



武汉大学

神经网络在语音识别中的应用

张晓明

杨剑锋

目录 >

- ▶ 腾讯叮当介绍
- ▶ 神经网络原理
- ▶ 语音识别原理
- ▶ 神经网络在语音识别中的应用

第一部分 >



腾讯叮当介绍



腾讯叮当是什么

腾讯AI助手品牌



产品和内部合作策略

侧重垂直领域解决方案，重点打造体验、服务、生活

- 丰富内容服务领域（音乐、资讯、有声读物、知识问答、出行、生活服务）
- 持续优化数据AI驱动的内容/服务推荐模型

底层技术上拉通公司内 AI 技术栈深度合作，应用层大力创新玩法和场景开拓

- BG 内智平、智创、研发联合项目
- BG 外和 AI Lab、优图在语音、图像、NLP、知识图谱等方向深度合作

对外合作输出策略

- 面向B端开放服务方案，发展合作伙伴（B2B2C）
- 面向C端发展个人助手规模，树立品牌形成数据积累
- 软硬结合树立标杆产品，拓展智能硬件品类（音箱、手表、车载、智能家居、机器人）

软件

对标 Alexa，构建面向新交互的完整服务平台，吸引合作伙伴和开发者

硬件

对标 Echo，BG 内部整合软硬件方案，推出智能音箱标杆产品

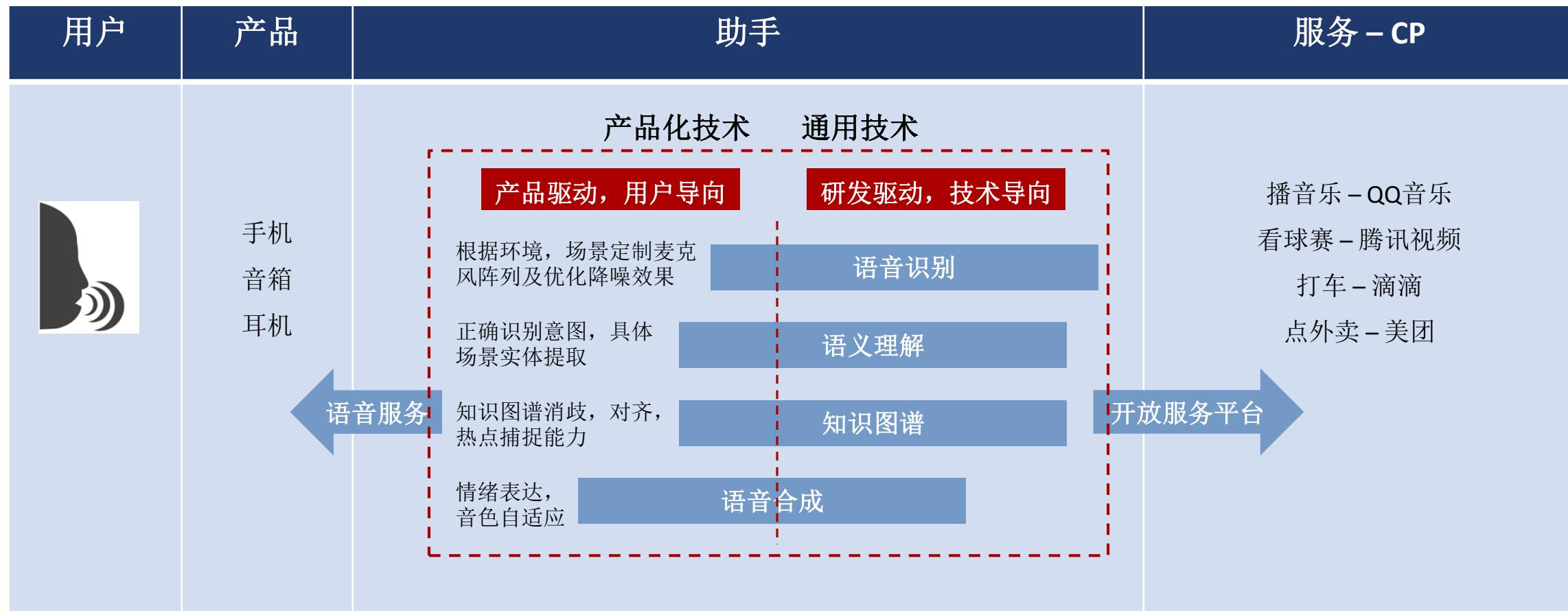


完整的AI技术和内容服务





腾讯叮当平台流程图





语音识别能力

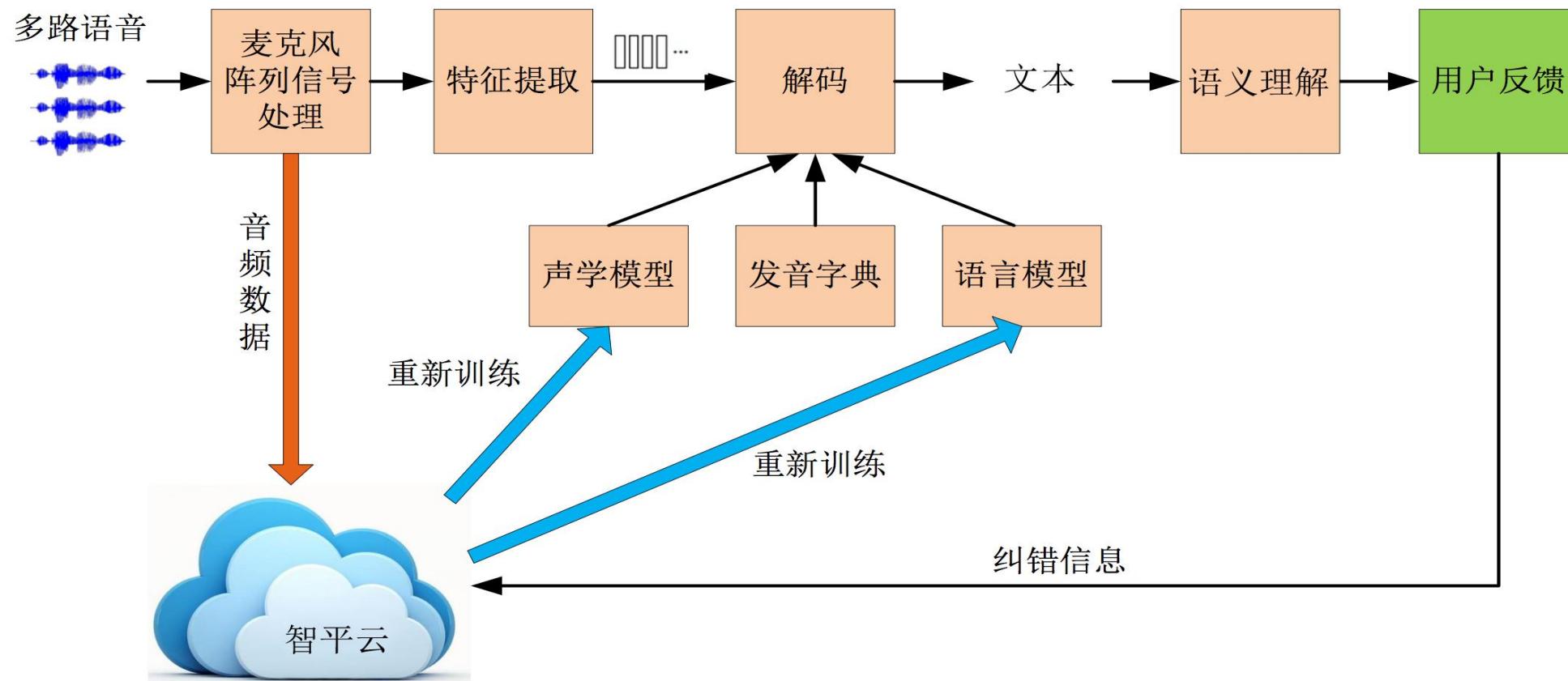
众测用户手机录制的音频，覆盖用户数大于200，文本为网上爬取的日常用语，环境安静基本无噪音								
众测用户录制数据	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	4634	42492	0.70%	1.04%	7.11%	8.86%	91.84%	40.06%
竞品1	4632	43326	0.25%	2.01%	6.48%	8.74%	91.51%	37.80%
竞品2	4630	43310	0.49%	1.00%	6.32%	7.82%	92.67%	33.22%
竞品3	4627	43313	0.34%	1.27%	6.03%	7.64%	92.70%	32.92%

