\mathbf{c}_{1:M} | \mathbf{x}_{1:N})$ ，选出下标序列 $\mathbf{c}_{1:M}$ 使得该条件概率最大

$$\begin{aligned} P(\mathbf{c}_{1:M} | \mathbf{x}_{1:N}) &= \prod_{m=1}^M P(c_m | c_{1:m-1}, \mathbf{x}_{1:N}) \\ &\approx \prod_{m=1}^M P(c_m | \mathbf{x}_{c_1}, \dots, \mathbf{x}_{c_{m-1}}, \mathbf{x}_{1:N}) \end{aligned}$$

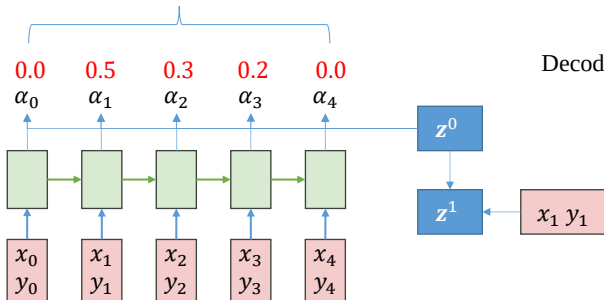
通过注意力分布来计算

指针网络

$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$: END



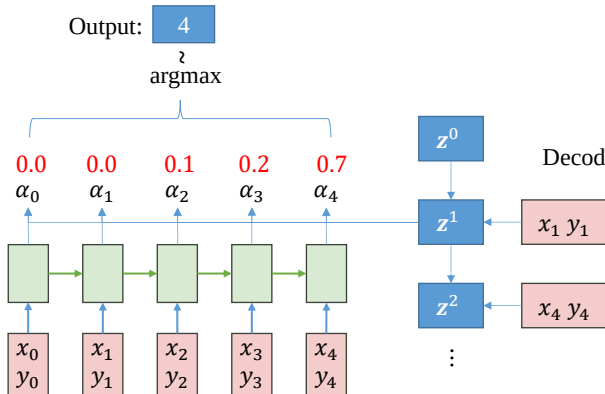
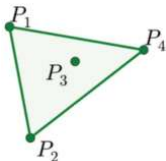
Output: $\begin{bmatrix} 1 \end{bmatrix}$
 $\hat{z}$
 argmax



Decoder的输出依赖于输入

Pointer Network

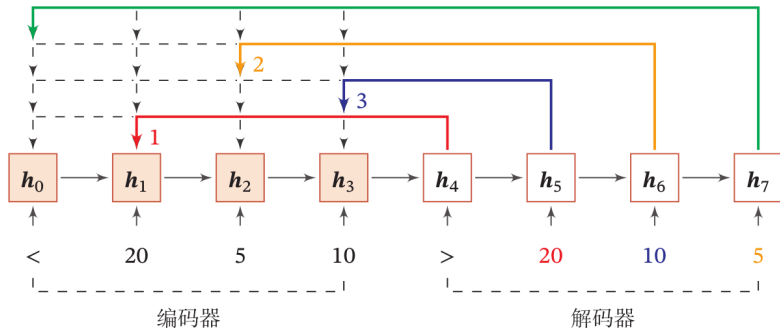
$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$: END



Decoder的输出依赖于输入

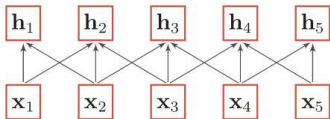
解码过程直到END具有最大attention权重

指针网络的更多应用—排序

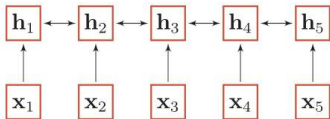


自注意力模型

- 当使用神经网络来处理一个变长的向量序列时，我们通常可以使用卷积网络或循环网络进行编码来得到一个相同长度的输出向量序列。



(a) 卷积网络

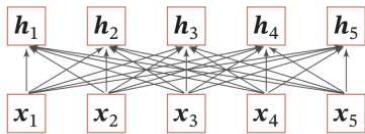


(b) 双向循环网络

只建模了输入信息的局部依赖关系

自注意力模型

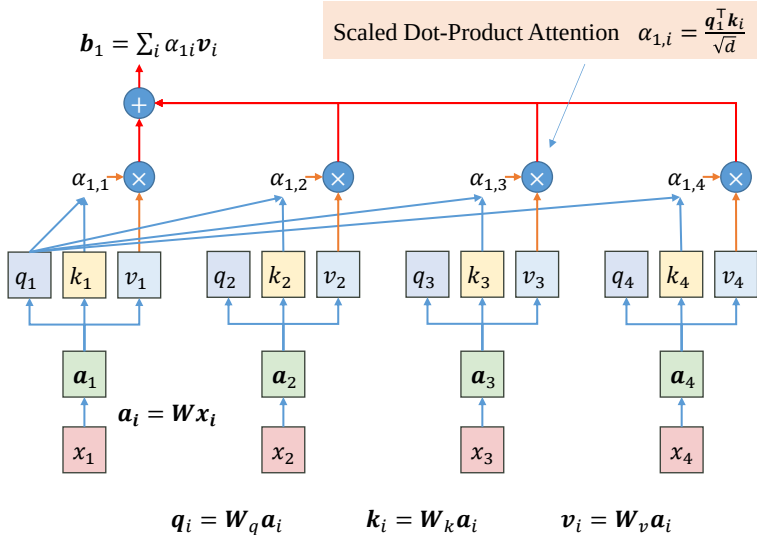
- 如何建立非局部 (Non-local) 的依赖关系
 - 全连接?



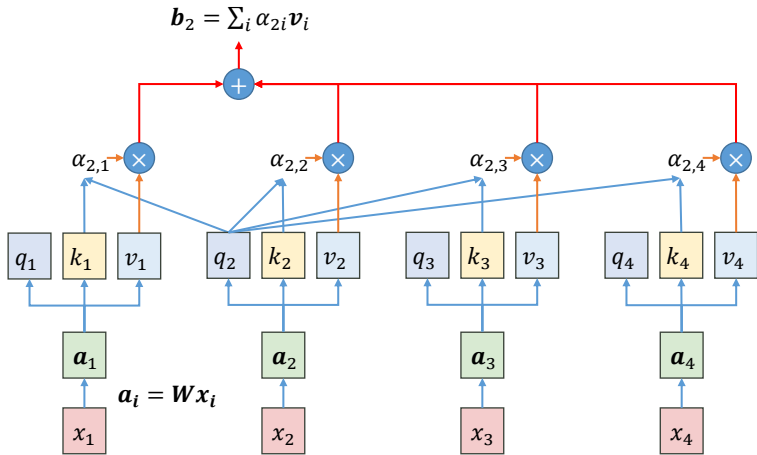
(a) 全连接模型

无法处理变长问题

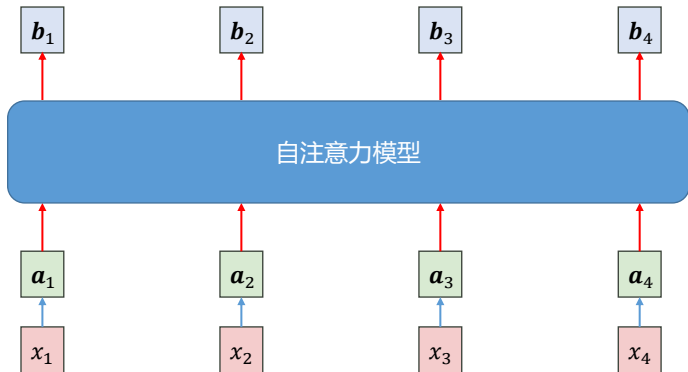
自注意力模型



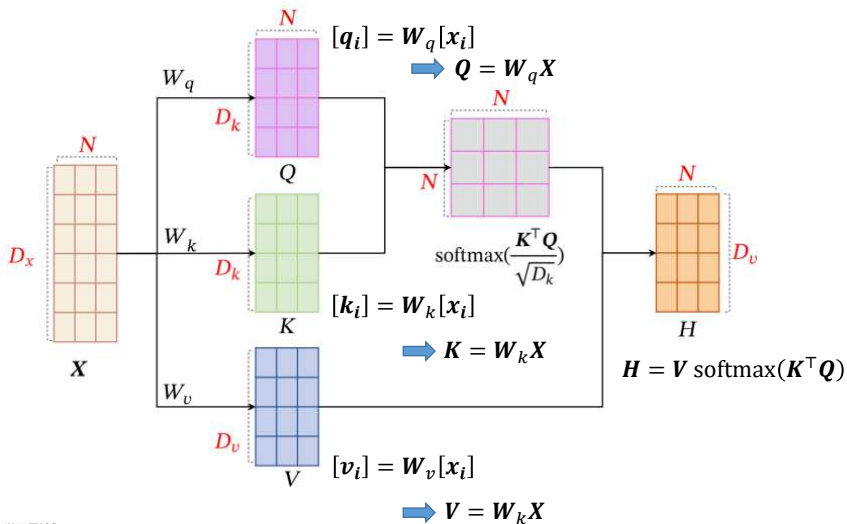
自注意力模型



自注意力模型



自注意力模型

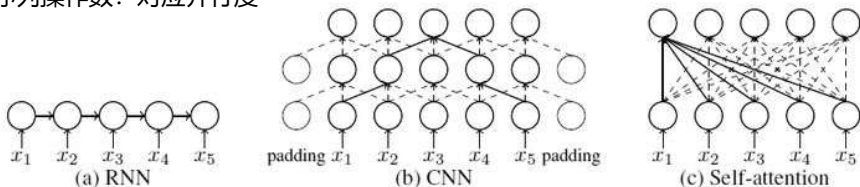


自注意力模型

k 卷积核大小 L 序列长度 d 维度

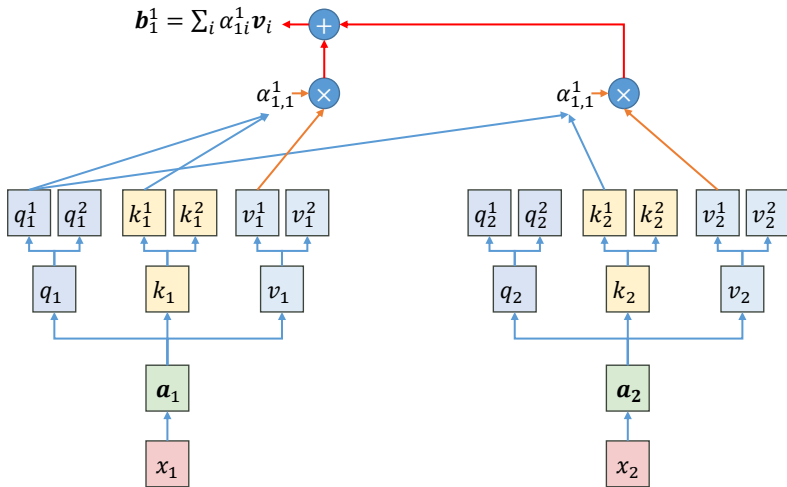
模型	每层复杂度	序列操作数	最大路径长度
CNN	$O(kLd^2)$	$O(1)$	$O(\log_k(L))$
RNN	$O(Ld^2)$	$O(L)$	$O(L)$
Self-attention	$O(L^2d)$	$O(1)$	$O(1)$

最大路径长度：两个结点传递信息所经历的最大路径长度
序列操作数：对应并行度

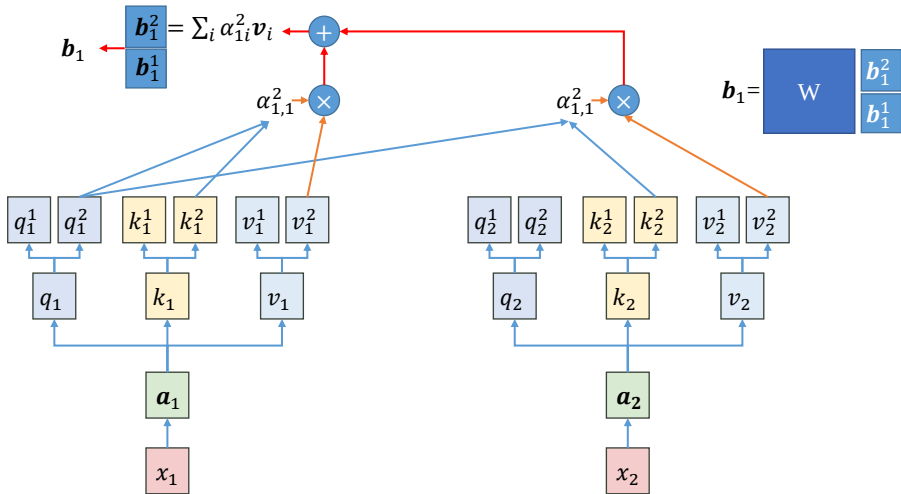


多头自注意力模型

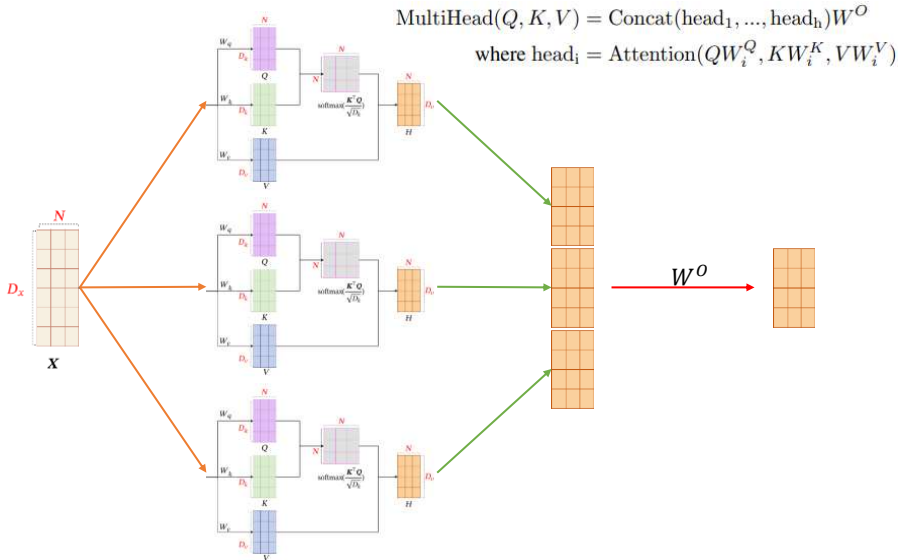
$$b_1^1 = \sum_i \alpha_{i1}^1 v_i$$



多头自注意力模型

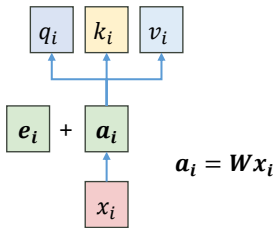
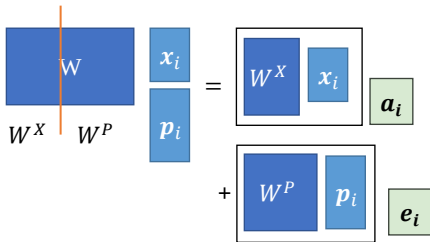


多头自注意力模型



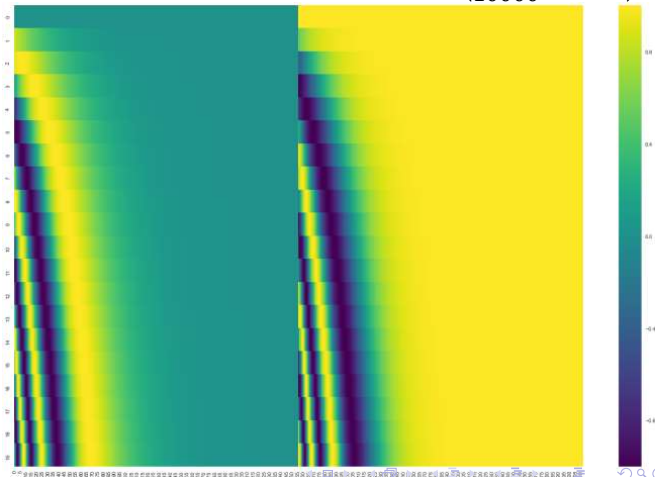
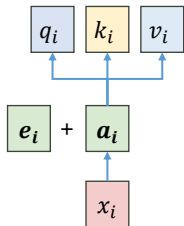
位置编码 (Positional Encoding)

- 在自注意力模型中并没有位置信息，对序列的任意置换不会影响输出结果
- 假设每个位置有个唯一的one-hot向量 p_i ，将 p_i 和 x_i 拼接作为新的输入



位置编码 (Positional Encoding)

- 使用正余弦函数编码位置，没有参数
- $$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$
- $$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$
- pos 位置, i 维度



位置编码 (Positional Encoding)

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$\sin(x+y) = \sin x \cos y + \cos x \sin y$$

$$\cos(x+y) = \cos x \cos y - \sin x \sin y$$

$$\alpha_k = \cos\left(\frac{k}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos+k,2i)} = \sin\left(\frac{pos+k}{10000^{2i/d_{model}}}\right) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)\alpha_k + \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)\beta_k$$

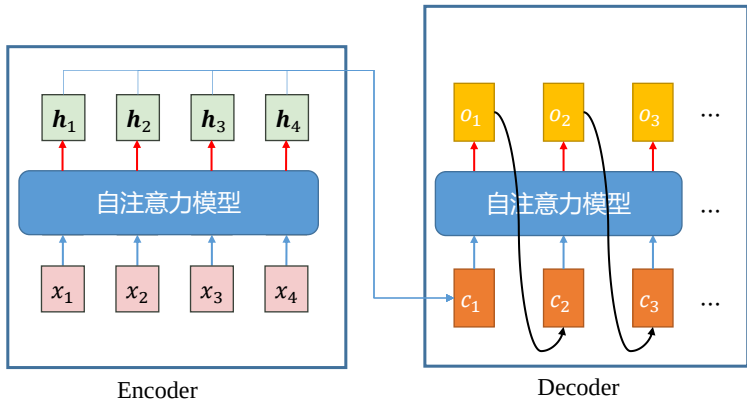
$$PE_{(pos+k,2i+1)} = \cos\left(\frac{pos+k}{10000^{2i/d_{model}}}\right) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)\alpha_k - \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)\beta_k$$

$$\begin{pmatrix} PE_{(pos+k,2i)} \\ PE_{(pos+k,2i+1)} \end{pmatrix} = \begin{pmatrix} \alpha_k & \beta_k \\ \alpha_k & -\beta_k \end{pmatrix} \begin{pmatrix} PE_{(pos,2i)} \\ PE_{(pos,2i+1)} \end{pmatrix}$$

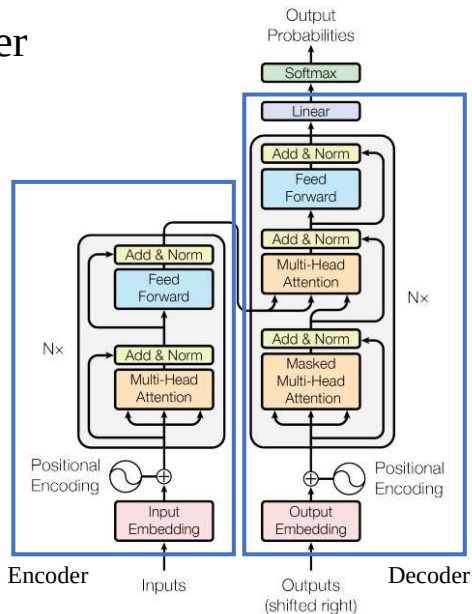
$$\beta_k = \sin\left(\frac{k}{10000^{2i/d_{model}}}\right)$$

每个固定的偏置 k , PE_{pos+k} 可以表示成 PE_{pos} 的线性函数

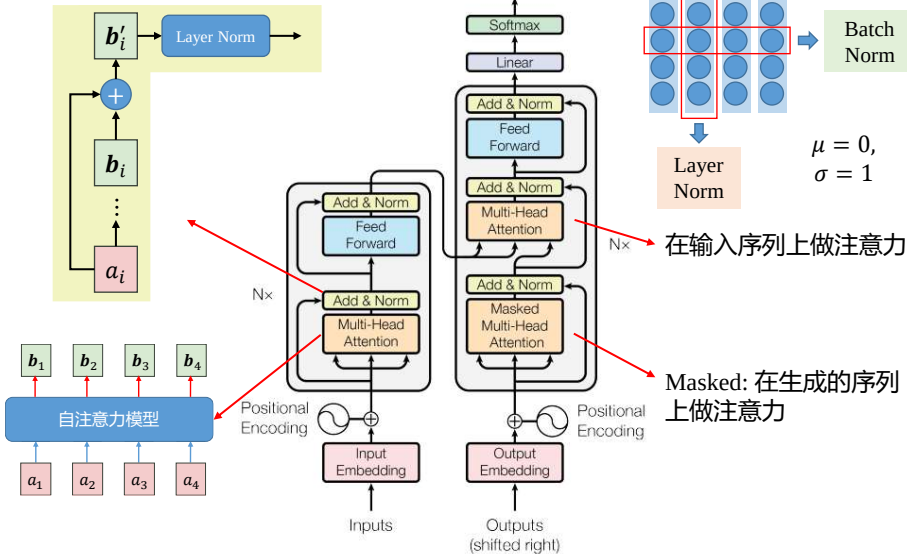
Seq2Seq with Self-attention



Transformer



Transformer

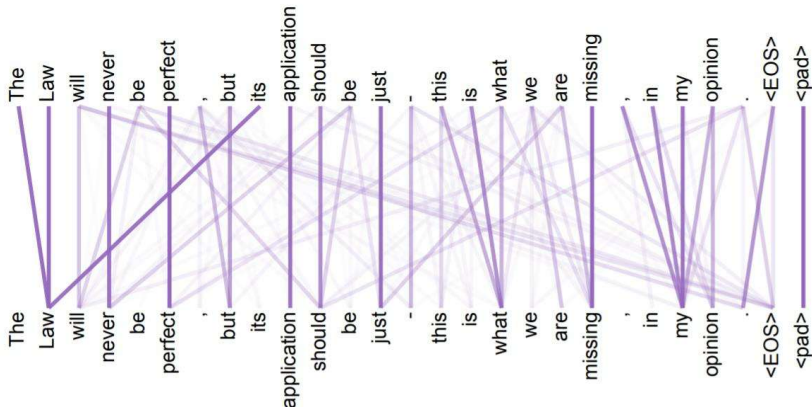


Transformer

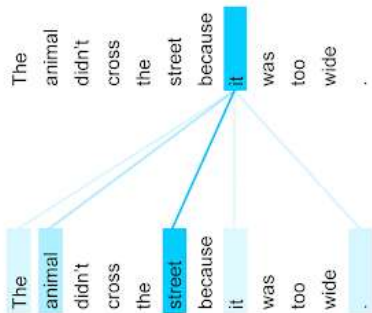
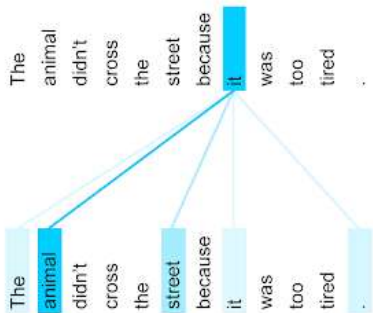


49

注意力可视化

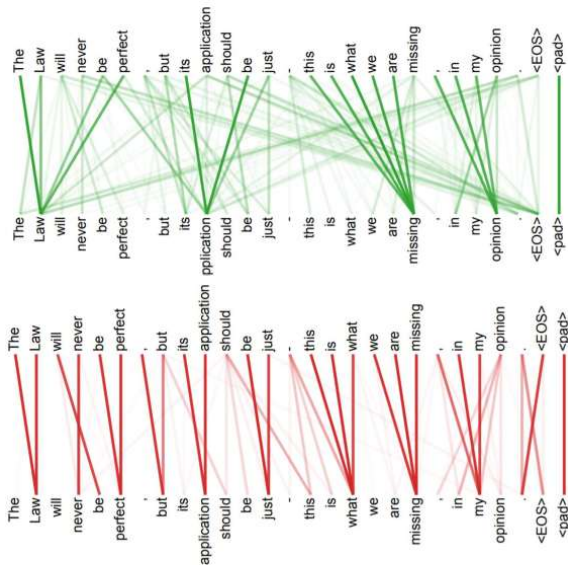


注意力可视化



The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads)

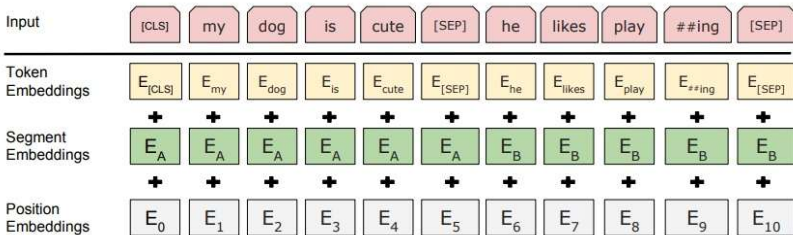
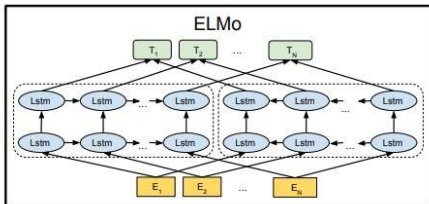
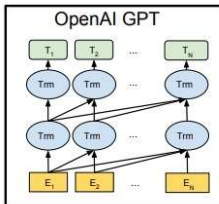
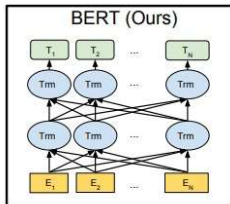
多头注意力可视化



预训练模型

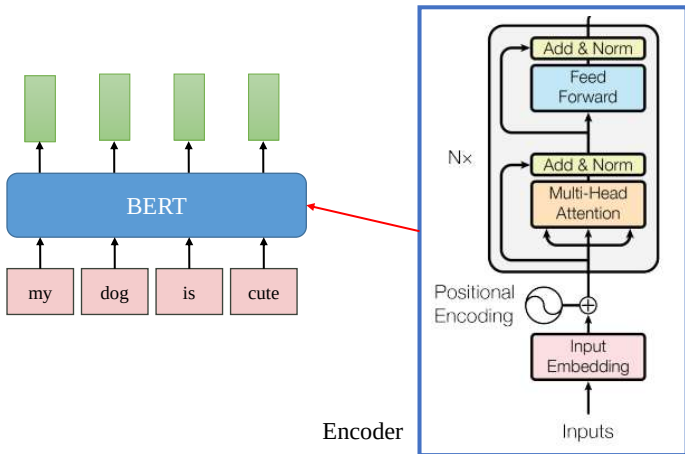
Transformer Encoder Transformer Decoder

BiLSTM



预训练模型—BERT

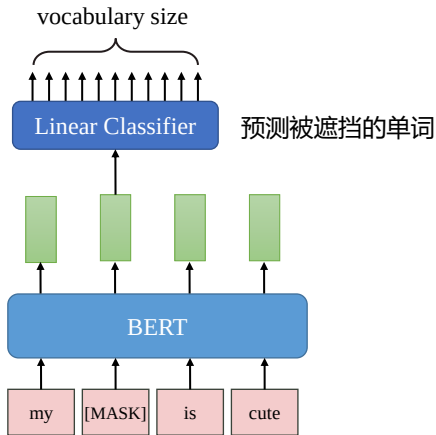
Bidirectional Encoder Representations from Transformers (BERT)



BERT的训练

• 任务1: Masked LM

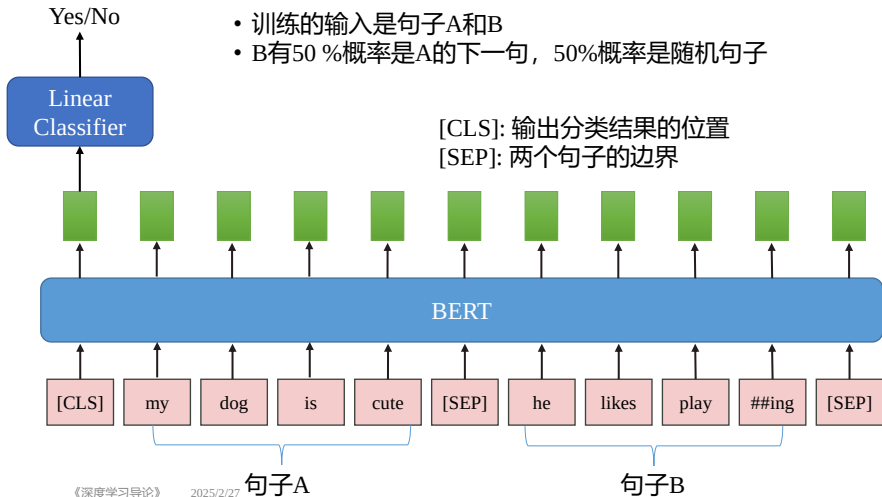
- 随机遮蔽15%的输入 token, 然后预测被遮住的 token
- 过多的[MASK] 引入会影响模型表现, 因为改变句子的语义
- 80%用[MASK], 10%用随机词, 10%保持词不动

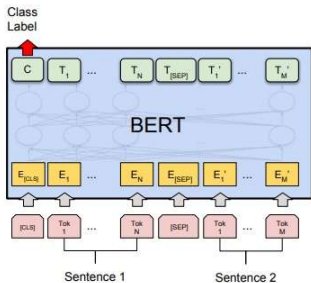


BERT的训练

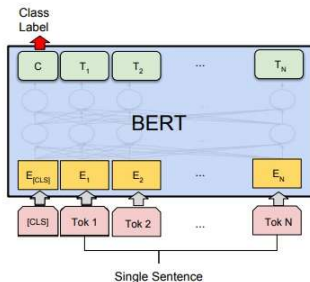
• 任务2: Next Sentence Prediction

- 训练的输入是句子A和B
- B有50 %概率是A的下一句，50%概率是随机句子

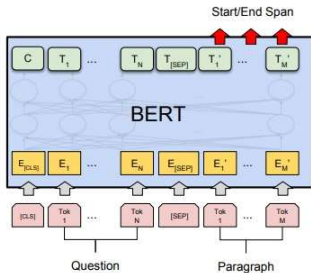




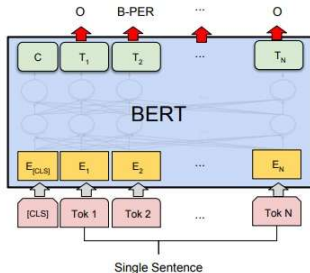
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



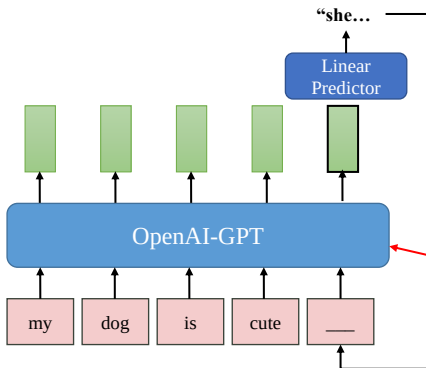
(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

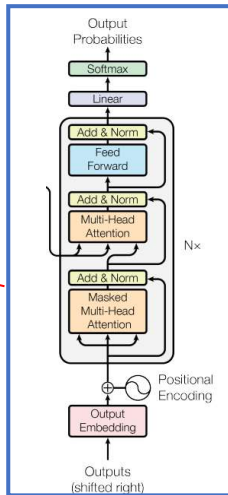
预训练模型—GPT

Generative Pretrained Transformers (GPT)



自回归(Auto-Regressive)生成式预训练模型

Decoder



Improving Language Understanding by Generative Pre-Training

预训练模型—GPT demo

Generative Pretrained Transformers (GPT)

⚡ Hosted inference API ⓘ

📄 Text Generation

Examples ▼

My dog is cute "

she grinned and lifted her hand to run it through her hair with one hand as she bent over the couch, pulling the leash forward with a creak as hers was caught. i sat down across from her and grabbed my

Compute

⌘+Enter

7.0

Computation time on Intel Xeon 3rd Gen Scalable cpu: 6.524 s

Try this on the huggingface : <https://huggingface.co/openai-gpt>

GPT 的预训练

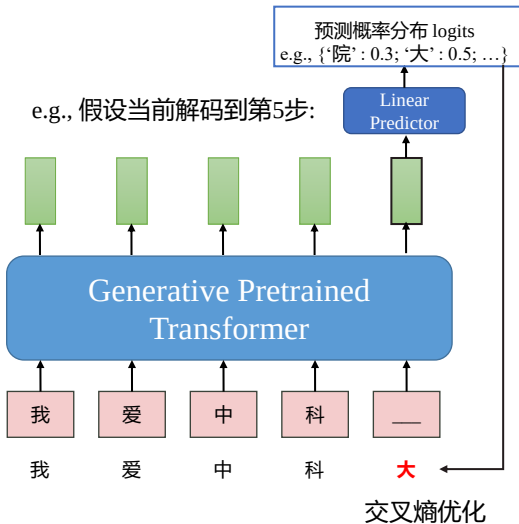
预训练任务: Language Modeling (LM)

根据上文预测下一个token, 并优化其预测概率:

$$\mathcal{L}(\mathcal{U}, \theta) = \sum_i p(u_i | u_{i-1}, u_{i-2}, \dots, u_{i-k}; \theta)$$

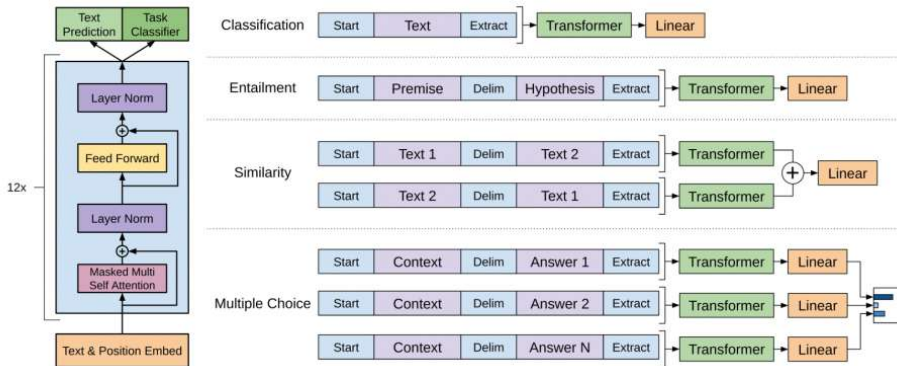
其中:

- $\mathcal{U} = \{u_1, u_2, \dots, u_n\}$ 代表用于训练的文本
- k 代表观察上文的window size
- θ 代表 GPT 的可学习参数
- $L(\cdot, \cdot)$ 代表待优化的LM目标函数

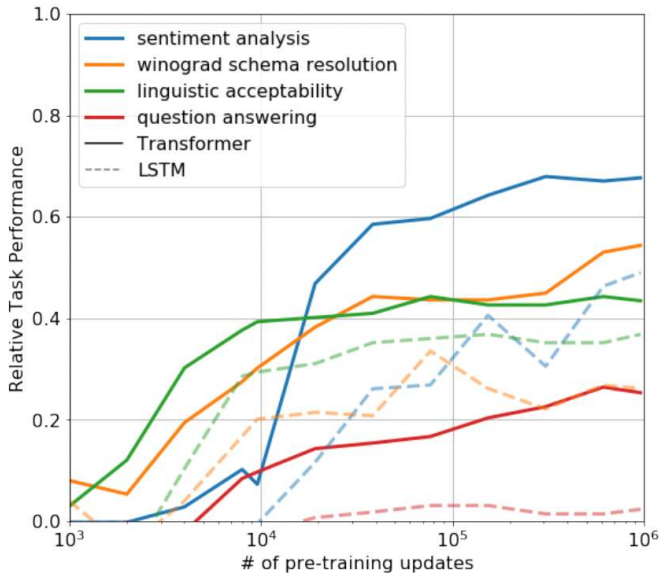


GPT 的应用

GPT 同样可以适用于各种各样的下游任务：



GPT 零样本(Zero-Shot)学习的能力



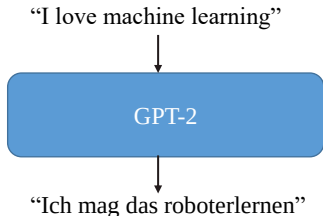
GPT-2

Language Models are Unsupervised Multitask Learners

利用提示 (Prompt) 进行多任务、
零样本学习

e.g., 输入如下:

“translate the following sentence to
German.”--- [prompt/instruction]



“I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I’m not a fool].

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: “**Mentez mentez, il en restera toujours quelque chose**,” which translates as, “**Lie lie and something will always remain**.”

“I hate the word ‘**perfume**,’” Burr says. ‘It’s somewhat better in French: ‘**parfum**.’

If listened carefully at 29:55, a conversation can be heard between two guys in French: “-**Comment on fait pour aller de l’autre coté? -Quel autre coté?**”, which means “- **How do you get to the other side? - What side?**”.

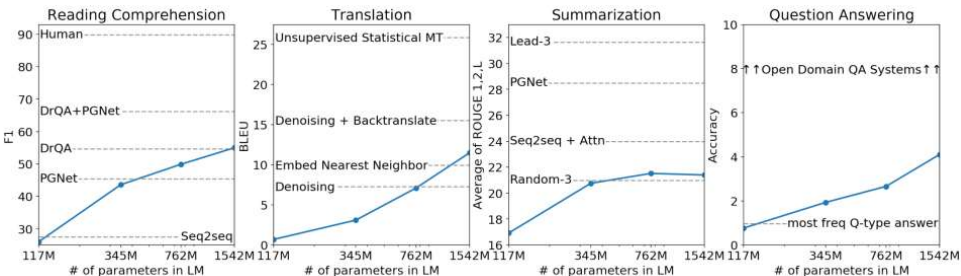
If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

“**Brevet Sans Garantie Du Gouvernement**”, translated to English: “**Patented without government warranty**”.

取自 WebText 文本训练数据集

GPT-2

实验: GPT-2 的零样本学习能力



在4个下游任务(阅读理解、翻译、摘要、问答)中,
Transformer(GPT-2) 的零样本学习性能都随着模型规模(参数量)的扩大而近似单调上升

- GPT-2的参数量从GPT-1的参数量(117M) 上升到1542M

GPT-3

Language Models are Few-Shot Learners

GPT-3的输入:

Translate to German:

Input: "Today is sunny";
Output: "Heute ist sonniges Wetter";
Input: "Mary likes hiking";
Output: "Maria reist gerne";

Input: "I love machine learning"

instruction (和GPT-2一样)

2-shot 样例
(demonstrations)

真正的待翻译语句

上下文“学习”
In-Context Learning

GPT-3

“Ich mag das roboterlernen”

Reference: what learning algorithm is in-context learning? investigations with linear models (ICLR'2023) <https://arxiv.org/abs/2211.15661>

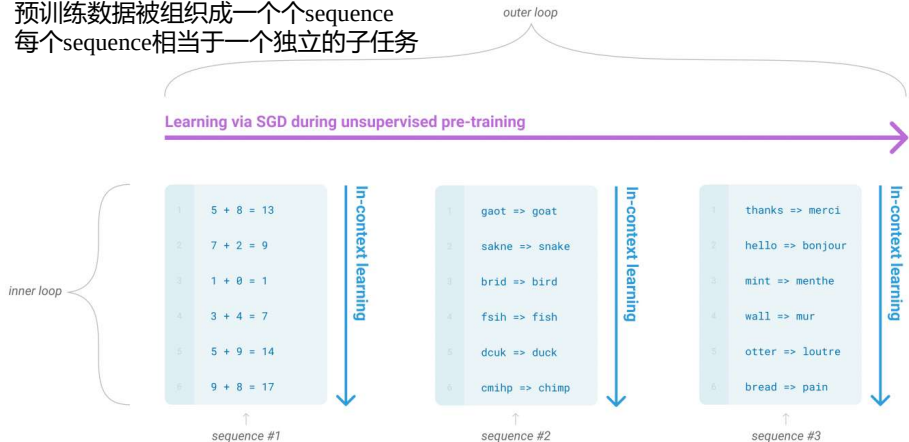
GPT-3

Language Models are Few-Shot Learners

预训练数据的组织方法: training episodes

预训练数据被组织成一个个sequence

每个sequence相当于一个独立的子任务

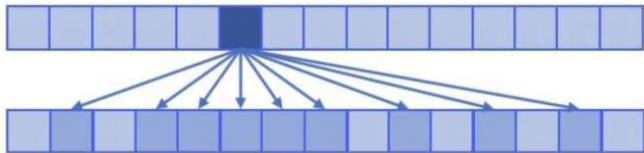


GPT-3

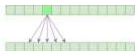
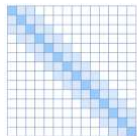
GPT-3 采用Sparse Self-Attention来有效处理长序列输入

Sparse Attention 相较于传统的 Self-Attention, 只去注意: (有效降低自注意力运算时间)

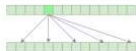
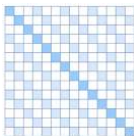
- 距当前 token 相对位置不超过 k 的 tokens (局部自注意力)
- 距当前 token 相对位置为 $n \cdot k$ 的 tokens ($n = 1, 2, 3, \dots$) (空洞自注意力)



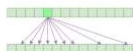
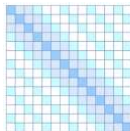
e.g., sparse attention ($k=2$)



local self-attention



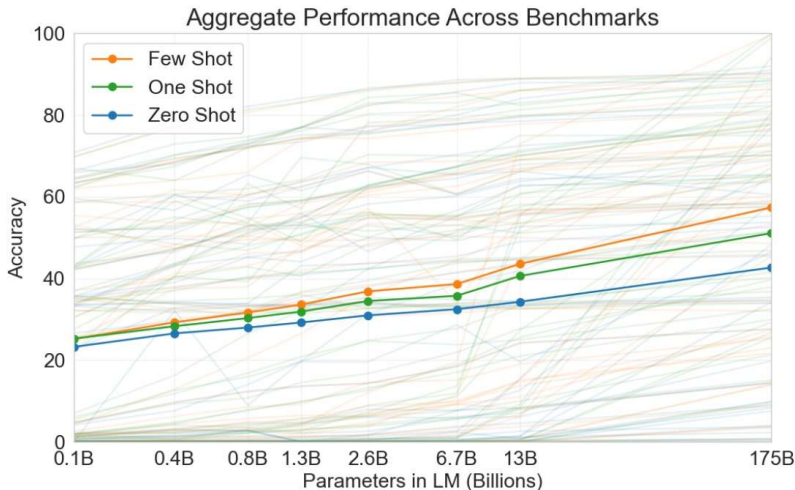
atrous self-attention



local and atrous self-attention

GPT-3

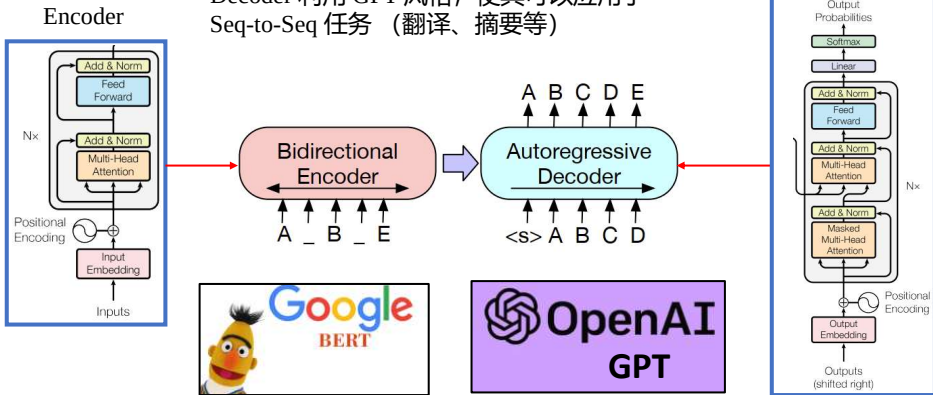
小样本(Few-Shot)、零样本(Zero-Shot) 学习能力随模型规模的变化趋势



预训练模型—BART (Facebook)

BART 同时使用了Transformer的 Encoder 和 Decoder

- Encoder 利用 BERT 风格的双向上下文表征
- Decoder 利用 GPT 风格，使其可以应用于 Seq-to-Seq 任务（翻译、摘要等）

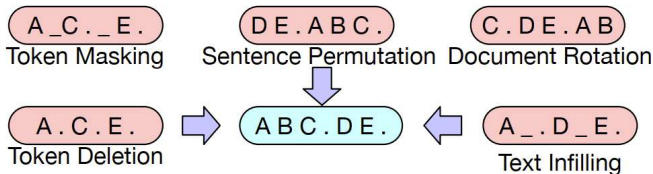


BART: **Denosing** Seq-to-Seq Pre-training for NLG, Translation and Comprehension (ACL'20)

预训练模型—BART

BART 初始实验的预训练任务有很多种:

Token Masking, Token Deletion, Sentence Permutation, Document Rotation, ...