网上爬取并通过人工标注的音频，覆盖用户数大于100，文本包含各种题材，比如百家讲坛、脱口秀、有声小说等，有1/3的数据包含背景音乐噪音								
网上爬取数据集	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	5119	139425	0.67%	4.15%	10.56%	15.37%	85.29%	80.58%
竞品1	5112	139338	0.38%	5.98%	9.29%	15.66%	84.72%	79.30%
竞品2	5121	139431	0.74%	2.79%	8.68%	12.21%	88.53%	72.95%
竞品3	5089	139226	0.61%	4.49%	7.97%	13.07%	87.54%	74.06%

带口音普通话	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	6352	52841	0.40%	1.46%	11.63%	13.49%	86.91%	52.79%
竞品1	6352	52841	0.12%	2.60%	6.81%	9.54%	90.58%	46.22%
竞品2	6352	52841	0.30%	1.02%	7.01%	8.33%	91.97%	37.88%
竞品3	6351	52832	0.25%	1.12%	7.19%	8.56%	91.69%	39.63%



远场识别重点优化



定制化的声学模型和语言模型带来更好的体验 ——
行业情况、交互方式、硬件特性、环境特性、内容焦点、个性化服务



远场识别能力

公司T450笔记本自带麦克风录制，距离大约3m，男女各10名，文本为音乐、天气、新闻、闹钟、电台5个场景，共941句

笔记本双麦录制	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	941	8203	0.80%	3.38%	13.63%	17.81%	82.99%	48.99%

讯飞6麦的麦克风阵列板子录制，距离大约3m，男女各12名，文本为音乐、天气、新闻、闹钟、电台5个场景，共1125句

讯飞6麦阵列录制	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	1125	11460	1.27%	2.67%	12.93%	16.87%	84.40%	55.20%



更自然的语音交互

[用户]: 灵犀灵犀、Hello google、Alexa.....

[系统]: 啾...

[用户]: 明天的天气

[系统]: 明天晴天，微风2级，30度—21度

[用户]: 灵犀灵犀、Hello google、Alexa.....

[系统]: 啾...

[用户]: 后天呢

[系统]: 后天多云转晴，微风3级，29度—22度

[用户]: 叮当，给我放首歌

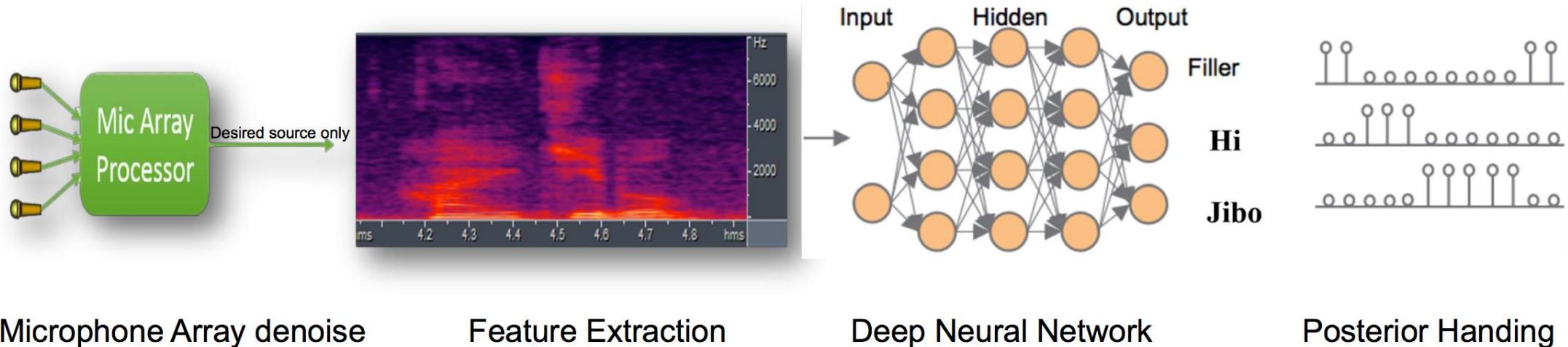
[系统]: hi，想听谁的歌呢

[用户]: 来首陈奕迅的歌

[系统]: 开始播放.....



更自然的语音唤醒



- **Two-Pass关键词识别算法**

- 初选+校验(减小计算量, 降低误识别率)

- **小尺寸模型**

- First-Pass模型的参数数量控制在200k以下

- **低计算量的远场特征**

- 每通道自动增益控制; 幂函数动态范围压缩

- **双音节唤醒词**

- 交互更加自然的, 对模型精度有更高的要求

- **当前识别精度 (双音节唤醒词@每小时0.5次误识)**

- 近场: ~98%
- 0dB 信噪比+混响: ~90%
- 小鸟音箱+背景音乐: ~70% (和前端阵列算法的适配进行中)



云端本地融合的车载识别系统

• 云端识别系统建立（需要场景适配）

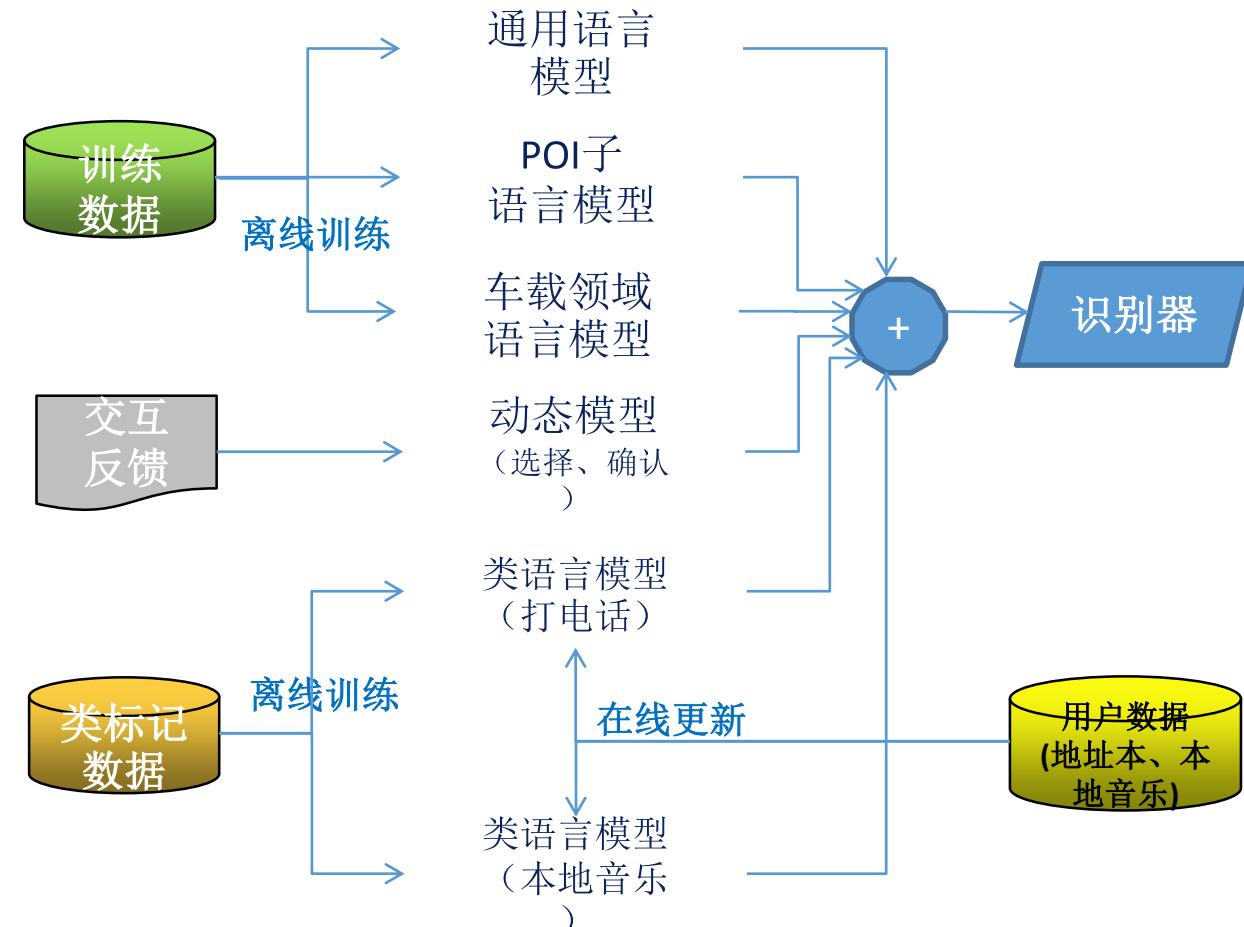
- 抗噪声学模型
 - 外部购买、在线收集车载噪声、语音训练数据
 - 车载环境的混响、噪声模拟训练数据
- 语言模型扩展
 - POI子领域语言模型（覆盖5000万POI条目）
 - 领域语言模型与背景语言模型插值调优
- 置信度优化、Runtime特征输出

• 客户端识别系统建立（保证离线运行）

- 声学模型压缩
 - 缩减深度模型的层数和各层节点数
 - SVD模型压缩（进行中）
- 语言模型优化
 - 通用语言模型裁剪、POI领域语言模型裁剪
 - 类语言模型（人名、本地歌曲等）动态更新

• 云端、客户端识别结果融合

- 基于规则的决策
- 基于runtime特征、云及端识别结果训练分类器(arbitration model)





车载环境下的识别能力

众测用户手机录制，真实车内环境，包含50个用户，文本为车载领域，一共2739多句								
众测用户真实车内人声	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	2739	18848	0.84%	3.07%	6.65%	10.56%	90.28%	34.83%

海天瑞声购买数据，4个mic不同距离录制（60码、80码、开窗），真实车内环境，包含10个用户（5男5女），文本为车载领域，一共8791句								
购买的真实车内人声	总句数	总字数	插入错误率	删除错误率	替换错误率	字错误率	字正确率	句错误率
智平	8791	78058	0.91%	4.56%	20.63%	26.10%	74.81%	74.50%

离线人名、歌曲名识别字错误率小于10%，然后再实际使用中加入文本和音素的多维结果返回



还需优化的一些问题

- 语音识别(词边界、发音)辅助语义消歧

“南京市长江大桥”
或者“南京市 长江 大桥”

- 语义理解容忍语音识别错误

“听披头士的哥”
-> play Beatles' music

- 语义理解上下文提升语音识别正确率

...
[用户]: 人民的名义什么时候播出
[系统]: 人民的名义每晚20点播出
[用户]: 那里面演李达康的演员叫什么名字
...

- 语义理解驱动对话进行信息再确认

[用户]: 爸爸去哪儿什么时候更新
[系统]: 爸爸去哪儿每周五晚22点开播
[用户]: 到时候提醒我
[系统]: 已成功为您增加周五晚22点的提醒，内容是
爸爸去哪儿开播了
[用户]: 提前10分钟提醒
[系统]: 4分钟还是10分钟?
[用户]: 10分钟
[系统]: 修改为周五晚21点50分提醒吗
[用户]: 确定
[系统]: 已成功为您增加周五晚21点50分提醒，内容
是爸爸去哪儿开播了

第二部分 >

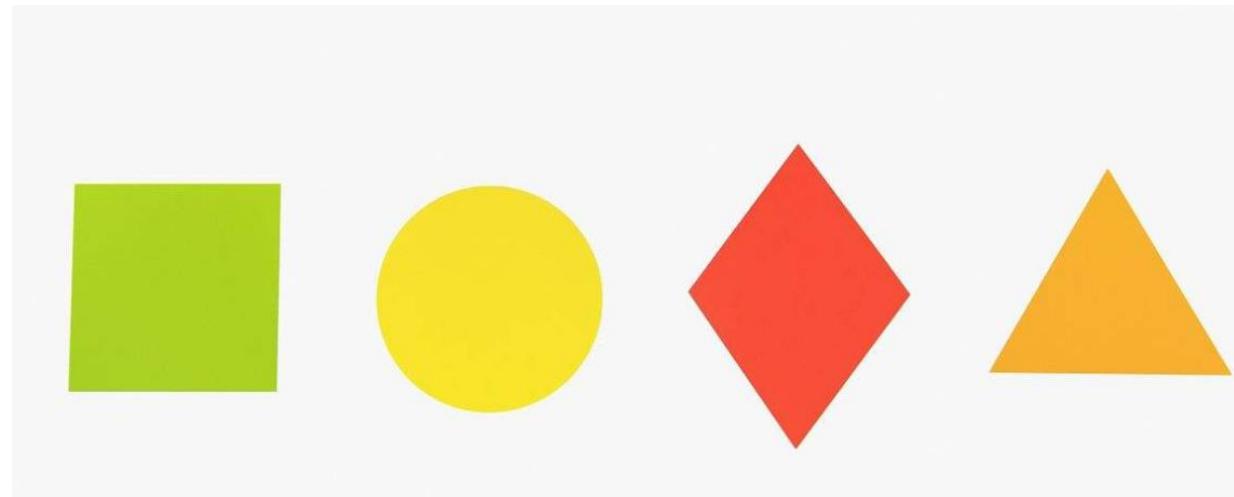


神经网络原理



神经网络能做什么？

- 判断图片中的几何形状



- 识别图中的数字

1 2 3 4 5 6
7 8 9 0 1 2

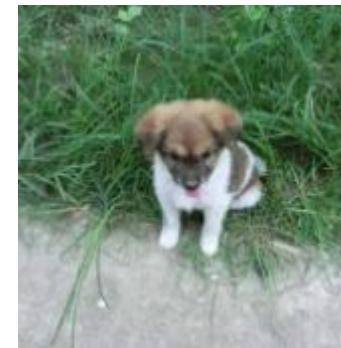


神经网络能做什么？

- 识别手写体数字

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

- 识别图片中的动物

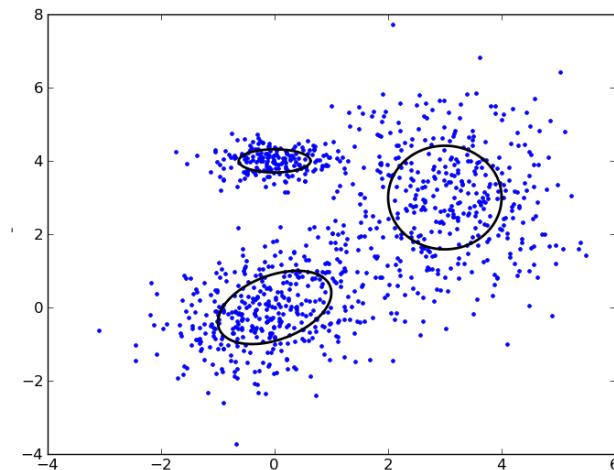
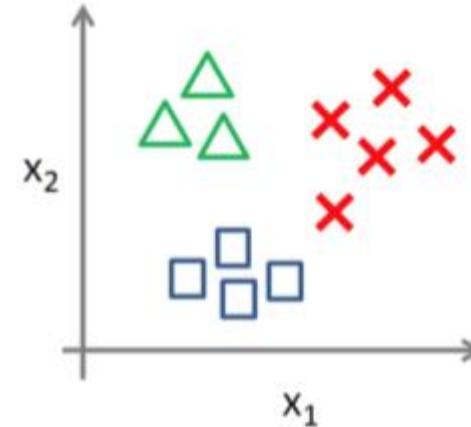