实验表明: Text Infilling 任务是 (在常见的下游任务中) 最为有效的预训练任务。

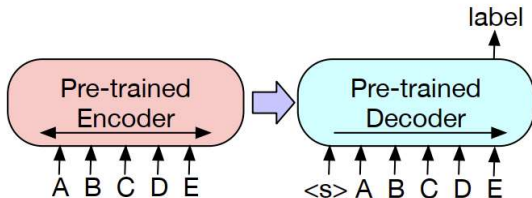
Text Infilling 任务:

- Encoder 采样若干个连续的文本片段 (片段长度从参数为3的泊松分布中采样) 做掩码处理 (每个片段仅用一个 [MASK] 代替)
- Decoder 要求自回归地完全复原本来的输入序列

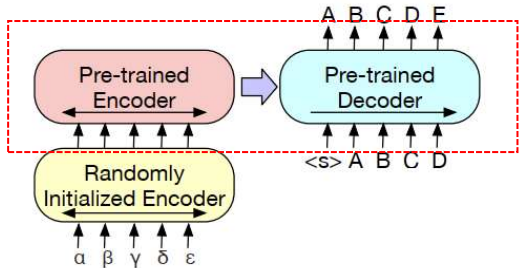
BART: **Denosing** Seq-to-Seq Pre-training for NLG, Translation and Comprehension (ACL'20)

预训练模型—BART

BART 用于文本分类
下游任务



BART 用于机器翻译
下游任务
(Foreigns to English)



Decoder

BERT、BART 和 GPT-1, 2, 3的比较

预训练模型	发布时间	参数量	结构 (Transformer)	预训练目标	迁移到 下游任务的方式
GPT-1	2018	110M	Decoder	LM	Supervised Fine-Tuning
BERT	2019	110M / 340M / 1270M	Encoder	MLM	Supervised Fine-Tuning
GPT-2	2019	1.5B	Decoder	LM	利用Prompt 做 Zero-Shot Learning
BART	2020	140M	Encoder Decoder	Text Infilling	Supervised Fine-Tuning
GPT-3	2020	175B	Decoder	LM	In-Context Learning 实现 Few-Shot Learning

神经推理

Sam walks into the kitchen.
Sam picks up an apple.
Sam walks into the bedroom.
Sam drops the apple.

Q: Where is the apple?

A. Bedroom

Brian is a lion.
Julius is a lion.
Julius is white.
Bernhard is green.

Q: What color is Brian?

A. White

Mary journeyed to the den.
Mary went back to the kitchen.
John journeyed to the bedroom.
Mary discarded the milk.

Q: Where was the milk before the den?

A. Hallway

大脑中的记忆

- 记忆：外界信息在人脑中的内部表示
- 记忆过程
 - 短期记忆：体现为神经元的活动，神经连接的暂时性强化
 - 工作记忆：和任务相关的容器，存放任务相关的短期记忆
 - 长期记忆：体现为神经元之间的连接形态
- 特点：联想记忆 (Associative Memory)
 - 基于内容寻址的存储

不严格的类比

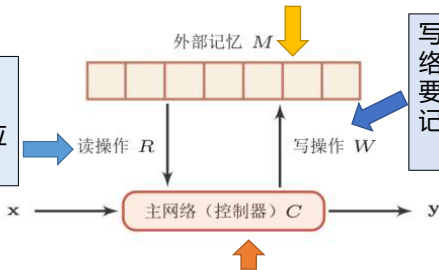
记忆周期	计算机	人脑	神经网络
短期	寄存器	短期记忆	状态（神经元活性）
中期	内存	工作记忆	外部记忆
长期	外存	长期记忆	可学习参数
存储方式	随机寻址	内容寻址	内容寻址为主

记忆增强神经网络

外部记忆单元用来存储信息，一般分为多个记忆片段，一般用向量 $\mathbf{M} = [m_1, \dots, m_N]$ 来表示，不需要全时参与主网络运算

写入模块W：根据主网络生成的查询向量 q_w 和要写入信息 a 来更新外部记忆 $m_i = R(m_i, q_w, a)$
 $\forall 1 \leq i \leq N$

读取模块R：根据主网络生成的查询向量 q_r ，从外部记忆单元中读取相应的信息 $r = R(m_{1:N}, q_r)$



负责信息处理，并与外界的交互（接受外界的输入信息并产生输出到外界）。主网络还同时通过读写模块和外部记忆进行交互

通过引入外部记忆将神经网络的参数和记忆容量的分离，即在少量增加网络参数的条件下可以大幅增加网络容量

记忆增强神经网络

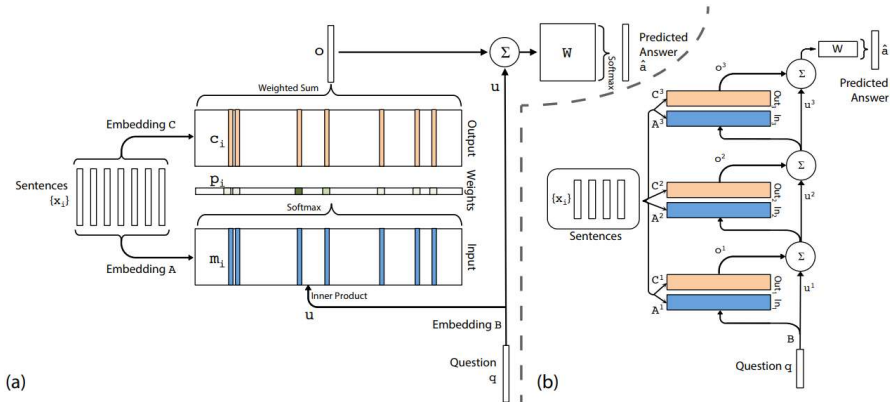
- 每个记忆片段按**内容寻址**的方式进行定位并进行读取或写入操作
- 按内容寻址通常使用**注意力机制**来实现

读取模块的通过
注意力机制实现:

$$\mathbf{r} = \sum_{i=1}^N \alpha_i \mathbf{m}_i$$

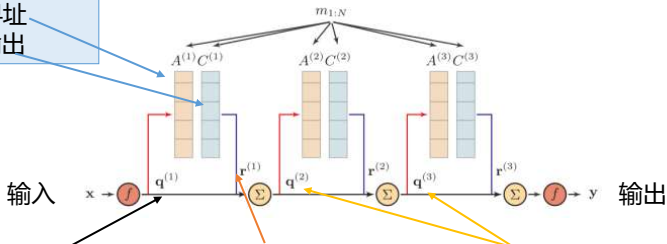
$$\alpha_i = \text{softmax} \left(s(\mathbf{m}_i, \mathbf{q}_r) \right)$$

端到端记忆网络



端到端记忆网络

两组记忆单元,
A用来寻址
C用来输出



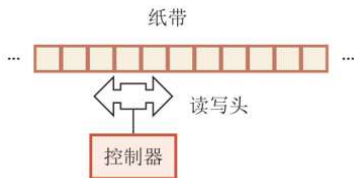
$$r = \sum_{i=1}^N \text{softmax}(a_i^T q) c_i$$

多跳操作 $q^{(k)} = q^{(k-1)} + r^{(k-1)}$
参数一般共享
外部记忆一般也共享使用

神经图灵机： Neural Turing Machine

- 图灵机

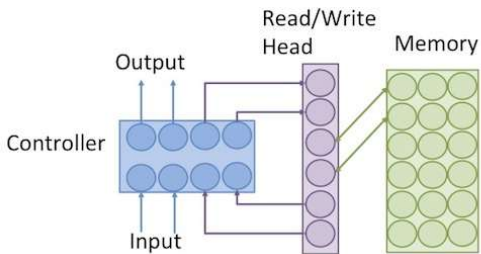
- 一种抽象数学模型，可以用来模拟任何可计算问题



1. **一条无限长的织带**：纸带上有一个个方格组成，每个方格可以存储一个符号；
2. **一个符号表**：纸带上可能出现的所有符号的集合，包含一个特殊的空白符
3. **一个读写头**：指向纸带上某个方格的指针，每次可以向左或右移动一个位置，并可以读取、擦除、写入当前方格中的内容；
4. **一个状态寄存器**：用来保存图灵机当前所处的状态，其中包含两个特殊的状态：起始状态和终止状态；
5. **一套控制规则**：根据当前机器所处的状态以及当前读写头所指的方格上的符号来确定读写头下一步的动作，令机器进入一个新的状态。

神经图灵机

- 组件
 - 控制器
 - 外部记忆
 - 读写操作
- 整个架构可微分



神经图灵机

控制器产生删除向量 e_t 和增加向量 a_t ，为要从外部记忆中删除的信息和要增加的信息

$$r_t = \sum_{i=1}^n \alpha_{t,i} m_{t,i}$$

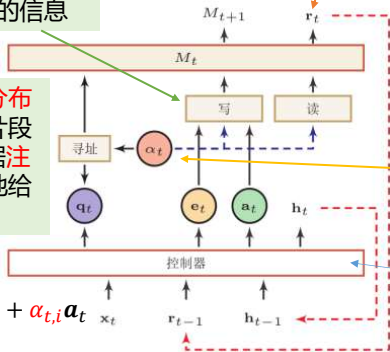
在时刻 t ，外部记忆记为 $M_t = [m_{t,1}, \dots, m_{t,n}]$ 通过注意力机制来寻址

$$\alpha_{t,i} = \text{softmax}(s(m_{t,i}, q_t))$$

控制器为一个前馈或循环神经网络

删除操作是根据注意力分布来按比例地在每个记忆片段中删除 e_t ，增加操作根据注意力分布来进行按比例地给每个记忆片段加入 a_t

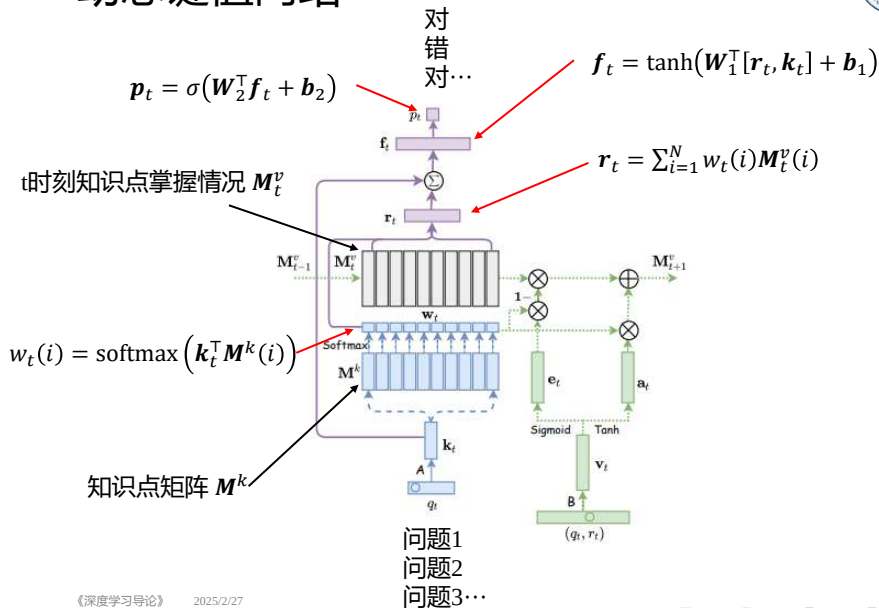
$$m_{t+1,i} = m_{t,i}(1 - \alpha_{t,i}e_t) + \alpha_{t,i}a_t$$



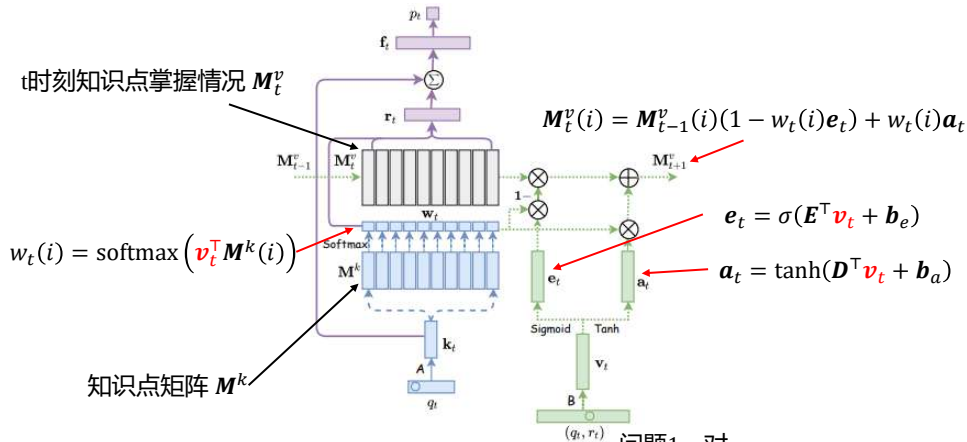
神经图灵机的一个应用

	问题1	问题2	问题3	问题4
学生A	对	对	对	?
学生B	错	对	对	?
学生C	对	错	错	?

动态键值网络

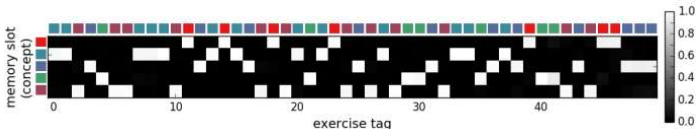


动态键值网络

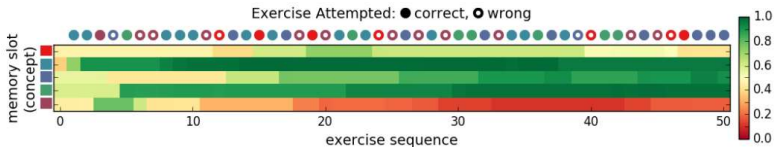


问题1, 对
问题2, 错
问题3, 对

动态键值网络



x轴表示题目，y轴表示知识点和题目的相关性



练习50道题之后,技能掌握情况。该学生掌握了第二、三、四个技能

作业

- 编写基于ELMo/BERT的预语言模型，利用少量的训练数据，微调模型用于文本分类。并和直接用RNN/Transformer训练的文本分类器进行对比，研究训练数据变化对性能带来的影响。
- 请助教准备相关数据集（参考文献如下）

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016, June). Hierarchical attention networks for document classification. In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies (pp. 1480-1489).



图卷积神经网络

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

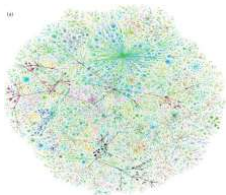
主页：<http://staff.ustc.edu.cn/~liandefu>



本章内容

- 图的定义和基本概念
- 图卷积神经网络
 - 预备知识
 - 谱方法的图卷积网络
 - 空间方法的图卷积网络
- 挑战与前沿
 - 过渡平滑
 - 采样问题
 - 标签问题

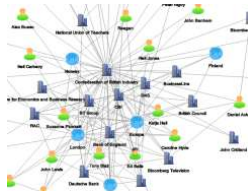
无处不在的图



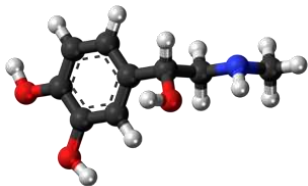
Internet



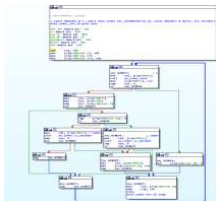
Social networks



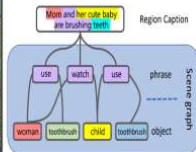
Networks of transactions



Biomedical graphs



Program graphs



Scene graphs



图的组成与表示

- 组成: $G = (V, E)$
 - 节点的集合 $V = \{v_1, v_2, \dots, v_n\}$
 - 边的集合 E
- 表示
 - $A \in R^{n \times n}$: 邻接矩阵 (adjacency matrix)
 - $X \in R^{n \times d}$: 节点属性矩阵 (node feature matrix)
 - $Y \in R^{n \times c}$: 节点标签矩阵 (node label matrix)

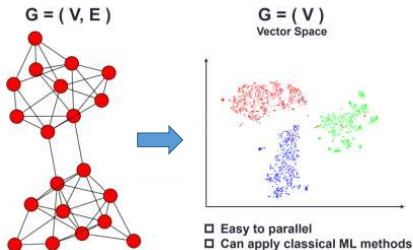
图表示学习

• 定义

- 给定一个图 $G = (V, E)$, 将图上的节点压缩成低维的向量表示 $R^{n \times k} (k \ll n)$.

• 性质

- 相邻节点具有相似的向量表示
- 具有相似属性的节点具有相似向量表示
- 节点的顺序变换对向量表示没有影响





图表示学习的应用

- 节点分类
 - 预测节点类别
- 链路预测
 - 预测两个节点是否有边相连
- 社区检测
 - 检测紧密连接的簇
- 图匹配
 - 计算两个子图的相似度
- 图分类
 - 预测一个图（子图）的类别

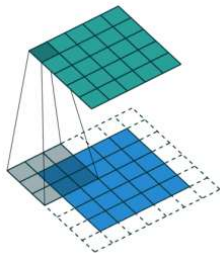
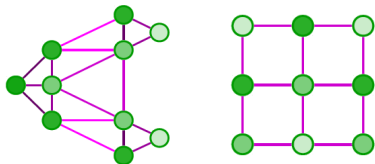


图表示学习方法的分类

- 基于矩阵分解的图表示学习方法
 - 拉普拉斯特征映射
 - 图分解
- 基于随机游走的图表示学习方法
 - DeepWalk
 - Node2Vec
- 基于图神经网络的表示学习方法

基于图神经网络表示学习的挑战

- 图 V.S 网络
 - 邻居节点的不均匀分布
 - 排列不变性
 - 边的额外属性
- 传统的深度学习 (CNN、RNN) 无法直接移植到图
 - 难以选取固定的卷积核来适应整个图的不规则性
 - 邻居节点数量的不确定和节点顺序的不确定
- 图神经网络既要考虑**节点信息**，也要考虑**结构信息**



基于图神经网络的表示学习方法

- 图神经网络本质上用来提取拓扑图的空间特征，主要包括两类方法
 - 基于空间域或顶点域vertex domain(spatial domain)的图卷积网络，可类比到直接在图片的像素点上进行卷积
 - 代表性方法包括：
 - Spectral Graph CNN (ICLR 2014)
 - ChebNet (NIPS 2016)
 - GCN (ICLR 2017)
 - 基于频域或谱域spectral domain的图卷积网络，可类比到对图片进行傅里叶变换后，再进行卷积
 - 代表性方法包括：
 - MPNN (NIPS 2017)
 - GraphSage (ICML 2017)
 - GAT (ICLR 2018)

图卷积神经网络的发展

空域可以类比到直接在图片的像素点上进行卷积

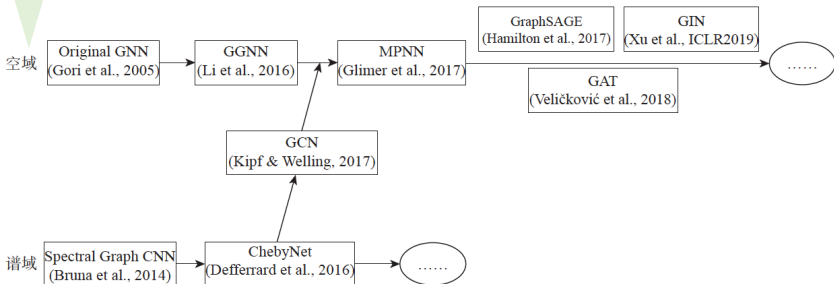
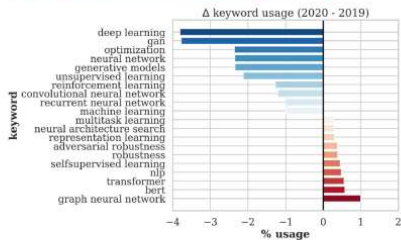
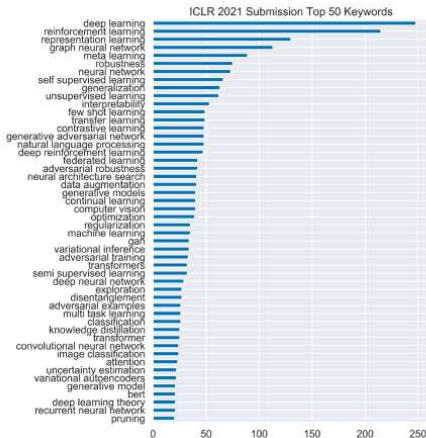


图 1.3 图神经网络的早期发展

谱域可以类比到对图片进行傅里叶变换后，再进行卷积

图卷积神经网络的发展



GNNs are currently experiencing their "ImageNet" moment



丰富的数据集生态系统



ogb.stanford.edu



PyTorch
geometric

<https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html>



TUDataset

graphlearning.io

Benchmarking Graph Neural Networks

github.com/graphdeeplearning/benchmarking-gnns

丰富的开发工具包



PyTorch
geometric

github.com/rustyls/pytorch_geometric

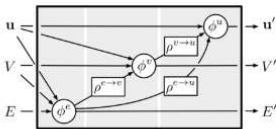


Spektral

[graphneural.network](https://github.com/graphneural.network)

DGL

dgl.ai



github.com/deepmind/graph_nets



github.com/deepmind/jraph

KDD CUP 2021: Large-Scale Graph ML



Sign up for [our Google group](#) to keep up to date with major changes.

[Overview](#) [Timeline](#) [Rules](#) [What's New](#) [MAG240M](#) [WikiKG90M](#) [PCQM4M](#) [Participate](#) [Leaderboards](#) [Paper](#)

Why a Large-Scale Graph ML Competiton?

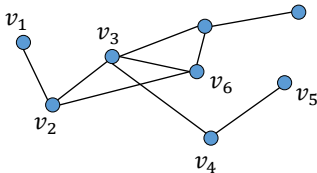
Machine Learning (ML) on graphs has attracted immense attention in recent years because of the prevalence of graph-structured data in real-world applications. Modern application domains include web-scale social networks, recommender systems, hyperlinked web documents, knowledge graphs (KGs), as well as molecule simulation data generated by the ever-increasing scientific computation. These domains involve large-scale graphs with billions of edges or a dataset with millions of graphs. Deploying accurate graph ML at scale will have a huge practical impact, enabling better recommendation results, improved web document search, more comprehensive KGs, and accurate ML-based drug and material discovery. However, community efforts to advance state-of-the-art in large-scale graph ML have been extremely limited. In fact, most of graph ML models have been developed and evaluated on extremely small datasets.

Handling large-scale graphs is challenging, especially for state-of-the-art expressive Graph Neural Networks (GNNs) because they make prediction on each node based on the information from many other nodes. Effectively training these models at scale requires sophisticated algorithms that are well beyond standard SGD over i.i.d. data. More recently, researchers improve model scalability by significantly simplifying GNNs, which inevitably limits their expressive power.

However, in deep learning, it has been demonstrated over and over again that one needs big expressive models and train them on big data to achieve the best performance. In graph ML, the trend has been the opposite—models get simplified and less expressive to be able to scale to large graphs. Thus, there is a massive opportunity to move the community to work with realistic and large-scale graph datasets and move the state of the field forward to where it needs to be.

图的基本概念

- 图的矩阵表示
 - 邻接矩阵 $A_{ij} = 1$, 如果 v_i 和 v_j 相邻



邻接矩阵

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

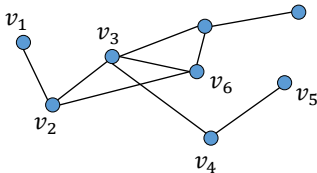
A

图的基本概念

• 图的矩阵表示

- 邻接矩阵 $A_{ij} = 1$, 如果 v_i 和 v_j 相邻
- 度矩阵 $D = \text{diag}(d(v_1), \dots, d(v_N))$

$$d(v_i) = \sum_{v_j \in \mathcal{N}_{v_i}} A_{ij}$$



度矩阵

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

D

邻接矩阵

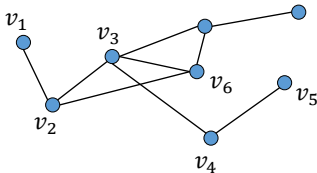
$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

A

图的基本概念

• 拉普拉斯矩阵的定义

$$\bullet L = D - A$$



度矩阵

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

D

邻接矩阵

$$- \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

A

拉普拉斯矩阵

$$= \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

L

图的基本概念

- 拉普拉斯矩阵的性质 $L = D - A$
- 1) $L \mathbf{1} = D\mathbf{1} - A\mathbf{1} = \mathbf{d} - \mathbf{d} = \mathbf{0}$ $\mathbf{d} = \text{diag}(D)$
- 2) $\mathbf{1}^T L = \mathbf{0}$
- 3) L 是半正定的。最小特征值为0，其特征向量为 $\mathbf{1}$

证明： 对于任意的向量 $\mathbf{z} \in \mathbb{R}^n$ ，那么

L 的特征值中0的个数等
同于原图的连通块数量

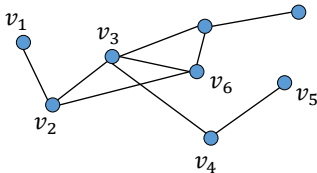
$$\begin{aligned}
 \mathbf{z}^T L \mathbf{z} &= \sum_i d_i z_i^2 - \sum_{ij} A_{ij} z_i z_j \\
 &= \frac{1}{2} \left(\sum_i d_i z_i^2 - 2 \sum_{ij} A_{ij} z_i z_j + \sum_j d_j z_j^2 \right) \\
 &= \frac{1}{2} \left(\sum_i \sum_j A_{ij} z_i^2 - 2 \sum_{ij} A_{ij} z_i z_j + \sum_j \sum_i A_{ij} z_j^2 \right) \\
 &= \frac{1}{2} \sum_i \sum_j A_{ij} (z_i - z_j)^2 \geq 0
 \end{aligned}$$

图的基本概念

• 归一化后的拉普拉斯矩阵

• 对称归一化: $L^{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

• 随机游走归一化: $L^{rw} = D^{-1}L = I - D^{-1}A$



度矩阵

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

D

邻接矩阵

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

A

拉普拉斯矩阵

$$L = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

L

图的基本概念

- 归一化后的拉普拉斯矩阵 L^{sym} 的性质:
- 1) L^{sym} 特征值的介于0到2之间

根据瑞利熵 $\lambda_{max}(L^{sym}) = \max_z \frac{\mathbf{z}^\top \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} \mathbf{z}}{\mathbf{z}^\top \mathbf{z}} = \max_f \frac{\mathbf{f}^\top \mathbf{L} \mathbf{f}}{\mathbf{f}^\top \mathbf{D} \mathbf{f}}$

$$0 \leq \frac{\mathbf{f}^\top \mathbf{L} \mathbf{f}}{\mathbf{f}^\top \mathbf{D} \mathbf{f}} = \frac{\frac{1}{2} \sum_i \sum_j A_{ij} (f_i - f_j)^2}{\sum_i f_i^2 d_i} \leq \frac{\sum_i \sum_j A_{ij} (f_i^2 + f_j^2)}{\sum_i f_i^2 d_i} = 2$$

$$(f_i - f_j)^2 \leq 2(f_i^2 + f_j^2)$$

当对于任意相邻的两个顶点
满足 $f_i + f_j = 0$ 时等式成立

此时该图为一个二部图

当 $\mathbf{f} = \mathbf{1}$, 即 $\mathbf{z} = \sqrt{\mathbf{d}}$ 时, 等式=0成立

图的基本概念

- 若 $\mathbf{z}$ 为图上的图信号，则下式表示图信号的光滑程度

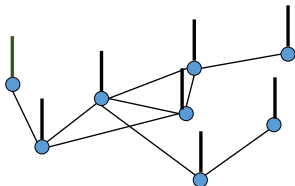
$$\mathbf{z}^T \mathbf{L} \mathbf{z} = \frac{1}{2} \sum_i \sum_j A_{ij} (z_i - z_j)^2$$

“Smoothness” of the signal $\mathbf{z}$

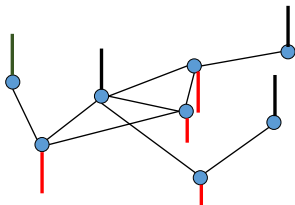
- 假设 $\mathbf{L} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^T$ 为其特征值分解，那么任意一个图信号 $\mathbf{z}$ 可以用特征向量表示即 $\mathbf{z} = \mathbf{U} \mathbf{\alpha}$ ，则该式也表示信号的频率

$$\mathbf{z}^T \mathbf{L} \mathbf{z} = \mathbf{\alpha}^T \mathbf{U}^T \mathbf{L} \mathbf{U} \mathbf{\alpha} = \mathbf{\alpha}^T \mathbf{\Lambda} \mathbf{\alpha} = \sum_i \lambda_i \alpha_i^2$$

“Frequency” of the signal $\mathbf{z}$



Low frequency graph signal



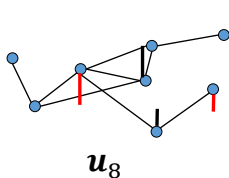
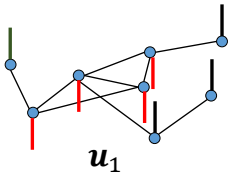
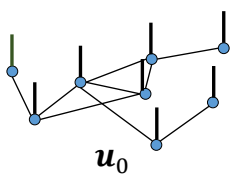
High frequency graph signal

图的基本概念

- 如果将特征向量作为图的信号，该信号的频率即为拉普拉斯矩阵对应的特征值

$$\underline{\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i} = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i$$

Frequency of the signal $\mathbf{u}_i$



High frequency

$$\mathbf{u}_0^T \mathbf{L} \mathbf{u}_0 = \lambda_0 = 0$$

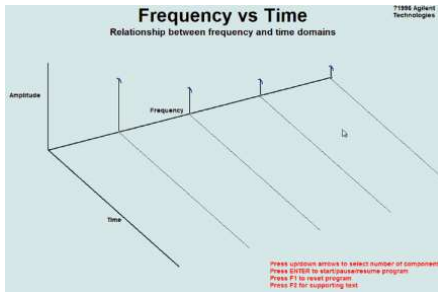
$$\mathbf{u}_1^T \mathbf{L} \mathbf{u}_1 = \lambda_1$$

$$\mathbf{u}_8^T \mathbf{L} \mathbf{u}_8 = \lambda_8$$

傅里叶变换和逆变换

$$F(\omega) = \mathcal{F}[f(x)] = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x} dx$$

$$f(x) = \mathcal{F}^{-1}[F(\omega)] = \int_{-\infty}^{+\infty} F(\omega)e^{i\omega x} dx$$



核心思想：用一组正交函数 $\cos$ 和 $\sin$ (或 $e^{-i\omega t}$)表示任意函数

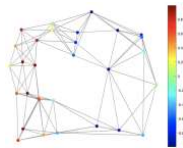
图傅里叶变换

- $L = U\Lambda U^T$ 为其特征值分解, U 的列向量类比于傅里叶变换中的基
- 对图上信号 f 的图傅里叶变换:

$$\mathcal{F}(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^n f(i)u_l(i) = \mathbf{u}_l^T \mathbf{f}$$

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1(1) & \mathbf{u}_1(2) & \dots & \mathbf{u}_1(N) \\ \mathbf{u}_2(1) & \mathbf{u}_2(2) & \dots & \mathbf{u}_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_N(1) & \mathbf{u}_N(2) & \dots & \mathbf{u}_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix}$$

$$\Rightarrow \hat{\mathbf{f}} = U^T \mathbf{f}$$

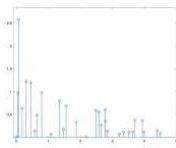


Spatial domain: $\mathbf{f}$

$$\hat{\mathbf{f}} = U^T \mathbf{f}$$



Decompose signal $\mathbf{f}$



Spectral domain: $\hat{\mathbf{f}}$

图傅里叶逆变换

- $L = U\Lambda U^T$ 为其特征值分解, U 的列向量类比于傅里叶变换中的基
- 图傅里叶逆变换:

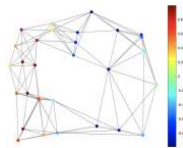
$$f(i) = \sum_{l=1}^n \hat{f}(\lambda_l) u_l(i) = \mathbf{u}(i)^T \hat{\mathbf{f}}$$

$$\begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1(1) & \mathbf{u}_2(1) & \dots & \mathbf{u}_N(1) \\ \mathbf{u}_1(2) & \mathbf{u}_2(2) & \dots & \mathbf{u}_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_1(N) & \mathbf{u}_2(N) & \dots & \mathbf{u}_N(N) \end{pmatrix} \begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix}$$



$$\mathbf{f} = \mathbf{U} \hat{\mathbf{f}}$$

任一信号 $\mathbf{f}$ 是不同频率的特征向量的组合

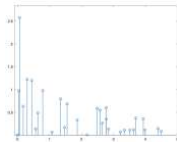


Spatial domain: $\mathbf{f}$

$$\mathbf{f} = \mathbf{U} \hat{\mathbf{f}}$$



Decompose signal $\mathbf{f}$



Spectral domain: $\hat{\mathbf{f}}$

谱域上的图卷积

- 空域上很难定义卷积-->转到谱域
- 从图信号处理的角度考虑图卷积



- 卷积公式 $f * g = \mathbb{F}^{-1}\{\mathbb{F}\{f\} \cdot \mathbb{F}\{g\}\}$

卷积定理：函数卷积的傅里叶变换是函数傅立叶变换的乘积

- 给一个图信号 x 和一个卷积核 g $x * g = U(U^\top x \odot U^\top g)$
- $U^\top g$ 当成整体的卷积核，用参数 θ 表示

$$x * g = U(U^\top x \odot U^\top g) = U(U^\top x \odot \theta) = U g_\theta U^\top x$$

用 g_θ 表示对角线为 θ 的矩阵

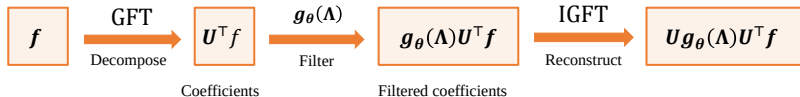
谱域图卷积神经网络

- 回顾:

$$\text{GFT: } \hat{f} = U^T f$$

$$\text{IGFT: } f = U \hat{f}$$

- 滤波信号: 空域信号 $\rightarrow$ 谱域信号 $\rightarrow$ 滤波 $\rightarrow$ 空域信号



最简单的谱域图卷积网络

- Spectral Graph CNN
 - 无参数化: g_θ 与 L 的特征向量无关

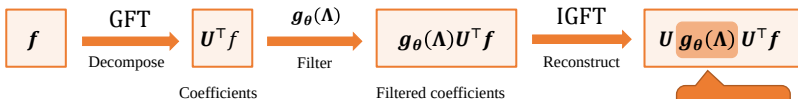
$$x * g = U g_\theta U^\top x = U \begin{pmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \ddots & \\ & & & \theta_n \end{pmatrix} U^\top x$$

- 缺点
 - 参数多, 总共有 n 个参数, n 为节点数
 - 需要对拉普拉斯矩阵进行特征分解, $O(n^3)$ 时间复杂度
 - 不能局部化: 每个节点上信号依赖于其他全部节点信号, 而不只是邻居

多项式卷积核 (ChebyNet)

- 局部化: L 对图信号 f 的操作 $L f$ 相当于在图上传播一步

常用归一化的 L ,
比如 L^{sym}



假设滤波器是拉普拉斯矩阵的函数

$g_\theta(L)$

- 多项式卷积核:

$$g_\theta(\Lambda) = \sum_{k=0}^K \theta_k \Lambda^k$$

$$U \hat{g}(\Lambda) U^T f = U \sum_{k=0}^K \theta_k \Lambda^k U^T f$$

$$L^k = U \Lambda^k U^T$$

$$= \sum_{k=0}^K \theta_k L^k f$$

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^K \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^K \theta_k \lambda_2^k & & \\ & & \dots & \\ & & & \sum_{k=0}^K \theta_k \lambda_N^k \end{bmatrix}$$

局部性: 实际上在图上传播了 K 步, 因此一个节点只影响它周围距离为 K 以内的邻居
效率高: 不需要进行特征值分解
参数少: 只有 $K+1$ 个参数

多项式卷积核 (ChebyNet)

- 基于切比雪夫多项式，由于其递归方式定义，可以进一步提升计算效率

- 切比雪夫多项式，通过如下递归定义

- $T_0(x) = 1$; $T_1(x) = x$
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1$$

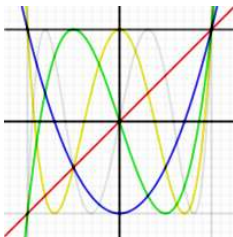
$$T_5(x) = 16x^5 - 20x^3 + 5x$$

$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$$

$$T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x$$

$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$

$$T_9(x) = 256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x$$



前六个第一类切比雪夫多项式的图像，其中 $-1/4 < x < 1/4$, $-1/4 < y < 1/4$; 按颜色依次是 T_0 , T_1 , T_2 , T_3 , T_4 , T_5 .

多项式卷积核 (ChebyNet)

- 切比雪夫多项式, 通过如下递归定义
 - $T_0(x) = 1; T_1(x) = x$
 - $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$
- 由于其定义域为 $[-1, 1]$, 因此 $\hat{g}(\Lambda)$ 通过如下方式定义

$$\hat{g}(\Lambda) = \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda}) \quad \text{其中 } \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{\max}} - I$$

- 在归一化之后, 任然可以不用计算特征值分解
- 有结论 $U\hat{g}(\Lambda)U^\top f = \sum_{k=0}^K \theta_k T_k(\tilde{L}) f$, 其中 $\tilde{L} = \frac{2L}{\lambda_{\max}} - I$

多项式卷积核 (ChebyNet)

- 通过数学归纳法证明
 - $T_0(x) = 1; T_1(x) = x$
 - $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

$$U\hat{g}(\Lambda)U^T f = \sum_{k=0}^K \theta_k T_k(\tilde{L}) f, \text{ 其中 } \tilde{L} = \frac{2L}{\lambda_{\max}} - I$$

- 容易验证, 当 $k = 0$ 显然有 $T_k(\tilde{L}) = UT_k(\tilde{\Lambda})U^T$

$$\text{当 } k = 1 \text{ 时 } \tilde{L} = \frac{2L}{\lambda_{\max}} - I = \frac{2U\Lambda U^T}{\lambda_{\max}} - UU^T = U\left(\frac{2\Lambda}{\lambda_{\max}} - I\right)U^T = U\tilde{\Lambda}U^T$$



结论成立

假设当 $n = k - 1$ 和 $n = k - 2$ 时 $T_n(\tilde{L}) = UT_n(\tilde{\Lambda})U^T$ 成立

$$\begin{aligned} T_k(\tilde{L}) &= 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L}) \\ &= 2U\tilde{\Lambda}U^T UT_{k-1}(\tilde{\Lambda})U^T - UT_{k-2}(\tilde{\Lambda})U^T \\ &= U(2\tilde{\Lambda}T_{k-1}(\tilde{\Lambda}) - T_{k-2}(\tilde{\Lambda}))U^T = U(T_k(\tilde{\Lambda}))U^T \end{aligned}$$



结论成立

多项式卷积核 (ChebyNet)

$$U\hat{g}(\Lambda)U^\top f = \sum_{k=0}^K \theta_k T_k(\tilde{L})f, \text{ 其中 } \tilde{L} = \frac{2L}{\lambda_{\max}} - I$$

一起计算会更快

- $T_0(x) = 1; T_1(x) = x$
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$



$$T_0(\tilde{L})f = f$$

$$T_1(\tilde{L})f = \tilde{L}f$$

$$T_k(\tilde{L})f = 2\tilde{L}T_{k-1}(\tilde{L})f - T_{k-2}(\tilde{L})f$$



$$F_0 = f$$

$$F_1 = \tilde{L}f$$

$$F_k = 2\tilde{L}F_{k-1} - F_{k-2}$$

归一化图拉普拉斯矩阵 L^{sym}
最大特征值约为2, $\lambda_{\max} \approx 2$

$$L^{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

$$\tilde{L} = \frac{2L^{sym}}{\lambda_{\max}} - I = L^{sym} - I = -D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

从ChebyNet到GCN

- 假设我们只取1阶切比雪夫多项式，且 $\lambda_{max} \approx 2$

$$y = \sum_{k=0}^1 \theta_k T_k(\tilde{L}) x \approx \theta_0 T_0(\tilde{L}) x + \theta_1 T_1(\tilde{L}) x \\ \approx \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

$$\text{取 } \theta' = \theta_0 = -\theta_1 \approx \theta' \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

- 此时节点只能被它周围的1阶邻接点所影响，但只需要叠加K层这样的图卷积层，就可以把节点的影响力扩展到K阶邻居节点

根据类似方法，证明可得 $I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ 的特征值在[0,2]之间，叠加多层图卷积层会多次迭代这个操作，可能造成数值不稳定和梯度爆炸的问题

$$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \xrightarrow{\text{做归一化}} \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \Rightarrow Y = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$$

$$\tilde{A} = A + I \\ \tilde{D} = D + I$$

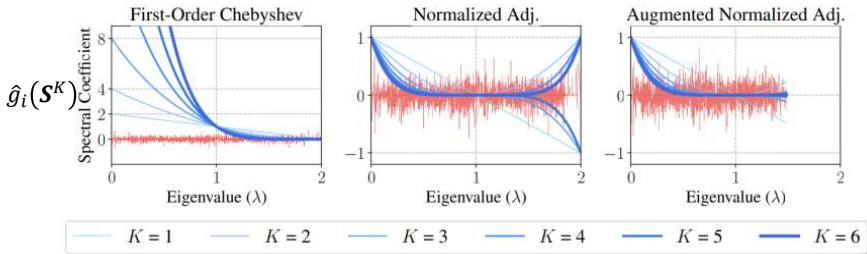
$X \in R^{n \times d}$: 节点属性矩阵

添加自环对谱系数的作用

$$\mathbf{S}_{1\text{-order}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

$$\mathbf{S}_{\text{adj}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

$$\tilde{\mathbf{S}}_{\text{adj}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$$



$$\hat{g}_i(\mathbf{S}_{1\text{-order}}^K) = (2 - \lambda_i)^K \quad \hat{g}_i(\mathbf{S}_{\text{adj}}^K) = (1 - \lambda_i)^K \quad \hat{g}_i(\tilde{\mathbf{S}}_{\text{adj}}^K) = (1 - \tilde{\lambda}_i)^K$$

λ_i 为 $\mathbf{L}^{\text{sys}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ 的第 i 个特征值 $\tilde{\lambda}_i$ 为 $\tilde{\mathbf{L}}^{\text{sys}} = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$ 的第 i 个特征值

添加自环对谱系数的作用

定理: 设 $\tilde{A} = A + \gamma I$, $\tilde{D} = D + \gamma I$, 其中 $\gamma > 0$ 。若 λ_i 为 $L^{sym} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ 的第 i 个特征值, $\tilde{\lambda}_i$ 为 $\tilde{L}^{sym} = I - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ 的第 i 个特征值, 那么

$$0 = \lambda_1 = \tilde{\lambda}_1 < \tilde{\lambda}_n < \lambda_n$$

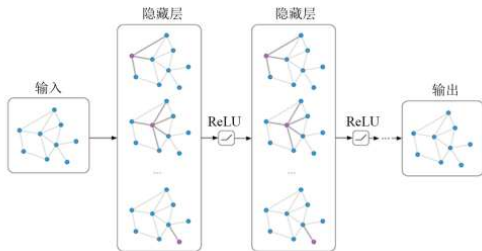
证明参见如下参考文献

从该定理可看出, 添加自环后, 归一化拉普拉斯矩阵的最大特征值会变小

从ChebyNet到GCN

- 叠加多层GCN得到最终的模型(常取两层)

$$\hat{Y} = f(X, A) = \text{Softmax} \left(\hat{A} \text{ReLU}(\hat{A} X W^0) W^1 \right)$$



其中

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

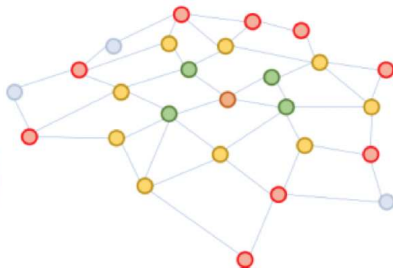
$$\tilde{A} = A + I_n$$

$$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

再看GCN

- 从矩阵形式到每个节点的更新公式

$$\begin{aligned} h_{v_i}^{l+1} &= \sigma \left(\sum_{j=1}^n \frac{1}{c_{ij}} h_{v_j}^l W^l \right) \\ &= \sigma \left(\sum_{v_k \in \mathcal{N}(v_i) \cup \{v_i\}} \frac{1}{c_{ik}} h_{v_k}^l W^l \right) \end{aligned}$$

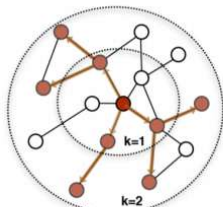


- GCN原来也是个空域上的图神经网络？

空域图神经网络：从空域上直接聚合邻居的信息

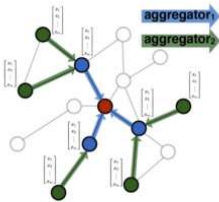
空域图神经网络—GraphSAGE

- 将空域图卷积神经网络拆分成汇聚和更新的两个过程，通过迭代地采样周围邻居的特征来更新得到向量表征

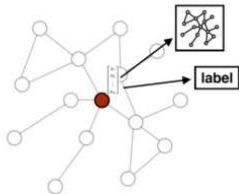


1. Sample neighborhood

选择固定大小的邻居
一阶邻居采样3个，
二阶邻居采样5个



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

聚合函数

- 均值聚合
- LSTM聚合
- 池化聚合

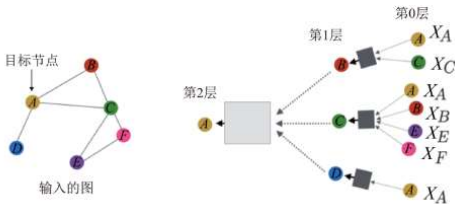
- 无监督目标函数：图重构
- 有监督目标函数：分类损失

空域图神经网络的一般形式

• 消息传递网络MPNN：信息传递(Message Passing Framework)

- 生成并转化周围邻居的信息
$$m_v^{t+1} = \sum_{w \in \mathcal{N}(v)} M_t(h_v^t, h_w^t, e_{vw})$$

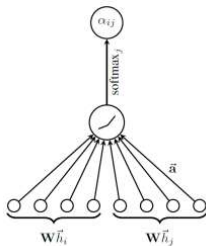
- 更新目标结点的状态
$$h_v^{t+1} = U(h_v^t, m_v^{t+1})$$



空域图卷积神经网络—GAT

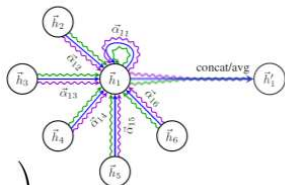
- 将自注意力的思想引入到图卷积网络中
- 某条边上连接的两个节点间注意力公式

$$a_{ij} = \frac{\exp(\text{LeakyReLU}(a[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(a[\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k]))}$$



- 节点的更新
- 多头注意力

$$\mathbf{h}'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} a_{ij} \mathbf{W} \mathbf{h}_j \right)$$



$$\mathbf{h}'_i = \parallel_{k=1}^K \left(\sum_{j \in \mathcal{N}_i} a_{ij}^k \mathbf{W}^k \mathbf{h}_j \right) \quad \mathbf{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} a_{ij}^k \mathbf{W}^k \mathbf{h}_j \right)$$

谱域GNN与空域GNN对比

谱域GNN

优点

- 与图信号处理(GSP)直接相关, 方便分析
- 容易引入GSP领域中更高级的filter达到特定的功能

缺点

- 通常计算复杂度高
- 通常不能迁移到别的图上 (not inductive learning)

空域GNN

优点

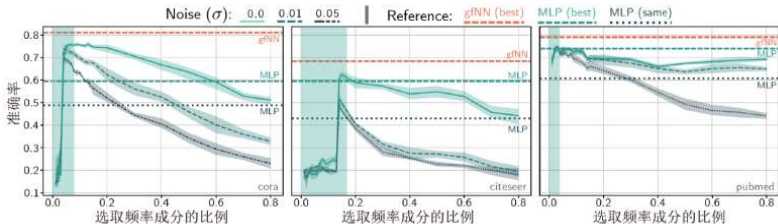
- 容易理解和实现, 统一的框架
- 容易导入边的信息或做出其他的改进

缺点:

- 过拟合
- ...

GCN的本质：低通滤波

- GNN是一个滤波器 $y = g_{\theta}(L)x = g_{\theta}(U\Lambda U^T)x = U g_{\theta}(\Lambda) U^T x$
- 给图信号加高斯噪声，再只取前k个基底进行图傅里叶(逆)变换，训练2层的MLP，在图上做节点分类 $\tilde{D}^{-1/2}U[:,k]U[:,k]^T\tilde{D}^{1/2}x$



- 取前20%的低频信号，分类效果最好
- 高频信号的保留不利于分类，添加的噪声越大，趋势越明显

GCN的本质：低通滤波

- GCN是低通滤波器

$$L^{sym} = U\Lambda U^T$$

$$D^{-1/2}AD^{-1/2}X = (I - L^{sym})X = U(I - \Lambda)U^T X$$

$$g(\Lambda) = I - \Lambda$$

$$\text{回顾: } \lambda_i(L^{sym}) \in [0, 2] \quad \rightarrow \quad g(\Lambda)_i \in [-1, 1]$$

在GCN中还添加自环 $\tilde{A} = A + I$ ，最大特征值会进一步收缩。因此，大部分图都是低通滤波

GCN的推广—图滤波神经网络 (gfNN)

- 单层图卷积网络可以写成一个低通滤波器加上一个单层前馈网络
- 形成先滤波再学习一般框架，可以采用各种低通滤波器

$$\mathbf{f} = f_{\text{mlp}}(f_A(\mathbf{X}))$$

- 拉普拉斯正则化的SSL (半监督学习)

$$\mathbf{y} = \arg \min_{\mathbf{y}} \|\mathbf{y} - \mathbf{x}\|_2^2 + \gamma \mathbf{y}^\top \mathbf{L}^{\text{sym}} \mathbf{y}$$

$$\Rightarrow \mathbf{y} = (\mathbf{I} + \gamma \mathbf{L}^{\text{sym}})^{-1} \mathbf{x} = \mathbf{U} (\mathbf{I} + \gamma \mathbf{\Lambda})^{-1} \mathbf{U}^\top \mathbf{x}$$

$$g(\mathbf{\Lambda}) = (\mathbf{I} + \gamma \mathbf{\Lambda})^{-1}$$

是一个低通滤波

当 $\gamma = 1$, 做一阶泰勒展开 $g(\mathbf{\Lambda}) \approx \mathbf{I} - \mathbf{\Lambda}$

GCN的推广—简化图神经网络 (SGC)

- 单层图卷积网络可以写成一个低通滤波器加上一个单层前馈网络
- 形成先滤波再学习一般框架，可以采用各种低通滤波器

$$f = f_{\text{mlp}}(f_A(\mathbf{X}))$$

- 简化图神经网络：省略每一层的非线性函数，那么

$$\begin{aligned} \mathbf{Z} &= \hat{\mathbf{A}}^m \mathbf{X} \mathbf{W}_1 \cdots \mathbf{W}_m & \hat{\mathbf{A}} &= \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \\ &= \hat{\mathbf{A}}^m \mathbf{X} \boldsymbol{\Theta} \end{aligned}$$

- 在所有线性层叠加完之后，加上非线性层来预测

$$\mathbf{Y} = \text{Softmax}(\mathbf{Z}) = \text{Softmax}(\hat{\mathbf{A}}^m \mathbf{X} \boldsymbol{\Theta})$$

- 这里的 $\hat{\mathbf{A}}^m \mathbf{X} = (\mathbf{I} - \tilde{\mathbf{L}}^{\text{sym}})^m \mathbf{X}$ 也对应一个低通滤波器

GCN的本质：拉普拉斯平滑

- 拉普拉斯平滑：让一个点和它周围得到点尽可能相似，每个节点的特征是其周围节点特征的均值。
- 考虑有添加自环的图 $\tilde{A} = A + I$ ，那么任一特征加上自身节点信息的拉普拉斯平滑

$$\hat{y}_i = (1 - \gamma)x_i + \gamma \sum_j \frac{\tilde{A}_{ij}}{\tilde{D}_i} x_j$$

- 矩阵形式 $\hat{Y} = X - \gamma \tilde{D}^{-1} \tilde{L} X = (I - \gamma \tilde{D}^{-1} \tilde{L}) X$
- 当 $\gamma = 1$ ，替换 $L^{rw} = \tilde{D}^{-1} \tilde{L}$ 为 $L^{sym} = \tilde{D}^{-1/2} \tilde{L} \tilde{D}^{-1/2}$

$$\begin{aligned} \hat{Y} &= (I - \tilde{D}^{-1/2} \tilde{L} \tilde{D}^{-1/2}) X = (I - \tilde{D}^{-1/2} (\tilde{D} - \tilde{A}) \tilde{D}^{-1/2}) X \\ &= \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X \end{aligned}$$

图卷积本质是一种特殊形式的拉普拉斯平滑

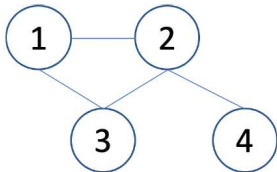
GCN的问题：过平滑

- 假设图是联通的，且不是二分图($\lambda_{\max} < 2$)
- 对任意的 w ，对层数 m 求极值：

$$\lim_{m \rightarrow +\infty} (I - \gamma L^{\text{sym}})^m w = c D^{-1/2} \mathbf{1} = c \sqrt{d}$$

- $I - \gamma L^{\text{sym}}$ 的特征值范围 $(-1, 1]$ ，因此只留下1对应的特征向量 $D^{-1/2} \mathbf{1}$

- 例子



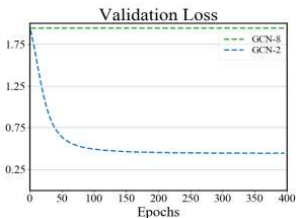
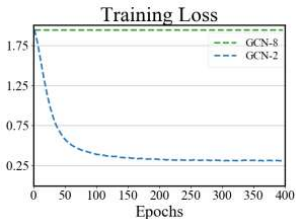
$$\hat{A} = \begin{bmatrix} 0.33333333 & 0.28867513 & 0.33333333 & 0. \\ 0.28867513 & 0.25 & 0.28867513 & 0.35355339 \\ 0.33333333 & 0.28867513 & 0.33333333 & 0. \\ 0. & 0.35355339 & 0. & 0.5 \end{bmatrix}$$

$$\hat{A}^{10} = \begin{bmatrix} 0.2505533 & 0.28847899 & 0.2505533 & 0.20304622 \\ 0.28847899 & 0.33340287 & 0.28847899 & 0.23608437 \\ 0.2505533 & 0.28847899 & 0.2505533 & 0.20304622 \\ 0.20304622 & 0.23608437 & 0.20304622 & 0.16876664 \end{bmatrix}$$

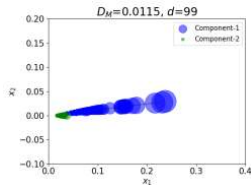
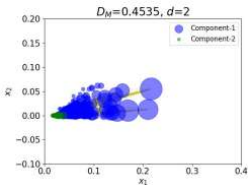
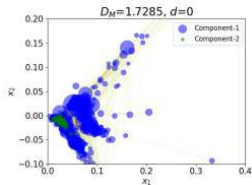
$$\hat{A}^{20} = \begin{bmatrix} 0.25000181 & 0.28867449 & 0.25000181 & 0.20412061 \\ 0.28867449 & 0.33333356 & 0.28867449 & 0.23570351 \\ 0.25000181 & 0.28867449 & 0.25000181 & 0.20412061 \\ 0.20412061 & 0.23570351 & 0.20412061 & 0.16667355 \end{bmatrix}$$

GCN的问题：过平滑

• GNN 过平滑问题



• 随着层数的增多，表征收敛到一个子空间

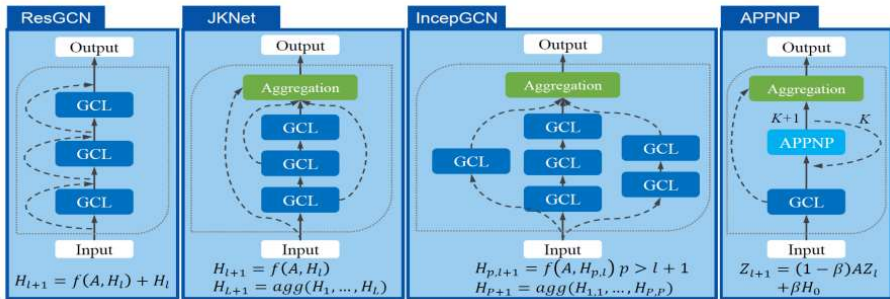


深层GNN—解决过平滑/过拟合

• 解决过平滑过拟合的原因

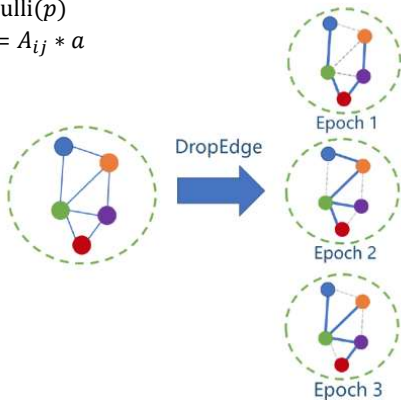
- ResGCN: 残差连接:
- JK-Net: Concatenate, Max pooling, LSTM
- IncepGCN
- APPNP: 增加根节点信息

$$Z^k = (1 - \alpha) \hat{A} Z^{k-1} + \alpha H$$



深层GNN—解决过平滑/过拟合

- DropEdge
 - 随机选择边 $a \sim \text{Bernoulli}(p)$
 - 稀疏化邻接矩阵 $A_{ij} = A_{ij} * a$
 - 类似dropout



深层GNN—解决过平滑/过拟合

- Pairnorm, 类似于BatchNorm
 - 尽可能保持正则化前后任何两个节点特征的总的距离保持不变

$$\text{TPSD}(\tilde{\mathbf{X}}) = \sum_{i,j \in [n]} \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2 = 2n^2 \left(\frac{1}{n} \sum_{i=1}^n \|\tilde{\mathbf{x}}_i\|_2^2 - \left\| \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{x}}_i \right\|_2^2 \right)$$

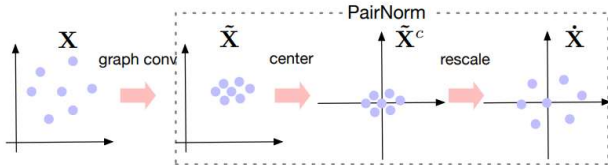
- 对GCN的输出
 - 中心化
 - 再拉伸

Center

$$\tilde{\mathbf{x}}_i^c = \tilde{\mathbf{x}}_i - \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{x}}_i$$

Rescale

$$\dot{\mathbf{x}}_i = s\sqrt{n} \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\|\tilde{\mathbf{X}}^c\|_F^2}}$$



图的池化—DiffPool

- 定义一个端到端的可微分层级池化方法
 - 对 $l-1$ 层的节点进行聚类得到粗化的图
 - 将粗化的图作为下一层 l 的输入网络结构
 - 联合学习GNN和节点聚类过程

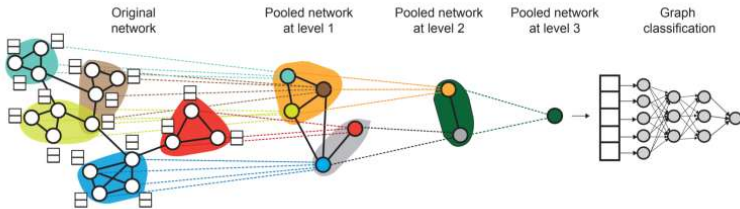


Figure 1: High-level illustration of our proposed method DIFFPOOL. At each hierarchical layer, we run a GNN model to obtain embeddings of nodes. We then use these learned embeddings to cluster nodes together and run another GNN layer on this coarsened graph. This whole process is repeated for L layers and we use the final output representation to classify the graph.

图的池化—DiffPool

• 如何根据聚类结果进行图池化？

- 假设已知 l 层的聚类结果 $\mathbf{S}^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$, 可得到下层 $l+1$ 的池化结果

$$(\mathbf{A}^{(l+1)}, \mathbf{X}^{(l+1)}) = \text{DIFFPOOL}(\mathbf{A}^{(l)}, \mathbf{Z}^{(l)})$$

其中:

$$\begin{aligned} \mathbf{X}^{(l+1)} &= \mathbf{S}^{(l)T} \mathbf{Z}^{(l)} \in \mathbb{R}^{n_l \times d} \\ \mathbf{A}^{(l+1)} &= \mathbf{S}^{(l)T} \mathbf{A}^{(l)} \mathbf{S}^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}} \end{aligned}$$

节点向量表征
邻接矩阵

• 如何学习分配矩阵？

- 对 l 层的输入特征学习节点特征

$$\mathbf{Z}^{(l)} = \text{GNN}_{l, \text{embed}}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)})$$

- 矩阵 $\mathbf{S}^{(l)}$ 用另外的 GNN+softmax 来学习

存在问题:

即时图本身是稀疏的, $\mathbf{S}^{(l)}$ 任然稠密, 可能会耗费大量计算时间、消耗巨大的内存

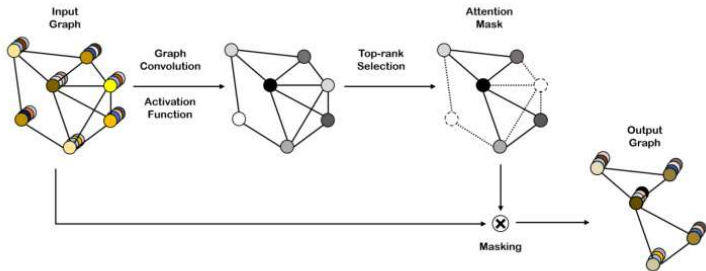
$$\mathbf{S}^{(l)} = \text{softmax}(\text{GNN}_{l, \text{pool}}(\mathbf{A}^{(l)}, \mathbf{X}^{(l)}))$$

图的池化—Top-k池化

- Top-k池化：选出具有代表性K个节点进行粗化，在图中保留这些粗化节点之间的连接，粗化后的图是原来的子图

$$y = \frac{X^{(l)} p^{(l)}}{\|p^{(l)}\|} \quad \text{idx} = \text{rank}(y, k) \quad A^{(l+1)} = A^{(l)}_{\text{idx}, \text{idx}} \quad X^{(l+1)} = X^{(l)}_{\text{idx}} \odot (\sigma(y_{\text{idx}}) \mathbf{1}^T)$$

$p^{(l)}$ 学习变量

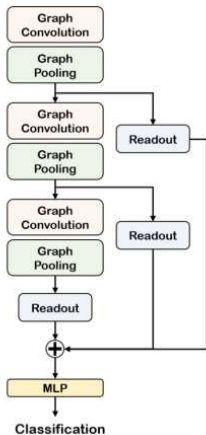


图的池化—SAGPool

- SAGPool:
 - 使用GNN来计算节点的重要性

$$\mathbf{y} = \sigma(\hat{\mathbf{A}}\mathbf{X}^l\boldsymbol{\theta}_{att}) \quad \text{idx} = \text{rank}(\mathbf{y}, k)$$

$$\mathbf{A}^{(l+1)} = \mathbf{A}_{\text{idx}, \text{idx}}^{(l)} \quad \mathbf{X}^{(l+1)} = \mathbf{X}_{\text{idx}}^{(l)} \odot (\mathbf{y}_{\text{idx}} \mathbf{1}^\top)$$



作业

- 给定三个图数据，实现GCN算法用于节点分类和链路预测，分析自环、层数、DropEdge、PairNorm、激活函数等因素的分类性能和链路预测性能的影响
- 请助教准备Cora、Citeseer、PPI的数据



无监督深度学习

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>



为什么无监督学习

- 有监督学习依赖于大量的**标记数据**
- 但获取数据标签开销巨大
- 若没有足够的有标签数据，应该怎么办呢？
- 可以利用无监督深度学习方法，利用大量未标记数据
 - 受限玻尔兹曼机
 - 自编码器



本章内容

- 玻尔兹曼机
 - 玻尔兹曼机 (Boltzmann Machine, BM)
 - 受限玻尔兹曼机 (Restricted Boltzmann Machine, RBM)
 - 深度玻尔兹曼机 (Deep Boltzmann Machine, DBM)
- 自编码器 (AutoEncoder)
 - PCA
 - 降噪自编码器 (Denoising AutoEncoder)
 - 稀疏自编码器 (Sparse AutoEncoder)

半监督学习

标记
数据



猫



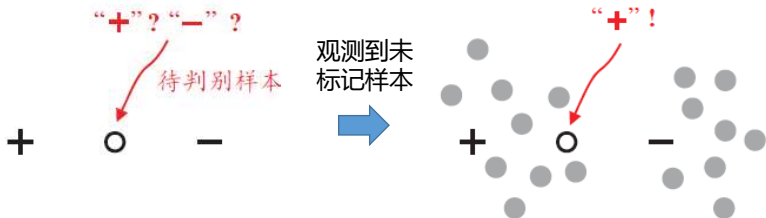
狗

未标记
数据



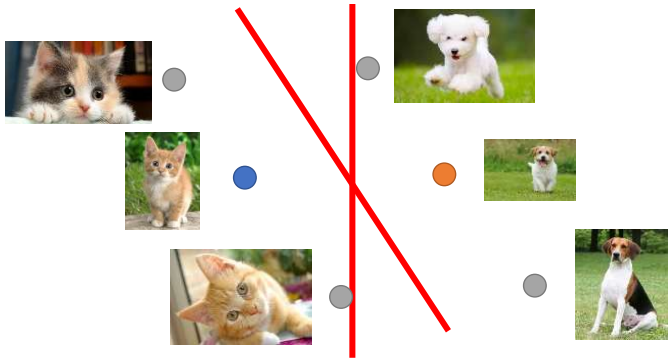
一堆猫和狗的图片，但不知哪些是狗，哪些是猫

半监督学习为何有效



未标记样本的分布能帮助有监督学习

半监督学习为何有效



Self-Taught Learning

标记
数据



猫



狗

未标记
数据



从互联网上抓取的成千上万的图片，但是和任务无关

Self-Taught Learning

标记
数据



手写数字

未标记
数据



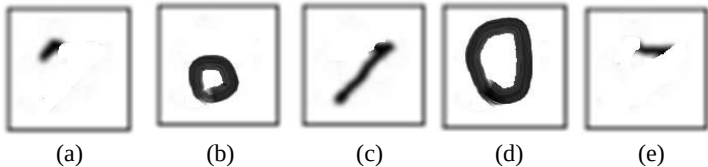
手写字符

隐变量



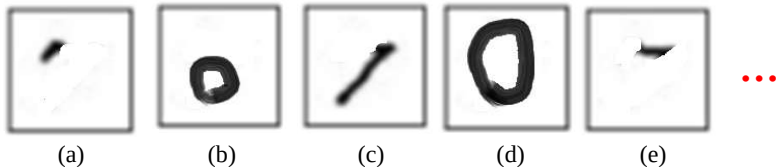
不管是数字还是字符，可能都是由笔画组成

隐变量

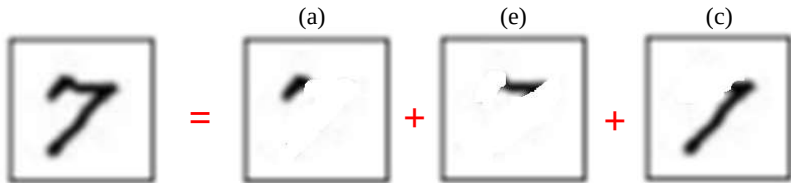


隐变量模型

词典



Sparse coding 或者 NMF



28 × 28的图片,
784维向量

[1 0 1 0 1 0 ...]

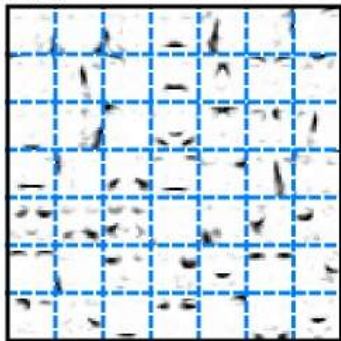
更短, 更简单的表示

隐变量模型

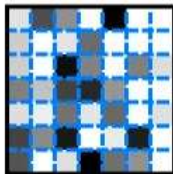
Original



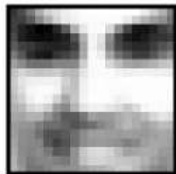
NMF



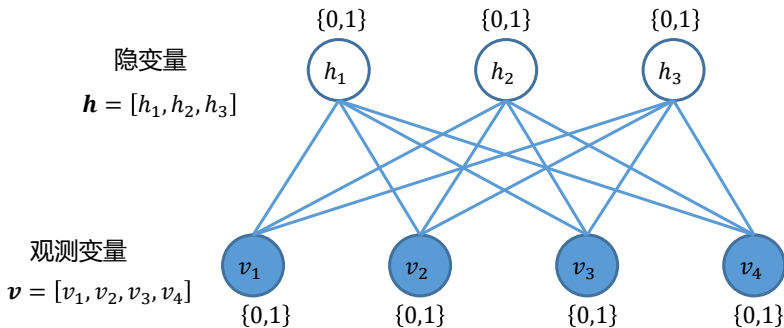
$\times$



$=$



RBM是含隐变量概率图模型



$$P(V = \mathbf{v}, H = \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})) \quad Z \text{ 为归一化系数}$$

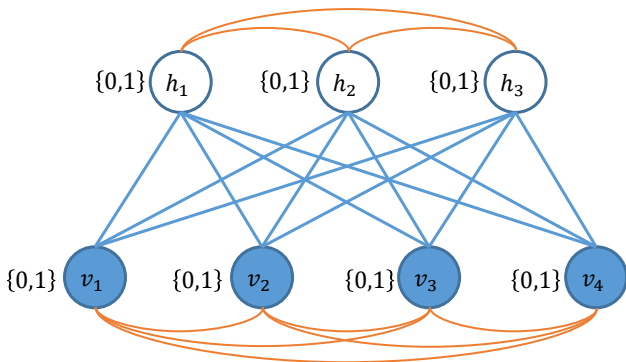
能量函数

$$E(\mathbf{v}, \mathbf{h}) = -(\mathbf{v}^T \mathbf{W} \mathbf{h} + \mathbf{a}^T \mathbf{v} + \mathbf{b}^T \mathbf{h})$$

模型参数

RBM是一种特殊的玻尔兹曼机

玻尔兹曼机



RBM特有的性质

- 在给定可观测变量时, **隐变量**之间相互条件独立

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad \text{其中 } P(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i w_{ij}v_i)$$

- 在给定隐变量时, **可观测变量**之间也相互条件独立

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad \text{其中 } P(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_j w_{ij}h_j)$$

RBM特有的性质

$$\begin{aligned}
 & \bullet \text{证明 } P(h_j = 1 | \mathbf{v}) = \sigma(b_j + \sum_i w_{ij} v_i) \quad E(\mathbf{v}, \mathbf{h}) = -(\mathbf{v}^\top \mathbf{W} \mathbf{h} + \mathbf{a}^\top \mathbf{v} + \mathbf{b}^\top \mathbf{h}) \\
 & \quad P(h_j = 1 | \mathbf{v}) \\
 & = P(h_j = 1 | \mathbf{v}, \mathbf{h}_{-j}) \\
 & = \frac{P(h_j = 1, \mathbf{v}, \mathbf{h}_{-j})}{P(h_j = 1, \mathbf{v}, \mathbf{h}_{-j}) + P(h_j = 0, \mathbf{v}, \mathbf{h}_{-j})} \\
 & = \frac{\exp(-E(h_j = 1, \mathbf{v}, \mathbf{h}_{-j}))}{\exp(-E(h_j = 1, \mathbf{v}, \mathbf{h}_{-j})) + \exp(-E(h_j = 0, \mathbf{v}, \mathbf{h}_{-j}))} \\
 & = \frac{1}{1 + \exp\left(-\left(E(h_j = 0, \mathbf{v}, \mathbf{h}_{-j}) - E(h_j = 1, \mathbf{v}, \mathbf{h}_{-j})\right)\right)} \\
 & = \frac{1}{1 + \exp\left(-\left(\sum_i v_i w_{ij} + b_j\right)\right)} \\
 & \quad E(\mathbf{v}, \mathbf{h}) = -(\mathbf{v}^\top \mathbf{W} \mathbf{h} + \mathbf{a}^\top \mathbf{v} + \mathbf{b}^\top \mathbf{h}) \\
 & \quad = -\sum_{i,j} v_i w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j \\
 & \quad = -\sum_{i,j' \neq j} v_i w_{ij'} h_{j'} - \sum_i a_i v_i - \sum_{j \neq j'} b_{j'} h_{j'} \\
 & \quad \quad - \sum_i v_i w_{ij} h_j - b_j h_j \\
 & \quad E(h_j = 0, \mathbf{v}, \mathbf{h}_{-j}) - E(h_j = 1, \mathbf{v}, \mathbf{h}_{-j}) \\
 & \quad = \sum_i v_i w_{ij} + b_j
 \end{aligned}$$

RBM特有的性质

- 在给定可观测变量时，**隐变量**之间相互条件独立

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad \text{其中 } P(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i w_{ij}v_i)$$

给定 $\mathbf{v}$ ，经过仿射变换用sigmoid得到 $\mathbf{h}$ 的概率分布

- 在给定隐变量时，**可观测变量**之间也相互条件独立

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad \text{其中 } P(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_j w_{ij}h_j)$$

给定 $\mathbf{h}$ ，经过仿射变换用sigmoid得到 $\mathbf{v}$ 的概率分布

RBM的训练 (无监督)

- 给定训练数据 $D = \{\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n\}$
- 通过最大化极大似然函数进行模型参数学习, 即

$$\operatorname{argmax}_{W, a, b} \prod_{k=1}^n P(\mathbf{x}^k)$$

$$\longleftrightarrow \operatorname{argmax}_{W, a, b} \sum_{k=1}^n \log P(\mathbf{x}^k) \approx \operatorname{argmax}_{W, a, b} \mathbb{E}_{\mathbf{x} \sim P_{data}} [\log P(\mathbf{x})]$$

$$\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n \text{ i.i.d. } \sim P_{data} \approx \operatorname{argmax}_{W, a, b} \mathbb{E}_{\mathbf{x} \sim P_{data}} \left[\log \frac{P(\mathbf{x})}{P_{data}(\mathbf{x})} \right]$$

$$\approx \operatorname{argmin}_{W, a, b} KL(P_{data} || P)$$

等价于优化从模型分布 P 到数据分布 P_{data} 的KL散度

RBM的训练 (无监督)

$$\operatorname{argmax}_{W,a,b} \prod_{k=1}^n P(\mathbf{x}^k) \quad \longleftrightarrow \quad \operatorname{argmax}_{W,a,b} \frac{1}{n} \sum_{k=1}^n \log P(\mathbf{x}^k)$$

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h}) = \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))}$$

$$\log P(\mathbf{x}) = \log \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})) - \log \sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))$$

梯度上升法
最大化对数似然

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta \frac{\partial \frac{1}{n} \sum_k \log P(\mathbf{x}^k)}{\partial w_{ij}^{t-1}}$$

如何求这个梯度呢？

RBM的训练 (无监督)

梯度上升法
最大化对数似然

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta \frac{\partial \frac{1}{n} \sum_k \log P(\mathbf{x}^k)}{\partial w_{ij}^{t-1}}$$

$$\log P(\mathbf{x}) = \underbrace{\log \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))}_{\mathbf{A}} - \log \sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))$$

$$\begin{aligned} \frac{\partial \mathbf{A}}{\partial w_{ij}} &= \frac{1}{\sum_{\mathbf{h}'} \exp(-E(\mathbf{x}, \mathbf{h}'))} \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})) \left(-\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial w_{ij}} \right) \\ &= \sum_{\mathbf{h}} \frac{\exp(-E(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{h}'} \exp(-E(\mathbf{x}, \mathbf{h}'))} x_i h_j \\ &= \mathbb{E}_{P(\mathbf{h}|\mathbf{x})} [x_i h_j] \\ &= P(h_j = 1 | \mathbf{x}) x_i \end{aligned}$$

RBM的训练 (无监督)

梯度上升法
最大化对数似然

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta \frac{\partial \frac{1}{n} \sum_k \log P(\mathbf{x}^k)}{\partial w_{ij}^{t-1}}$$

$$\log P(\mathbf{x}) = \log \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})) - \underbrace{\log \sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))}_{\mathbf{B}}$$

$$\frac{\partial \mathbf{B}}{\partial w_{ij}} = \frac{1}{\sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))} \sum_{\mathbf{x}} \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})) \left(-\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial w_{ij}} \right)$$

$$= \sum_{\mathbf{x}} \sum_{\mathbf{h}} \frac{\exp(-E(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))} x_i h_j$$

$$= \mathbb{E}_{P(\mathbf{x}, \mathbf{h})} [x_i h_j]$$

$$= \mathbb{E}_{P(\mathbf{x})} [P(h_j = 1 | \mathbf{x}) x_i]$$

$$\frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} = P(h_j = 1 | \mathbf{x}) x_i - \mathbb{E}_{P(\mathbf{x})} [P(h_j = 1 | \mathbf{x}) x_i]$$

RBM的训练 (无监督)

梯度上升法
最大化对数似然

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta \frac{\partial \frac{1}{n} \sum_k \log P(\mathbf{x}^k)}{\partial w_{ij}^{t-1}}$$

$$\frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} = P(h_j = 1 | \mathbf{x}) x_i - \mathbb{E}_{P(\mathbf{x})}[P(h_j = 1 | \mathbf{x}) x_i]$$

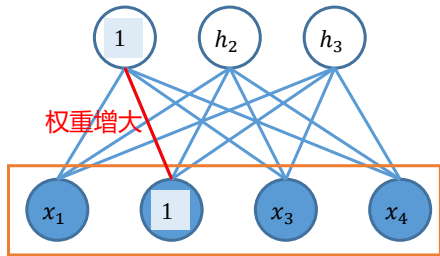
$$\rightarrow w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta \left(\frac{1}{n} \sum_k P(h_j = 1 | \mathbf{x}^k) x_i^k - \mathbb{E}_{P(\mathbf{x})}[P(h_j = 1 | \mathbf{x}) x_i] \right)$$

$$\rightarrow w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta (\mathbb{E}_{P_{data}(\mathbf{x})}[P(h_j = 1 | \mathbf{x}) x_i] - \mathbb{E}_{P(\mathbf{x})}[P(h_j = 1 | \mathbf{x}) x_i])$$

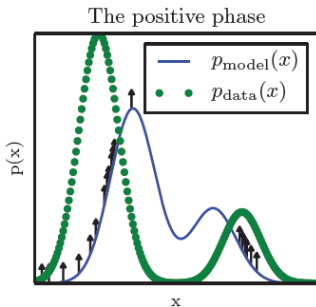
若 $P_{data} = P$, 那么参数不再更新

RBM的训练 (无监督)

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta (\underbrace{\mathbb{E}_{P_{data}(x)}[P(h_j = 1|x)x_i]}_{\text{positive phase}} - \mathbb{E}_{P(x)}[P(h_j = 1|x)x_i])$$



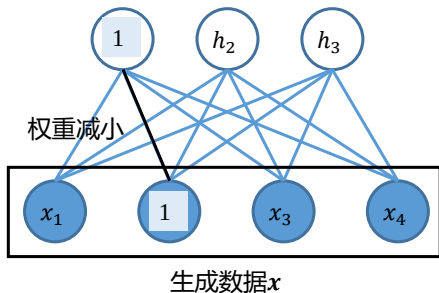
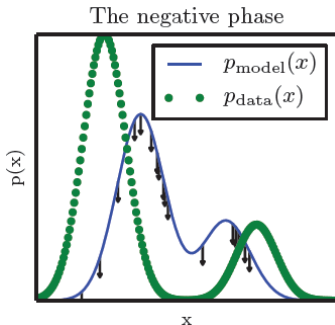
观测数据 x



增大从数据中采样得到的 $\log \tilde{p}(x)$

RBM的训练 (无监督)

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta \left(\mathbb{E}_{P_{data}(x)} [P(h_j = 1|x)x_i] - \underbrace{\mathbb{E}_{P(x)} [P(h_j = 1|x)x_i]}_{\text{The negative phase}} \right)$$



降低从模型中采样得到的 $\log \tilde{p}(x)$

RBM的训练 (无监督)

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta (\mathbb{E}_{P_{data}(x)} [P(h_j = 1|x)x_i] - \underbrace{\mathbb{E}_{P(x)} [P(h_j = 1|x)x_i]})$$

这个期望难以直接计算

可以通过Gibbs采样
来近似该期望

$$\mathbf{x}^1 \sim P(\mathbf{x})$$

$$\mathbf{x}^2 \sim P(\mathbf{x})$$

$$\vdots$$

$$\mathbf{x}^L \sim P(\mathbf{x})$$

$$\approx \frac{1}{L} \sum_l x_i^l P(h_j = 1|x^l)$$

RMB的观测变量概率 $P(\mathbf{x})$

$$\text{记 } \mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$$

$$\begin{aligned} P(\mathbf{x}) &= \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h}) = \sum_{\mathbf{h}} \frac{\exp(-E(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{h}', \mathbf{x}'} \exp(-E(\mathbf{x}', \mathbf{h}'))} \\ &= \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))}{\sum_{\mathbf{x}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{x}', \mathbf{h}'))} \stackrel{\text{def}}{=} \frac{F(\mathbf{x})}{\sum_{\mathbf{x}'} F(\mathbf{x}')} \end{aligned}$$

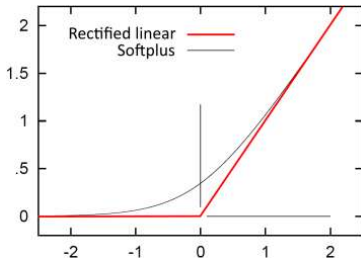
$$\begin{aligned} &= \sum_{\mathbf{h}} \exp\left(\sum_i z_i h_i\right) \\ &= \sum_{\mathbf{h}} \prod_i \exp(z_i h_i) \\ &= \prod_i \left(\sum_{h_i} \exp(z_i h_i)\right) \\ &= \prod_i (1 + \exp(z_i)) \end{aligned}$$

$$\begin{aligned} F(\mathbf{x}) &= \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})) = \sum_{\mathbf{h}} \exp(\mathbf{x}^\top \mathbf{W} \mathbf{h} + \mathbf{a}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}) \\ &= \exp(\mathbf{a}^\top \mathbf{x}) \sum_{\mathbf{h}} \exp((\mathbf{x}^\top \mathbf{W} + \mathbf{b}^\top) \mathbf{h}) \\ &= \exp\left(\mathbf{a}^\top \mathbf{x} + \sum_i \log(1 + \exp(\mathbf{x}^\top \mathbf{W}_{:,i} + b_i))\right) \end{aligned}$$

RMB的观测变量概率 $P(\mathbf{x})$

$$P(\mathbf{x}) = \frac{F(\mathbf{x})}{\sum_{\mathbf{x}'} F(\mathbf{x}')} \quad F(\mathbf{x}) = \exp \left(\mathbf{a}^\top \mathbf{x} + \sum_j \log(1 + \exp(\mathbf{x}^\top \mathbf{W}_{:,j} + b_j)) \right)$$

$$= \exp(\mathbf{a}^\top \mathbf{x} + \mathbf{1}^\top \text{softplus}(\mathbf{W}^\top \mathbf{x} + \mathbf{b}))$$



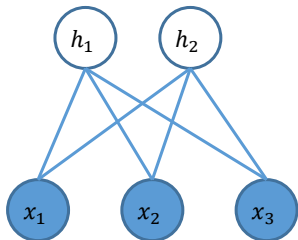
RBM的训练 (无监督)

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta (\mathbb{E}_{P_{data}(x)} [P(h_j = 1|x)x_i] - \mathbb{E}_{P(x)} [P(h_j = 1|x)x_i])$$

$$P(x) \propto \exp(\mathbf{a}^\top \mathbf{x} + \mathbf{1}^\top \text{softplus}(\mathbf{W}^\top \mathbf{x} + \mathbf{b}))$$

这个期望难以直接计算

Gibbs采样—从 $P(x)$ 中采样



随机初始化 x^0

For $l = 1$ to L

$$x_1^l \sim P(x_1 | x_2^{l-1}, x_3^{l-1})$$

$$x_2^l \sim P(x_2 | x_1^l, x_3^{l-1})$$

$$x_3^l \sim P(x_3 | x_1^l, x_2^l)$$

采样得到一个样本 x^l

End

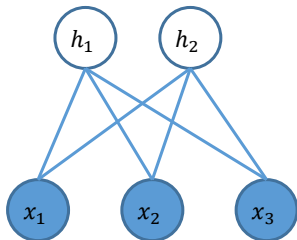
概率计算有点麻烦!!!