分类问题

- 二分类问题
- 多分类问题
- 典型如ImageNet图像识别
- 高斯模型等传统解决方法

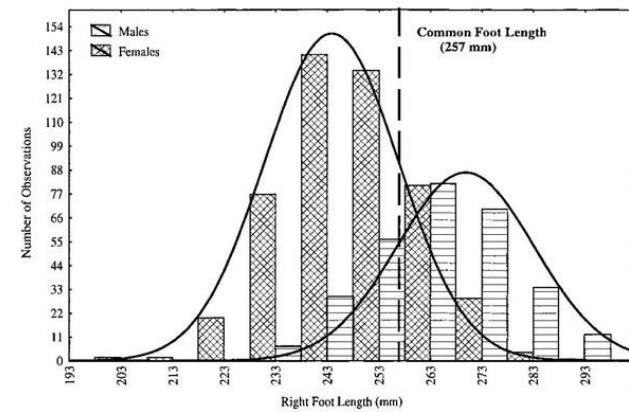
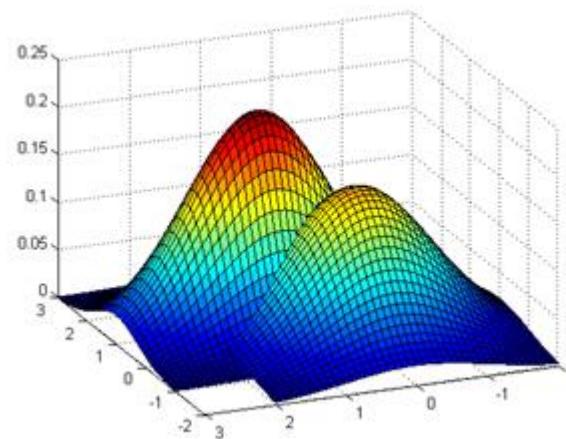
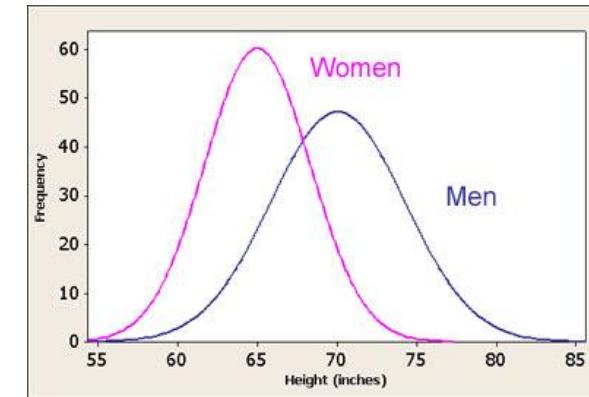
Multi-class classification:





高斯模型

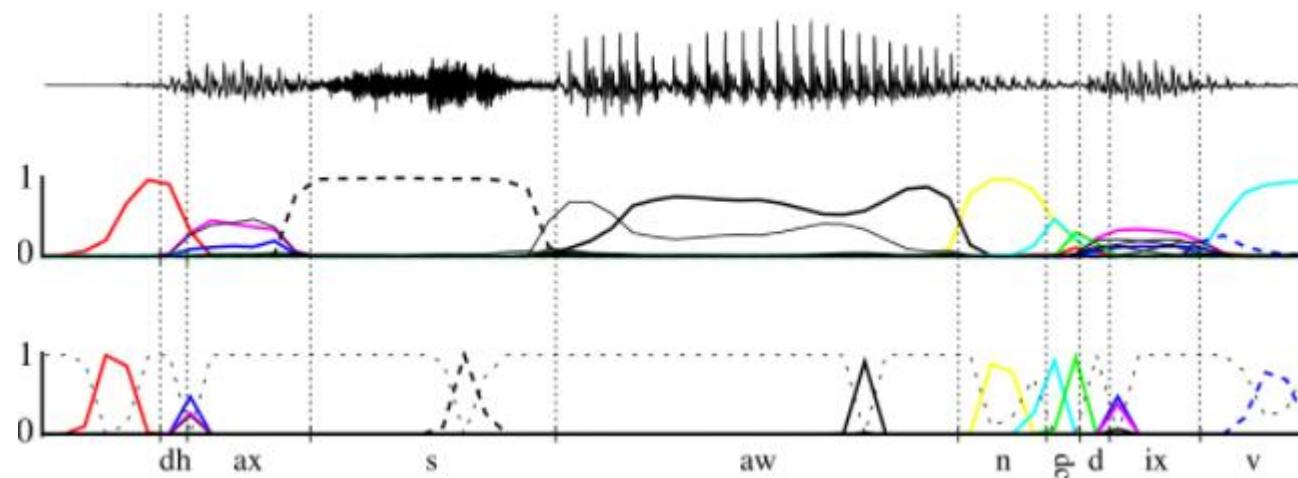
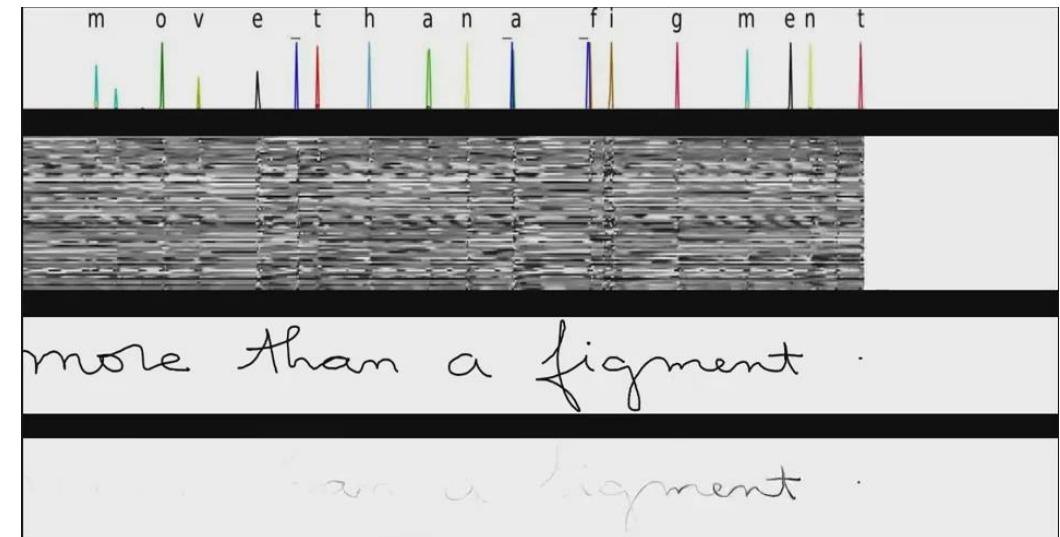
- 通过概率来量化
- 多用于分类问题
- 比如男女分类
- 知道了身高
- 知道了脚长
- 知道了头发长度
-





序列分类问题

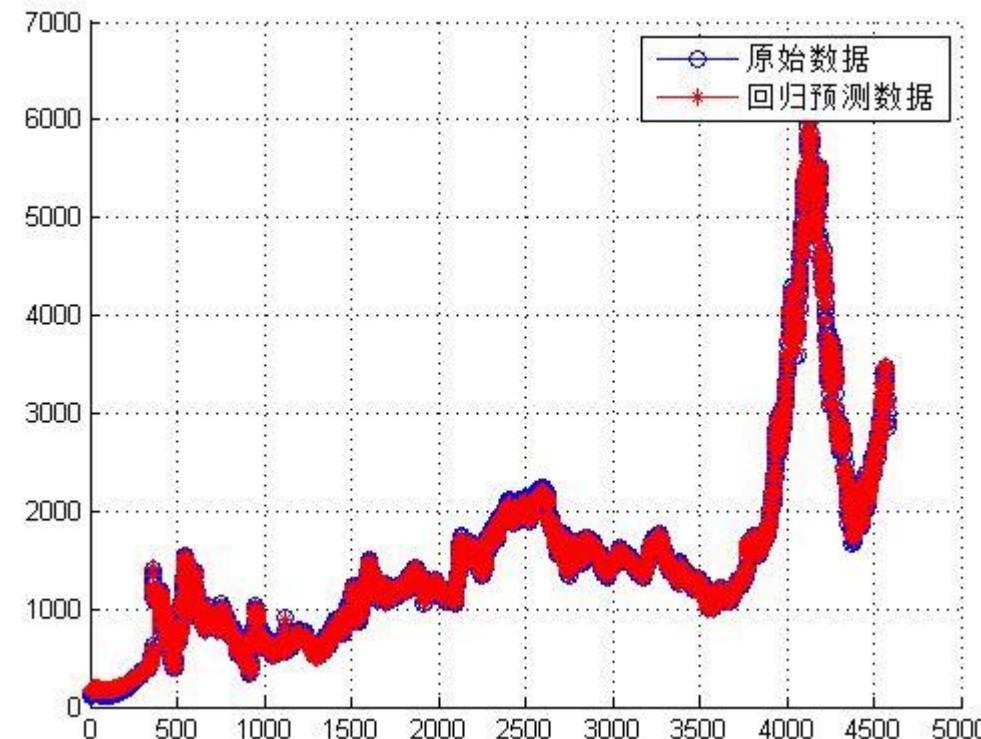
- 问题的输入是序列A
- 问题的输出是序列B
- B的长度小于或者等于A
- 手写识别问题
- 语音识别问题





回归问题

- 给定一个输入
- 预测对应的输出

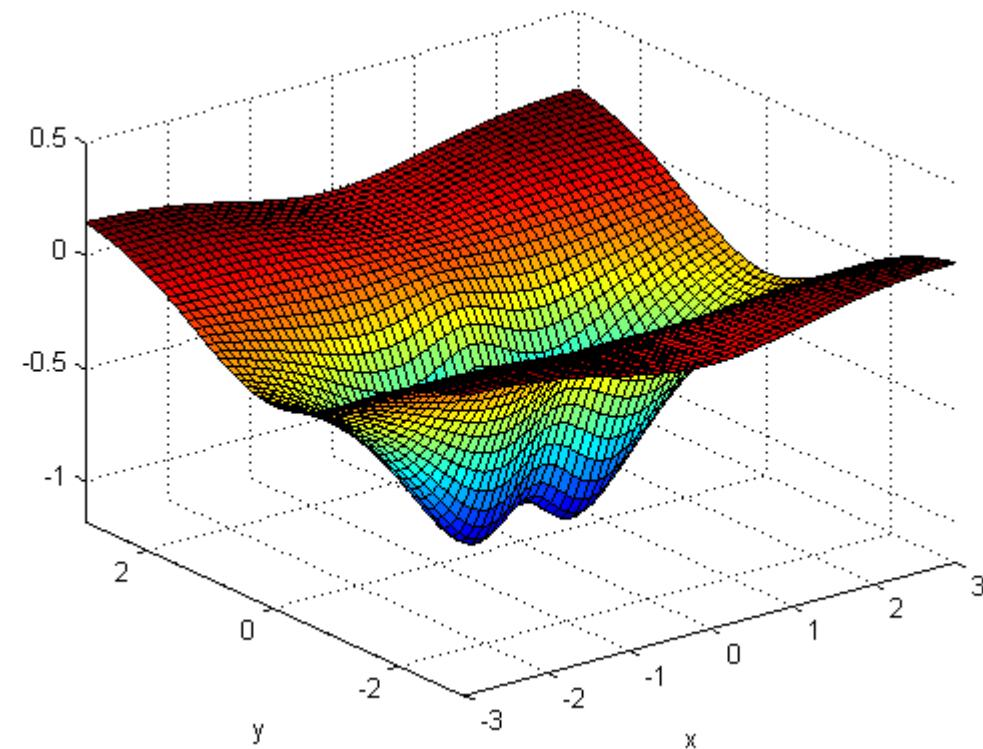




分类和回归的共同点

- 寻找一个函数
- 输入一般是在一个多维空间上

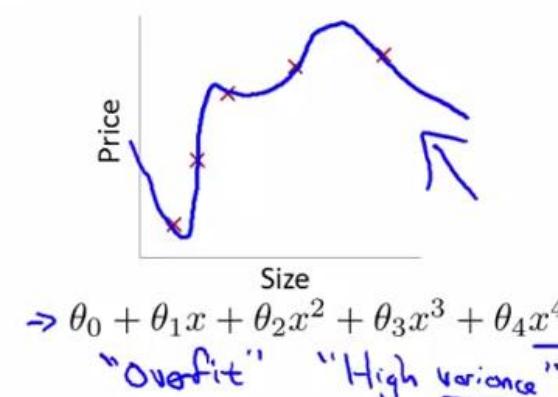
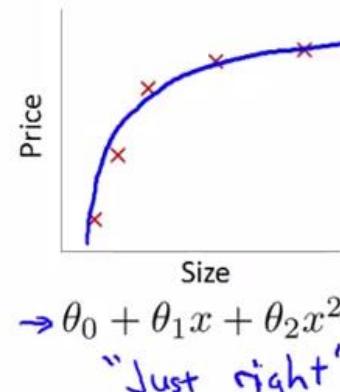
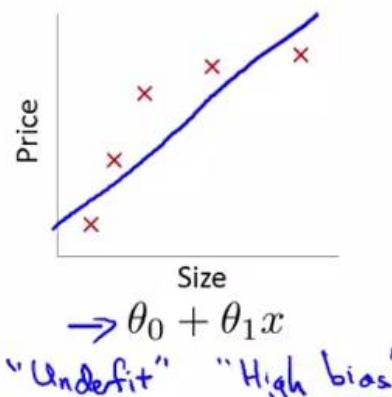
$$y = f(x)$$





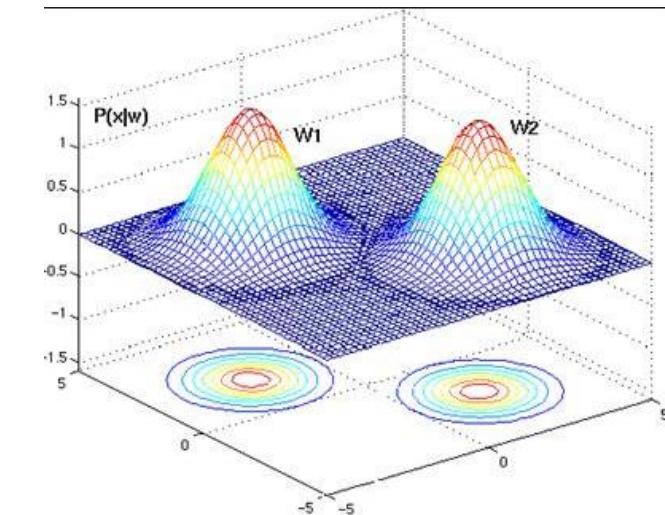
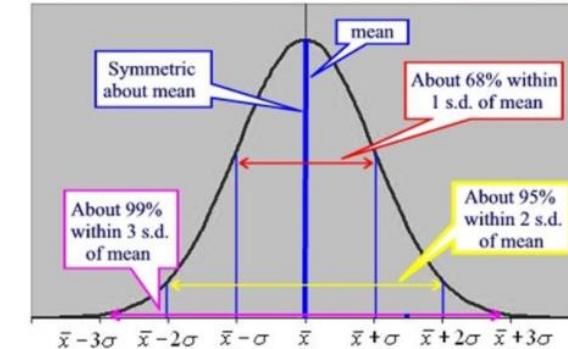
传统机器学习的一些做法

- 假定函数的形式
- 通过训练数据来找函数参数



Normal (Gaussian) distribution

- Probability density function (PDF) $f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$