RBM的训练 (无监督)

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta (\mathbb{E}_{P_{data}(x)} [P(h_j = 1|x)x_i] - \mathbb{E}_{P(x)} [P(h_j = 1|x)x_i])$$

这个期望难以直接计算

Gibbs采样—从 $P(\mathbf{x}, \mathbf{h})$ 中采样



随机初始化 $\mathbf{x}^0$

For $l = 1$ to L

$$h_1^l \sim P(h_1 | x_1^l, x_2^l, x_3^l, \cancel{h_2^{l-1}})$$

$$h_2^l \sim P(h_2 | x_1^l, x_2^l, x_3^l, \cancel{h_1^{l-1}})$$

$$x_1^l \sim P(x_1 | \cancel{x_2^{l-1}}, \cancel{x_3^{l-1}}, h_1^{l-1}, h_2^{l-1})$$

$$x_2^l \sim P(x_2 | \cancel{x_1^{l-1}}, \cancel{x_3^{l-1}}, h_1^{l-1}, h_2^{l-1})$$

$$x_3^l \sim P(x_3 | \cancel{x_1^{l-1}}, \cancel{x_2^{l-1}}, h_1^{l-1}, h_2^{l-1})$$

采样得到一个样本 $\mathbf{x}^l$

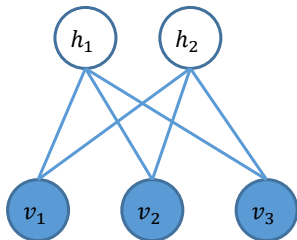
End

RBM的训练 (无监督)

$$w_{ij}^t \leftarrow w_{ij}^{t-1} + \eta (\mathbb{E}_{P_{data}(x)} [P(h_j = 1|x)x_i] - \underbrace{\mathbb{E}_{P(x)} [P(h_j = 1|x)x_i]})$$

这个期望难以直接计算

Gibbs采样—从 $P(\mathbf{x}, \mathbf{h})$ 中采样



随机初始化 $\mathbf{x}^0$

For $l = 1$ to L

$$h_1^l \sim P(h_1 | x_1^{l-1}, x_2^{l-1}, x_3^{l-1})$$

$$h_2^l \sim P(h_2 | x_1^{l-1}, x_2^{l-1}, x_3^{l-1})$$

$$x_1^l \sim P(x_1 | h_1^l, h_2^l) \quad \mathbf{h}^l \sim P(\mathbf{h} | \mathbf{x}^l)$$

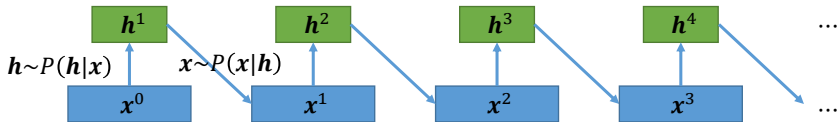
$$x_2^l \sim P(x_2 | h_1^l, h_2^l)$$

$$x_3^l \sim P(x_3 | h_1^l, h_2^l) \quad \mathbf{x}^l \sim P(\mathbf{x} | \mathbf{h}^{l-1})$$

采样得到一个样本 $\mathbf{x}^l$

End

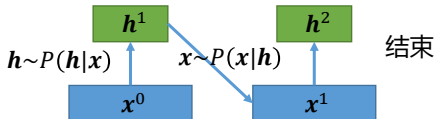
RBM的训练 (无监督)



x^0 一般初始化为某个数据 x

$$\mathbb{E}_{P(x)}[P(h_j = 1|x)x_i] \approx \frac{1}{N} \sum_l^L P(h_j = 1|x^l)x_i^l$$

RBM的训练 (Contrastive Divergence)



$$\mathbb{E}_{P(x)}[P(h_j = 1|x)x_i] \approx P(h_j = 1|x^1)x_i^1$$

RBM的变种

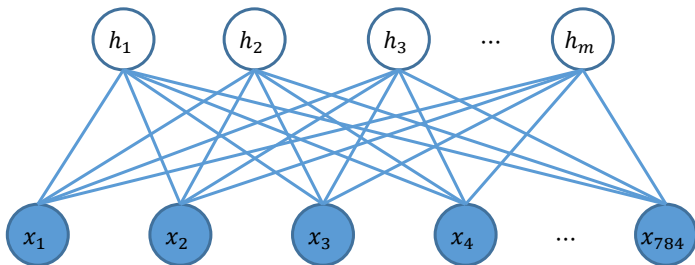
- “高斯-伯努利” 受限玻尔兹曼机
 - 隐变量是离散，但观测变量是连续的

$$E(\mathbf{v}, \mathbf{h}) = - \sum_i \frac{(v_i - \mu_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_i \sum_j \frac{v_i}{\sigma_i} w_{ij} h_j$$

- “伯努利-高斯” 受限玻尔兹曼机
 - 隐变量是连续，但观测变量是离散的

$$E(\mathbf{v}, \mathbf{h}) = - \sum_i a_i v_i - \sum_j \frac{(h_j - \mu_j)^2}{2\sigma_j^2} - \sum_i \sum_j v_i w_{ij} \frac{h_j}{\sigma_j}$$

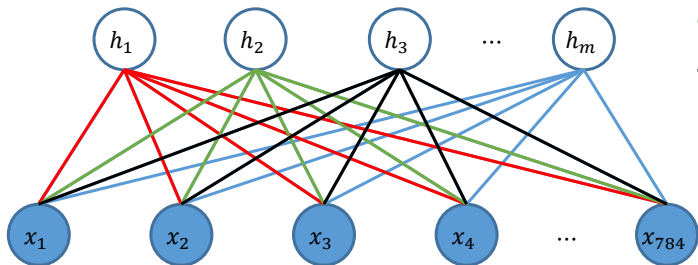
RBM用于手写字符识别



每个观测变量是
图中的每个像素
(黑色还是白色)



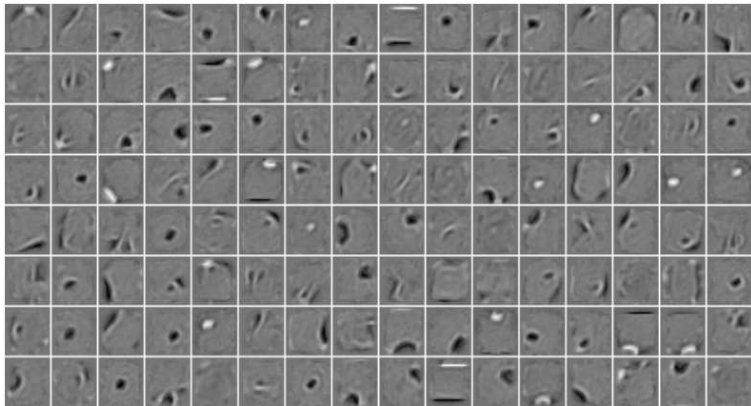
RBM用于手写字符识别



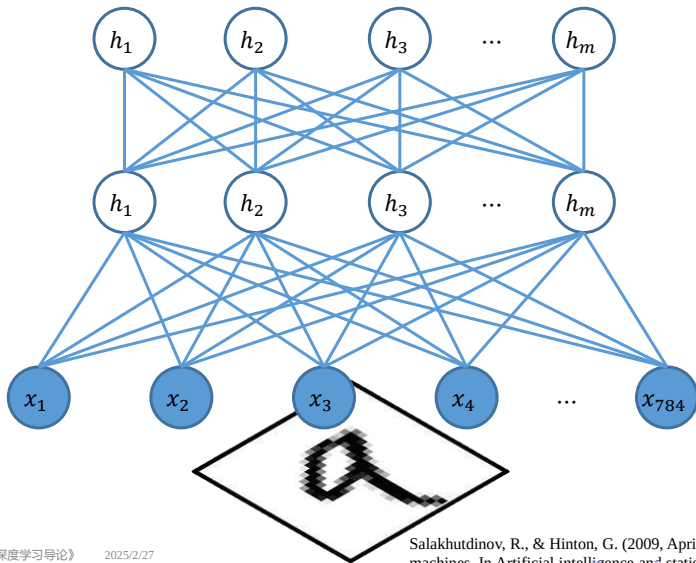
每个观测变量是
图中的每个像素
(黑色还是白色)



RBM用于手写字符识别

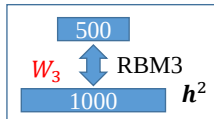
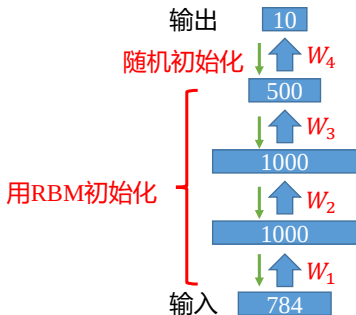


Deep Boltzmann Machines

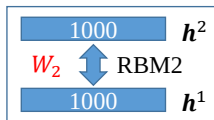


RBM用于预训练

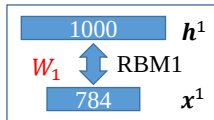
通过后向传播来做微调



从RBM2采样得到 h^2



从RBM1采样得到 h^1



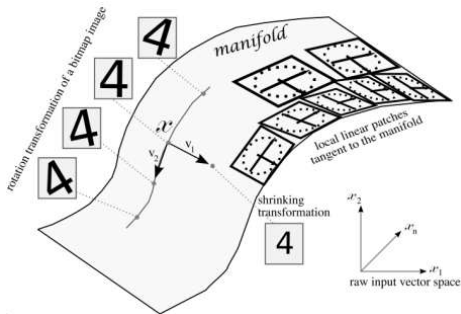
AutoEncoder

- 在MINST中，用 $28 \times 28 = 784$ 维矩阵表示每个数字
- 但不是每个784维的矩阵都表示数字



这些数字图像分布在784维空间内嵌的低维流形空间 (manifold) 上

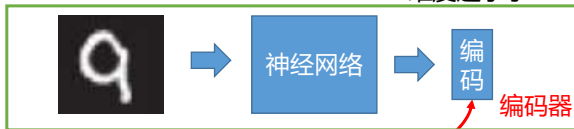
因此可以用更紧凑的向量来表示这些数字图像



AutoEncoder

类似于RBM的 $P(h|x)$

维度远小于784



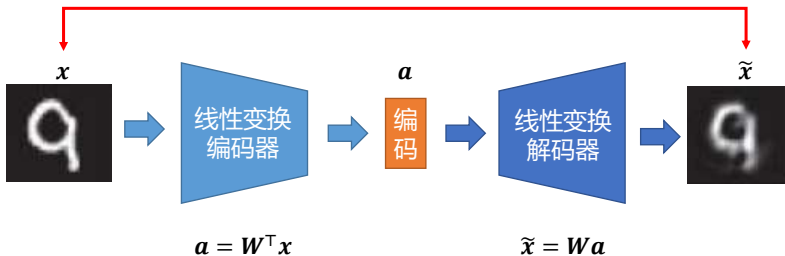
联合训练



RBM还有 $P(x|h)$, 可以从编码重构出原始的对象

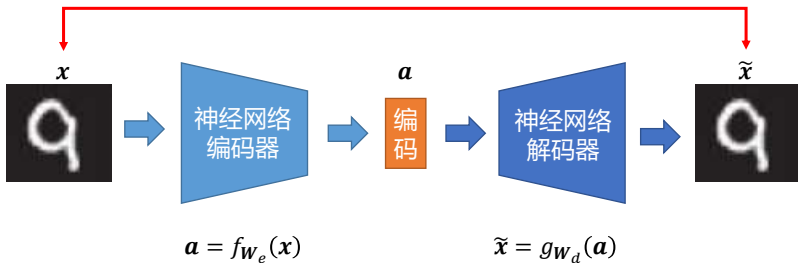
PCA回顾

最小化输入和输出之间的差异 $(x - \tilde{x})^2$



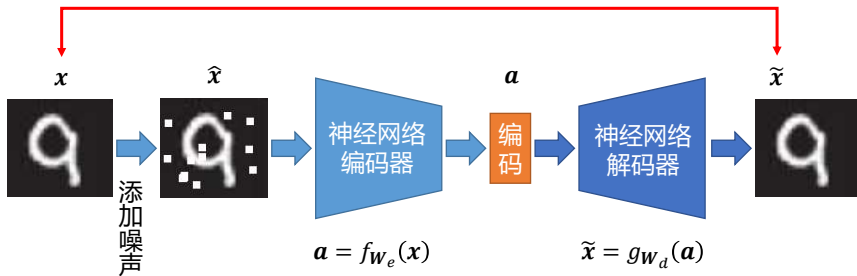
AutoEncoder

最小化输入和输出之间的差异 $(x - \tilde{x})^2$



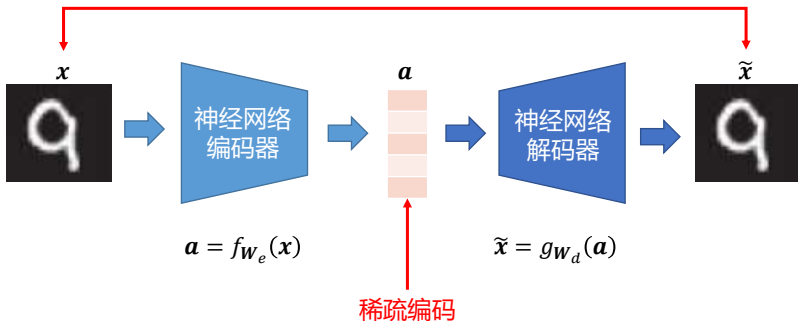
Denoising AutoEncoder

最小化输入和输出之间的差异 $(x - \tilde{x})^2$



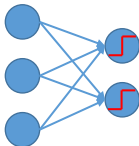
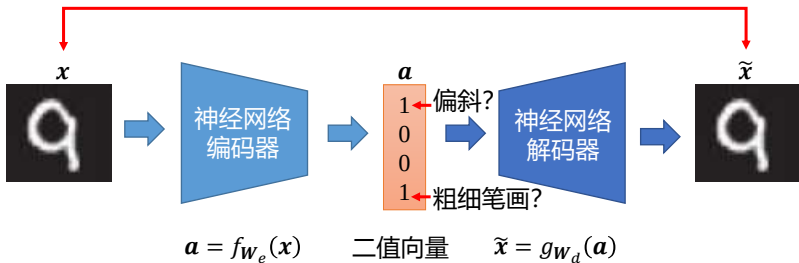
Sparse AutoEncoder

最小化输入和输出之间的差异 $(x - \tilde{x})^2$ + 稀疏约束 $\rho(a)$

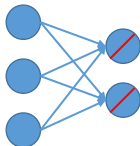


Discrete AutoEncoder

最小化输入和输出之间的差异 $(x - \tilde{x})^2$



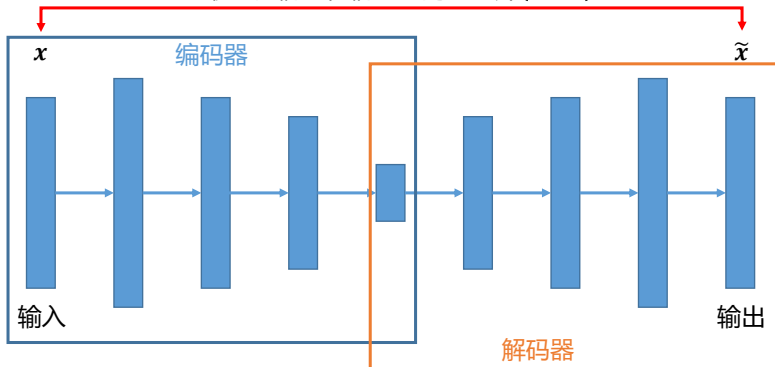
前向



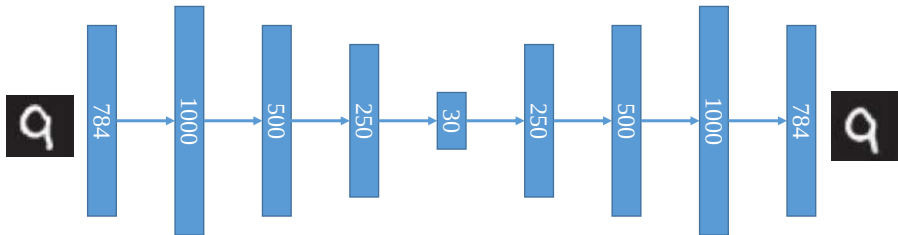
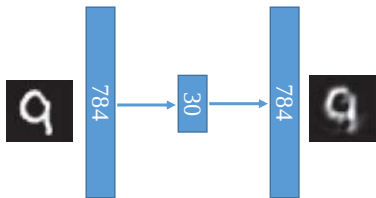
后向

Deep AutoEncoder

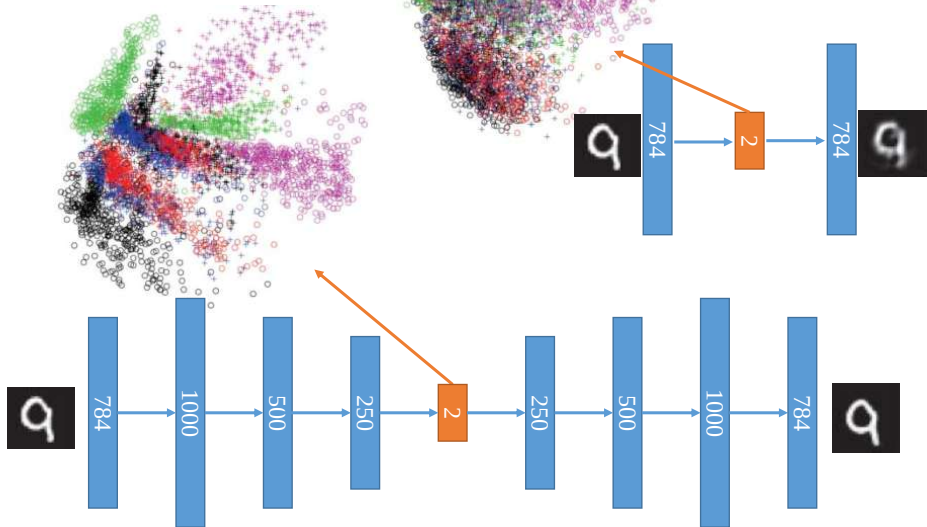
最小化输入和输出之间的差异 $(x - \tilde{x})^2$



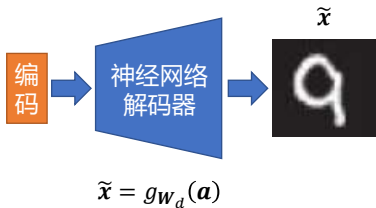
Deep AutoEncoder



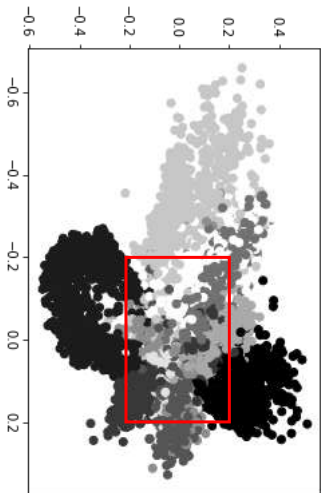
Deep AutoEncoder



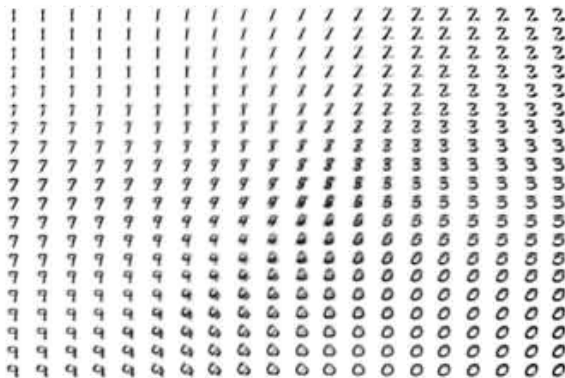
用AutoEncoder生成图片



用AutoEncoder生成图片

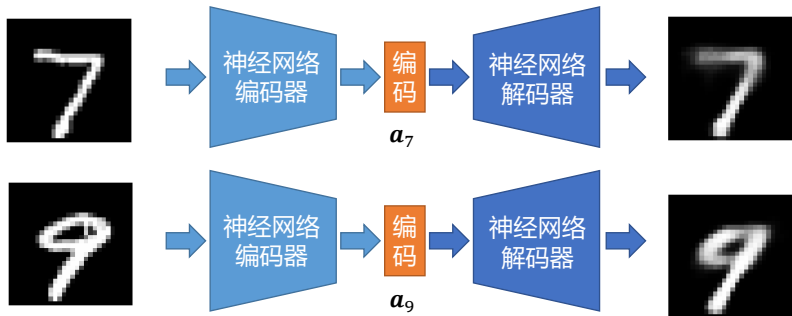


在红色区域内均匀采样400个坐标，作为解码器输入



通过L2正则编码器将图片投影到2维空间

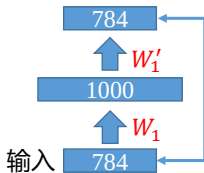
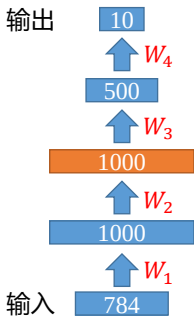
用AutoEncoder生成图片



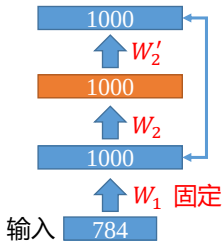
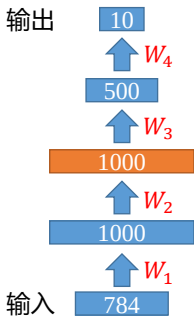
$$a = \alpha a_7 + (1 - \alpha) a_9$$



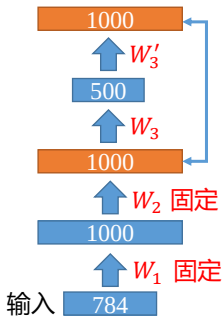
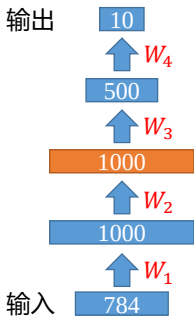
AutoEncoder用于预训练



AutoEncoder用于预训练

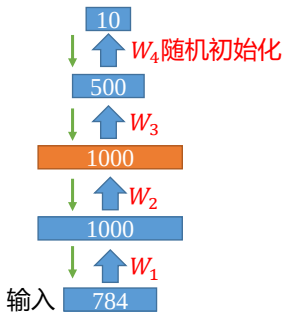
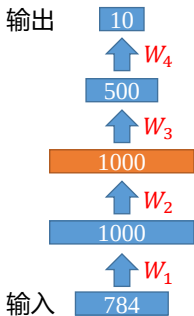


AutoEncoder用于预训练



AutoEncoder用于预训练

通过后向传播来做微调



AutoEncoder用于相似图片检索

dist: 0.0



用迈克杰克逊的头
像进行检索

dist: 3064.2



dist: 3094.1

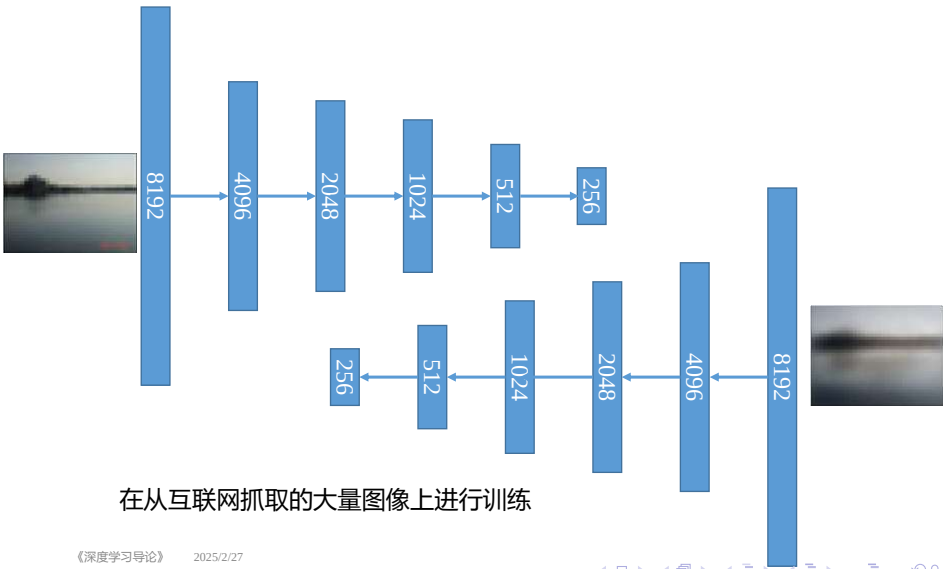


dist: 3132.4



按照图片像素相似
度，最相似的图片

AutoEncoder用于相似图片检索



AutoEncoder用于相似图片检索

dist: 0.0



用迈克杰克逊的头
像进行检索

dist: 61



dist: 64



dist: 65



dist: 66



dist: 67



dist: 67



按照图片的256维
的编码向量，最相
似的图片



谢谢





无监督深度学习 —生成模型 (1)

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>

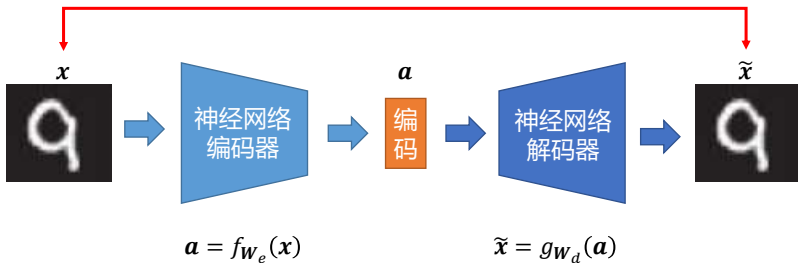


本章内容

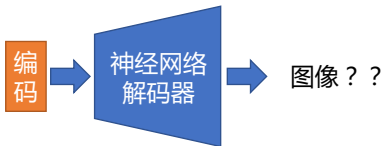
- 变分自编码器 (VAE)
- 对抗生成网络 (GAN)

回顾：AutoEncoder

最小化输入和输出之间的差异 $(x - \tilde{x})^2$

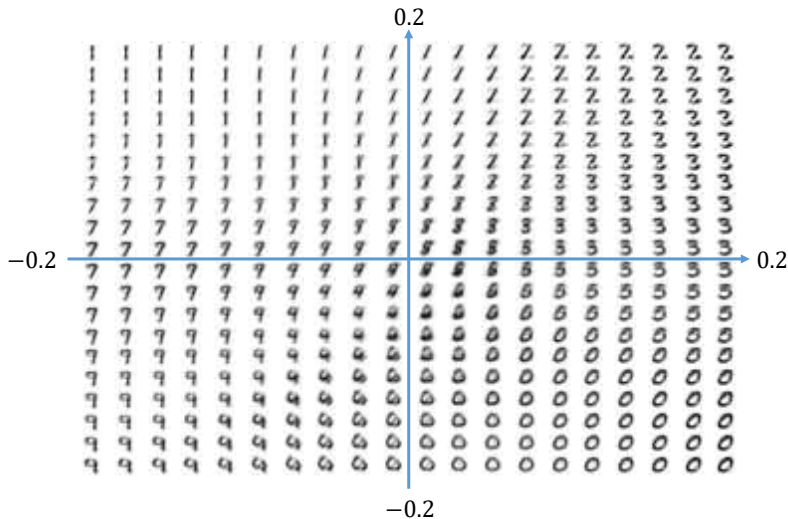


随机生成的向量作为编码



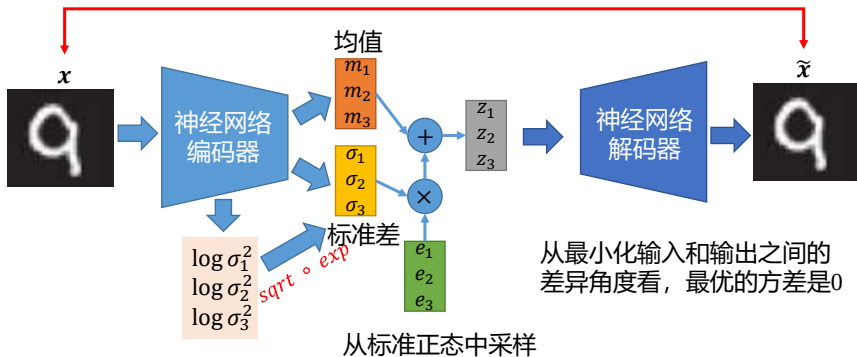


回顾: AutoEncoder



Variational AutoEncoder (VAE)

最小化输入和输出之间的差异 $(x - \tilde{x})^2$



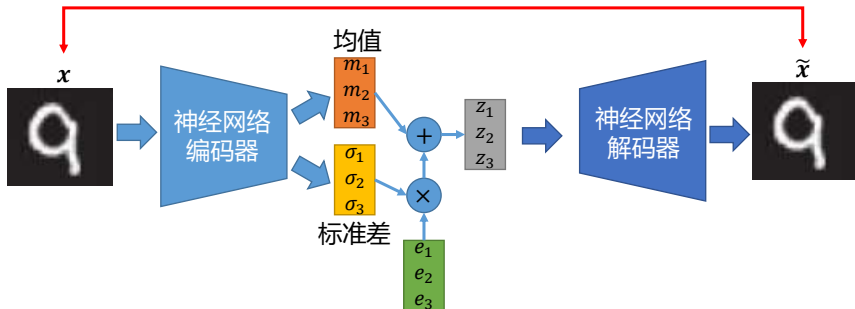


Variational AutoEncoder (VAE)

$$\text{记 } \sigma_i^2 = \exp(\epsilon_i)$$

$$\text{同时最小化 } \sum_{i=1}^3 (\underbrace{\sigma_i^2 - (1 + \log \sigma_i^2)}_{\text{自动学习方差}}) + \underbrace{(m_i)^2}_{\text{正则化, 使其不离0远}}$$

最小化输入和输出之间的差异 $(x - \tilde{x})^2$



从标准正态中采样



Variational AutoEncoder (VAE)

$$\sigma_i^2 = \exp(\epsilon_i)$$

同时最小化 $\sum_{i=1}^3 (\exp(\epsilon_i) - (1 + \epsilon_i) + \frac{(m_i)^2}{2})$

自动学习方差

正则化, 使其不离0远

的差异 $(x - \tilde{x})^2$

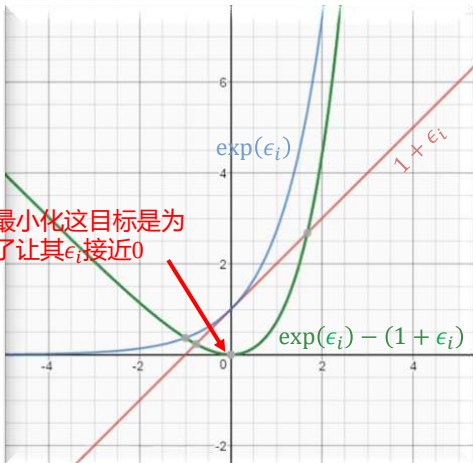
$\tilde{x}$

神经网络
解码器



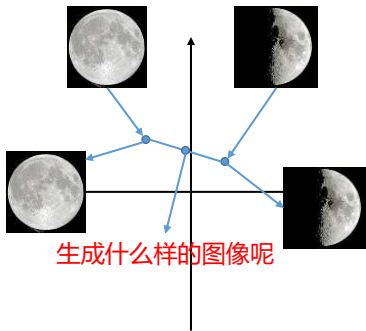
z_1
 z_2
 z_3

中采样



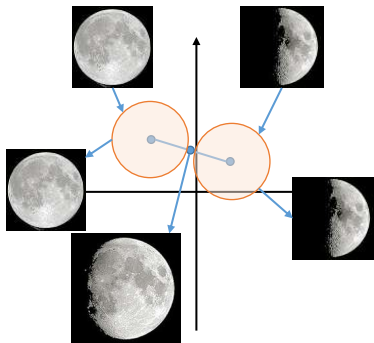
Variational AutoEncoder (VAE)

AutoEncoder



生成什么样的图像呢

Variational AutoEncoder



VAE V.S. AE

编码

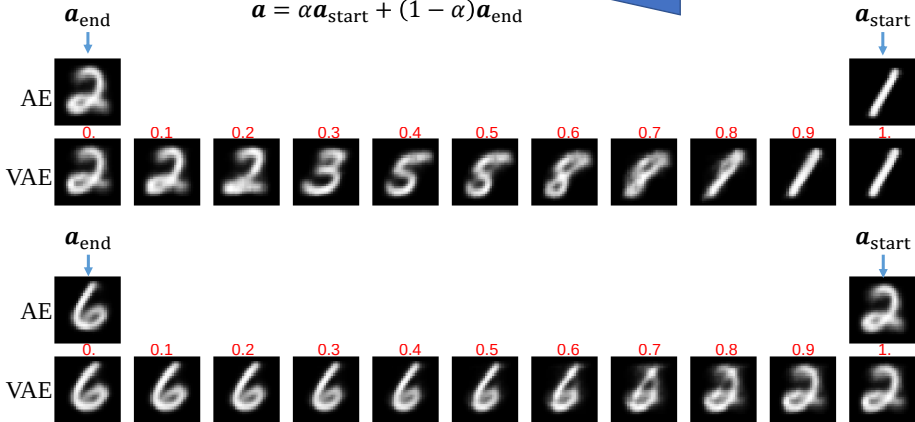


神经网络
解码器



?

$$\mathbf{a} = \alpha \mathbf{a}_{\text{start}} + (1 - \alpha) \mathbf{a}_{\text{end}}$$



高斯混合模型

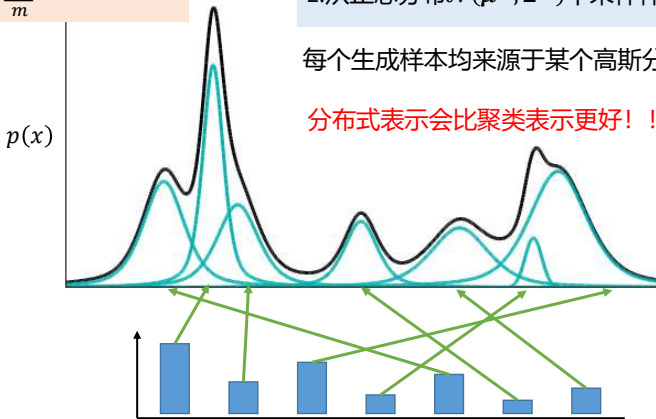
$$p(x) = \sum_m P(m)p(x|m)$$

样本生成过程

1. 先从多项分布中 $P(m)$ 采样一个 m
2. 从正态分布 $\mathcal{N}(\mu^m, \Sigma^m)$ 中采样样本 x

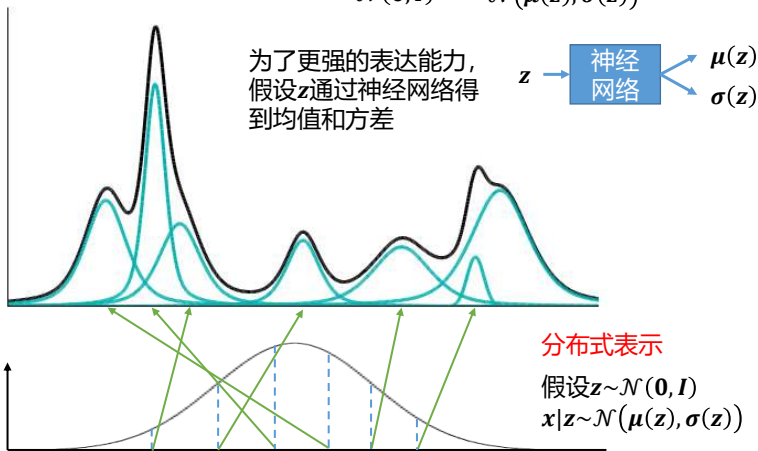
每个生成样本均来源于某个高斯分布

分布式表示会比聚类表示更好!!!



无限高斯混合模型

$$P(x) = \int \underbrace{P(z)}_{\mathcal{N}(0, I)} \underbrace{P(x|z)}_{\mathcal{N}(\mu(z), \sigma(z))} dz$$



VAE最大化似然概率

$$P(x) = \int \underbrace{P(z)}_{\mathcal{N}(\mathbf{0}, I)} \underbrace{P(x|z)}_{\mathcal{N}(\mu(z), \sigma(z))} dz$$



优化目标: $\log P(x) = \log \int P(z)P(x|z)dz$

优化参数: 神经网络中的参数 Θ

当 $Q(z) = P(z|x, \Theta)$ 时等式成立

$$\ln P(x|\Theta) = \ln \int Q(z) \frac{P(x, z|\Theta)}{Q(z)} dz \geq \underbrace{\int Q(z) \ln \frac{P(x, z|\Theta)}{Q(z)} dz}_{\mathcal{L}(Q, \Theta)}$$

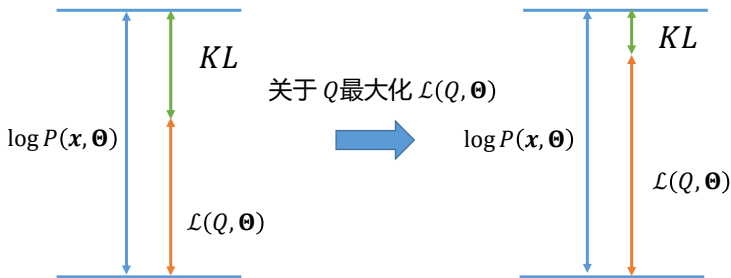
由于 $P(z|x, \Theta)$ 通常难以计算, 假设其为正态分布, 记为 $Q(z|x, \Phi)$

用神经网络近似其参数



VAE最大化似然概率

$$\mathcal{L}(Q, \Theta) = \int Q(z|x, \Phi) \ln \frac{P(x, z|\Theta)}{Q(z|x, \Phi)} = \ln P(x|\Theta) - KL(Q(z|x, \Phi) || P(z|x, \Theta))$$



关于 Q 最大化 $\mathcal{L}(Q, \Theta)$ 会使得 $Q(z|x, \Phi)$ 与 $P(z|x, \Theta)$ 差异减小
即 $Q(z|x, \Phi)$ 是 $P(z|x, \Theta)$ 的一个好的近似

VAE最大化似然概率

$$\begin{aligned}
 \ln P(x|\Theta) &= \ln \int Q(z|x, \Phi) \frac{P(x, z|\Theta)}{Q(z|x, \Phi)} \\
 &\geq \underbrace{\int Q(z|x, \Phi) \ln \frac{P(x, z|\Theta)}{Q(z|x, \Phi)}}_{\mathcal{L}(Q, \Theta)} = \int Q(z|x, \Phi) \ln \frac{P(x, z|\Theta)}{Q(z|x, \Phi)} \\
 &= \int Q(z|x, \Phi) \ln \frac{P(x|z, \Theta)P(z)}{Q(z|x, \Phi)} \\
 &= \int Q(z|x, \Phi) \ln P(x|z, \Theta) + \underbrace{\int Q(z|x, \Phi) \ln \frac{P(z)}{Q(z|x, \Phi)}}_{-KL(Q(z|x, \Phi)||P(z))}
 \end{aligned}$$

VAE最大化似然概率

$$\mathcal{L}(Q, \Theta) = \int Q(\mathbf{z}|\mathbf{x}, \Phi) \ln P(\mathbf{x}|\mathbf{z}, \Theta) - KL(Q(\mathbf{z}|\mathbf{x}, \Phi) || P(\mathbf{z}))$$

$$\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\exp(-1/2(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}}$$

用神经网络近似其参数 $\mathbf{x} \rightarrow$  $\mu(\mathbf{x})$ and $\log \sigma^2(\mathbf{x})$

$$KL(\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) || \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}))$$

$$= \int \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \log \frac{\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2)}{\mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})} \quad \mathbf{z} \text{ 为 } D \text{ 维向量}$$

$$= \int \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) \left(-\frac{1}{2} \log |\text{diag}(\boldsymbol{\sigma}^2)| - \frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^\top \text{diag}(\boldsymbol{\sigma}^2)^{-1} (\mathbf{z} - \boldsymbol{\mu}) + \frac{1}{2} \mathbf{z}^\top \mathbf{z} \right)$$

$$= -\frac{1}{2} \log |\text{diag}(\boldsymbol{\sigma}^2)| - \frac{D}{2} + \frac{1}{2} \text{tr}(\boldsymbol{\mu} \boldsymbol{\mu}^\top + \text{diag}(\boldsymbol{\sigma}^2))$$

$$= -\frac{1}{2} \sum_i \log \sigma_i^2 - \frac{D}{2} + \frac{1}{2} (\sum_i \mu_i^2 + \sum_i \sigma_i^2) \quad \mathbb{E}_{\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})}[(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})]$$

$$= \frac{1}{2} \sum_i (-1 - \log \sigma_i^2 + \mu_i^2 + \sigma_i^2)$$

$$= \text{tr}(\boldsymbol{\Sigma}^{-1} \mathbb{E}_{\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top])$$

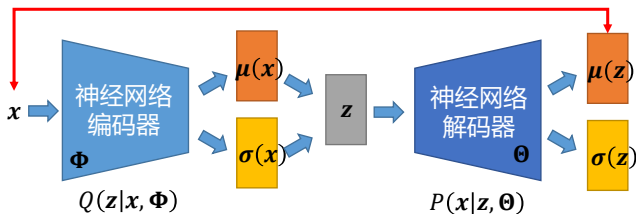
$$= \text{tr}(\boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma})$$

$$= D$$

VAE最大化似然概率

$$\begin{aligned}
 \text{最大化 } \mathcal{L}(Q, \Theta) &= \underbrace{\int Q(z|x, \Phi) \ln P(x|z, \Theta)}_{\mathbb{E}_{Q(z|x, \Phi)} [\ln P(x|z, \Theta)]} - \underbrace{KL(Q(z|x, \Phi) || P(z))}_{\frac{1}{2} \sum_i (-1 - \log \sigma_i^2 + \mu_i^2 + \sigma_i^2)} \\
 &\approx \frac{1}{L} \sum_l \ln P(x|z_l, \Theta) \quad \begin{matrix} z_l \sim Q(z|x, \Phi) \\ \sim \mathcal{N}(\mu(x), \sigma(x)) \end{matrix}
 \end{aligned}$$

最小化输入和输出之间的差异 $(x - \tilde{x})^2$ 同时 **最小化**

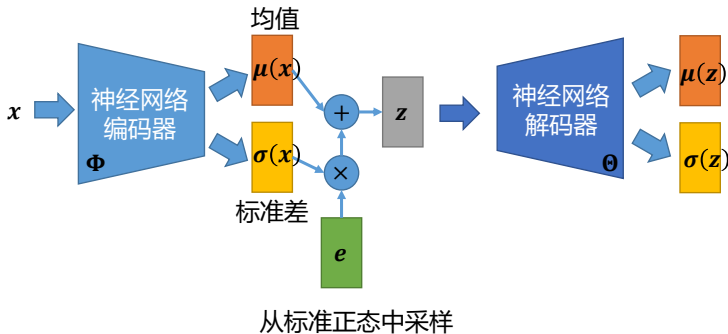


直接从 $\mathcal{N}(\mu(x), \sigma(x))$ 采样编码会导致梯度无法回传

VAE的重采样技术

直接从 $\mathcal{N}(\mu(x), \sigma(x))$ 采样 z 等价于 从 $\mathcal{N}(\mathbf{0}, I)$ 采样 e 后, 运用如下变换

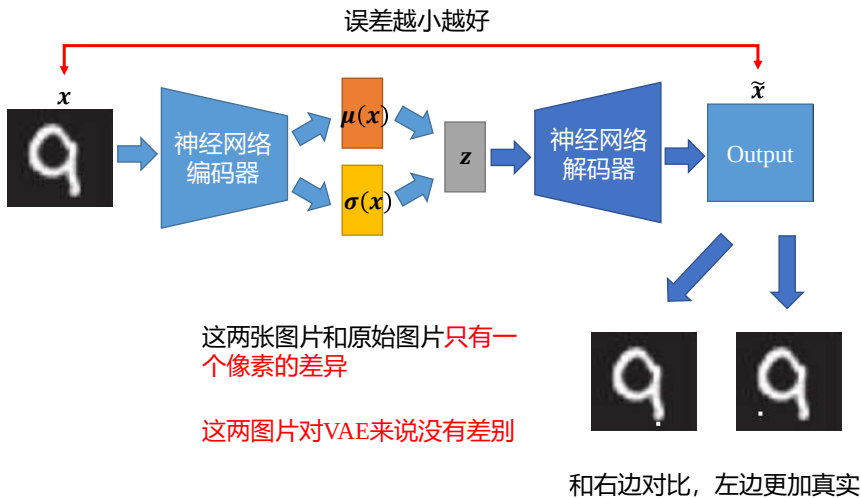
$$z = e \odot \sigma(x) + \mu(x)$$



VAE生成出
来的图像



VAE的问题

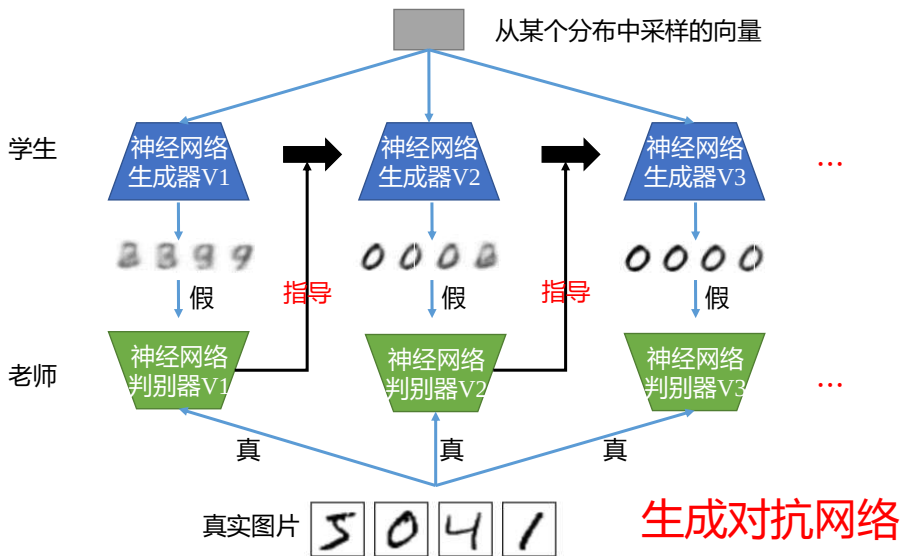


VAE只是记住了数据库里面的图片，并没有学习到如何生成新图片

VAE生成的图片



解码器训练的新思路—引入判别器

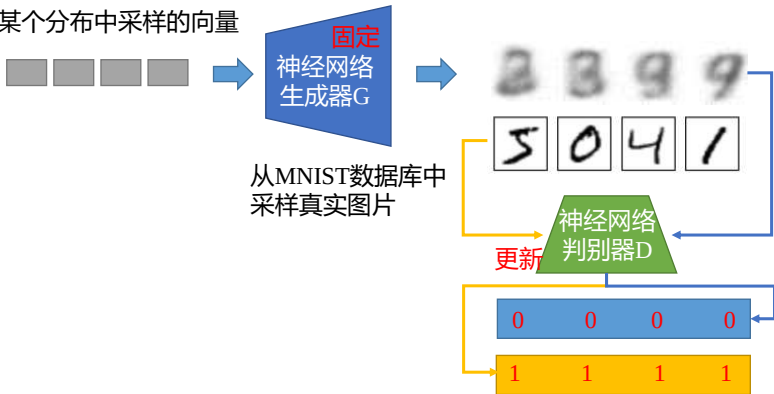


生成对抗网络的算法

- 初始化生成器和判别器
- 在每个训练步
 - 1) 固定生成器，更新判别器D

学会给真实图片高分，给生成图片低分

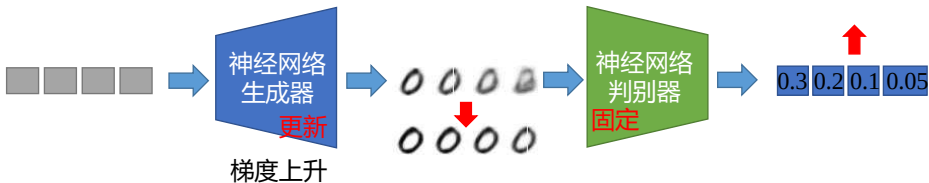
从某个分布中采样的向量



生成对抗网络的算法

- 初始化生成器和判别器
- 在每个训练步
- 1) 固定判别器，更新生成器G

生成得分更高的图片
让判别器更易混淆的图片



生成对抗网络的算法

- 随机初始化判别器参数 θ_d 和生成器参数 θ_g
- While not convergent:

更新D

- 从数据库中采样 m 样本 $\{x^1, x^2, \dots, x^m\}$
- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 用生成器生成样本 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$, $\tilde{x}^i = G(z^i)$
- 更新判别器参数 θ_d , 通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$$

$$\theta_d \leftarrow \theta_d + \eta \nabla \tilde{\mathcal{L}}(\theta_d)$$

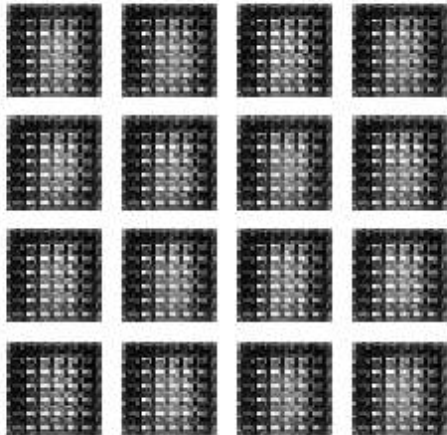
更新G

- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 更新生成器参数 θ_g , 通过最大化如下目标

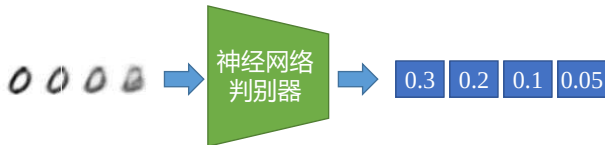
$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m \log (D(G(z^i)))$$

$$\theta_g \leftarrow \theta_g + \eta \nabla \tilde{\mathcal{L}}(\theta_g)$$

生成对抗网络用于MNIST数据集



判别器D的更多理解——假设只有判别器



输入一个对象 x

输出一个标量，代表输入 x 的好坏

能否用判别器生成对象呢？ 答案是可以的！！

判别器D的更多理解—假设只有判别器

- 假设 $D(x)$ 已经训练的很好，那么可以通过如下方式生成图像

$$x^* = \arg \max_{x \in \mathcal{X}} D(x)$$

- 即找到 判别器 打分最高（最真实）的图片
 - 可以枚举所有的 $x \in \mathcal{X}$ 找最高分的 x^*
 - 也可以通过梯度上升法 $x^t \leftarrow x^t + \eta D(x^t)$