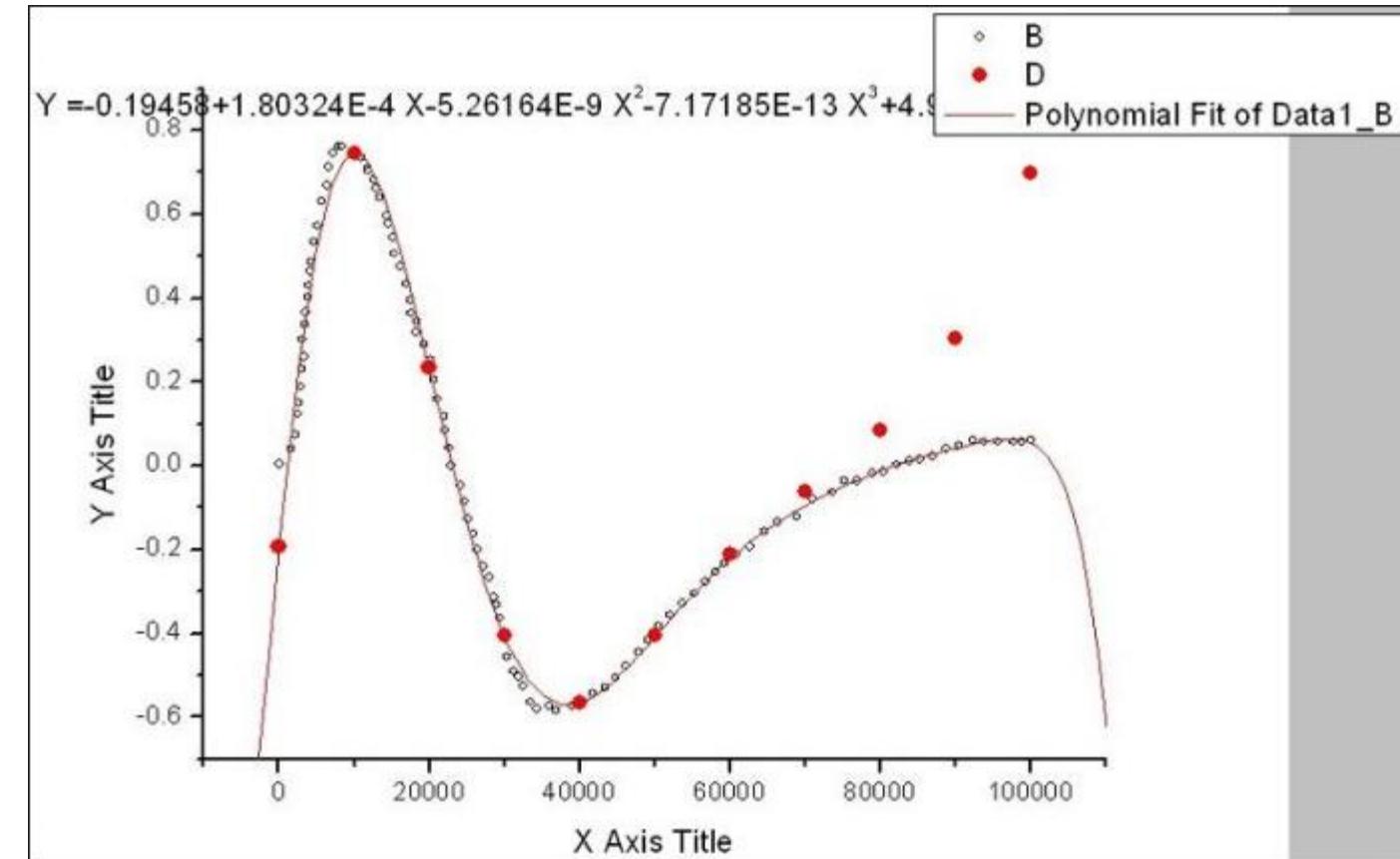




神经网络如何解决

- 回顾一下基本问题
- 复杂的函数可以通过简单函数去拟合
- 多项式拟合是其中一种形式
- 信号处理中的傅里叶变换是另外一种
- 神经网络是一种更通用的形式...

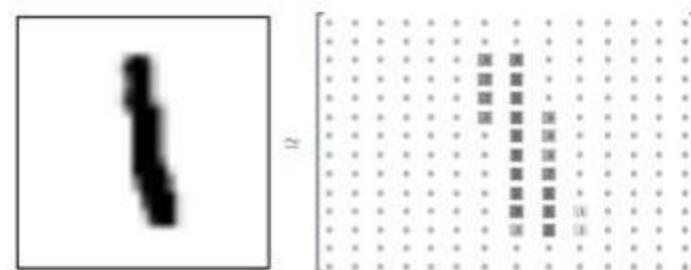
$$y = f(x)$$





从数字识别来认识神经网络

- 如何识别下图中的数字？
 - 寻找 $y = f(x)$
 - x 是一个数字对应的像素
 - y 取值是0、1、2...9



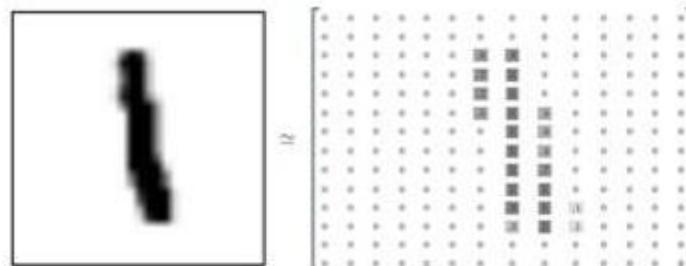
$y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$y = f(x) \text{ ?}$$

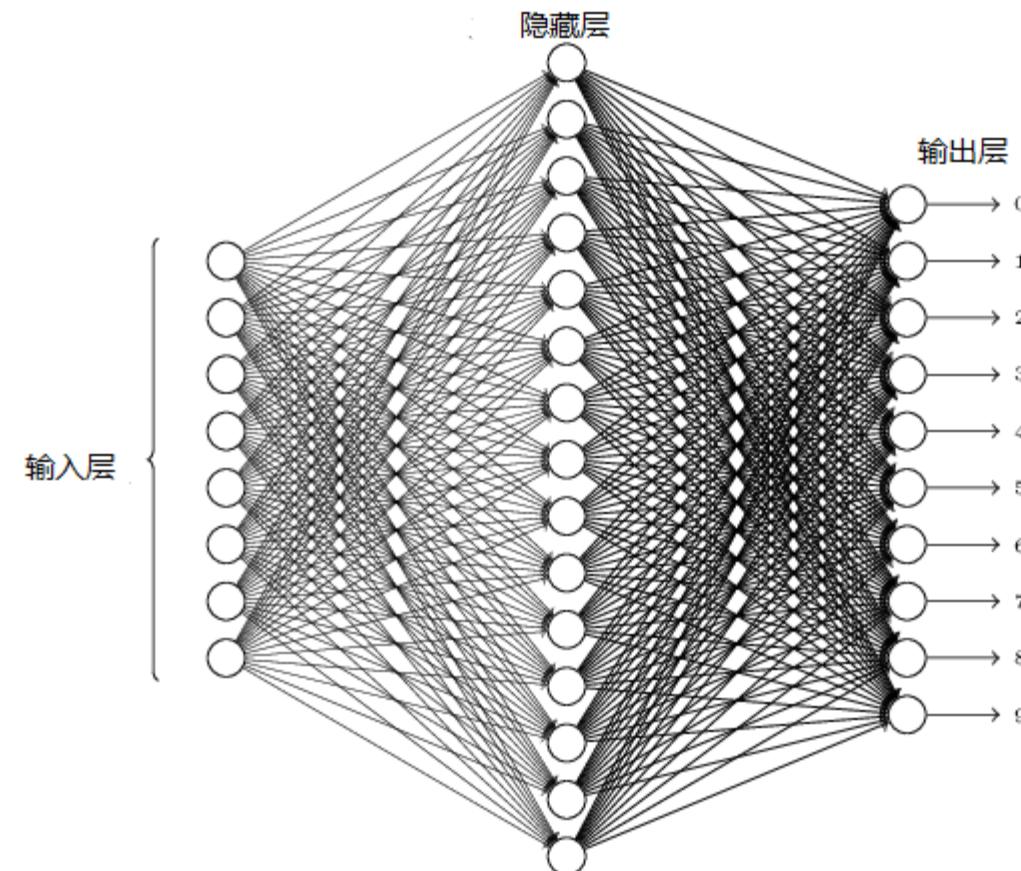
0	4	1	9	2	1	3	1	4	3
5	3	6	1	7	2	8	6	9	4
0	9	1	1	2	4	3	2	7	3
8	6	9	0	5	6	0	7	6	1
8	7	9	3	9	8	5	9	3	3
0	7	4	9	8	0	9	4	1	4
4	6	0	4	5	6	1	0	0	1
7	1	6	3	0	2	1	1	7	9
0	2	6	7	8	3	9	0	4	6
7	4	6	8	0	7	8	3	1	5