但是如何学习判别器呢？

判别器D的更多理解——假设只有判别器

- 在没有生成器的情况下，如何学习判别器呢

- 我们只有真实的图像，如果只用真实数据训练，

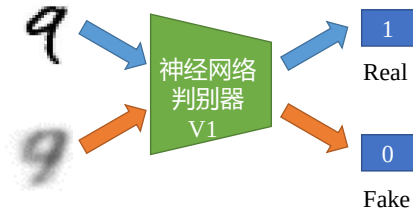
那么判别器会学习到总是输出1



- 因此，判别器学习需要负样本，即不真实样本

判别器D的更多理解—假设只有判别器

- 要训练好判别器，负样本非常关键



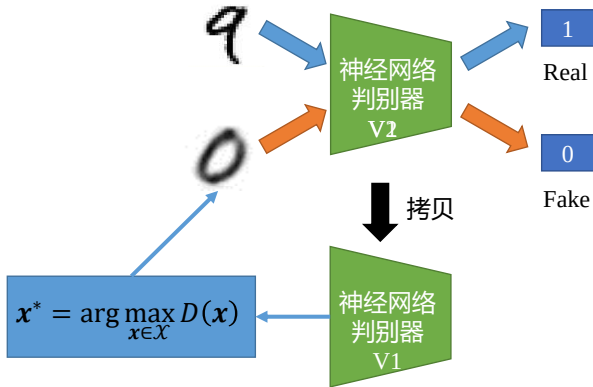
在训练完后，输入



输出这么高的分
是因为训练时负样本太烂了，
使得判别器会误认为已经非常好了

判别器D的更多理解—假设只有判别器

- 要训练好判别器，负样本非常关键



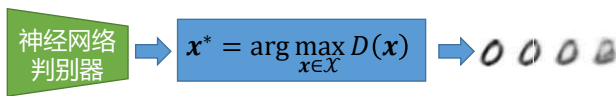
判别器D的更多理解—假设只有判别器

在没有生成器的情况下判别器D的训练算法

- 给定真实样本集，随机生成负样本集
- While not convergent
 - 学习判别器D，使得给正样本高分，负样本低分



- 用判别器D 生成负样本集

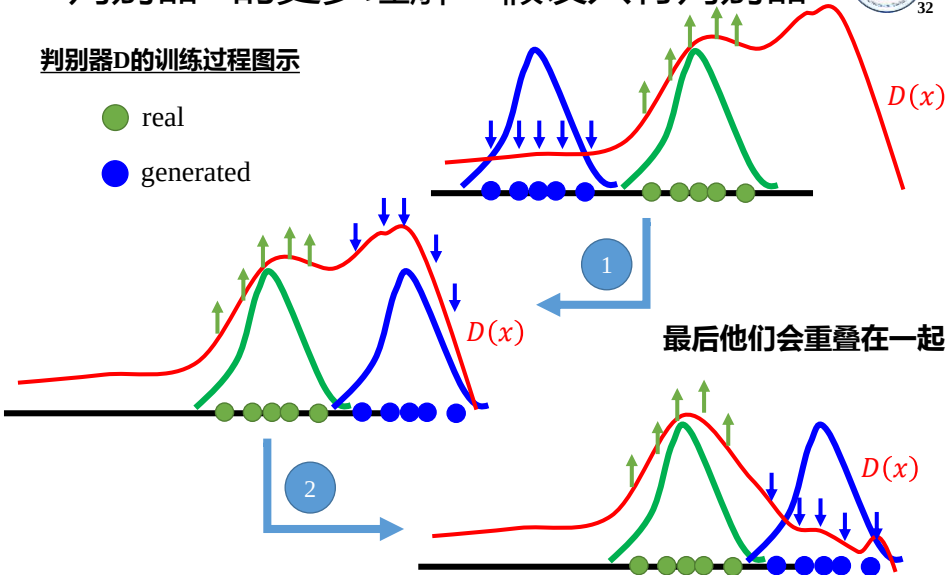


判别器D的更多理解—假设只有判别器

判别器D的训练过程图示

● real

● generated



判别器+生成器

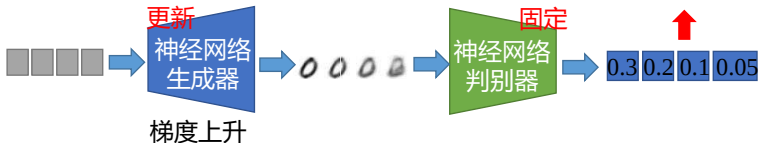
GAN的训练算法

- 给定真实样本集，随机生成负样本集
- While not convergent
 - 学习判别器D，使得给正样本高分，负样本低分



- 用判别器D 生成负样本集

$$x^* = \arg \max_{x \in \mathcal{X}} D(x)$$



GAN的优势

生成器

- 优点
 - 容易生成样本
- 缺点
 - 缺乏全局观

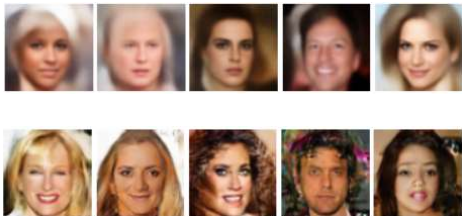
判别器

- 优点
 - 有全局意识
- 缺点
 - 难生成

综合了生成器和判别器的优点，弥补了各自的缺点

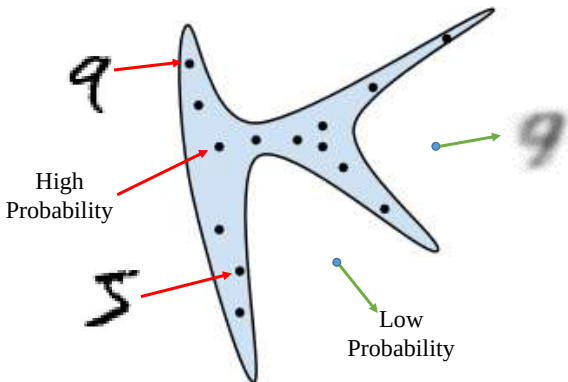
通过生成器快速生成样本，通过判别器提升生成器的全局观

GAN和VAE的生成质量对比



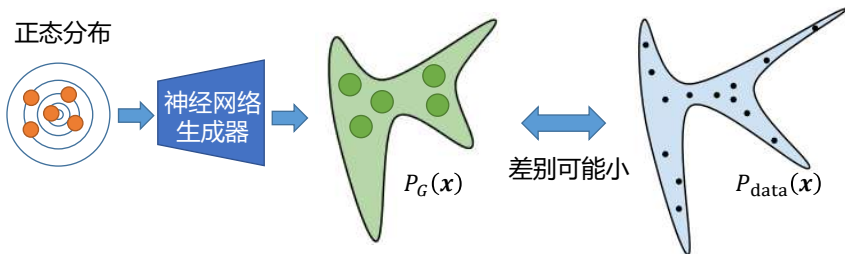
GAN背后的数学

- GAN的目标是找到真实的数据分布



GAN背后的数学

- 生成器G实际上隐式地定义了一个分布 $P_G(x)$ ，因为从生成器中生成样本相当于从分布 $P_G(x)$ 采样



$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$

P_G 和 P_{data} 间的距离，但是应该如何计算呢？

GAN背后的数学

- 虽然不知道 $P_G(x)$ 和 $P_{data}(x)$ 的具体形式，但是都很容易从中采样
- 判别器的训练就是将给正样本（从 $P_{data}(x)$ 采样的样本）高分，给负样本（从 $P_G(x)$ 中采样的）低分，其目标函数如下

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))]$$

- 在给定G的情况下，最优的判别器 $D^* = \arg \max_D V(G, D)$

其中 $\max_D V(G, D)$ 与 P_G 和 P_{data} 间的散度（JS散度）是相关的

GAN背后的数学

★ 从 $P_{\text{data}}(x)$ 采样
★ 从 $P_G(x)$ 采样



39

• 直观上理解

$$D^* = \arg \max_D V(G, D)$$



$P_G(x)$ 和 $P_{\text{data}}(x)$ 差异小



样本更难区分，所以
 $\max_D V(G, D)$ 更小



$P_G(x)$ 和 $P_{\text{data}}(x)$ 差异大



样本很容易区分
 $\max_D V(G, D)$ 更大

GAN背后的数学

$$\begin{aligned} V(G, D) &= E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))] \\ &= \int P_{data}(x) \log D(x) dx + \int P_G(x) \log(1 - D(x)) dx \\ &= \int [P_{data}(x) \log D(x) + P_G(x) \log(1 - D(x))] dx \end{aligned}$$

$$\frac{\partial V(G, D)}{\partial D(x)} = P_{data}(x) \frac{1}{D(x)} + P_G(x) \frac{1}{1 - D(x)} (-1) = 0$$

$$\Rightarrow \frac{1 - D(x)}{D(x)} = \frac{P_G(x)}{P_{data}(x)} \Rightarrow D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

GAN背后的数学

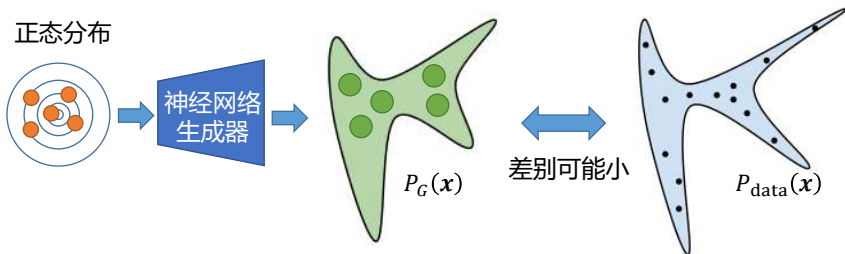
$$M = \frac{1}{2}(P + Q)$$

$$\text{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

$$\begin{aligned} V(G, D^*) &= E_{x \sim P_{data}}[\log D^*(x)] + E_{x \sim P_G}[\log(1 - D^*(x))] \\ &= \int P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} dx + \int P_G(x) \log \left(\frac{P_G(x)}{P_{data}(x) + P_G(x)} \right) dx \\ &= \int P_{data}(x) \log \frac{P_{data}(x)}{1/2(P_{data}(x) + P_G(x))} dx \\ &\quad + \int P_G(x) \log \left(\frac{P_G(x)}{1/2(P_{data}(x) + P_G(x))} \right) dx - 2 \log 2 \\ &= KL \left(P_{data}(x) \parallel \frac{P_{data}(x) + P_G(x)}{2} \right) + KL \left(P_G(x) \parallel \frac{P_{data}(x) + P_G(x)}{2} \right) - 2 \log 2 \\ &= 2\text{JS}(P_{data}(x) \parallel P_G(x)) - 2 \log 2 \end{aligned}$$

GAN背后的数学

- 生成器G实际上隐式地定义了一个分布 $P_G(x)$ ，因为从生成器中生成样本相当于从分布 $P_G(x)$ 采样



$$G^* = \arg \min_G \text{Div}(P_G, P_{data})$$



$$G^* = \arg \min_G \max_D V(G, D)$$

P_G 和 P_{data} 间的距离，但是应该如何计算呢？

GAN背后的数学

$$V(G, D^*) = E_{x \sim P_{data}} [\log D^*(x)] + E_{x \sim P_G} [\log(1 - D^*(x))]$$

GAN的训练算法

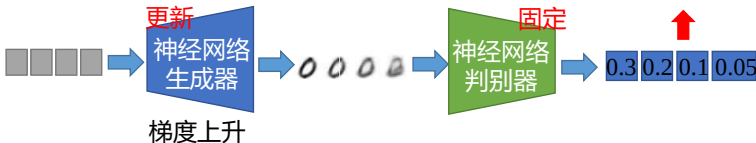
- 给定真实样本集，随机生成负样本集
- While not convergent
 - 学习判别器D，使得给正样本高分，负样本低分

$$D^* = \arg \max_D V(G, D)$$



- 用判别器D生成负样本集

$$G^* = \arg \min_G \max_D V(G, D)$$



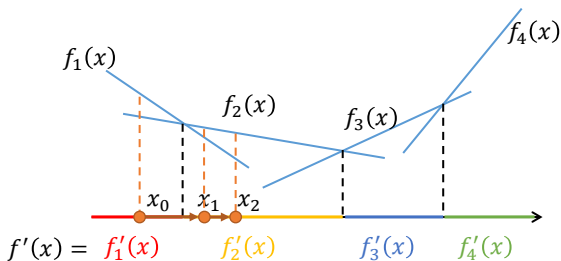
GAN背后的数学

$$G^* = \arg \min_G \max_D V(G, D)$$

- 为了找到最优的 G^* ，可以通过梯度下降法进行优化

$$\theta_G \leftarrow \theta_G - \eta \frac{\partial \left(\max_D V(G, D) \right)}{\partial \theta_G}$$

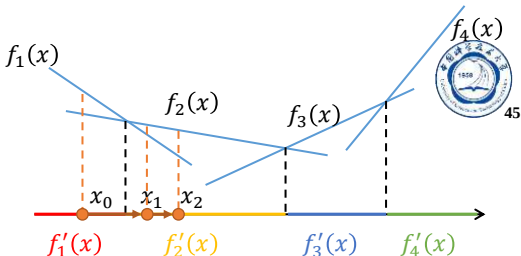
$$f(x) = \max\{f_1(x), f_2(x), f_3(x), f_4(x)\} \quad f'(x) = ?$$



GAN背后的数学

GAN的训练算法

- 初始化 G_0
- 找到 D_0^* 使得 $V(G_0, D)$ 最大化



$V(G_0, D_0^*)$ 是分布 $P_{G_0}(x)$ 和 $P_{data}(x)$ 的JS散度

- $\theta_{G_1} \leftarrow \theta_{G_0} - \eta \frac{\partial V(G_0, D_0^*)}{\partial \theta_G} \rightarrow G_1$ 减小了 $P_{G_0}(x)$ 和 $P_{data}(x)$ 的JS散度
- 找到 D_1^* 使得 $V(G_1, D)$ 最大化

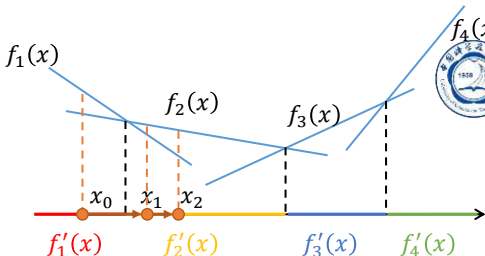
$V(G_1, D_1^*)$ 是分布 $P_{G_1}(x)$ 和 $P_{data}(x)$ 的JS散度

- $\theta_{G_2} \leftarrow \theta_{G_1} - \eta \frac{\partial V(G_1, D_1^*)}{\partial \theta_G} \rightarrow G_2$ 减小了 $P_{G_1}(x)$ 和 $P_{data}(x)$ 的JS散度
- ...

GAN背后的数学

GAN的训练算法

- 初始化 G_0
- 找到 D_0^* 使得 $V(G_0, D)$ 最大化



$V(G_0, D_0^*)$ 是分布 $P_{G_0}(x)$ 和 $P_{data}(x)$ 的JS散度

- $\theta_{G_1} \leftarrow \theta_{G_0} - \eta \frac{\partial V(G_0, D_0^*)}{\partial \theta_G}$ $\rightarrow$ G_1 减小了 $P_{G_0}(x)$ 和 $P_{data}(x)$ 的JS散度
- $\theta_{G_2} \leftarrow \theta_{G_1} - \eta \frac{\partial V(G_1, D_0^*)}{\partial \theta_G}$ $\rightarrow$ G_2 每次 D^* 的求解需要不断迭代直到收敛, 而 G 只更新一次, G 的更新效率太低
- $\theta_{G_3} \leftarrow \theta_{G_2} - \eta \frac{\partial V(G_2, D_0^*)}{\partial \theta_G}$ $\rightarrow$ G_3
- ...

?

只能学习率较小且更新次数不多

否则 $V(G_i, D_0^*)$ 无法准确衡量 $P_{G_i}(x)$ 和 $P_{data}(x)$ 的JS散度

生成对抗网络的算法实现技巧

- 随机初始化判别器参数 θ_d 和生成器参数 θ_g
- While not convergent:

出于效率考虑, 对 D^* 的求解只迭代了 K 步, 因而只得到 $\max_D V(G, D)$ 的下界

更新D

重复
K次

- 从数据库中采样 m 样本 $\{x^1, x^2, \dots, x^m\}$
- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 用生成器生成样本 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$
- 更新判别器参数 θ_d , 通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$$

$$\theta_d \leftarrow \theta_d + \eta \nabla \tilde{\mathcal{L}}(\theta_d)$$

更新G

只做
一次

- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 更新生成器参数 θ_g , 通过最小化如下目标

$$\tilde{\mathcal{L}} = \cancel{\frac{1}{m} \sum_{i=1}^m \log D(x^i)} + \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^i)))$$

$$\theta_g \leftarrow \theta_g - \eta \nabla \tilde{\mathcal{L}}(\theta_g)$$

生成对抗网络的算法实现技巧

在更新G的时候，最小化如下目标

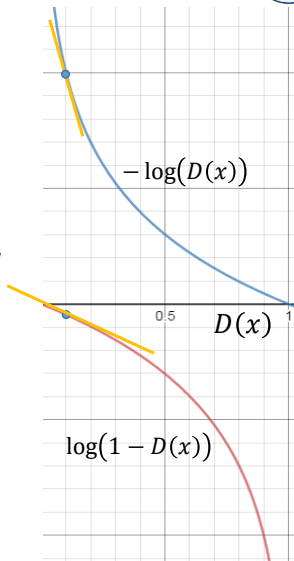
$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^i)))$$

但是在初始时候会很慢，因为G比较差，生成的样本很容易被D分辨为假的，也就是说 $D(G(z^i))$ 是趋于0的， $\log(1 - D(G(z^i)))$ 的梯度很小

在更新G的时候，最小化如下目标

$$\tilde{\mathcal{L}} = -\frac{1}{m} \sum_{i=1}^m \log(D(G(z^i)))$$

Log D技巧



生成对抗网络的算法实现技巧

- 随机初始化判别器参数 θ_d 和生成器参数 θ_g
- While not convergent:

出于效率考虑, 对 D^* 的求解只迭代了 K 步, 因而只得到 $\max_D V(G, D)$ 的下界

更新D

重复
K次

- 从数据库中采样 m 样本 $\{x^1, x^2, \dots, x^m\}$
- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 用生成器生成样本 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$
- 更新判别器参数 θ_d , 通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log (1 - D(\tilde{x}^i))$$

$$\theta_d \leftarrow \theta_d + \eta \nabla \tilde{\mathcal{L}}(\theta_d)$$

更新G

只做
一次

- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 更新生成器参数 θ_g , 通过最小化如下目标

$$\tilde{\mathcal{L}} = -\frac{1}{m} \sum_{i=1}^m \log (D(G(z^i)))$$

$$\theta_g \leftarrow \theta_g - \eta \nabla \tilde{\mathcal{L}}(\theta_g)$$

GAN背后的数学

- 不用Log D技巧，生成网络的目标函数为

$$V(G, D^*) = 2\text{JS}(P_{data}(\mathbf{x})||P_G(\mathbf{x})) - 2\log 2$$

- 但用Log D技巧时，生成网络的目标函数为

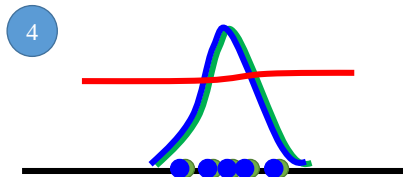
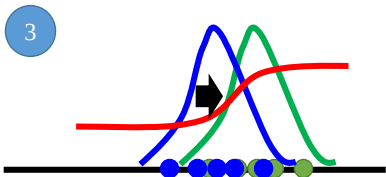
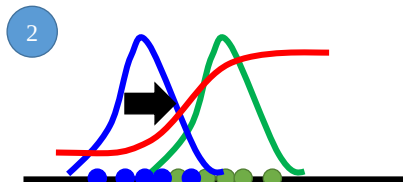
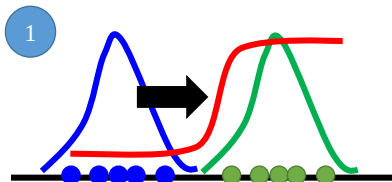
$$V(G, D^*) \propto \text{KL}(P_G(\mathbf{x})||P_{data}(\mathbf{x})) - 2\text{JS}(P_{data}(\mathbf{x})||P_G(\mathbf{x}))$$

同时最小化生成分布与真实分布的KL散度，并最大化两者的JS散度，

这和直觉相冲突，在数值上则会导致梯度不稳定

GAN训练的直观理解

P_{data}
 P_G
 D





谢谢





无监督深度学习 —生成模型 (2)

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>



本章内容

- f-GAN
- WGAN
- Conditional GAN
- Cycle-GAN

对抗生成网络 (GAN)

$$V(G, D) = E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_G} [\log(1 - D(x))]$$

GAN的训练算法

- 给定真实样本集，随机生成负样本集
- While not convergent
 - 学习判别器D，使得给正样本高分，负样本低分

$$D^* = \arg \max_D V(G, D)$$

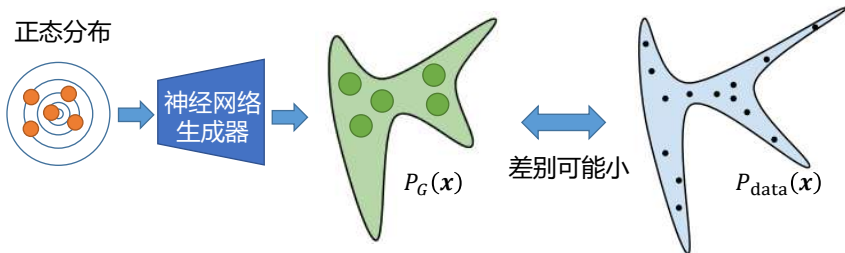


- 用判别器D生成负样本集

$$G^* = \arg \min_G V(G, D^*)$$



对抗生成网络 (GAN)



$$V(G, D) = E_{\mathbf{x} \sim P_{data}} [\log D(\mathbf{x})] + E_{\mathbf{x} \sim P_G} [\log(1 - D(\mathbf{x}))]$$

$$G^* = \arg \min_G \max_D V(G, D)$$

$$V(G, D^*) = 2\text{JS}(P_{data}(\mathbf{x}) || P_G(\mathbf{x})) - 2 \log 2$$

对抗生成网络 (GAN)

- 能否用其他的概率间的度量呢？比如KL, Reverse KL等
- f-GAN提供了一个统一的框架，可以使用除了JS散度外很多度量

Name	$D_f(P Q)$	Generator $f(u)$
Total variation	$\frac{1}{2} \int p(x) - q(x) dx$	$\frac{1}{2} u - 1 $
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$
Reverse Kullback-Leibler	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} dx$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)} dx$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int (\sqrt{p(x)} - \sqrt{q(x)})^2 dx$	$(\sqrt{u} - 1)^2$
Jeffrey	$\int (p(x) - q(x)) \log \left(\frac{p(x)}{q(x)} \right) dx$	$(u-1) \log u$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$-(u+1) \log \frac{1+u}{2} + u \log u$
Jensen-Shannon-weighted	$\int p(x) \pi \log \frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi)q(x) \log \frac{q(x)}{\pi p(x)+(1-\pi)q(x)} dx$	$\pi u \log u - (1-\pi+\pi u) \log(1-\pi+\pi u)$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx - \log(4)$	$u \log u - (u+1) \log(u+1)$

$$D_f(P_{data}||P_G) = \max_D \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [f^*(D(x))] \}$$

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1 - \log(-t)$
Pearson χ^2	$\frac{1}{4}t^2 + t$
Neyman χ^2	$2 - 2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$
Jensen-Shannon	$-\log(2 - \exp(t))$
Jensen-Shannon-weighted	$(1-\pi) \log \frac{1-\pi}{1-\pi \exp(t)}$
GAN	$-\log(1 - \exp(t))$

f-divergence

f-divergence的定义

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx \quad f(\cdot) \text{是凸函数}, f(1) = 0$$

若对于任意的 x , 都有 $p(x) = q(x)$

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx = \int q(x) f(1) dx = 0$$

由Jensen 不等式

$$\begin{aligned} &\geq f\left(\int q(x) \frac{p(x)}{q(x)} dx\right) \\ &= f(1) \\ &= 0 \end{aligned}$$



f-divergence

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

- $f(x) = x \log x$

$$D_f(P||Q) = \int q(x) \frac{p(x)}{q(x)} \log\left(\frac{p(x)}{q(x)}\right) dx = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \quad \text{KL散度}$$

- $f(x) = -\log x$

$$D_f(P||Q) = \int q(x) -\log\left(\frac{p(x)}{q(x)}\right) dx = \int q(x) \log\left(\frac{q(x)}{p(x)}\right) dx \quad \text{Reverse KL散度}$$

- $f(x) = -(x+1) \log \frac{x+1}{2} + x \log x$ JS散度

$$\begin{aligned} D_f(P||Q) &= - \int (p(x) + q(x)) \log \frac{p(x) + q(x)}{2q(x)} dx + \int p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \\ &= \int q(x) \log \frac{2q(x)}{p(x) + q(x)} dx + \int p(x) \log \left(\frac{2p(x)}{p(x) + q(x)} \right) dx \\ &= 2 \text{JS}(p(x)||q(x)) \end{aligned}$$



f-divergence

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

- $f(x) = x \log x - (x + 1) \log(x + 1)$

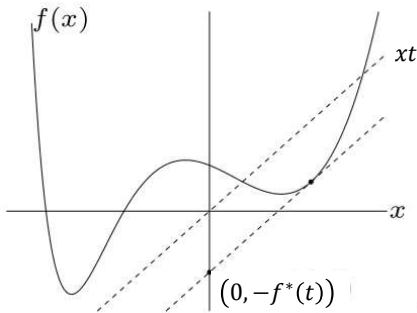
$$\begin{aligned} D_f(P||Q) &= - \int (p(x) + q(x)) \log \frac{p(x) + q(x)}{q(x)} dx + \int p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \\ &= \int q(x) \log \frac{q(x)}{p(x) + q(x)} dx + \int p(x) \log \left(\frac{p(x)}{p(x) + q(x)} \right) dx \\ &= \int q(x) \log \frac{q(x)}{(p(x) + q(x))/2} dx + \int p(x) \log \left(\frac{p(x)}{(p(x) + q(x))/2} \right) dx - 2 \log 2 \\ &= 2 \text{JS}(p(x)||q(x)) - 2 \log 2 \end{aligned}$$

GAN使用的散度

共轭函数

- 每个函数 f 都有一个共轭函数 f^*

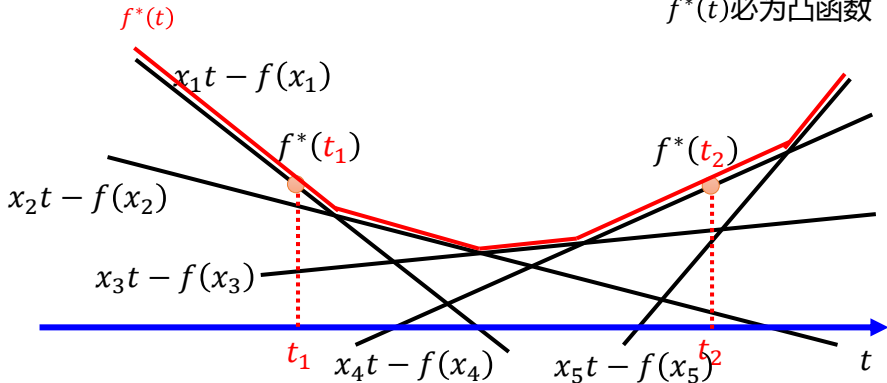
$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$



共轭函数

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$f^*(t)$ 必为凸函数



共轭函数

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f(x) = x \log x \quad \longleftrightarrow \quad f^*(t) = \exp(t - 1) \quad t \in \mathbb{R}$$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - x \log x\}$$

$$\text{令 } g(x) = xt - x \log x$$

$$\text{计算导数} \quad \rightarrow \quad g'(x) = t - 1 - \log x$$

$$\text{令导数等于0} \quad \rightarrow \quad x = \exp(t - 1)$$

$$\text{代入 } g(x) \quad \rightarrow \quad \max_{x \in \text{dom}(f)} g(x) = \exp(t - 1)$$

共轭函数

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f(x) = -\log x \quad \longleftrightarrow \quad f^*(t) = -1 - \log(-t) \quad t < 0$$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt + \log x\}$$

$$\text{令 } g(x) = xt + \log x$$

$$\text{计算导数} \quad \rightarrow \quad g'(x) = t + 1/x$$

$$\text{令导数等于0} \quad \rightarrow \quad x = -1/t$$

$$\text{代入 } g(x) \quad \rightarrow \quad \max_{x \in \text{dom}(f)} g(x) = -1 - \log(-t)$$

共轭函数

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f(x) = -(x+1) \log \frac{x+1}{2} + x \log x \quad \longleftrightarrow \quad f^*(t) = -\log(2 - \exp(t))$$

$$t < \log 2$$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt + (x+1) \log(x+1)/2 - x \log x\}$$

$$\text{令 } g(x) = xt + (x+1) \log(x+1)/2 - x \log x$$

$$\text{计算导数} \quad \rightarrow \quad g'(x) = t + \log(x+1)/2 - \log x$$

$$\text{令导数等于0} \quad \rightarrow \quad x = \frac{1}{2e^{-t} - 1}$$

$$\text{代入 } g(x) \quad \rightarrow \quad \max_{x \in \text{dom}(f)} g(x) = -t - \log(2e^{-t} - 1) = -\log(2 - \exp(t))$$

共轭函数

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f(x) = -(x+1)\log(x+1) + x\log x \iff f^*(t) = -\log(1 - \exp(t))$$

$$t < 0$$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt + (x+1)\log(x+1) - x\log x\}$$

$$\text{令 } g(x) = xt + (x+1)\log(x+1) - x\log x$$

$$\text{计算导数} \implies g'(x) = t + \log(x+1) - \log x$$

$$\text{令导数等于0} \implies x = \frac{1}{e^{-t} - 1}$$

$$\text{代入 } g(x) \implies \max_{x \in \text{dom}(f)} g(x) = -t - \log(e^{-t} - 1) = -\log(1 - \exp(t))$$

f-GAN

$$f^{**} = f$$

$$f * (t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\} \quad \longleftrightarrow \quad f(x) = \max_{t \in \text{dom}(f^*)} \{xt - f^*(t)\}$$

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$$

$$= \int q(x) \max_{t \in \text{dom}(f^*)} \left\{ \frac{p(x)}{q(x)} t - f^*(t) \right\} dx$$

引入函数 D

$$D: x \mapsto \text{dom}(f^*)$$

$$\geq \max_D \int q(x) \left\{ \frac{p(x)}{q(x)} D(x) - f^*(D(x)) \right\} dx$$

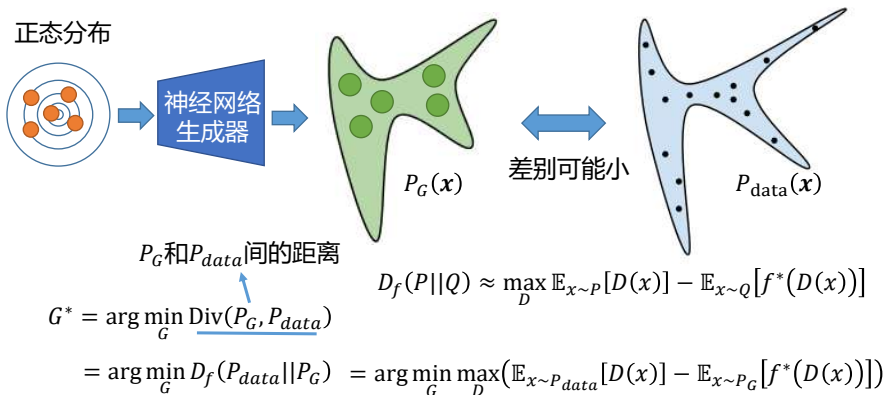
在一些情况下等式可以成立

$$= \max_D \int p(x) D(x) - q(x) f^*(D(x)) dx$$

$$= \max_D \mathbb{E}_{x \sim P}[D(x)] - \mathbb{E}_{x \sim Q}[f^*(D(x))]$$

f-GAN

- 生成器G实际上隐式地定义了一个分布 $P_G(x)$ ，因为从生成器中生成样本相当于从分布 $P_G(x)$ 采样

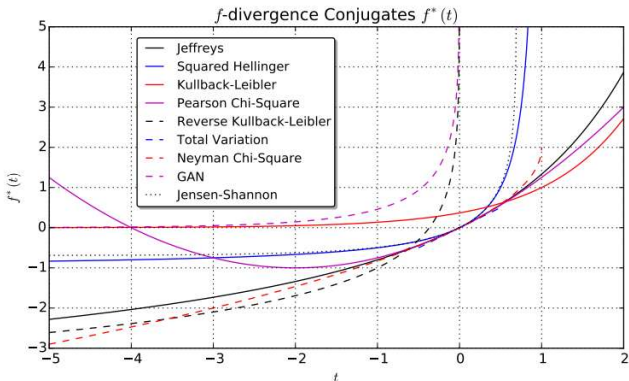


f-GAN

$$D: x \mapsto \text{dom}(f^*)$$

$$G^* = \arg \min_G \max_D (\mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [f^*(D(x))])$$

大部分 f^* 均为单调递增函数



f-GAN

$$G^* = \arg \min_G \max_D (\mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [f^*(D(x))])$$
$$D: x \mapsto \text{dom}(f^*)$$

- 不同的函数 f ，共轭函数有所不同，共轭函数的定义域也不一样
- 引入激活函数 $g(\cdot)$ ，针对 f 共轭函数的定义域来设计（一般单调递增），从而用 $g(V(x))$ 来表示 $D(x)$ ，使得 $V(x)$ 输出不再有约束

$$G^* = \arg \min_G \max_D (\mathbb{E}_{x \sim P_{data}} [g(V(x))] - \mathbb{E}_{x \sim P_G} [f^*(g(V(x)))])$$

GAN是f-GAN的特列

$$G^* = \arg \min_G \max_D \left(\mathbb{E}_{x \sim P_{data}} [g(V(x))] - \mathbb{E}_{x \sim P_G} [f^*(g(V(x)))] \right)$$

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx = 2 \text{JS}(p(x)||q(x)) - 2 \log 2$$

$$f(x) = -(x+1) \log(x+1) + x \log x \quad \longleftrightarrow \quad f^*(t) = -\log(1 - \exp(t))$$

$$t < 0$$

选择 $g(v) = -\log(1 + \exp(-v))$

$$g(V(x)) = \log \frac{1}{1 + \exp(-V(x))}$$

$$\log D(x)$$

$$\begin{aligned} f^*(g(V(x))) &= -\log \left(1 - \exp \left(\log \frac{1}{1 + \exp(-V(x))} \right) \right) \\ &= -\log \left(1 - \frac{1}{1 + \exp(-V(x))} \right) \end{aligned}$$

$$\log(1 - D(x))$$

f-GAN中使用KL散度

$$G^* = \arg \min_G \max_D \left(\mathbb{E}_{x \sim P_{data}} [g(V(x))] - \mathbb{E}_{x \sim P_G} [f^*(g(V(x)))] \right)$$

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx = KL(P||Q)$$

$$f(x) = x \log x \quad \longleftrightarrow \quad f^*(t) = \exp(t - 1) \quad t \in \mathbb{R}$$

选择 $g(v) = v$

$$g(V(x)) = V(x)$$

$$V^*(x) = 1 + \log \frac{P_{data}(x)}{P_G(x)} \quad f^*(g(V(x))) = \exp(V(x) - 1)$$

$$G^* = \arg \min_G \max_D \left(\mathbb{E}_{x \sim P_{data}} [V(x)] - \mathbb{E}_{x \sim P_G} [\exp(V(x) - 1)] \right)$$

f-GAN中使用Reverse KL散度

$$G^* = \arg \min_G \max_D \left(\mathbb{E}_{x \sim P_{data}} [g(V(x))] - \mathbb{E}_{x \sim P_G} [f^*(g(V(x)))] \right)$$

$$D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx = \text{Reverse KL } (P||Q)$$

$$f(x) = -\log x \quad \longleftrightarrow \quad f^*(t) = -1 - \log(-t) \quad t < 0$$

选择 $g(v) = -\exp(-v)$

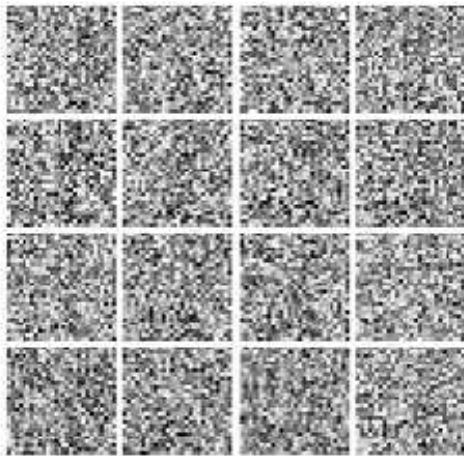
$$g(V(x)) = -\exp(-V(x))$$

$$V^*(x) = \log \frac{P_{data}(x)}{P_G(x)}$$

$$f^*(g(V(x))) = -1 - \log(\exp(-V(x))) = -1 + V(x)$$

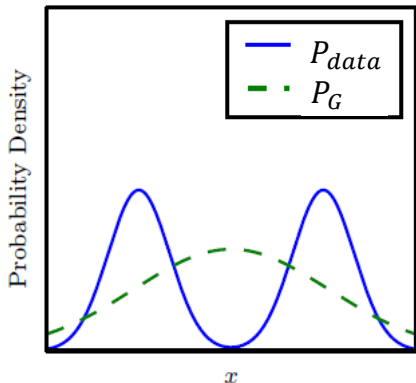
$$G^* = \arg \min_G \max_D \left(\mathbb{E}_{x \sim P_{data}} [-\exp(-V(x))] - \mathbb{E}_{x \sim P_G} [V(x) - 1] \right)$$

GAN模型中的模型坍塌



模型坍塌的原因 (一)

$$KL = \int P_{data} \log \frac{P_{data}}{P_G} dx$$



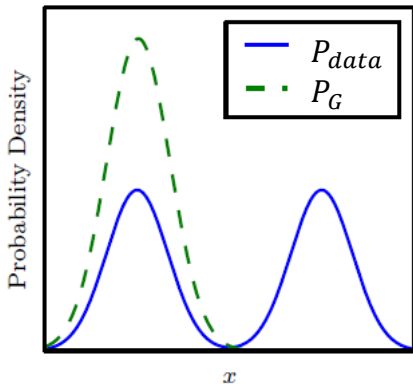
极大似然估计等价于
最小化 $KL(P_{data} || P_G)$

- 当 $P_G(x) \rightarrow 0$ 而 $P_{data}(x) > 0$ 时, 贡献为无穷大
 - “生成器要覆盖到所有的观测数据”, 但并没有捕捉到峰值, 因此生成数据质量不高

模型坍塌的原因（一）

$$\text{Reverse KL} = \int P_G \log \frac{P_G}{P_{data}} dx$$

GAN优化的JS散度更接近Reverse KL散度



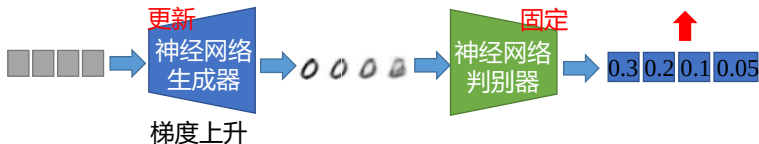
最小化 $KL(P_G || P_{data})$

- 当 $P_{data}(x) \rightarrow 0$ 而 $P_G(x) > 0$ 时, 贡献为无穷大
 - “生成器生成了不真实的样本”, 惩罚巨大
- 当 $P_G(x) \rightarrow 0$ 而 $P_{data}(x) > 0$ 时, 贡献为0
 - “生成器没能生成真实的样本”, 惩罚微小

模型坍塌的原因（二）

- 如果在训练G的时候到极致，那么G最优解和初始随机数无关

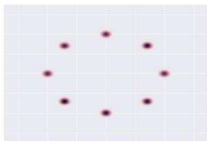
$$x^* = \arg \max_{x \in \mathcal{X}} D(x)$$



- 在训练D的时候，G生成出来的模式打低分，即给低的概率

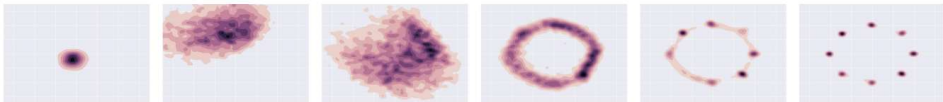


模型坍塌的原因 (二)

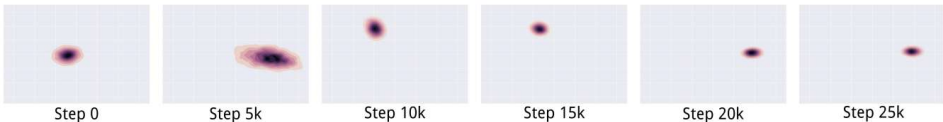


目标: 8个mode

理想情况



实际情况



给定D时，训练G到极致，只生成一个mode下数据

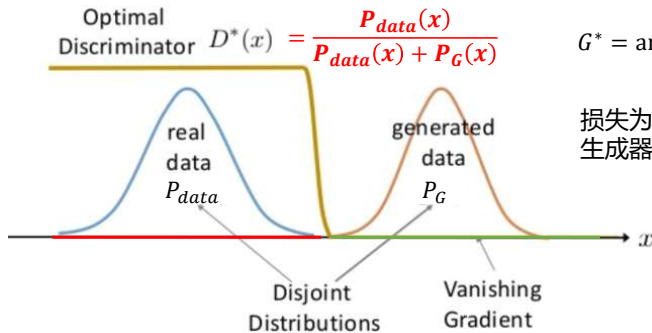
G会认为上次生成出来的数据是不真实的，会跳到其他mode

GAN中梯度消失问题

$$V(G, D^*) = E_{x \sim P_{data}} [\log D^*(x)] + E_{x \sim P_G} [\log(1 - D^*(x))]$$

$$= \int P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} dx + \int P_G(x) \log \left(\frac{P_G(x)}{P_{data}(x) + P_G(x)} \right) dx$$

$$= 0$$

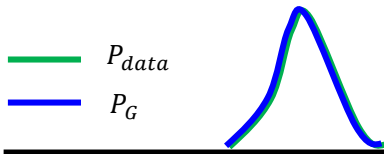
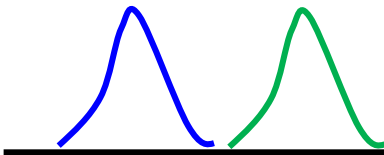
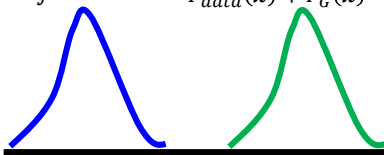


$$G^* = \arg \min_G V(G, D^*)$$

损失为常数，梯度为0，
生成器无法更新

GAN中梯度消失问题

$$V(G, D^*) = \int P_{data}(x) \log \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} dx + \int P_G(x) \log \left(\frac{P_G(x)}{P_{data}(x) + P_G(x)} \right) dx$$



P_{data}

P_G

$$V(G, D^*) = 0$$



如果直接优化
JS散度, 那么
G无法更新

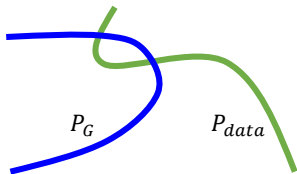
$$V(G, D^*) = 0$$



$$V(G, D^*) = -2 \log 2$$

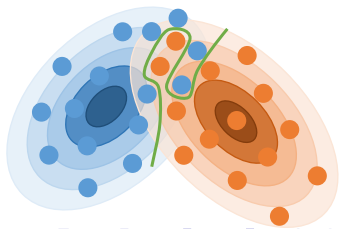
JS散度是不适合的

- 大部分情况 P_G 和 P_{data} 不交叉的
- (1) 数据分布在高维空间中的低维流形交叉部分几乎可以忽略



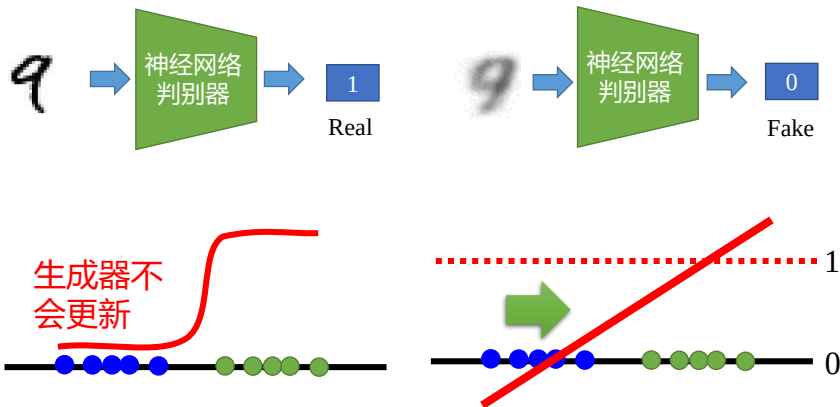
- (2) 即使有交叉部分，GAN基于采样方法训练，由于判别器能力强大，可以让分类错误率为0，等价于 P_G 和 P_{data} 不交叉

弱化判别器



Least Square GAN (LSGAN)