手写数字识别

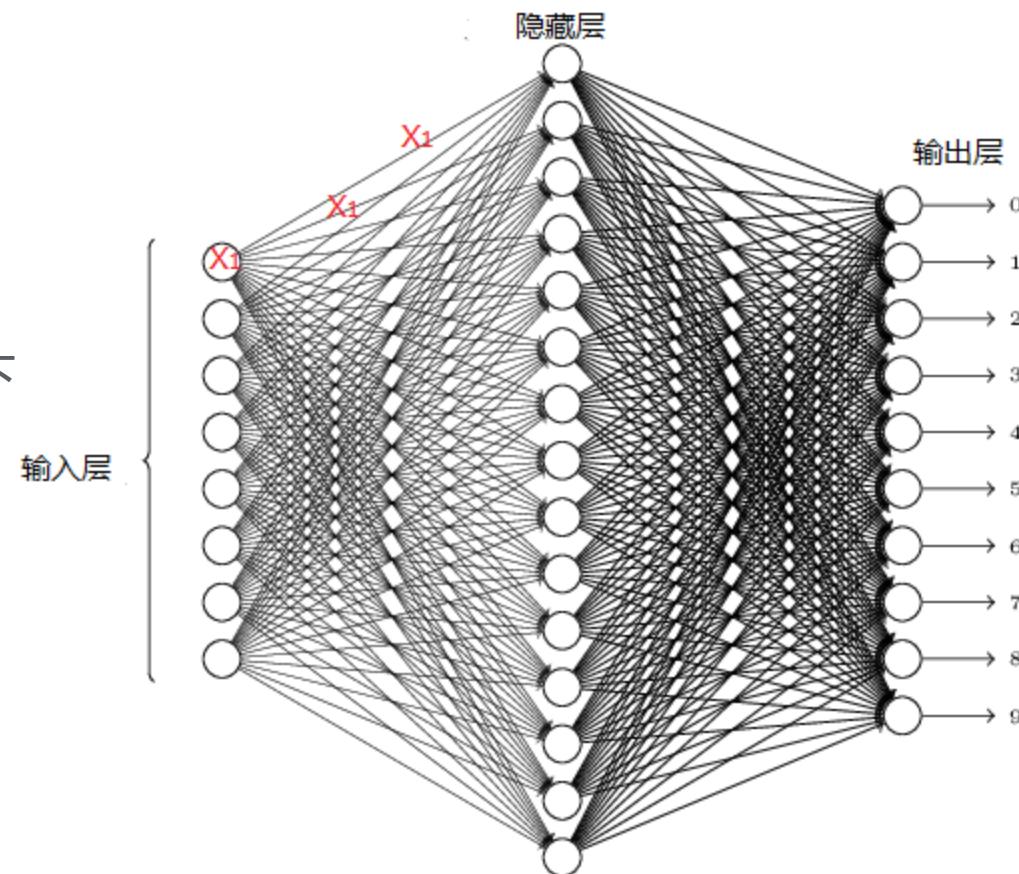


- 输入层对应各个像素的值
- 隐藏层数、节点数可调整
- 输出层10个节点

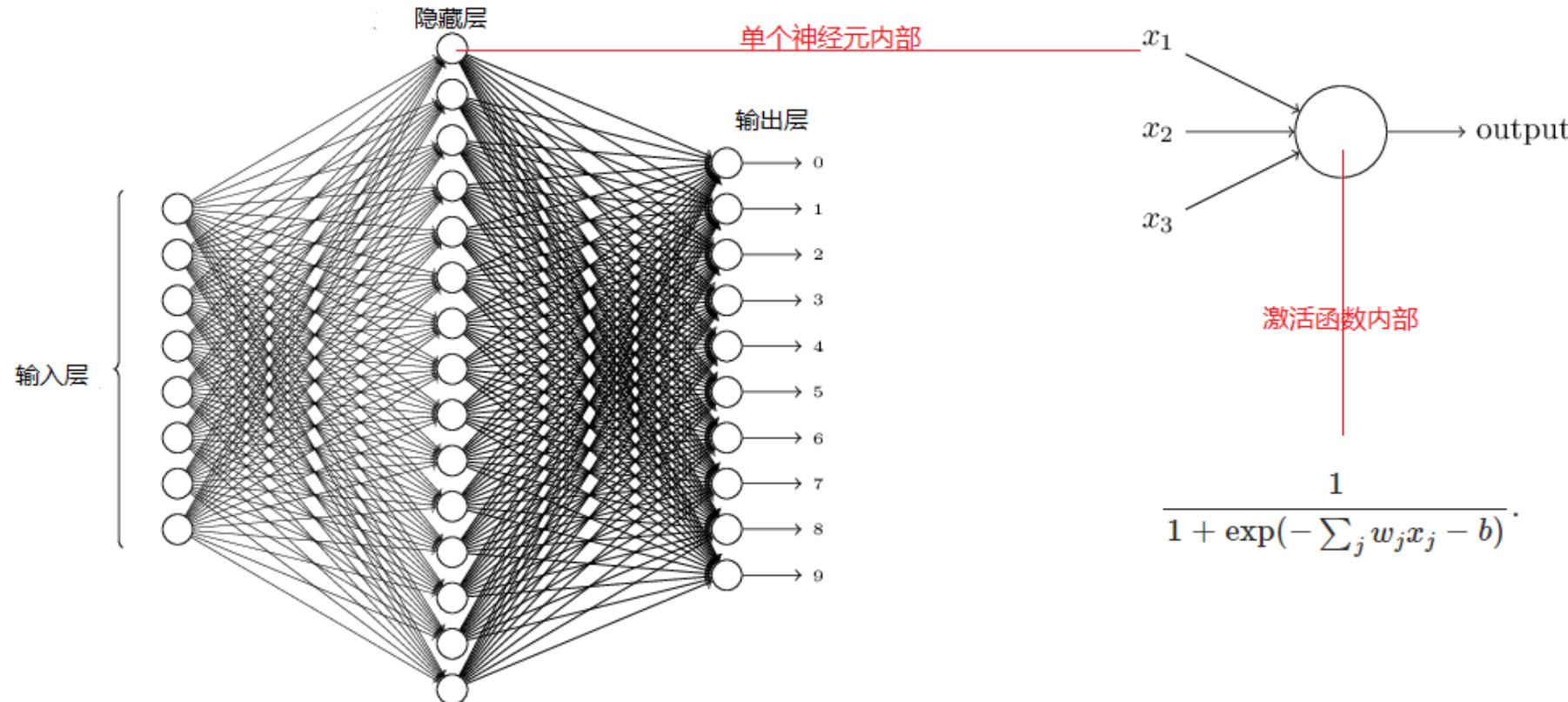


神经网络输入层

- 输入层每个输入的值加权后传递到下一层



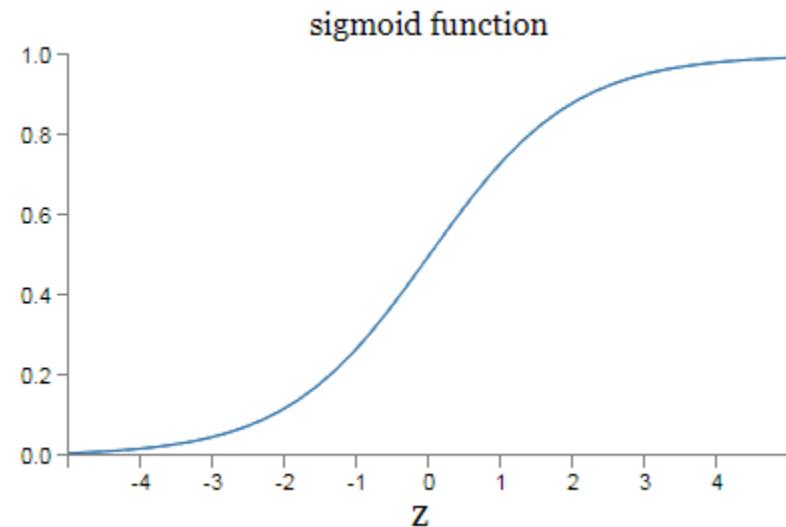
神经网络隐藏层



神经网络激活函数

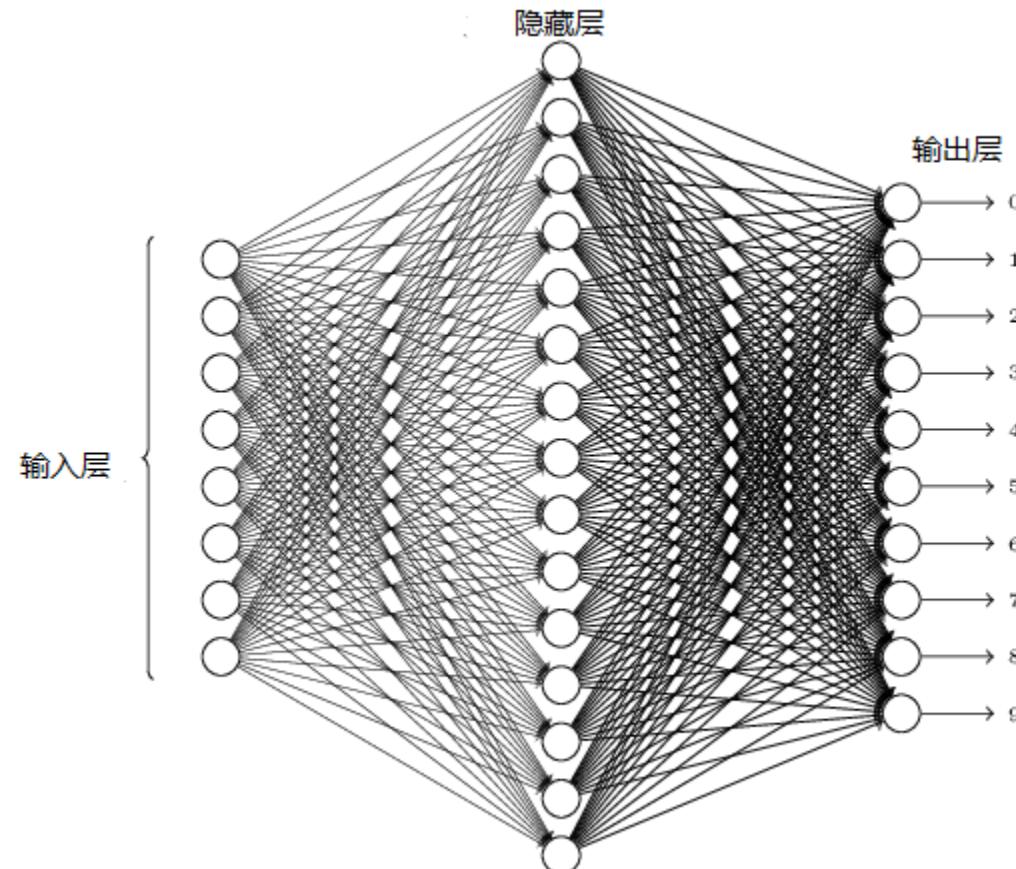
- 激活函数可以有多种选择
- 激活函数需要非线性
- 激活函数需要方便求导
- Sigmoid激活函数是常用之一

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}.$$



神经网络输出层

- 输出层也是sigmod神经元
- sigmod函数输出范围在0、1之间
- 如果隐藏层足够多
- 如果每层的神经元足够多
- 如果每个神经元参数 (w, b) 选择正确
- 输出层输出最大的神经元，对应识别结果





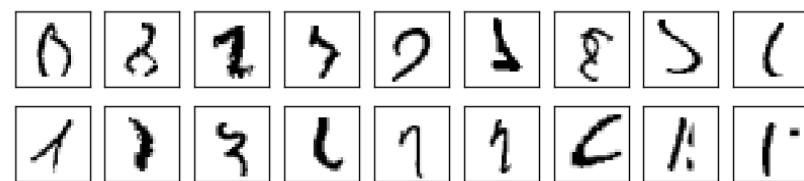
神经网络数字识别效果

- 60,000条训练数据



网络配置	识别准确率
784,30,10	96.48