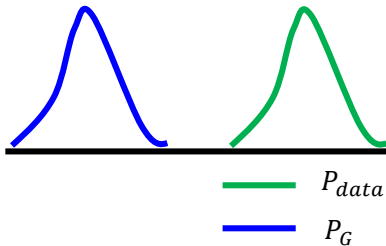
- 把sigmoid函数替换成线性函数



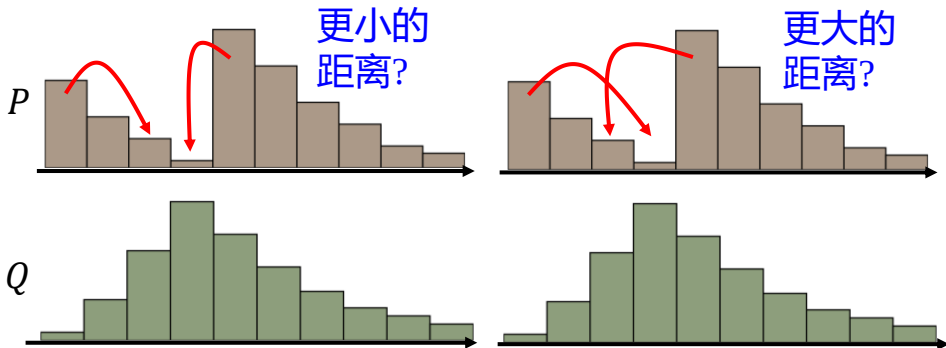
推土机距离衡量分布间的差异

- 将分布想象为两个有一定存土量的土堆，那么推土机距离就是将一个土堆 转换为另一个土堆所需的最小总工作量
- 工作量的定义是 泥土量（比例）乘以它移动的距离

等价于移土平均距离



推土机距离衡量分布间的差异

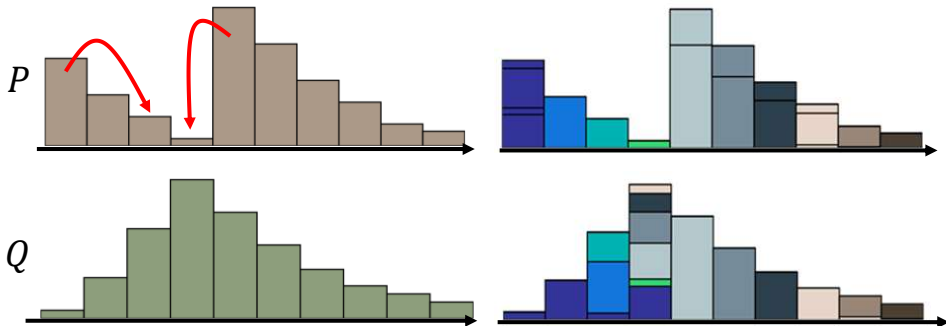


有很多可能的推土计划

用最短移土距离的推土计划来定义推土机距离

推土机距离衡量分布间的差异

这个例子中最好的推土计划



有很多可能的推土计划

用最短移土距离的推土计划来定义推土机距离



推土计划是一个矩阵

矩阵中元素 $\gamma(x_p, x_q)$ 是从一个位置到运送到另外位置的土量

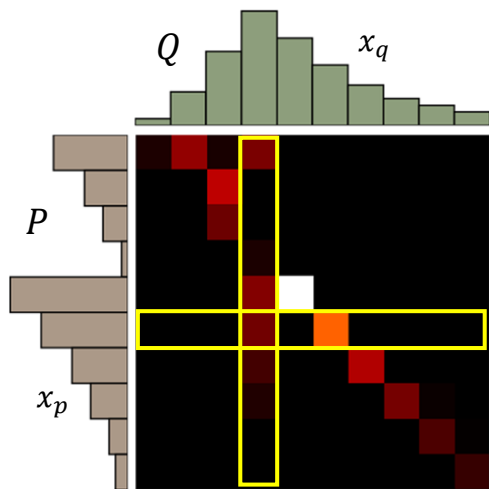
推土计划 γ 的总工作量

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

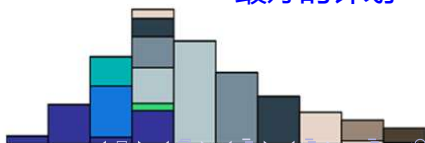
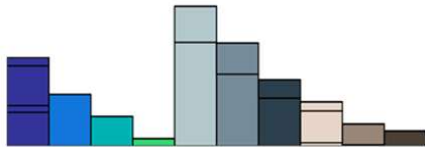
推土机距离定义为:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

最好的计划



推土计划 γ
所有可能的计划 Π



WGAN

- 推土机距离很难计算

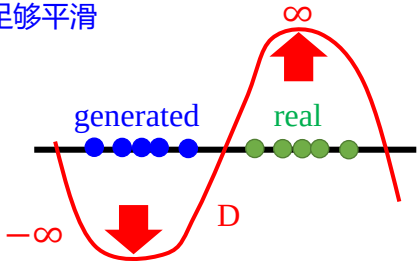
$$\min_{\gamma \in \Pi} B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

- 但可以Kantorovich-Rubinstein duality用 $V(G, D)$ 来优化计算

$$V(G, D) = \max_{\|D\|_L \leq 1} \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] \}$$

判别器D必须足够平滑

如果没有足够平滑的约束，这个优化会无法收敛



WGAN

- 推土机距离很难计算

$$\min_{\gamma \in \Pi} B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

- 但可以Kantorovich-Rubinstein duality用 $V(G, D)$ 来优化计算

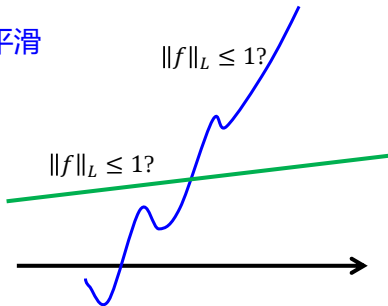
$$V(G, D) = \max_{\|D\|_L \leq 1} \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] \}$$

判别器D必须足够平滑

$\|f\|_L \leq K$ K 为利普希茨常数

↔ $\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|$

输出的变化不能超过输入变化的 K 倍



WGAN

- 推土机距离很难计算

$$\min_{\gamma \in \Pi} B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

- 但可以Kantorovich-Rubinstein duality用 $V(G, D)$ 来优化计算

$$V(G, D) = \max_{\|D\|_L \leq 1} \{ \mathbb{E}_{x \sim P_{data}} [D(x)] - \mathbb{E}_{x \sim P_G} [D(x)] \}$$

判别器D必须足够平滑 如何实现这个平滑约束呢？

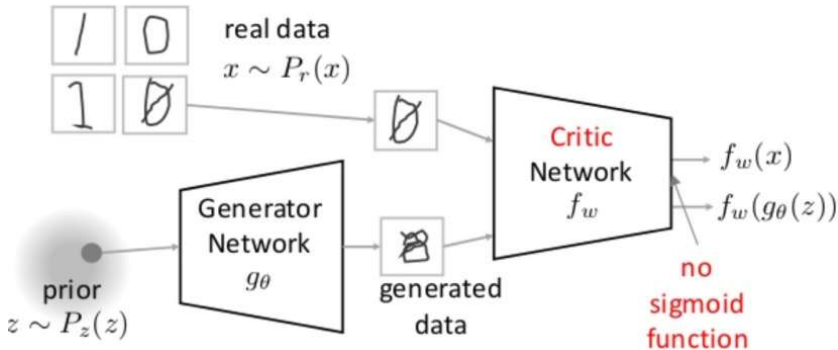
简单的方法是做Gradient Clipping：强制所有的参数在 $[-k, k]$ 之间

在参数更新的时候，如果参数比 k 大，那么截断为 k ；
如果参数比 $-k$ 小，那么截断为 $-k$

WGAN

$$\min_{\theta} \max_{w \in [-k, k]^l} \mathbb{E}_{x \sim P_r} [f_w(x)] - \mathbb{E}_{z \sim P_z} [f_w(g_{\theta}(z))]$$

k-Lipschitz Constraint



WGAN的算法

- 随机初始化判别器参数 θ_d 和生成器参数 θ_g
- While not convergent:

更新D

重复
K次

- 从数据库中采样 m 样本 $\{x^1, x^2, \dots, x^m\}$
- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 用生成器生成样本 $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}, \tilde{x}^i = G(z^i)$
- 更新判别器参数 θ_d , 通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m D(x^i) + \frac{1}{m} \sum_{i=1}^m -D(\tilde{x}^i)$$

$$\theta_d \leftarrow \theta_d + \eta \nabla \tilde{\mathcal{L}}(\theta_d) \quad \leftarrow \text{Weight Clipping}$$

更新G

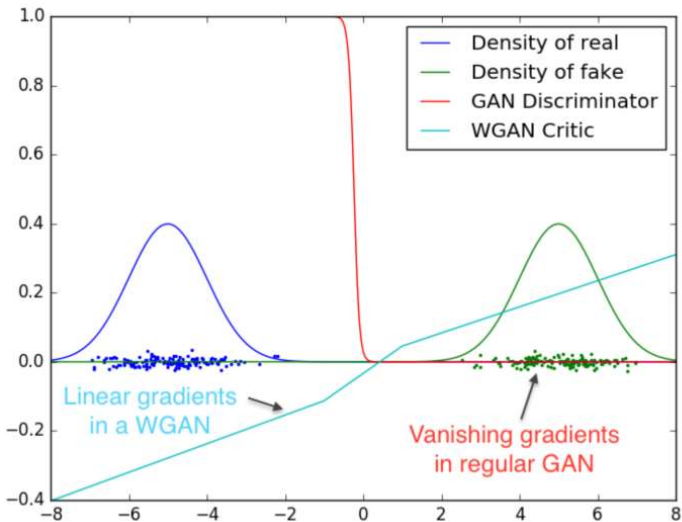
只做
一次

- 从某个分布中采样 m 个噪声样本 $\{z^1, z^2, \dots, z^m\}$
- 更新生成器参数 θ_g , 通过最大化如下目标

$$\tilde{\mathcal{L}} = \frac{1}{m} \sum_{i=1}^m D(G(z^i))$$

$$\theta_g \leftarrow \theta_g + \eta \nabla \tilde{\mathcal{L}}(\theta_g)$$

WGAN



WGAN

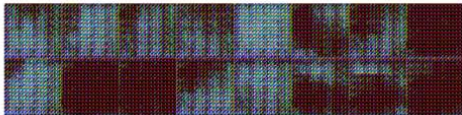


WGAN



DCGAN

~~batch normalization~~
~~constant number of filters at every layer~~



DCGAN

LSGAN

Original
WGANImproved
WGAN

G: CNN, D: CNN



G: CNN (no normalization), D: CNN (no normalization)



G: CNN (tanh), D: CNN(tanh)

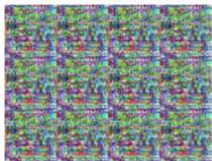


DCGAN

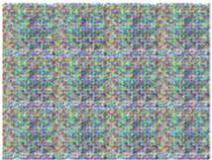
LSGAN

Original
WGANImproved
WGAN

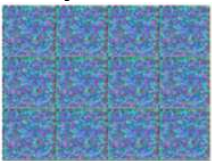
G: MLP, D: CNN



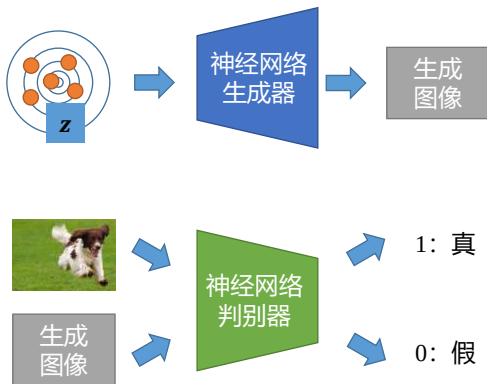
G: CNN (bad structure), D: CNN



G: 101 layer, D: 101 layer



Conditional GAN



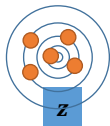
Conditional GAN (有监督)

Text-to-Image

情境

小狗在奔跑

c



神经网络
生成器

生成
图像

a dog is running



a bird is flying



小狗在奔跑

c



小狗在奔跑

c



生成
图像

猫在抓老鼠

c'



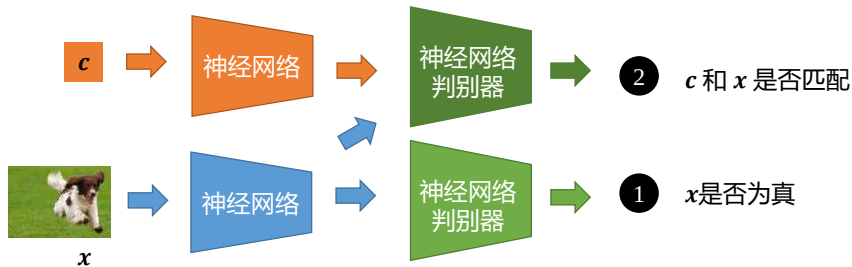
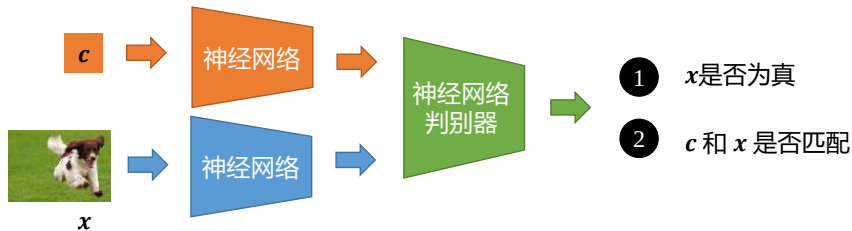
神经网络
判别器

1: 真

0: 假

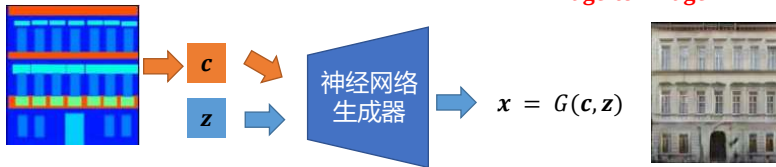
0: 假

Conditional GAN (有监督) — 判别器



Conditional GAN (有监督)

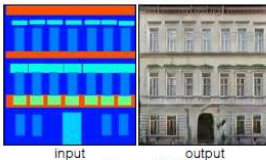
Image-to-Image



Labels to Street Scene



Labels to Facade



BW to Color



Aerial to Map



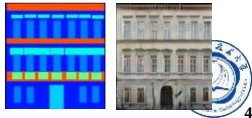
Day to Night



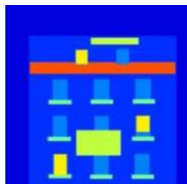
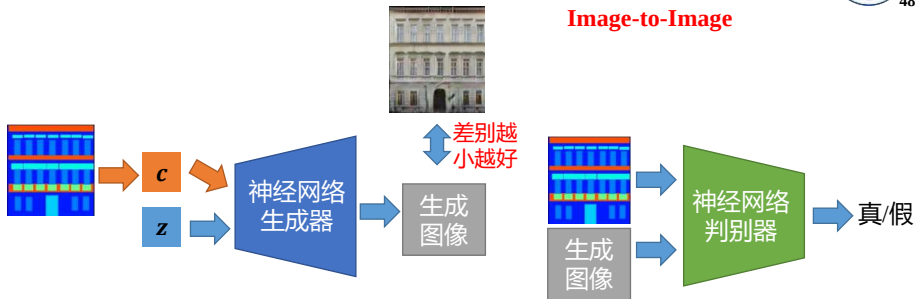
Edges to Photo



Conditional GAN (有监督)



48



输入



输入输出相似



GAN

Conditional GAN (有监督)

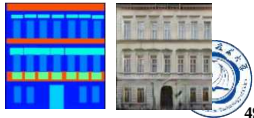
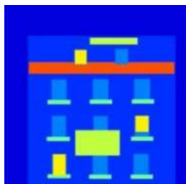
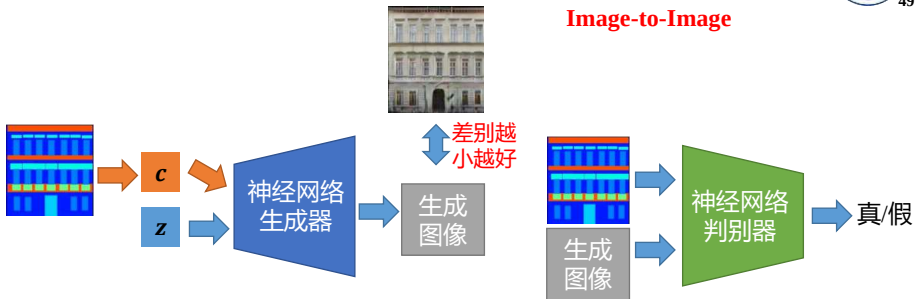


Image-to-Image



输入



输入输出相似



GAN

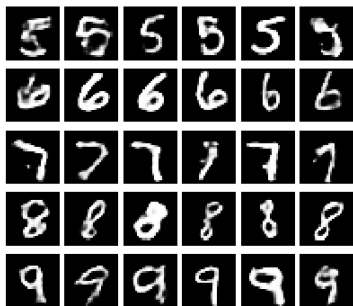
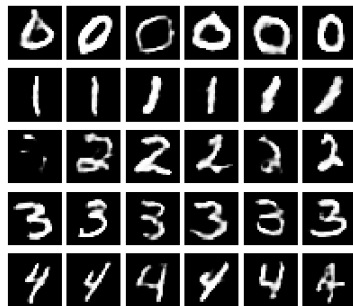


GAN+相似

Conditional GAN应用—生成特定数字图



50



Conditional GAN应用—人脸控制

0-18

19-29

30-39

40-49

50-59

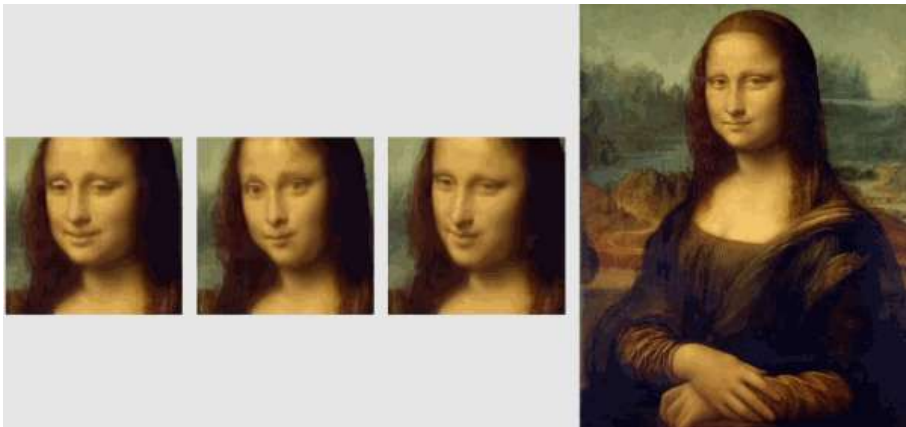
60+



Conditional GAN应用—Talking Head Generation



52



Conditional Generation (无监督)



Domain
X



神经网络
生成器



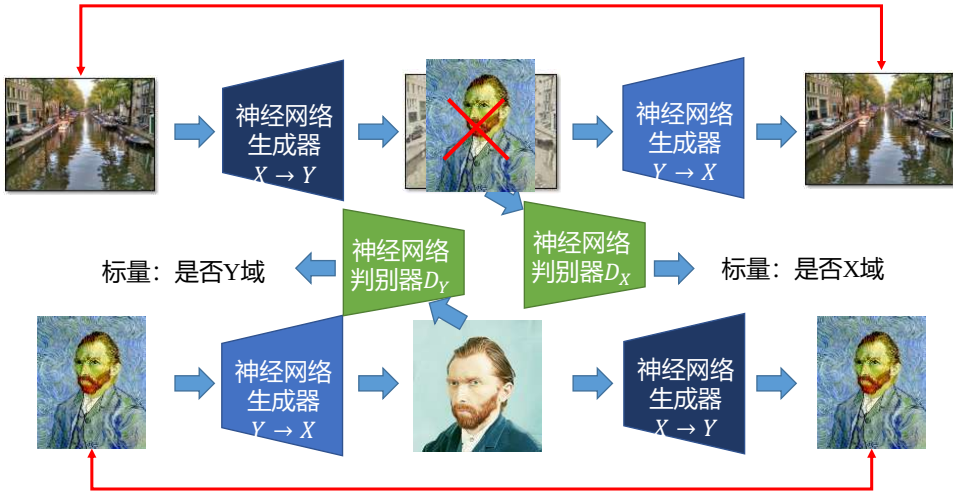
Domain
Y



在**没有配对数据**的情况下，如何将一个 domain 的对象转换到另外一个 domain 里

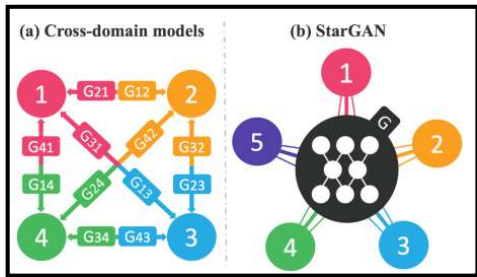
Cycle GAN

误差越小越好

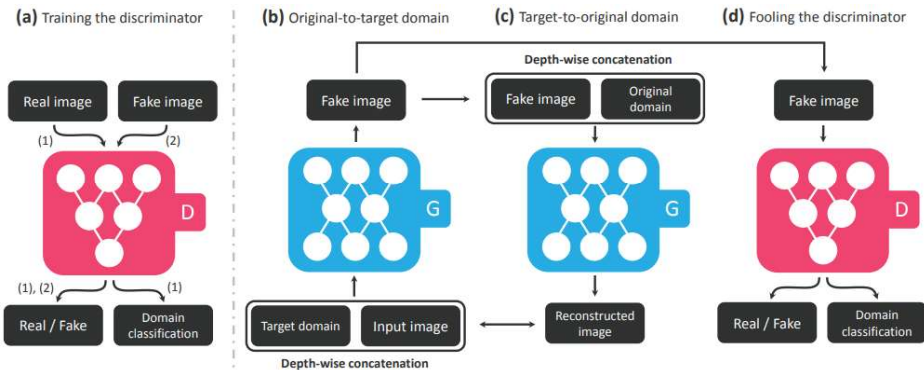


StarGAN

<https://arxiv.org/abs/1711.09020>



55



StarGAN的应用



Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

Cycle GAN

<https://selfie2anime.com/>



57



Cycle GAN



基于Conditional Generation的生成式摘要



59

X域



Y域



文档

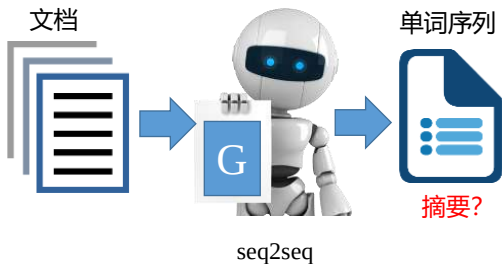
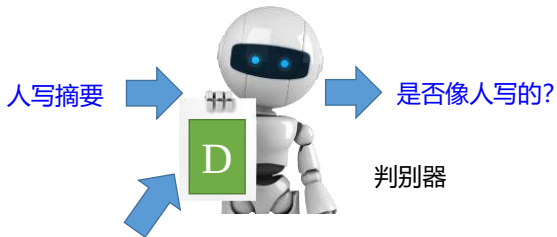


摘要

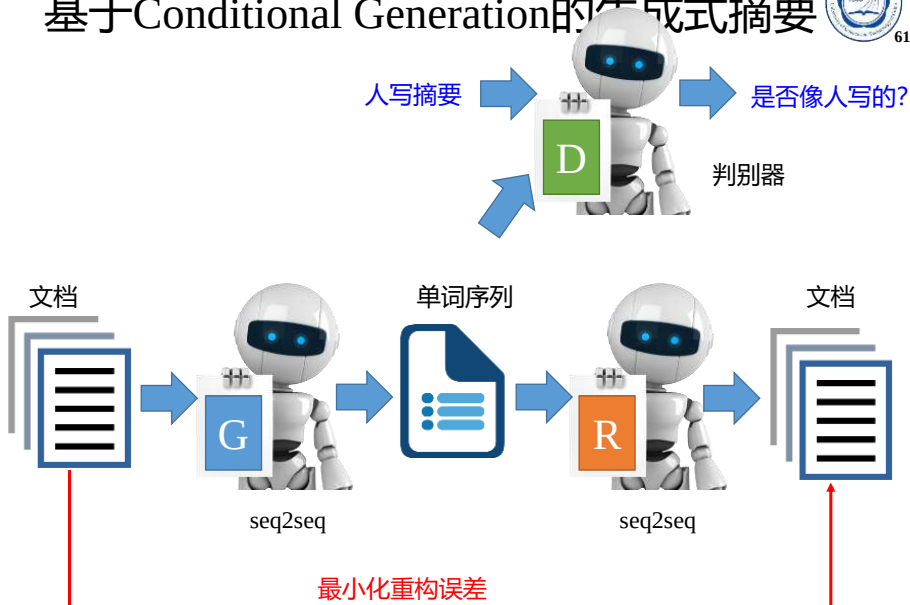
基于Conditional Generation的生成式摘要



60



基于Conditional Generation的生成式摘要



基于Conditional Generation的机器翻译

X域

Y域



文档



摘要





谢谢





元学习 (Meta-Learning)

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

主页：<http://staff.ustc.edu.cn/~liandefu>



本章内容

- 基于优化的方法
- 基于度量的方法
- 基于模型的方法

为什么要元学习

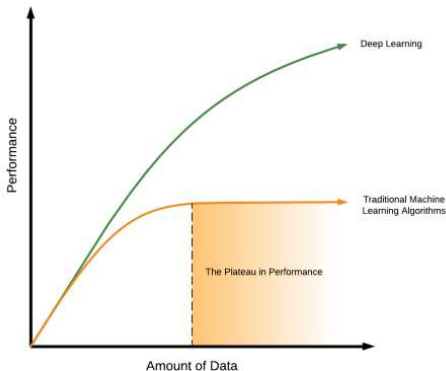
- 使用数据进行有效扩展：

传统机器学习

- 通常需要**更复杂的模型**来提高准确性

深度学习

- 可以通过使用**更多数据**来有效提高准确性



人类学习新的concept 和skill的过程需要的训练样本量
远远小于机器学习模型的需求量

什么是元学习

- 元学习 (Meta Learning) = 学习 如何学习 (Learn to Learn)
- 用少量的数据快速的学习新的概念和技能



任务1: 语音识别
任务2: 图像分割
任务3: 文本分类
任务4: 股票预测
任务5: 艺术创作
:
任务n: 蛋白质折叠

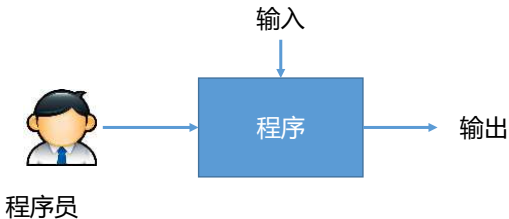


算法认为应该在
任务 $n+1$: 疾病诊断
表现更好
因为它学会了一些技能

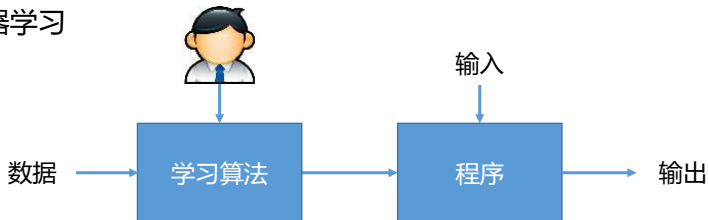
元学习就是说为新任务快速学习一个新模型

什么是机器学习

- 传统编程



- 机器学习

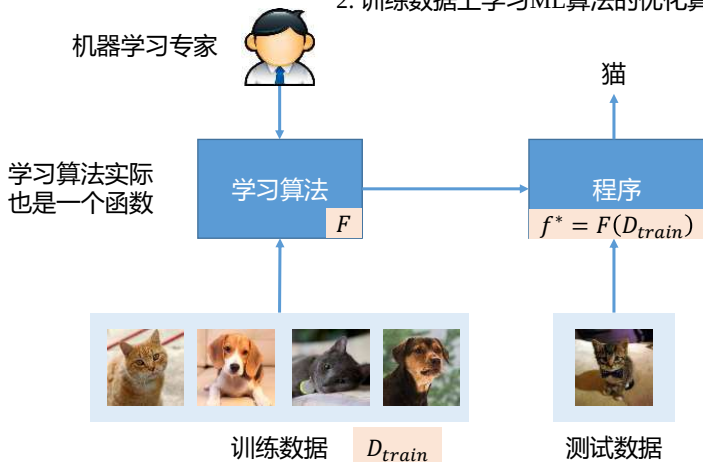


什么是元学习

$f^* = F(D_{train})$ 能否直接从数据中学习 F

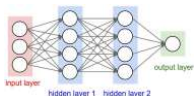
这个 F 包含两部分 (都是人为设定) :

1. 机器学习算法 (如神经网络)
2. 训练数据上学习 ML 算法的优化算法 (e.g., Adam)



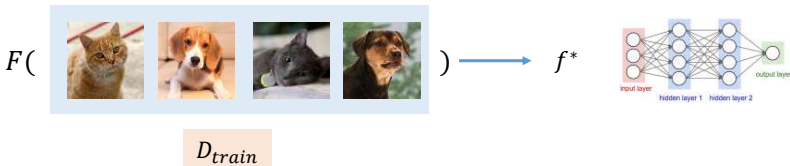
什么是元学习

- 机器学习：学习如何根据数据找拟合函数 f

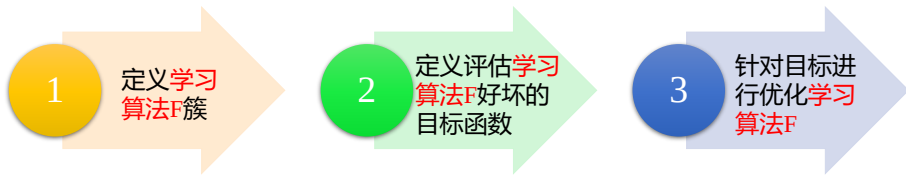


$$f(\text{Image of a cat}) \longrightarrow \text{猫}$$

- 元学习：学习如何根据数据找学习算法 F ，该学习算法 F 可以根据训练数据拟合函数 f

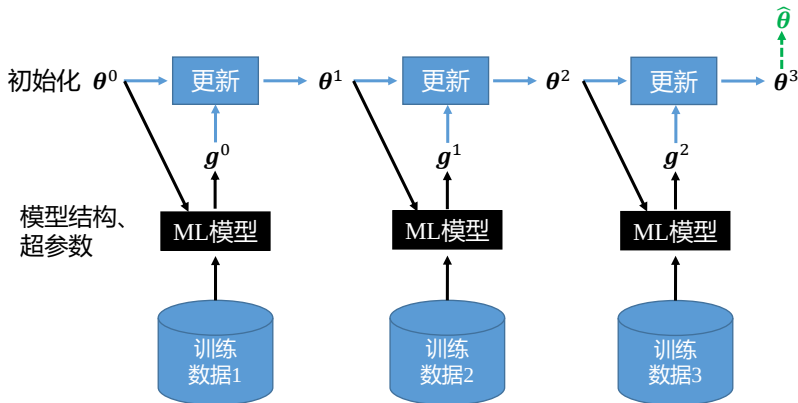


什么是元学习



什么是元学习

- 定义学习算法簇

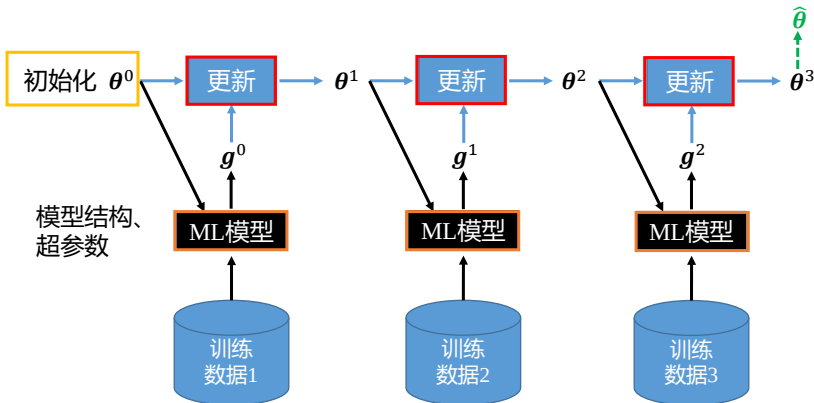


什么是元学习

颜色框中不同的设置会导致不同的学习算法

- 定义学习算法簇

颜色框中的模块设置目前都是由专家完成的





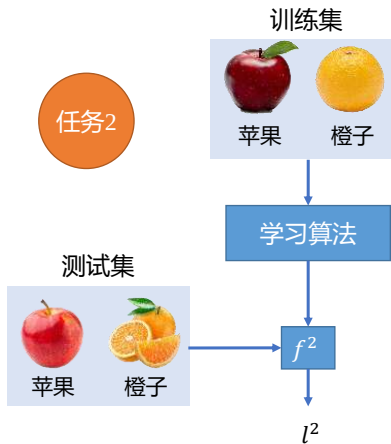
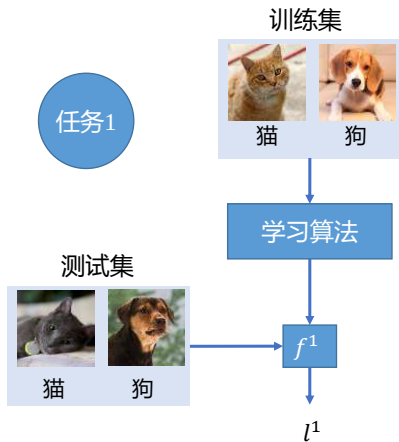
什么是元学习

- 定义评估学习算法好坏的评价指标

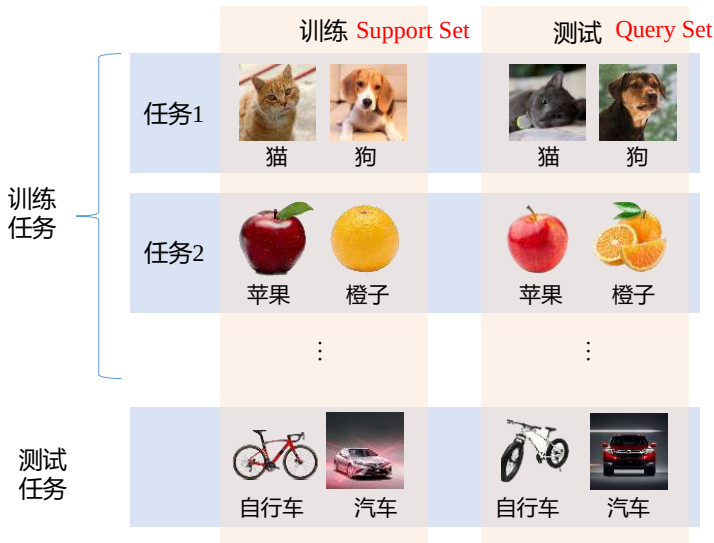
$$L(F) = \sum_{n=1}^N l^n$$

N为任务数

l^n 为在第n个任务的测试集上loss



元学习的训练与测试



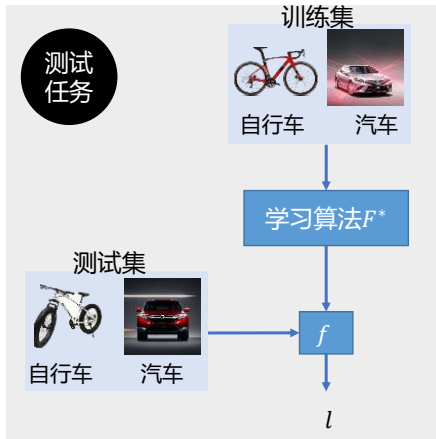
元学习

- 定义评价学习算法好坏的目标函数

$$L(F) = \sum_{n=1}^N l^n$$

- 寻找最优学习算法 F^*

$$F^* = \arg \min_F L(F)$$



元学习的形式化

- 从已知label集合中采样一个子集: $L \subset \mathcal{L}$ (对应一个任务)
- 采样得到对应的训练集 S^L 和测试集 B^L : $S^L \subset \mathcal{D}$ 且 $B^L \subset \mathcal{D}$
- 元学习的优化目标如下

$$F^* = \arg \min_F \mathbb{E}_{L \subset \mathcal{L}} \left[\mathbb{E}_{S^L \subset \mathcal{D}, B^L \subset \mathcal{D}} \left[\sum_{(x,y) \in B^L} P_F(y|x, S^L) \right] \right]$$

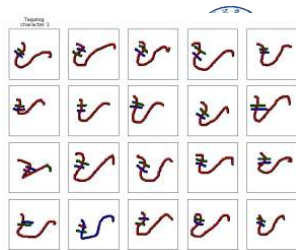
红色部分是**相较于监督学习的目标函数**所增加的部分

元学习的常见应用范例—小样本学习

- N-ways K-shot 分类
- 在每个训练和测试任务
 - 训练集 (support set) 中, 只有 N 个类, 每个类中有 K 个样本
 - 测试集 (query set) 中, 同样 N 个类, 每个类 x 个样本 (x 任意取值)

元学习关键数据集—Omniglot

- 1623 字符 (类别)
- 每个字符 (类别) 有20个样本



元学习的常见应用范例—小样本分类

- N-ways K-shot 分类
- 在每个训练和测试任务
 - 训练集 (support set) 中, 只有 N 个类, 每个类中有 K 个样本
 - 测试集 (query set) 中, 同样 N 个类, 每个类 x 个样本 (x 任意取值)

Omniglot 上的
20 ways 1 shot

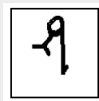
Omniglot 共1623 字符 (类别)
每个字符 (类别) 有20个样本

一个任务

每个字符代表
一个类, 共20
个类

𐀀	𐀁	𐀂	𐀃	𐀄
𐀅	𐀆	𐀇	𐀈	𐀉
𐀊	𐀋	𐀌	𐀍	𐀎
𐀏	𐀐	𐀑	𐀒	𐀓

训练集
(Support set)



测试集
(Query set)

元学习的形式化

- 从已知label集合中采样一个子集: $L \subset \mathcal{L}$ (对应一个任务)
- 采样得到对应的训练集 S^L 和测试集 B^L : $S^L \subset \mathcal{D}$ 且 $B^L \subset \mathcal{D}$
- 元学习的优化目标如下

$$F^* = \arg \min_F \mathbb{E}_{L \subset \mathcal{L}} \left[\mathbb{E}_{S^L \subset \mathcal{D}, B^L \subset \mathcal{D}} \left[\sum_{(x,y) \in B^L} P_F(y|x, S^L) \right] \right]$$

	基于模型的	基于度量的	基于优化的
核心理念	RNN; memory	Metric Learning	Gradient Descent
如何建模 $P_F(y x, S^L)$	$f_\theta(x, S^L)$	$\sum_{(x,y) \in S^L} k_\theta(x, x_i) y_i$	$P_{g_\phi(\theta, S^L)}(y x)$

基于优化的元学习

- MAML
 - Chelsea Finn, Pieter Abbeel, and Sergey Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”, ICML, 2017
- Reptile
 - Alex Nichol, Joshua Achiam, John Schulman, On First-Order Meta-Learning Algorithms, arXiv, 2018
- Gradient Descent as LSTM

OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING

Sachin Ravi* and Hugo Larochelle
Twitter, Cambridge, USA
{sachinr, hugo}@twitter.com

Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz¹, Misha Denil¹, Sergio Gómez Colmenarejo¹, Matthew W. Hoffman¹,
David Pfau¹, Tom Schaul¹, Brendan Shillingford^{1,2}, Nando de Freitas^{1,2,3}

¹Google DeepMind ²University of Oxford ³Canadian Institute for Advanced Research

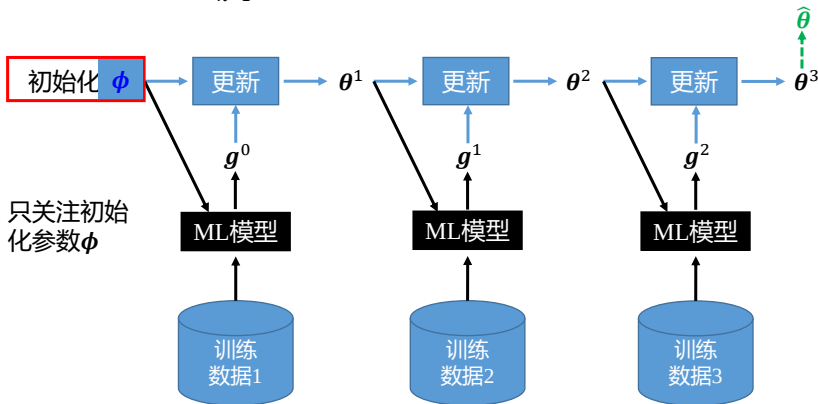
MAML

$\hat{\theta}^n$ 从第n个任务学到的模型参数

$\hat{\theta}^n$ 依赖于初始值 ϕ

损失函数
$$L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$$

$l^n(\hat{\theta}^n)$ 第n个任务在测试集上的损失



MAML

$\hat{\theta}^n$ 从第n个任务学到的模型参数

$\hat{\theta}^n$ 依赖于初始值 ϕ

损失函数 $L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$

$l^n(\hat{\theta}^n)$ 第n个任务在测试集上的损失

如何优化参数 ϕ 呢? 梯度下降法

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

Model Pre-training 损失函数

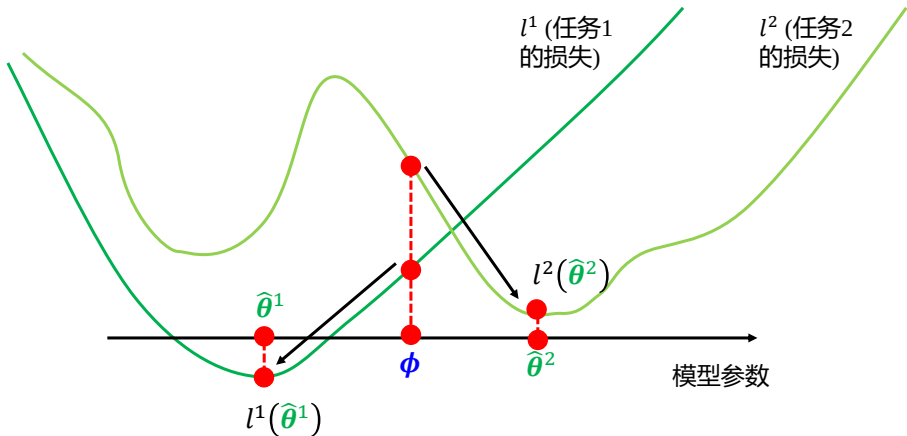
$$L(\phi) = \sum_{n=1}^N l^n(\phi)$$

MAML

MAML不关心 ϕ 在每个训练任务上的表现

MAML优化的是用 ϕ 在每个任务上更新后 (即 $\hat{\theta}^n$) 的表现

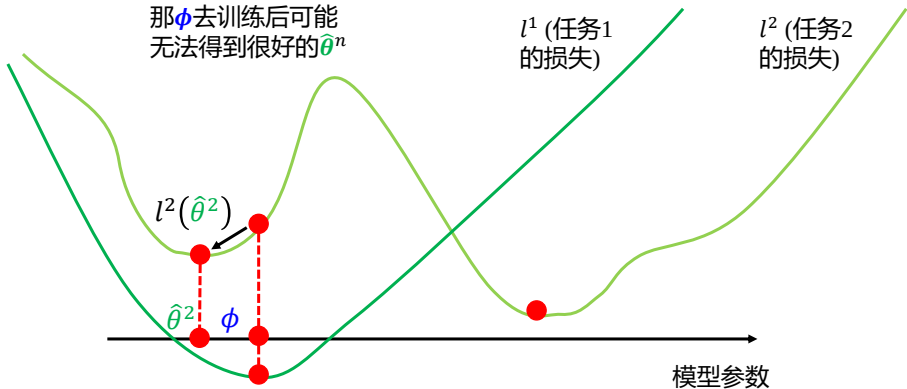
损失函数
$$L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$$



Model Pre-training

损失函数 $L(\phi) = \sum_{n=1}^N l^n(\phi)$

Pre-training的目标: 找到所有任务上都最好的 ϕ



MAML

$\hat{\theta}^n$ 从第n个任务学到的模型参数

$\hat{\theta}^n$ 依赖于初始值 ϕ

损失函数 $L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$

$l^n(\hat{\theta}^n)$ 第n个任务在测试集上的损失

如何优化参数 ϕ 呢? 梯度下降法

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

优化 ϕ 使得在训练后可以取得好的性能

潜力

Model Pre-training 损失函数

$$L(\phi) = \sum_{n=1}^N l^n(\phi)$$

优化 ϕ 使得在当前所有任务上取得好的性能

现实

MAML

$\hat{\theta}^n$ 从第 n 个任务学到的模型参数

$\hat{\theta}^n$ 依赖于初始值 ϕ

损失函数
$$L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$$

$l^n(\hat{\theta}^n)$ 第 n 个任务在测试集上的损失

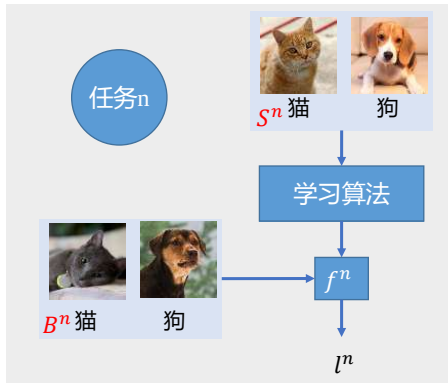
如何优化参数 ϕ 呢? 梯度下降法

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

只考虑一步更新 $\hat{\theta}^n \leftarrow \phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n)$

该任务的在查询集 B^n 上的损失 $l^n(\hat{\theta}^n, B^n)$

→ $l^n(\phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n), B^n)$



MAML

$\hat{\theta}^n$ 从第n个任务学到的模型参数

$\hat{\theta}^n$ 依赖于初始值 ϕ

损失函数
$$L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$$

$l^n(\hat{\theta}^n)$ 第n个任务在测试集上的损失

如何优化参数 ϕ 呢? 梯度下降法

$$\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$$

只考虑一步更新 $\hat{\theta}^n \leftarrow \phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n)$

该任务的在查询集 B^n 上的损失 $l^n(\hat{\theta}^n, B^n)$

$$l^n(\phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n), B^n)$$

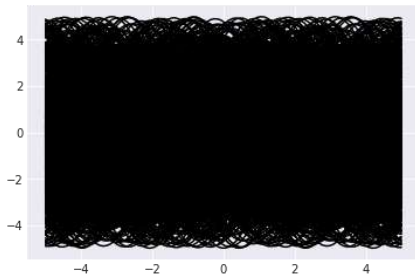
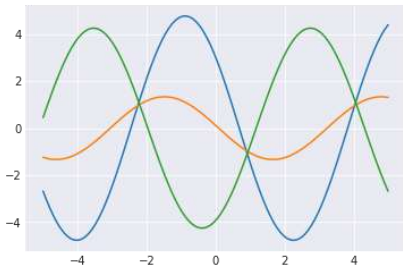
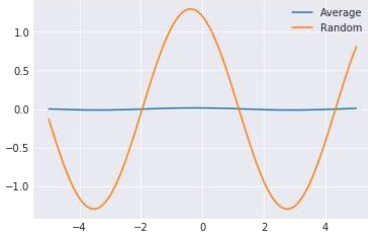
为什么只做一步更新呢?

- 快!
- 效果好!
- 数据少!
- 算法实际应用可多步更新!

Toy Example

• 每个任务：

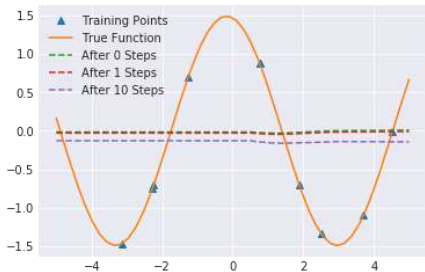
- 随机采样 a 和 b 的值
- 从正弦函数 $y = a \sin(x + b)$ 采样 K 个样本点
- 基于 K 个样本用神经网络估计该函数



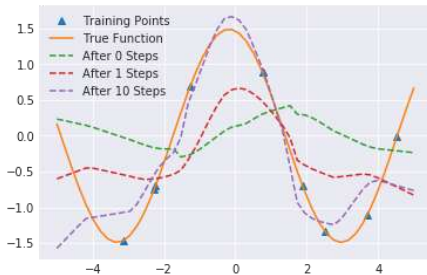
1000个不同的 a 和 b

Toy Example

Model Pre-training



MAML



Omniglot & Mini-ImageNet

	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
Omniglot (Lake et al., 2011)				
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	—	—
MAML, no conv (ours)	$89.7 \pm 1.1\%$	$97.5 \pm 0.6\%$	—	—
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	$98.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$

	5-way Accuracy	
	1-shot	5-shot
MiniImagenet (Ravi & Larochelle, 2017)		
fine-tuning baseline	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
nearest neighbor baseline	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
matching nets (Vinyals et al., 2016)	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
meta-learner LSTM (Ravi & Larochelle, 2017)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML, first order approx. (ours)	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
MAML (ours)	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$



MAML

$$\hat{\theta}^n \leftarrow \phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n) \Rightarrow \hat{\theta}_i^n \leftarrow \phi_i - \varepsilon \frac{\partial l^n(\phi, S^n)}{\partial \phi_i}$$

损失函数 $L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n) = \sum_{n=1}^N l^n(\phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n), B^n)$

如何优化参数 ϕ 呢? 梯度下降法 $\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$

$$\begin{aligned} \nabla_{\phi} L(\phi) &= \sum_{n=1}^N \nabla_{\phi} l^n(\hat{\theta}^n, B^n) \\ &= \sum_{n=1}^N (\nabla_{\phi} \hat{\theta}^n)^{\top} \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) = \sum_{n=1}^N \left(I - \varepsilon \nabla_{\phi}^2 l^n(\phi, S^n) \right)^{\top} \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) \end{aligned}$$

$$\nabla_{\phi} \hat{\theta}^n = \begin{bmatrix} \frac{\partial \hat{\theta}_1^n}{\partial \phi_1} & \cdots & \frac{\partial \hat{\theta}_1^n}{\partial \phi_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{\theta}_d^n}{\partial \phi_1} & \cdots & \frac{\partial \hat{\theta}_d^n}{\partial \phi_d} \end{bmatrix} \quad \frac{\partial \hat{\theta}_i^n}{\partial \phi_j} = \begin{cases} -\frac{\partial^2 l^n(\phi, S^n)}{\partial \phi_i \partial \phi_j}, & \text{if } i \neq j \\ 1 - \frac{\partial^2 l^n(\phi, S^n)}{\partial \phi_i \partial \phi_j}, & \text{if } i = j \end{cases}$$



MAML

$$\hat{\theta}^n \leftarrow \phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n) \Rightarrow \hat{\theta}_i^n \leftarrow \phi_i - \varepsilon \frac{\partial l^n(\phi, S^n)}{\partial \phi_i}$$

损失函数 $L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n) = \sum_{n=1}^N l^n(\phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n), B^n)$

如何优化参数 ϕ 呢? 梯度下降法 $\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$

$$\begin{aligned} \nabla_{\phi} L(\phi) &= \sum_{n=1}^N \nabla_{\phi} l^n(\hat{\theta}^n, B^n) \approx \sum_{n=1}^N \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) \\ &= \sum_{n=1}^N (\nabla_{\phi} \hat{\theta}^n)^{\top} \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) \approx \sum_{n=1}^N \left(I - \varepsilon \nabla_{\phi}^2 l^n(\phi, S^n) \right)^{\top} \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) \end{aligned}$$

$$\nabla_{\phi} \hat{\theta}^n = \begin{bmatrix} \frac{\partial \hat{\theta}_1^n}{\partial \phi_1} & \cdots & \frac{\partial \hat{\theta}_1^n}{\partial \phi_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{\theta}_d^n}{\partial \phi_1} & \cdots & \frac{\partial \hat{\theta}_d^n}{\partial \phi_d} \end{bmatrix} \quad \frac{\partial \hat{\theta}_i^n}{\partial \phi_j} = \begin{cases} -\frac{\partial^2 l^n(\phi, S^n)}{\partial \phi_i \partial \phi_j}, & \text{if } i \neq j \\ 1 - \frac{\partial^2 l^n(\phi, S^n)}{\partial \phi_i \partial \phi_j}, & \text{if } i = j \end{cases}$$

MAML

$$\hat{\theta}^n \leftarrow \phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n)$$

$$\text{损失函数 } L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n) = \sum_{n=1}^N l^n(\phi - \varepsilon \nabla_{\phi} l^n(\phi, S^n), B^n)$$

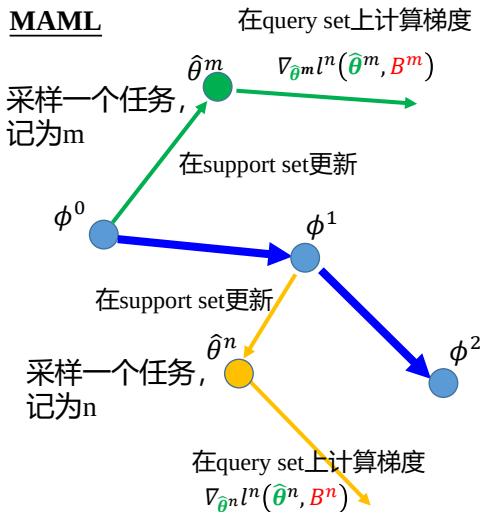
如何优化参数 ϕ 呢? 梯度下降法 $\phi \leftarrow \phi - \eta \nabla_{\phi} L(\phi)$

$$\nabla_{\phi} L(\phi) = \sum_{n=1}^N \nabla_{\phi} l^n(\hat{\theta}^n, B^n) \approx \sum_{n=1}^N \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n)$$

products, which is supported by standard deep learning libraries such as TensorFlow (Abadi et al., 2016). In our experiments, we also include a comparison to dropping this backward pass and using a first-order approximation, which we discuss in Section 5.2.

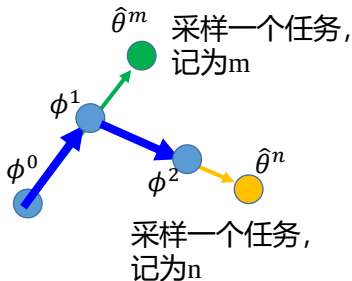
MAML的实际实现

MAML



损失函数 $L(\phi) = \sum_{n=1}^N l^n(\phi)$

Pre-training模型



基于优化的元学习

- MAML
 - Chelsea Finn, Pieter Abbeel, and Sergey Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”, ICML, 2017
- Reptile
 - Alex Nichol, Joshua Achiam, John Schulman, On First-Order Meta-Learning Algorithms, arXiv, 2018
- Gradient Descent as LSTM

OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING

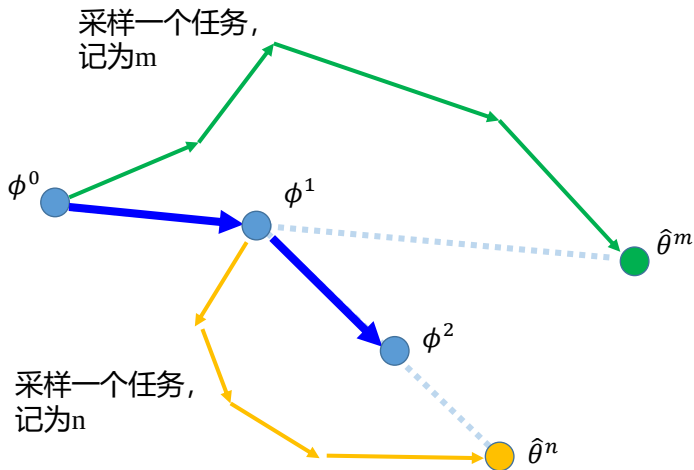
Sachin Ravi* and Hugo Larochelle
Twitter, Cambridge, USA
{sachinr, hugo}@twitter.com

Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz¹, Misha Denil¹, Sergio Gómez Colmenarejo¹, Matthew W. Hoffman¹,
David Pfau¹, Tom Schaul¹, Brendan Shillingford^{1,2}, Nando de Freitas^{1,2,3}

¹Google DeepMind ²University of Oxford ³Canadian Institute for Advanced Research

Reptile



Reptile

$\hat{\theta}^n$ 从第n个任务学到的模型参数

$\hat{\theta}^n$ 依赖于初始值 ϕ

损失函数 $L(\phi) = \sum_{n=1}^N l^n(\hat{\theta}^n)$

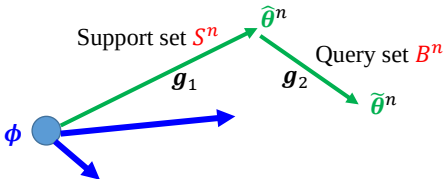
$l^n(\hat{\theta}^n)$ 第n个任务在测试集上的损失

记 $g_1 = \nabla_{\phi} l^n(\phi, S^n)$ $\hat{\theta}^n \leftarrow \phi - \varepsilon g_1$ $g_2 = \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n)$ $\tilde{\theta}^n \leftarrow \hat{\theta}^n - \varepsilon g_2$

$$g_{\text{FOMAML}} = \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) = g_2$$

$$g_{\text{MAML}} = \left(I - \varepsilon \nabla_{\phi}^2 l^n(\phi, S^n) \right)^{\top} \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) = g_2 - \varepsilon H g_2 \quad H = \nabla_{\phi}^2 l^n(\phi, S^n)$$

$$g_{\text{Reptile}} = \frac{(\tilde{\theta}^n - \phi)}{\varepsilon} = g_1 + g_2$$

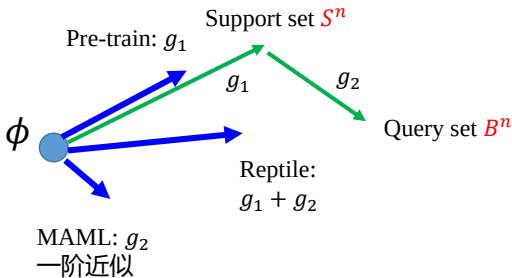


Reptile

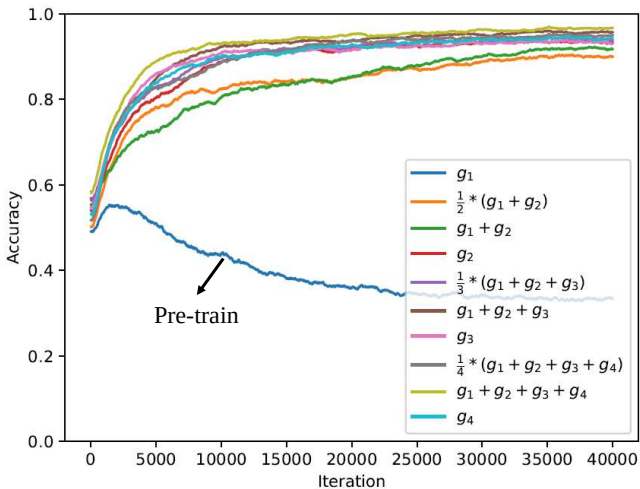
$$\text{记 } g_1 = \nabla_{\phi} l^n(\phi, S^n) \quad g_2 = \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) \quad \hat{\theta}^n \leftarrow \phi - \varepsilon g_1 \quad \tilde{\theta}^n \leftarrow \hat{\theta}^n - \varepsilon g_2$$

$$g_{\text{FOMAML}} = \nabla_{\hat{\theta}^n} l^n(\hat{\theta}^n, B^n) = g_2$$

$$g_{\text{Reptile}} = \frac{(\tilde{\theta}^n - \phi)}{\varepsilon} = g_1 + g_2$$



Reptile



基于优化的元学习

- MAML
 - Chelsea Finn, Pieter Abbeel, and Sergey Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”, ICML, 2017
- Reptile
 - Alex Nichol, Joshua Achiam, John Schulman, On First-Order Meta-Learning Algorithms, arXiv, 2018
- Gradient Descent as LSTM

OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING

Sachin Ravi* and Hugo Larochelle
Twitter, Cambridge, USA
{sachinr,hugo}@twitter.com

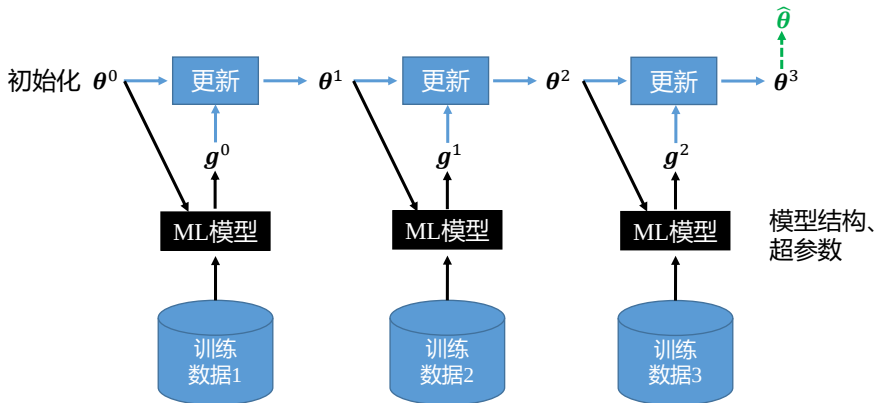
Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz¹, Misha Denil¹, Sergio Gómez Colmenarejo¹, Matthew W. Hoffman¹,
David Pfau¹, Tom Schaul¹, Brendan Shillingford^{1,2}, Nando de Freitas^{1,2,3}

¹Google DeepMind ²University of Oxford ³Canadian Institute for Advanced Research

Gradient Descent as LSTM

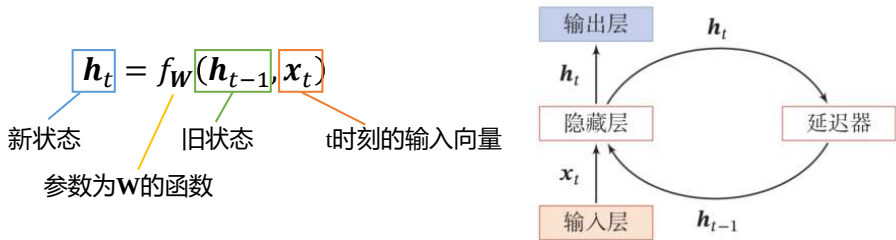
- 学习过程很像RNN，参数就是隐状态



循环神经网络 (Recurrent Neural Network)

41

- 循环神经网络通过使用带自反馈的神经元，能够处理任意长度的时序数据。



状态 h_t “存储” 直到t时刻的历史数据 $\{x_1, \dots, x_t\}$

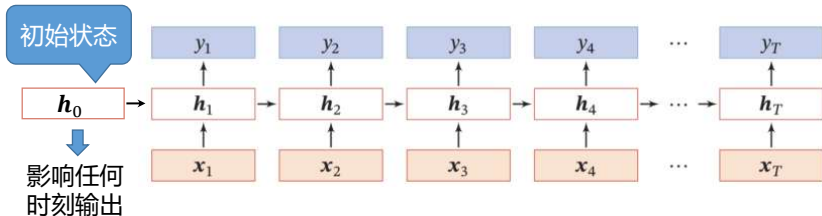
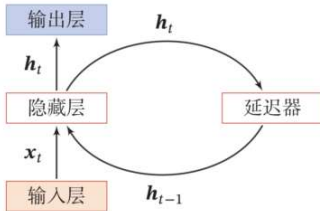
$$h_t = f_W(f_W(f_W(h_{t-3}, x_{t-2}), x_{t-1}), x_t)$$

按时间展开

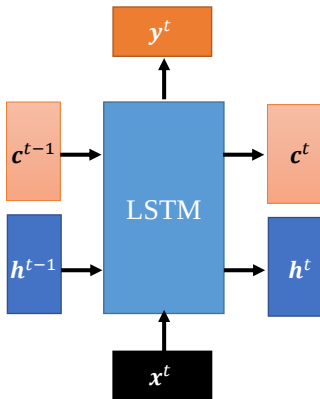
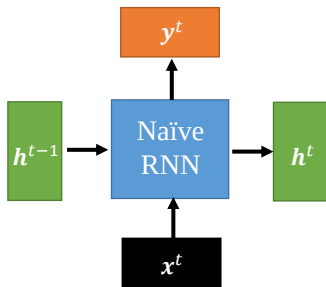
$$\boxed{h_t} = f_w(\boxed{h_{t-1}}, \boxed{x_t})$$

新状态 旧状态 t时刻的输入向量

参数为 w 的函数



LSTM

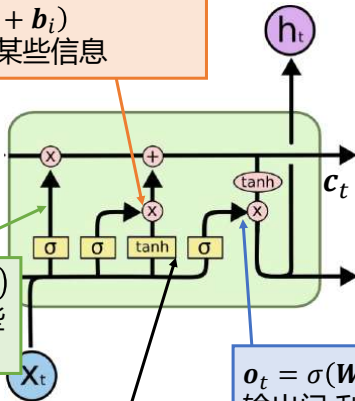


c 变化很慢, 通过加法更新
 h 变化很快, 可能完全不一样

长短期记忆神经网络 (LSTM)

$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
输入门:记忆现在的某些信息

$$h_t = o_t \odot \tanh(c_t)$$



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
遗忘门:选择忘记过去某些信息

$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$:
输出门:利用多少存储信息输出

$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$
感知层, 确定关键和新的信息

Gradient Descent as LSTM

- 可以更加简洁的描述为

$$\begin{bmatrix} \tilde{c}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{h}_{t-1} \\ -\nabla_{\theta} L \end{bmatrix} + \mathbf{b} \right)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} - \eta \nabla_{\theta} L$$



$$\boldsymbol{\theta}^t = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \odot \boldsymbol{\theta}^{t-1} + \begin{bmatrix} \eta \\ \eta \\ \vdots \\ \eta \end{bmatrix} \odot -\nabla_{\theta} L$$

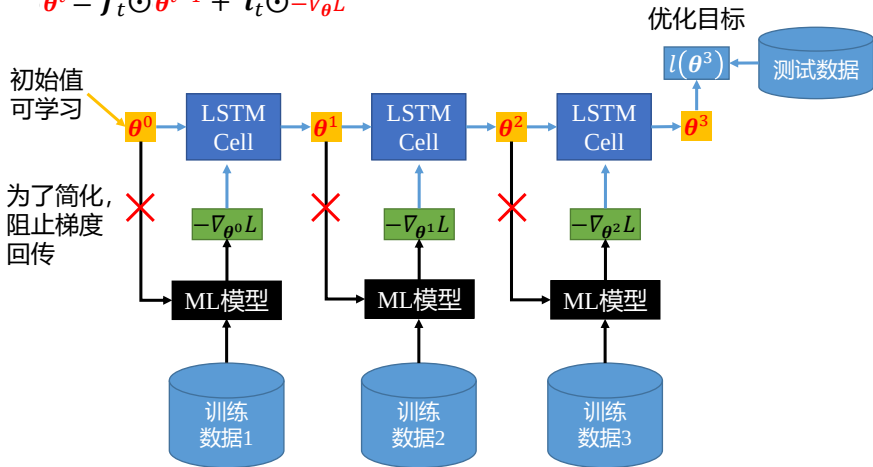
是否能自动从 $-\nabla_{\theta} L$ 等其它信息中自动确定 $\mathbf{f}_t$ 、 $\mathbf{i}_t$

正则化系数 $\mathbf{f}_t$

自适应学习率 $\mathbf{i}_t$

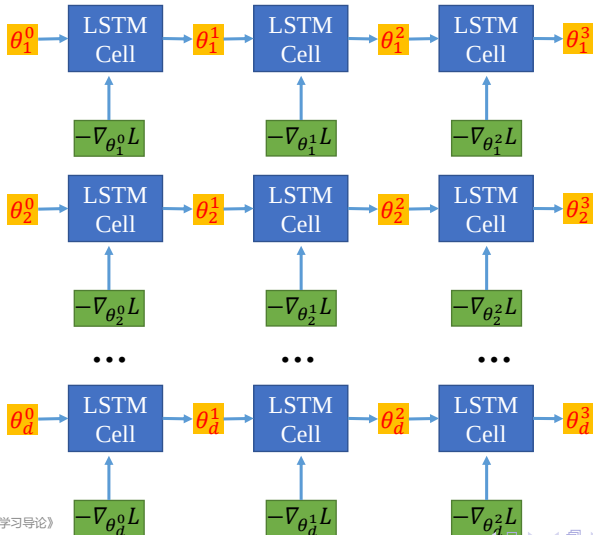
Gradient Descent as LSTM

$$\theta^t = f_t \odot \theta^{t-1} + i_t \odot -\nabla_{\theta} L$$



Gradient Descent as LSTM (实际实现)

LSTM只使用一个cell, 不同参数 (共d个参数) 间共享



- LSTM模型不会大
- 传统SGD算法中, 所有参数更新公式
- 训练和测试时模型架构可以不同

Gradient Descent as LSTM (改进)

自适应学习率—RMSProp

$$\begin{aligned} w^1 &\leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 & \sigma^0 &= g^0 \\ w^2 &\leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 & \sigma^1 &= \sqrt{\alpha(\sigma^0)^2 + (1 - \alpha)g^1^2} \\ w^3 &\leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 & \sigma^2 &= \sqrt{\alpha(\sigma^1)^2 + (1 - \alpha)g^2^2} \\ &\vdots & & \\ w^{t+1} &\leftarrow w^t - \frac{\eta}{\sigma^t} g^t & \sigma^t &= \sqrt{\alpha(\sigma^{t-1})^2 + (1 - \alpha)g^t^2} \end{aligned}$$

指数加权的
移动平均

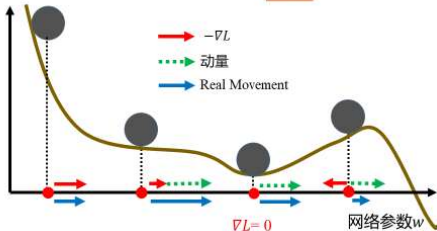
$$(\sigma^t)^2 = (\alpha)^t g_0^2 + (1 - \alpha) \sum_{i=1}^t g_i^2$$

动量法—动机

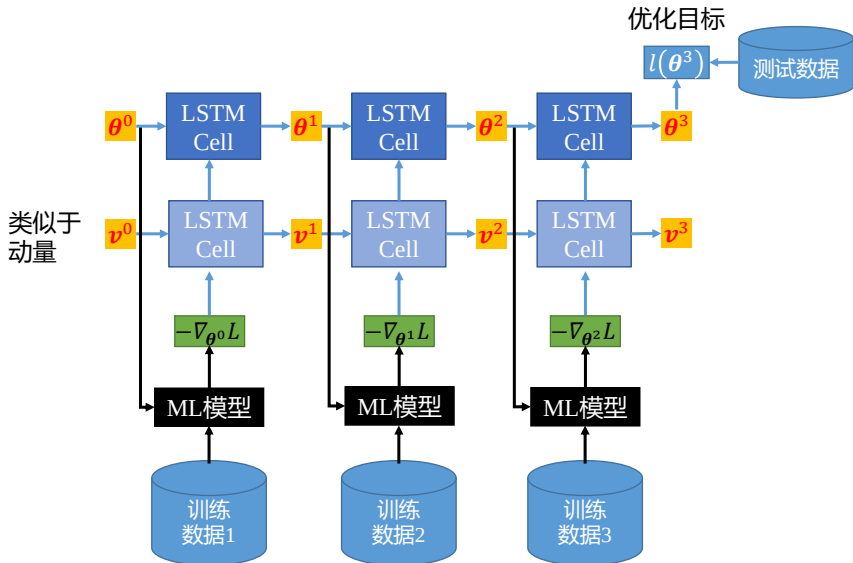
一个物体的动量指的是该物体在它运动方向上保持运动的趋势，是该物体的质量和速度的乘积

$$\text{Movement} = -\nabla L + \text{动量}$$

损失 $\mathcal{L}$



Gradient Descent as LSTM (改进)

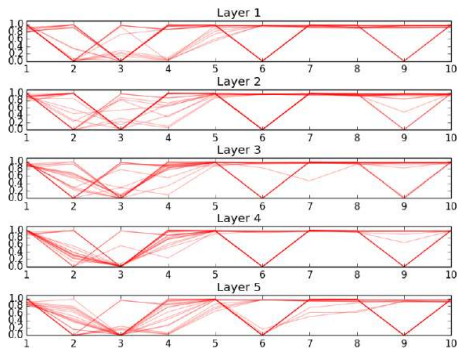


Gradient Descent as LSTM (实验结果)

$$\theta^t = f_t \odot \theta^{t-1} + i_t \odot -\nabla_{\theta} L$$



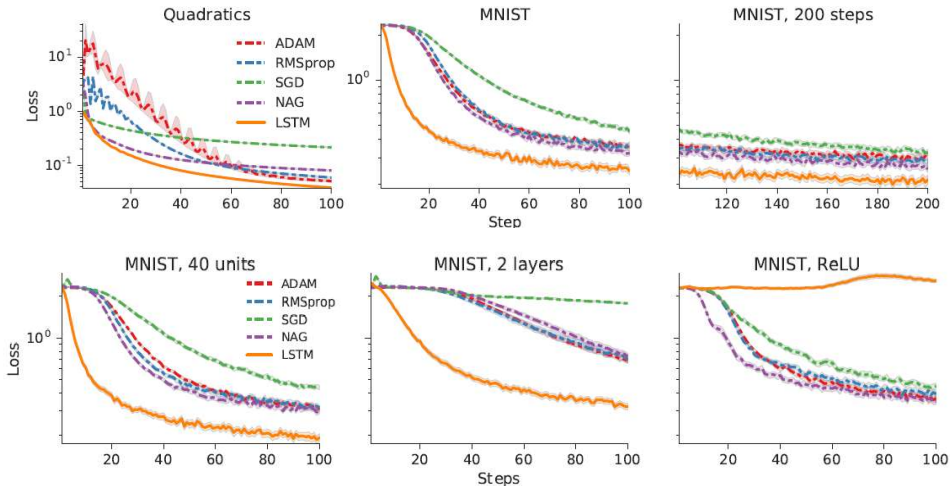
(a) Forget gate values for 1-shot meta-learner



(b) Input gate values for 1-shot meta-learner

Gradient Descent as LSTM (实验结果)

51



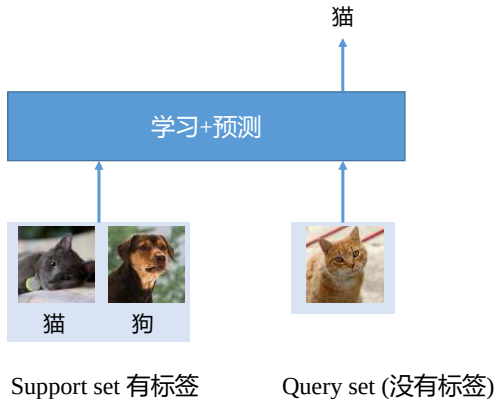
元学习的形式化

- 从已知label集合中采样一个子集: $L \subset \mathcal{L}$ (对应一个任务)
- 采样得到对应的训练集 S^L 和测试集 B^L : $S^L \subset \mathcal{D}$ 且 $B^L \subset \mathcal{D}$
- 元学习的优化目标如下

$$F^* = \arg \min_F \mathbb{E}_{L \subset \mathcal{L}} \left[\mathbb{E}_{S^L \subset \mathcal{D}, B^L \subset \mathcal{D}} \left[\sum_{(x,y) \in B^L} P_F(y|x, S^L) \right] \right]$$

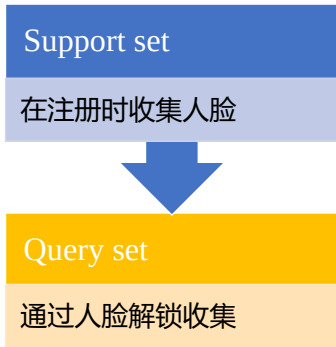
	基于模型的	基于度量的	基于优化的
核心理念	RNN; memory	Metric Learning	Gradient Descent
如何建模 $P_F(y x, S^L)$	$f_\theta(x, S^L)$	$\sum_{(x,y) \in S^L} k_\theta(x, x_i) y_i$	$P_{g_\phi(\theta, S^L)}(y x)$

基于度量的元学习

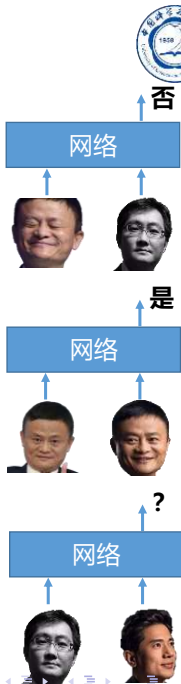
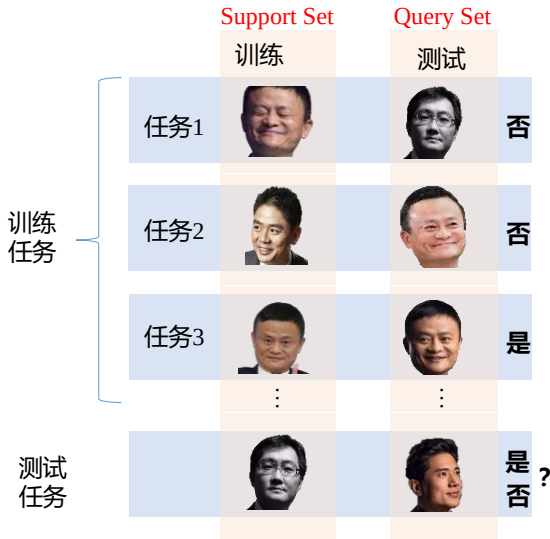


人脸识别是小样本学习

- 在每个任务中



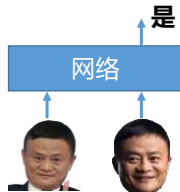
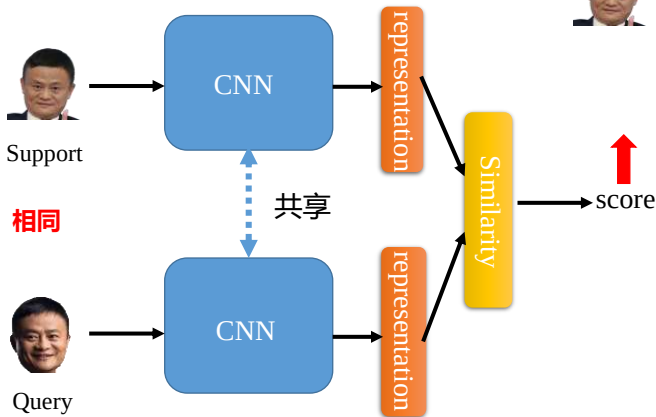
基于度量的元学习



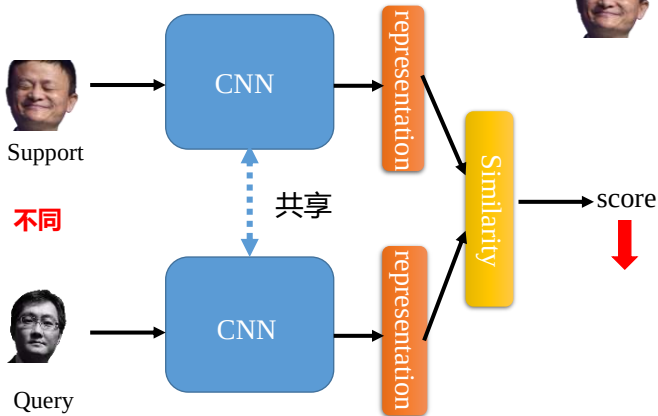
Siamese Network



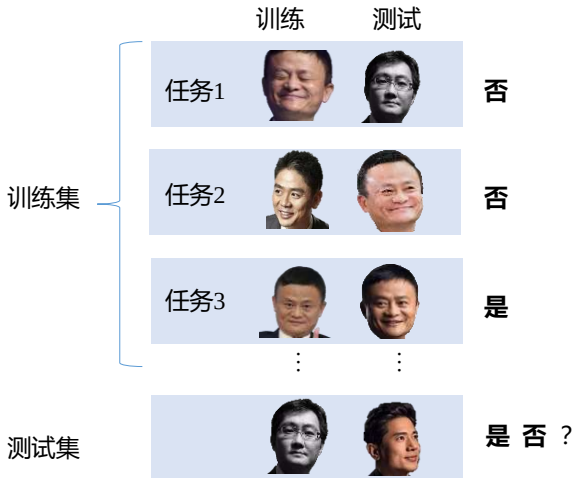
56



Siamese Network

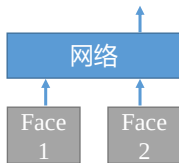


Siamese Network的直观解释



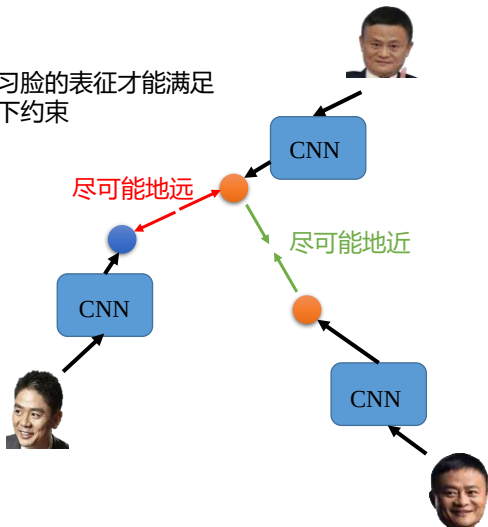
二分类任务

是否是同一张脸



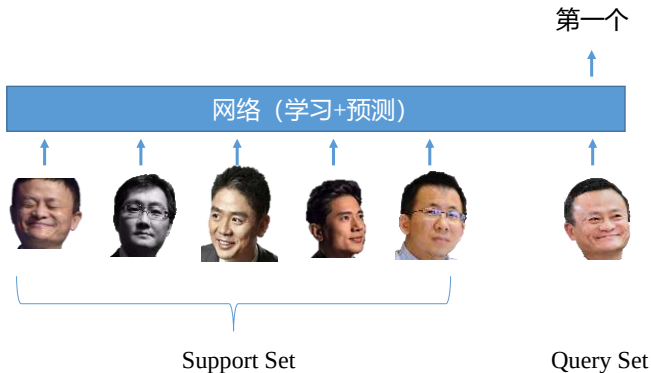
Siamese Network的直观解释

学习脸的特征才能满足
如下约束



N-way Few-shot Learning

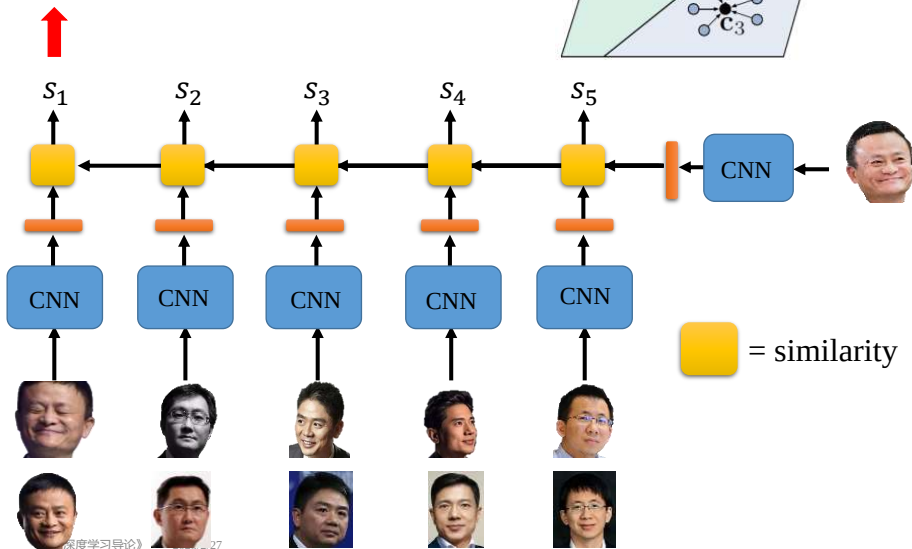
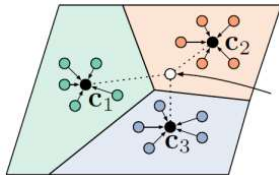
- 5-ways 1-shot



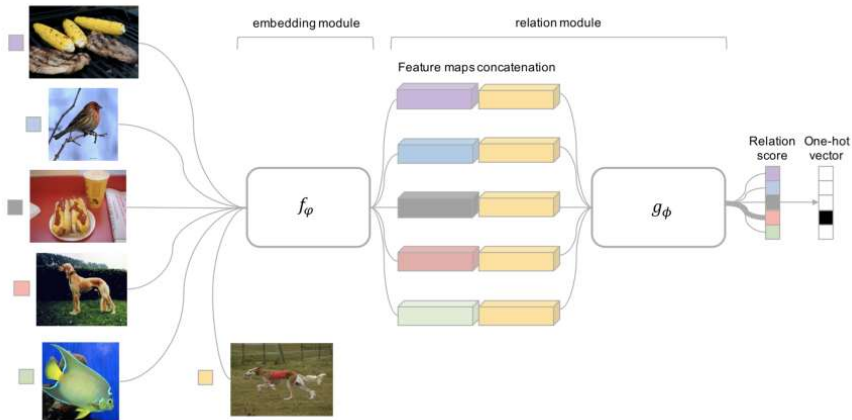
Prototypical Network



62



Relation Network



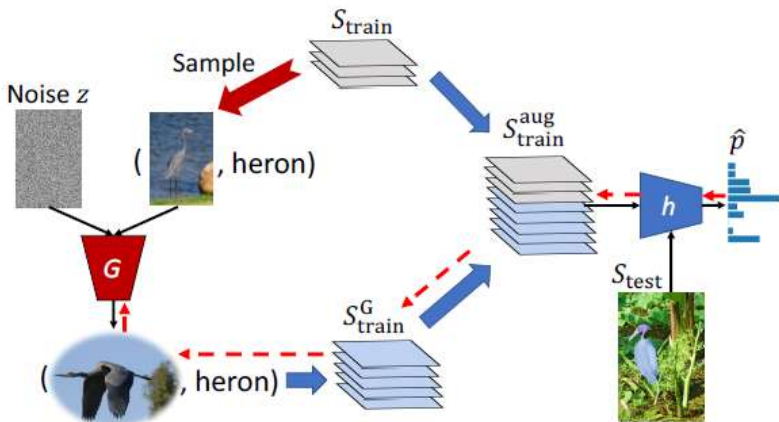
Few-shot learning for imaginary data



<https://arxiv.org/abs/1801.05401>

Few-shot learning for imaginary data

65





谢谢





深度强化学习简介

主讲：连德富 特任教授 | 博士生导师

邮箱：liandefu@ustc.edu.cn

手机：13739227137

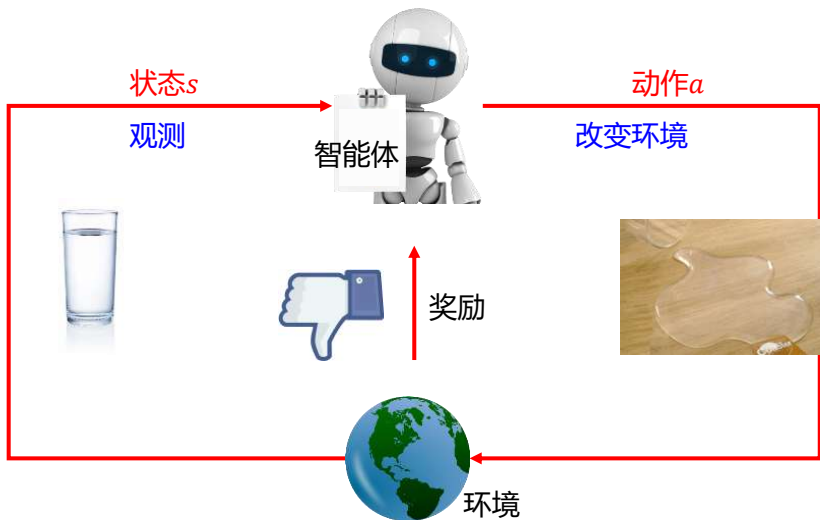
主页：<http://staff.ustc.edu.cn/~liandefu>



本章内容

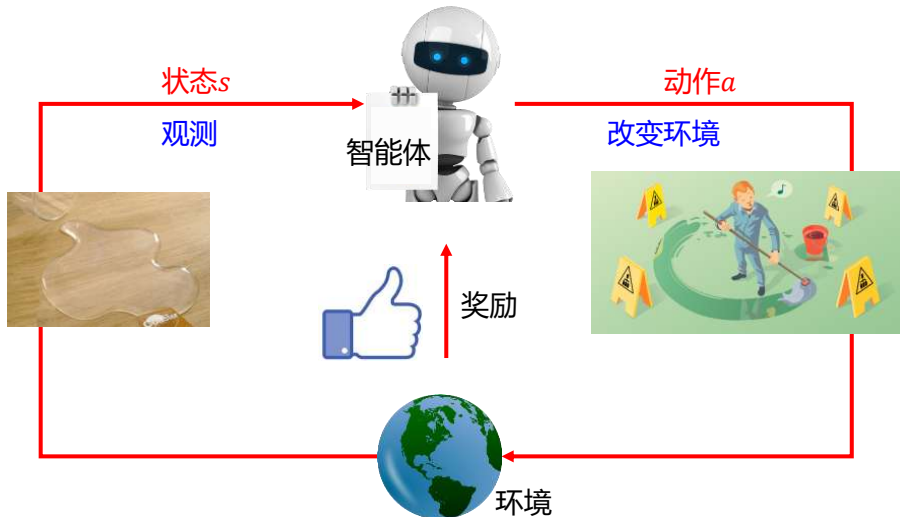
- 强化学习基础
- 深度强化学习
 - 策略梯度
 - DQN
 - Actor-Critic
 - PPO/TRPO

强化学习基础



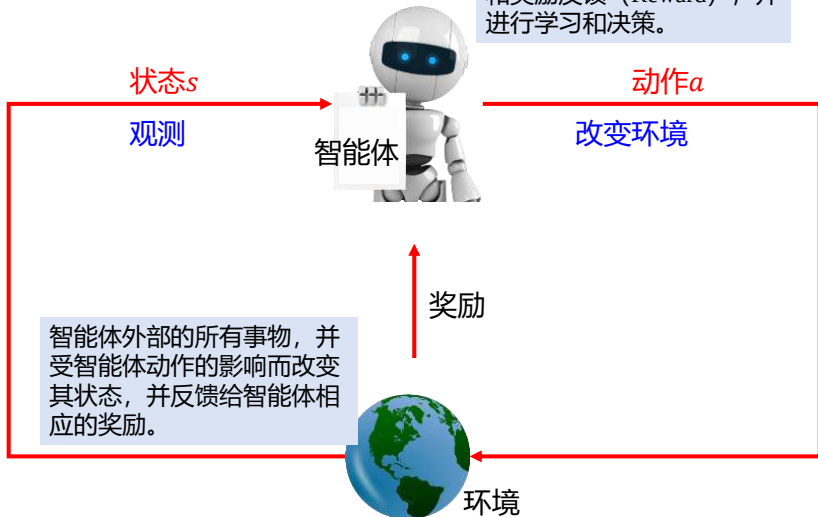
强化学习基础

智能体学习如何采取行动以最大化期望奖励

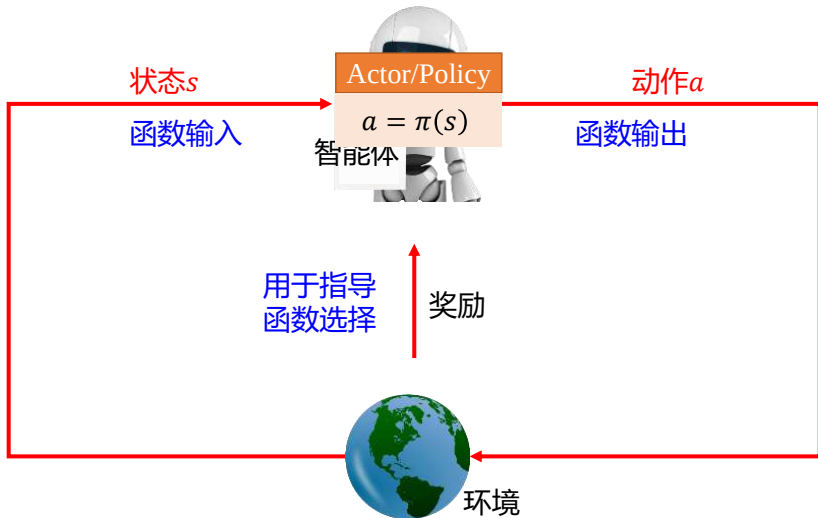


强化学习基础

感知外界环境的状态 (State) 和奖励反馈 (Reward), 并进行学习和决策。

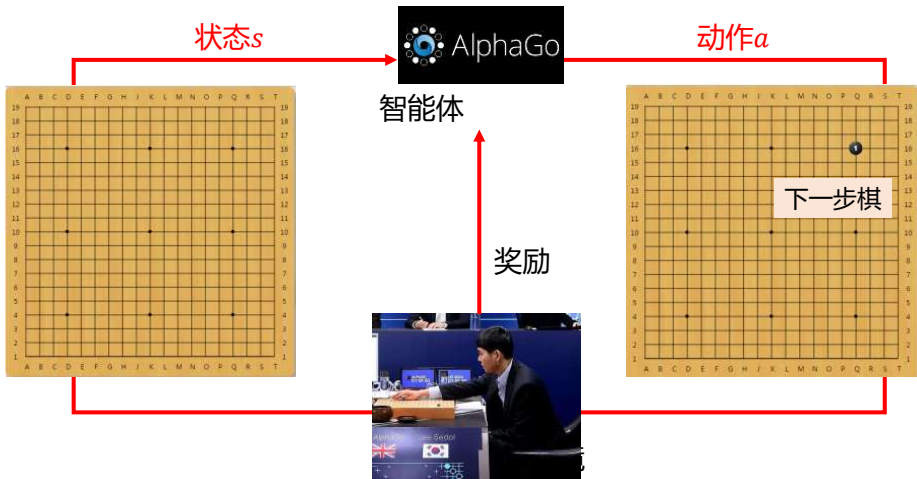


强化学习基础



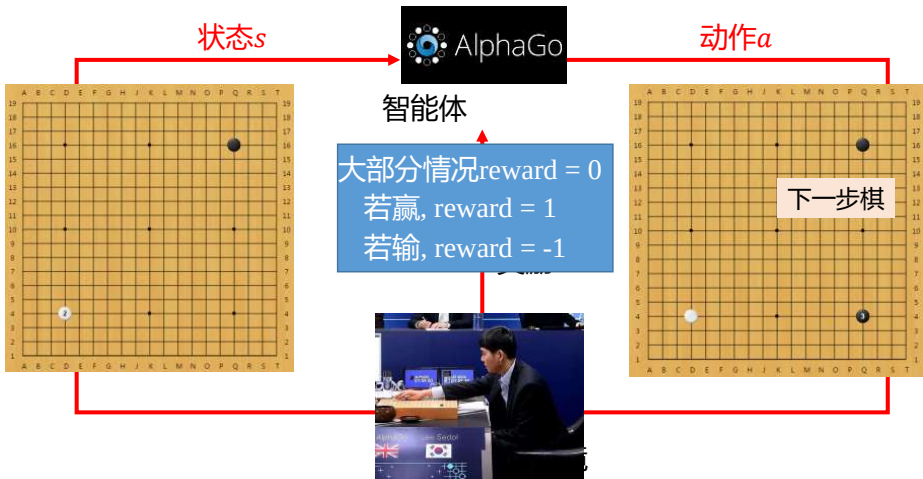
强化学习案例—下围棋

7



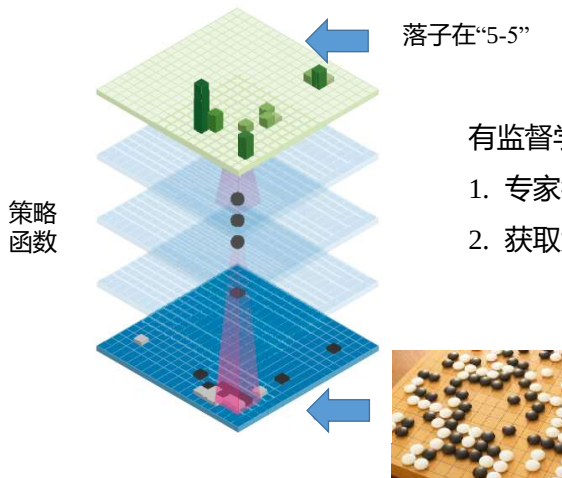
强化学习案例—下围棋

智能体学习如何采取行动以最大化期望奖励



强化学习案例—下围棋

• 有监督学习



有监督学习的挑战：

1. 专家很难给出正确的落子位置
2. 获取大量数据的成本很高

强化学习案例—玩游戏（太空侵略者）

终止状态：所有敌人被消灭或者飞船被摧毁

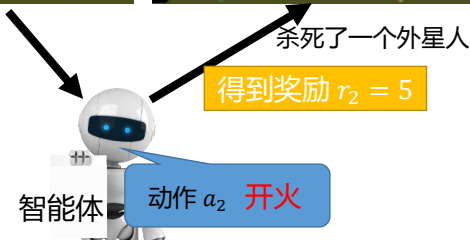
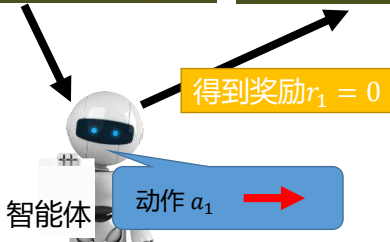
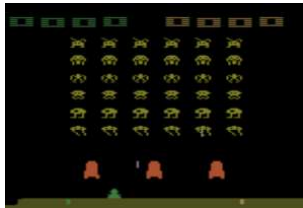


强化学习案例—玩游戏（太空侵略者）

开始状态 s_1

状态 s_2

状态 s_3

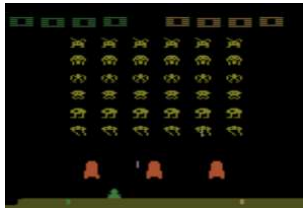


强化学习案例—玩游戏（太空侵略者）

开始状态 s_1



状态 s_2



状态 s_3



在很多轮之后



进入终止态

从交互开始到结束过程
称为一个回合 (episode)

得到奖励 r_T



智能体

动作 a_T

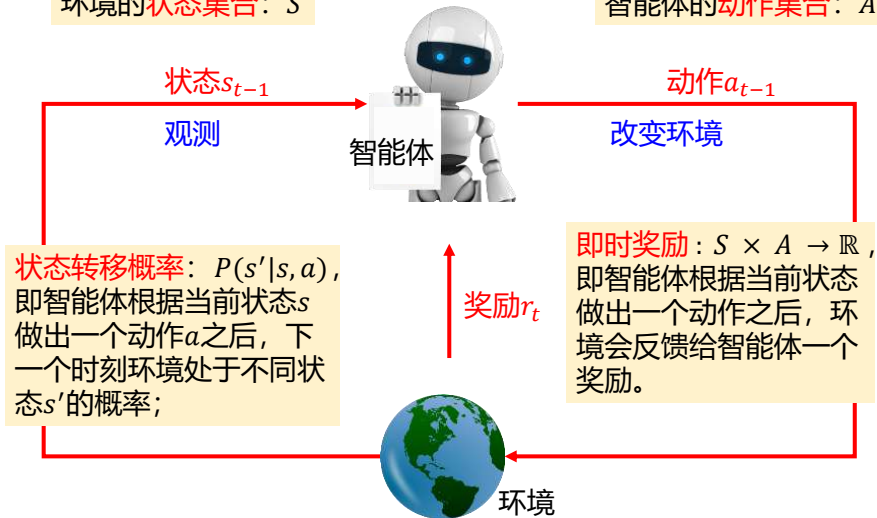
强化学习目标：学习最大化
每个回合的期望累积得分

$$E[r_1 + r_2 + \dots + r_T]$$

强化学习中的基本要素

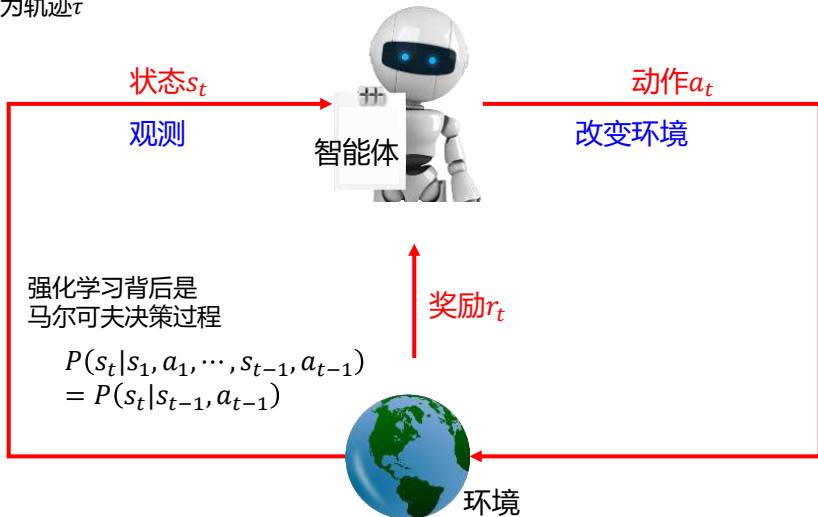
环境的**状态集合**： S

智能体的**动作集合**： A



强化学习基础

交互历史, $\tau = s_1, a_1, r_1, \dots, s_t, a_t, r_t, \dots, s_T, a_T, r_T$
称为轨迹 τ

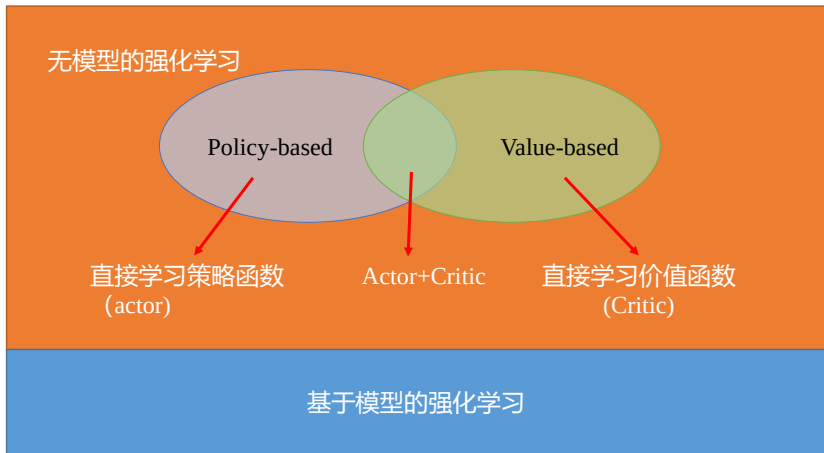


强化学习的特性

- 奖励会有延迟
 - 在太空侵略者游戏中，只有“开火”动作有奖励；但是在开火前的左右移动是非常重要的
 - 在下围棋的时候，可能会牺牲短期的奖励以获得长期奖励
- 智能体的动作会影响到后续接收到的数据

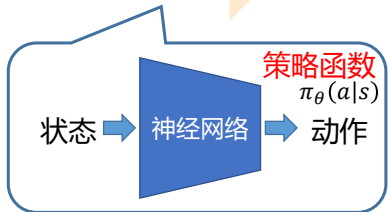
强化学习的分类

模型：状态转移概率和奖励函数



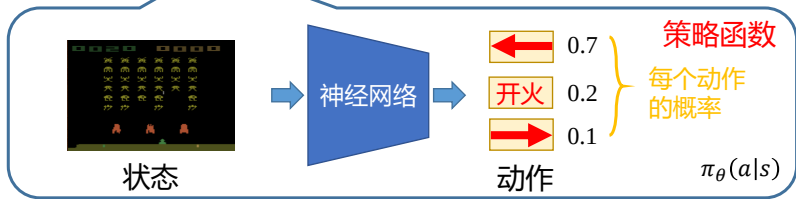
基于策略的方法—直接学习策略函数

- 深度学习的三个步骤



基于策略的方法—直接学习策略函数

- 深度学习的三个步骤



基于策略的方法—评估策略好坏

- 深度学习的三个步骤



- 用策略函数 $\pi_{\theta}(a|s)$ 去和环境交互，一个回合的轨迹 τ
 $s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_T, a_T, r_T$ 其奖励 $R(\tau) = \sum_{t=1}^T r_t$
- 由于 $\pi_{\theta}(a|s)$ 的随机性，强化学习最大化期望回报

$$\bar{R}_{\theta} = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[R(\tau)] = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[\sum_{t=1}^T r_t]$$

基于策略的方法—评估策略好坏

轨迹 $\tau = s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_T, a_T, r_T$

$$P_{\theta}(\tau) = P(s_1)P_{\theta}(a_1|s_1)P(r_1, s_2|s_1, a_1)P_{\theta}(a_2|s_2)P(r_2, s_3|s_2, a_2) \dots$$

$$= P(s_1) \prod_{t=1}^T \underbrace{P_{\theta}(a_t|s_t)}_{\text{由策略函数确定}} \underbrace{P(r_t, s_{t+1}|s_t, a_t)}_{\text{模型}}$$

和策略函数无关,
由环境确定

由策略函数确定

期望回报 $\bar{R}_{\theta} = \mathbb{E}_{\tau \sim P_{\theta}(\tau)}[R(\tau)]$

$$= \sum_{\tau} R(\tau) P_{\theta}(\tau) \approx \frac{1}{N} \sum_{n=1}^N R(\tau^n)$$

从分布 $P_{\theta}(\tau)$ 中采样 N 个轨迹
 $\{\tau^1, \tau^2, \dots, \tau^N\}$

用 $\pi_{\theta}(a|s)$ 和环境交互 N 次,
得到 N 条轨迹

基于策略的方法—策略更新

- 深度学习的三个步骤



- $\theta^* = \arg \max_{\theta} \bar{R}_{\theta}$

- 用梯度上升法求解

Start with θ^0

$$\theta^1 \leftarrow \theta^0 + \eta \nabla \bar{R}_{\theta^0}$$

$$\theta^2 \leftarrow \theta^1 + \eta \nabla \bar{R}_{\theta^1}$$

.....

基于策略的方法—策略梯度

$$\bar{R}_\theta = \sum_{\tau} R(\tau) P_\theta(\tau) \quad \rightarrow \quad \nabla \bar{R}_\theta = ?$$

$$\begin{aligned} \nabla \bar{R}_\theta &= \sum_{\tau} R(\tau) \nabla P(\tau|\theta) = \sum_{\tau} R(\tau) P_\theta(\tau) \frac{\nabla P_\theta(\tau)}{P_\theta(\tau)} \\ &= \sum_{\tau} R(\tau) P_\theta(\tau) \nabla \log P_\theta(\tau) \\ &\approx \frac{1}{N} \sum_{n=1}^N R(\tau^n) \nabla \log P_\theta(\tau) \end{aligned}$$

用 $\pi_\theta(a|s)$ 和环境交互N次,
得到N条轨迹

$$\{\tau^1, \tau^2, \dots, \tau^N\}$$

基于策略的方法—策略梯度

轨迹 $\tau = s_1, a_1, r_1, \dots, s_{t-1}, a_{t-1}, r_{t-1}, \dots, s_T, a_T, r_T$

$$P_{\theta}(\tau) = \underbrace{P(s_1)}_{\text{和策略函数无关}} \prod_{t=1}^T \underbrace{P_{\theta}(a_t|s_t)}_{\text{由策略函数确定}} \underbrace{P(r_t, s_{t+1}|s_t, a_t)}_{\text{由策略函数确定}}$$

和策略函数无关

由策略函数确定

$$\log P_{\theta}(\tau) = \log P(s_1) + \sum_{t=1}^T \log P_{\theta}(a_t|s_t) + \log P(r_t, s_{t+1}|s_t, a_t)$$

$$\nabla \log P_{\theta}(\tau) = \sum_{t=1}^T \nabla \log P(a_t|s_t, \theta)$$

基于策略的方法—策略梯度

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}} \quad \nabla \log P_{\theta}(\tau) = \sum_{t=1}^T \nabla \log P_{\theta}(a_t | s_t)$$

$$\begin{aligned} \nabla \bar{R}_{\theta} &\approx \frac{1}{N} \sum_{n=1}^N R(\tau^n) \nabla \log P_{\theta}(\tau) = \frac{1}{N} \sum_{n=1}^N R(\tau^n) \sum_{t=1}^{T_n} \nabla \log P_{\theta}(a_t^n | s_t^n) \\ &= \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} R(\tau^n) \nabla \log P_{\theta}(a_t^n | s_t^n) \end{aligned}$$

如果在轨迹 τ^n 中，智能体在状态 s_t^n 时采取了动作 a_t^n

- 如果 $R(\tau^n) > 0$ 更新参数 θ ，以增加概率 $P(a_t^n | s_t^n)$
- 如果 $R(\tau^n) < 0$ 更新参数 θ ，以降低概率 $P(a_t^n | s_t^n)$

基于策略的方法—策略梯度

用策略函数 π_θ 和环境交互，得到轨迹

$$\begin{aligned} \tau^1: & \begin{pmatrix} (s_1^1, a_1^1, r_1^1) \\ (s_2^1, a_2^1, r_2^1) \\ \vdots \end{pmatrix} R(\tau^1) \\ & \vdots \\ \tau^2: & \begin{pmatrix} (s_1^2, a_1^2, r_1^2) \\ (s_2^2, a_2^2, r_2^2) \\ \vdots \end{pmatrix} R(\tau^2) \end{aligned}$$

收集数据

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

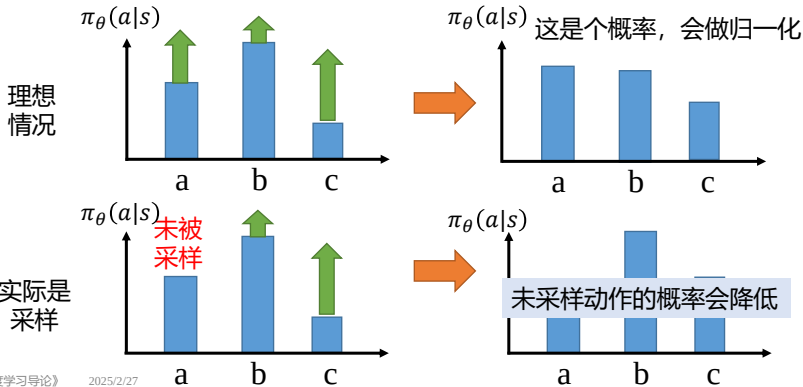
$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} R(\tau^n) \nabla \log P_\theta(a_t^n | s_t^n)$$

更新模型

基于策略的方法——带基线的策略梯度

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p_{\theta}(a_t^n | s_t^n)$$



基于策略的方法—带基线的策略梯度

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} R(\tau^n) \nabla \log P_{\theta}(a_t^n | s_t^n)$$

路径梯度的方差很大

导致训练不稳定

$$T^1 = \{\tau_1^1, \tau_1^2, \dots, \tau_1^N\} \longrightarrow \nabla \bar{R}_{\theta}^1$$

$$T^2 = \{\tau_2^1, \tau_2^2, \dots, \tau_2^N\} \longrightarrow \nabla \bar{R}_{\theta}^2$$

$$T^3 = \{\tau_3^1, \tau_3^2, \dots, \tau_3^N\} \longrightarrow \nabla \bar{R}_{\theta}^3$$

⋮

⋮

$\sum_i (\nabla \bar{R}_{\theta}^i - \text{mean}[\nabla \bar{R}_{\theta}])^2$
会很大

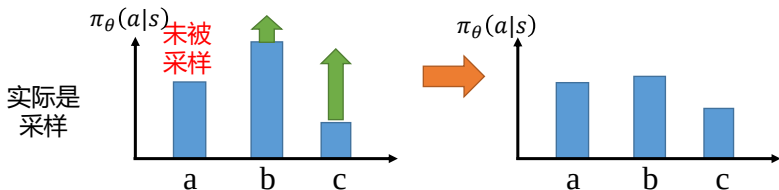
这是在高维空间中使用蒙特卡罗方法的的通病

基于策略的方法—带基线的策略梯度

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (R(\tau^n) - b(s_t^n)) \nabla \log P_{\theta}(a_t^n | s_t^n)$$

不会改
变期望



基于策略的方法——带基线的策略梯度

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (R(\tau^n) - b(s_t^n)) \nabla \log P_{\theta}(a_t^n | s_t^n)$$

不会改变期望

可以减小路径梯度的方差

在 s_1 下采样 a_1, a_2, a_3 对应轨迹的 $R(\tau)$ 分别是200,210,220, 设 $b(s_1) = 210$

在 s_2 下采样 a_1, a_2, a_3 对应轨迹的 $R(\tau)$ 分别是105,110,115, 设 $b(s_2) = 110$

200,210,220,105,110,115

-10,0,10,-5,0,5

方差 >

方差

基于值函数的方法

- 值函数并不直接确定动作，只是评价策略 π_θ 的好坏
- 不过可以从值函数中推导出策略函数

背后的方法就是Q-learning

基于值函数的方法

- 状态值函数 $V^\pi(s)$

表示在观测到状态 s 之后的期望累计收益 $V^\pi(s) = \mathbb{E}_\tau[R(\tau)|s_t = s]$



$V^\pi(s)$ 大

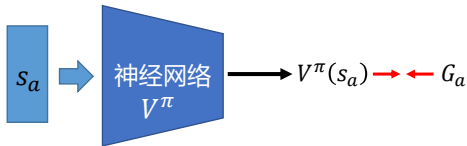


$V^\pi(s)$ 小

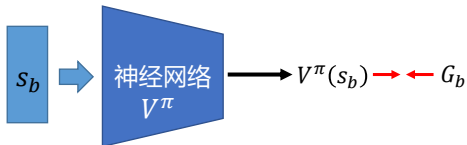
估计状态值函数 $V^\pi(s)$

- 基于蒙特卡洛的方法

在看到状态 s_a 之后，直到回合结束时的累计奖励记为 G_a

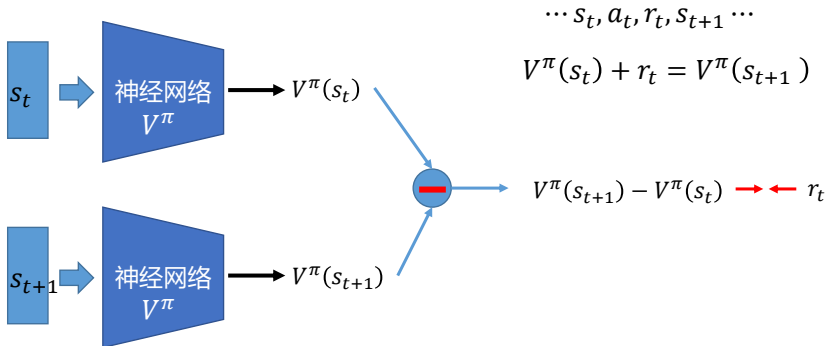


在看到状态 s_b 之后，直到回合结束时的累计奖励记为 G_b

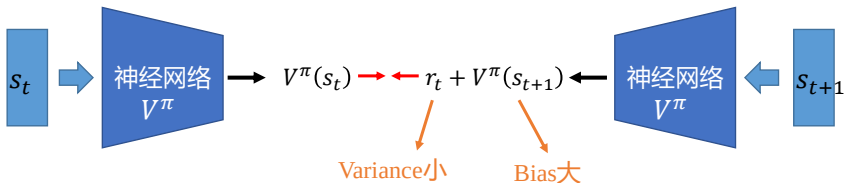
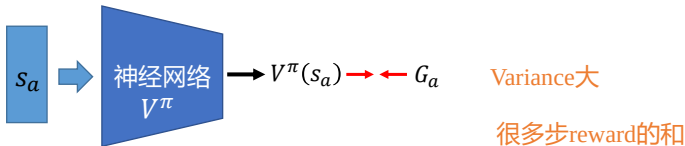


估计状态值函数 $V^\pi(s)$

- 基于时间差分的方法 (Temporal-difference)



估计状态值函数 $V^\pi(s)$



估计状态值函数 $V^\pi(s)$

- 假设由如下8个回合

$s_a, r = 0, s_b, r = 0, \text{END}$

MC $V^\pi(s_b) = 3/4$

$s_b, r = 1, \text{END}$

$s_b, r = 1, \text{END}$

$s_b, r = 1, \text{END}$

$s_b, r = 1, \text{END}$

$s_b, r = 1, \text{END}$

$s_b, r = 1, \text{END}$

$s_b, r = 0, \text{END}$

MC $V^\pi(s_a) = 0$

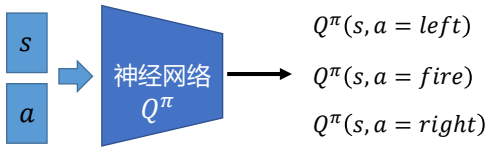
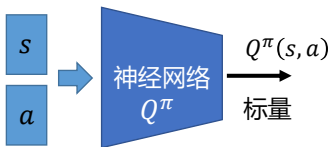
TD $V^\pi(s_a) = V^\pi(s_b) + r = 3/4$

估计状态动作值函数 $Q^\pi(s, a)$

- 状态动作值函数 $Q^\pi(s, a)$

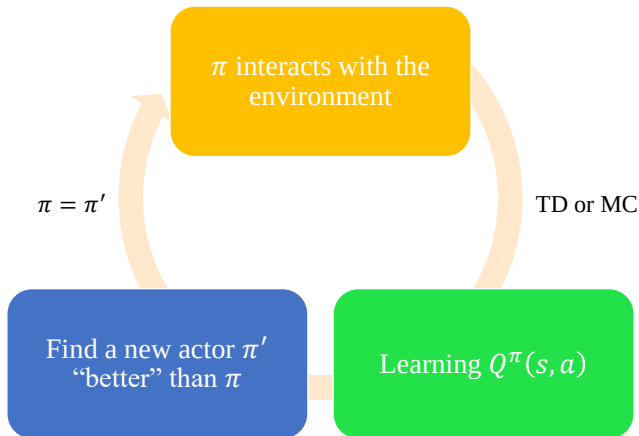
表示在观测到状态 s 且执行动作 a 之后的期望累计收益

$$Q^\pi(s, a) = \mathbb{E}_\tau[R(\tau) | s_t = s, a_t = a]$$

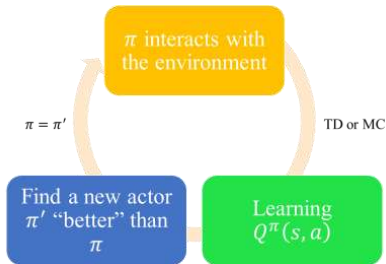


如果动作空间是离散的

Q-learning



Q-learning



- 给定 $Q^\pi(s, a)$, 如何找到更好的策略 π' , 即满足 $V^{\pi'}(s) \geq V^\pi(s), \forall s \in S$

$$\pi'(s) = \arg \max_a Q^\pi(s, a)$$

π' 没有额外的参数, 只依赖于 Q

Q-learning

$$\bullet \pi'(s) = \arg \max_a Q^\pi(s, a)$$

$$V^\pi(s) = Q^\pi(s, \pi(s))$$

$$\leq \max_a Q^\pi(s, a)$$

$$= Q^\pi(s, \pi'(s))$$

$$= \mathbb{E}[r_t + V^\pi(s_{t+1}) | s_t = s, a_t = \pi'(s_t)]$$

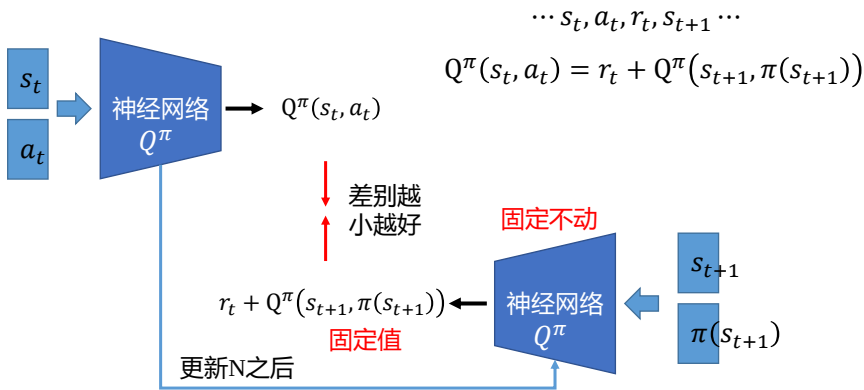
$$V^\pi(s) \leq \max_a Q^\pi(s, a) \leq \mathbb{E}[r_t + Q^\pi(s_{t+1}, \pi'(s_{t+1})) | s_t = s, a_t = \pi'(s_t)]$$

$$\leq \mathbb{E}[r_t + r_{t+1} + V^\pi(s_{t+2}) | s_t = s, \pi'(s_t), s_{t+1}, \pi'(s_{t+1})]$$

$$\leq \mathbb{E}[r_t + r_{t+1} + \dots + r_{t+n} | s_t = s, \pi'(s_t), s_{t+1}, \dots, \pi'(s_{t+n}), s_{t+n}]$$

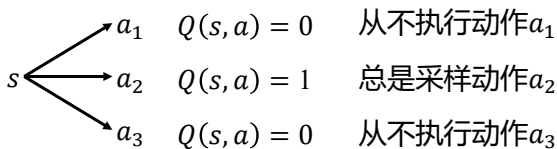
$$\leq V^{\pi'}(s)$$

Q-learning (Target network)



Exploration

- 策略函数是基于Q函数 $\pi'(s) = \arg \max_a Q^\pi(s, a)$



对于数据收集是不利的

Epsilon Greedy

$$\pi'(s) = \begin{cases} \arg \max_a Q^\pi(s, a), & \text{以概率 } 1 - \epsilon \\ \text{random}, & \text{以概率 } \epsilon \end{cases}$$

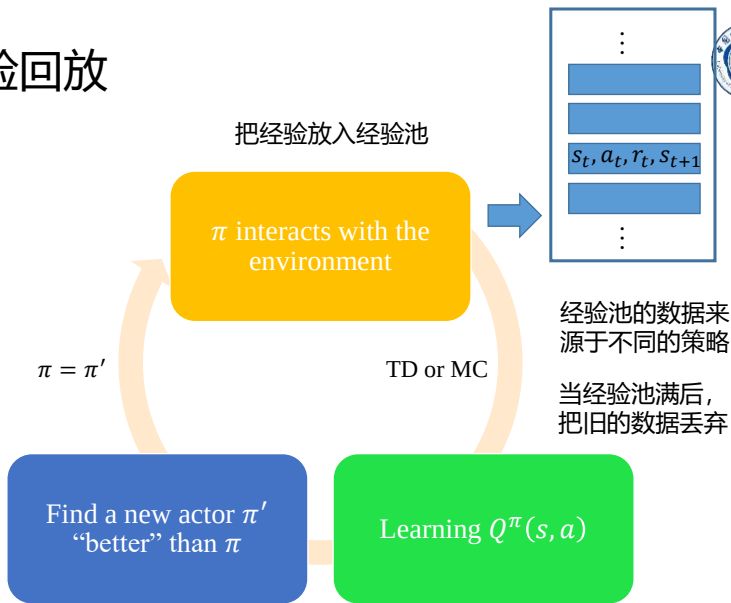
Boltzmann Exploration

$$P(a|s) = \frac{\exp(Q(s, a))}{\sum_a \exp(Q(s, a))}$$

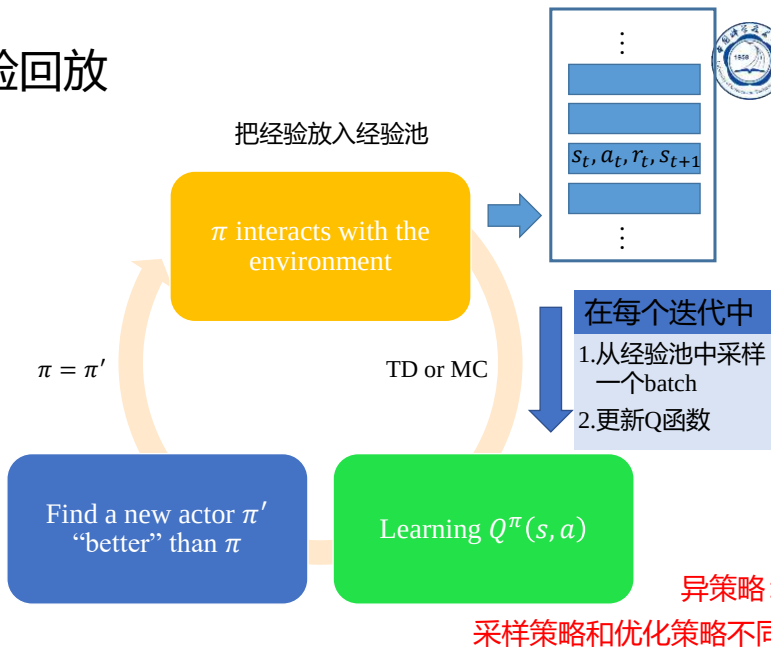
经验回放



42

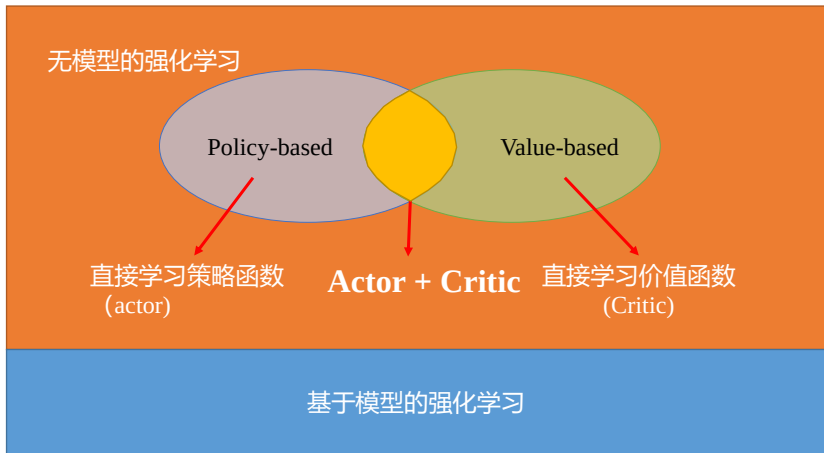


经验回放



强化学习的分类

模型：状态转移概率和奖励函数



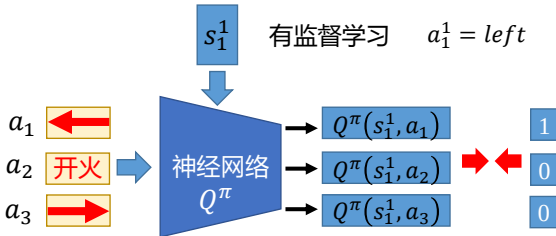
策略梯度优化目标和多分类的关系

用策略函数 π_θ 和环境交互，得到轨迹

$$\begin{aligned} & (s_1^1, a_1^1, r_1^1) \\ \tau^1: & (s_2^1, a_2^1, r_2^1) \quad R(\tau^1) \\ & \vdots \\ & (s_1^2, a_1^2, r_1^2) \\ \tau^2: & (s_2^2, a_2^2, r_2^2) \quad R(\tau^2) \\ & \vdots \end{aligned}$$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} R(\tau^n) \nabla \log P_\theta(a_t^n | s_t^n)$$



$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \nabla \log P_\theta(a_t^n | s_t^n)$$

Actor-Critic

$\times 3$	$\times -2$	$\times -2$	$\times -7$	$\times -2$	$\times -2$
(s_a, a_1)	(s_b, a_2)	(s_c, a_3)	(s_a, a_2)	(s_b, a_2)	(s_c, a_3)
$+5$	$+0$	-2	-5	$+0$	-2
$R = +3$			$R = -7$		

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (R(\tau^n) - b(s_t^n)) \nabla \log P_\theta(a_t^n | s_t^n)$$

$$\sum_{k=t}^{T_n} r_k^n \longrightarrow \sum_{k=t}^{T_n} \gamma^{k-t} r_k^n$$

$$R(\tau_{t:T_n}^n)$$

$\gamma \in [0,1]$ 是折扣率

- 当 γ 接近于0时, 智能体更在意短期回报;
- 当 γ 接近于1时, 长期回报变得更重要

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} \left(\underbrace{\sum_{k=t}^{T_n} \gamma^{k-t} r_k^n}_{R(\tau_{t:T_n}^n)} - \underbrace{b(s_t^n)}_{V^\pi(s_t^n)} \right) \nabla \log P_\theta(a_t^n | s_t^n)$$

$R(\tau_{t:T_n}^n)$ 表示在状态 s_t^n 下采取动作 a_t^n 的累计收益

$$\mathbb{E}[R(\tau_{t:T_n}^n)] = Q^\pi(s_t^n, a_t^n) \quad V^\pi(s_t^n) = \mathbb{E}_{a_t^n \sim \pi(s_t^n)}[Q^\pi(s_t^n, a_t^n)]$$

→
$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (Q^\pi(s_t^n, a_t^n) - V^\pi(s_t^n)) \nabla \log P_\theta(a_t^n | s_t^n)$$

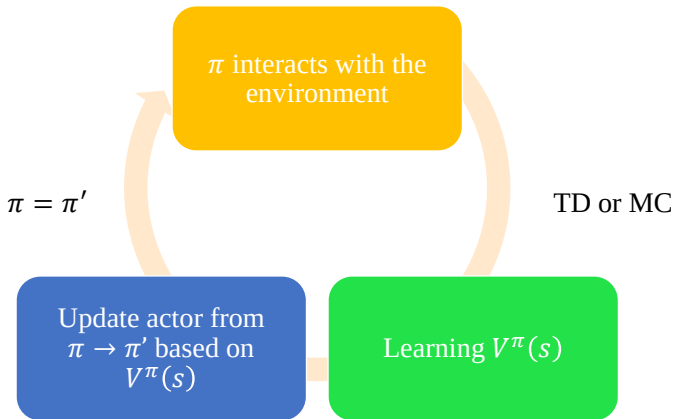
需要估计Q网络和V网络

$$Q^\pi(s_t^n, a_t^n) = \mathbb{E}[r_t^n + \gamma V^\pi(s_{t+1}^n)] \approx r_t^n + \gamma V^\pi(s_{t+1}^n)$$

→
$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (r_t^n + \gamma V^\pi(s_{t+1}^n) - V^\pi(s_t^n)) \nabla \log P_\theta(a_t^n | s_t^n)$$

直接一个V网络就搞定

Advantage Actor-Critic



$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} (r_t^n + V^\pi(s_{t+1}^n) - V^\pi(s_t^n)) \nabla \log P_\theta(a_t^n | s_t^n)$$

Advantage Function, 记为 $A(s_t, a_t)$

输入: 状态空间 $\mathcal{S}$, 动作空间 $\mathcal{A}$;

可微分的策略函数 $\pi_{\theta}(a|s)$;

可微分的状态值函数 $V_{\phi}(s)$;

折扣率 γ , 学习率 $\alpha > 0, \beta > 0$;

1 随机初始化参数 θ, ϕ ;

2 repeat

3 初始化起始状态 s ;

4 $\lambda = 1$;

5 repeat

6 在状态 s , 选择动作 $a = \pi_{\theta}(a|s)$;

7 执行动作 a , 得到即时奖励 r 和新状态 s' ;

8 $\delta \leftarrow r + \gamma V_{\phi}(s') - V_{\phi}(s)$;

9 $\phi \leftarrow \phi + \beta \delta \frac{\partial}{\partial \phi} V_{\phi}(s)$;

10 $\theta \leftarrow \theta + \alpha \lambda \delta \frac{\partial}{\partial \theta} \log \pi_{\theta}(a|s)$;

11 $\lambda \leftarrow \gamma \lambda$;

12 $s \leftarrow s'$;

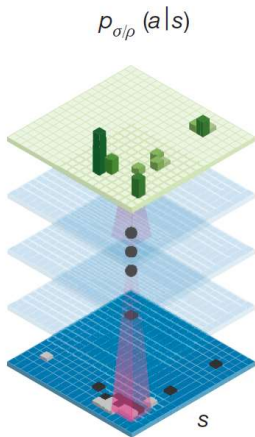
13 until s 为终止状态;

14 until θ 收敛;

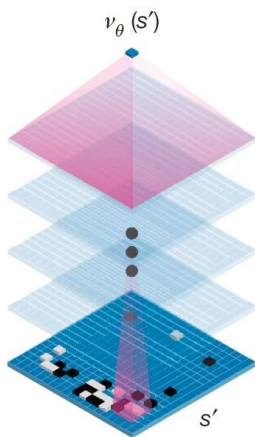
输出: 策略 π_{θ}

AlphaGo

Policy network

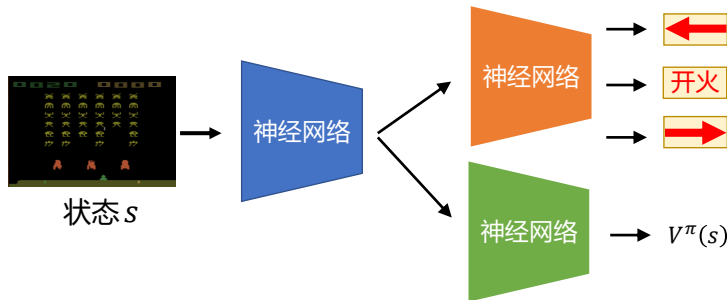


Value network



Advantage Actor-Critic的实现技巧

- Actor (策略网络) 和Critic (网络) 可以参数共享



- 对策略函数的概率输出施加熵正则，以支持更多的探索过程

从同策略到异策略

- 同策略：优化的策略和交互的策略是相同的
- 异策略：优化的策略和交互的策略是相同的

$$\nabla \bar{R}_\theta = E_{\tau \sim P_\theta(\tau)} [R(\tau) \nabla \log P_\theta(\tau)]$$

- 用 π_θ 收集数据，每当更新 θ 时，我们须重新采样数据
- 目标：用从 $\pi_{\theta'}$ 收集的数据来训练 θ 。由于 θ' 固定，可以多次重用历史数据

从同策略到异策略

$$\nabla \bar{R}_\theta = E_{\tau \sim P_\theta(\tau)} [R(\tau) \nabla \log P_\theta(\tau)]$$

- 用 π_θ 收集数据，每当更新 θ 时，我们须重新采样数据
- 目标：用从 $\pi_{\theta'}$ 收集的数据来训练 θ 。由于 θ' 固定，可以多次重用历史数据

$$\nabla \bar{R}_\theta = E_{\tau \sim P_{\theta'}(\tau)} \left[\frac{P_\theta(\tau)}{P_{\theta'}(\tau)} R(\tau) \nabla \log P_\theta(\tau) \right]$$

- 从 $\pi_{\theta'}$ 收集数据
- 用收集到的数据来多次训练 π_θ

从同策略到异策略

- 在策略梯度中，每次的梯度为

$$\begin{aligned} & \mathbb{E}_{s_t} \left[\mathbb{E}_{a_t \sim \pi^\theta} \left[A^\theta(s_t, a_t) \nabla \log p_\theta(a_t^n | s_t^n) \right] \right] && A^\theta(s_t, a_t) \text{表示在策略} \pi_\theta \text{下的 Advantage function} \\ &= \mathbb{E}_{s_t} \left[\mathbb{E}_{a_t \sim \pi^{\theta'}} \left[\frac{p_\theta(a_t^n | s_t^n)}{p_{\theta'}(a_t^n | s_t^n)} \overbrace{A^{\theta'}(s_t, a_t)}^{\text{这项来源于和环境交互}} \nabla \log p_\theta(a_t^n | s_t^n) \right] \right] \end{aligned}$$

- 优化的目标 $J^{\theta'}(\theta)$

$$J^{\theta'}(\theta) = \mathbb{E}_{s_t} \left[\mathbb{E}_{a_t \sim \pi^{\theta'}} \left[\frac{p_\theta(a_t | s_t)}{p_{\theta'}(a_t | s_t)} A^{\theta'}(s_t, a_t) \right] \right] \quad \text{用} \pi_{\theta'} \text{收集的数据何时不能用?}$$

添加约束： $\pi_{\theta'}$ 和 π_θ 的差别不能太大

$\pi_{\theta'}$ 和 π_θ 差别过大的时候

由于它们都是概率，可以用KL散度来衡量

从同策略到异策略

$$\nabla f(x) = f(x) \nabla \log f(x)$$

Proximal Policy Optimization (PPO)

$$J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\pi_{\theta}, \pi_{\theta'})$$

$$J^{\theta'}(\theta) = \mathbb{E}_{s_t} \left[\mathbb{E}_{a_t \sim \pi^{\theta'}} \left[\frac{p_{\theta}(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t) \right] \right]$$

TRPO (Trust Region Policy Optimization)

$$J_{TRPO}^{\theta'}(\theta) = J^{\theta'}(\theta) \quad KL(\pi_{\theta}, \pi_{\theta'}) < \delta$$



谢谢