784,30, 30,10	96.90
784,30, 30,10	96.57
784,30, 30, 30,10	96.53

- 10,000条测试数据



- 如何寻找网络参数?



神经网络训练

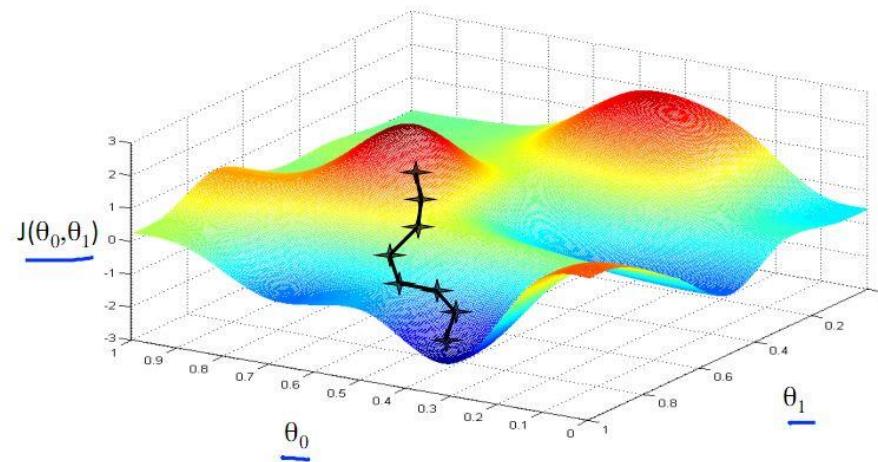
- 神经网络的输入定义为 x
- 前面的例子中， x 是一个 784 维的 vector
- 神经网络期望的输出定义为 y
- $y = y(x)$
- y 是一个 10 维的 vector
- 比如数字 6 对应的 y 是

$$y(x) = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0)^T$$

- 对于输入 x ，神经网络实际输出为 α
- 定义神经网络的损失函数如下：

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

- $C(w, b)$ 就是网络训练的目标
- 关于权重 w 及 b 的函数
- 网络训练就是找 $C(w, b)$ 的最小值
- 低维空间中的值问题，可以通过梯度法



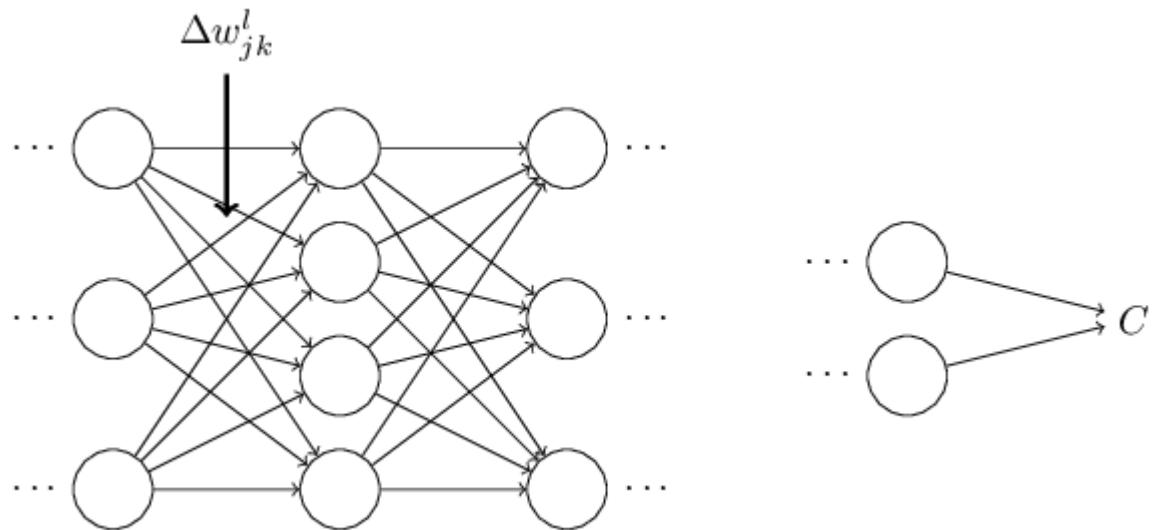


神经网络训练

- $C(w, b)$ 的优化一样可以使用梯度法
- $C(w, b)$ 包含几千万个变量，如何求梯度？
- 简单的方案如下：

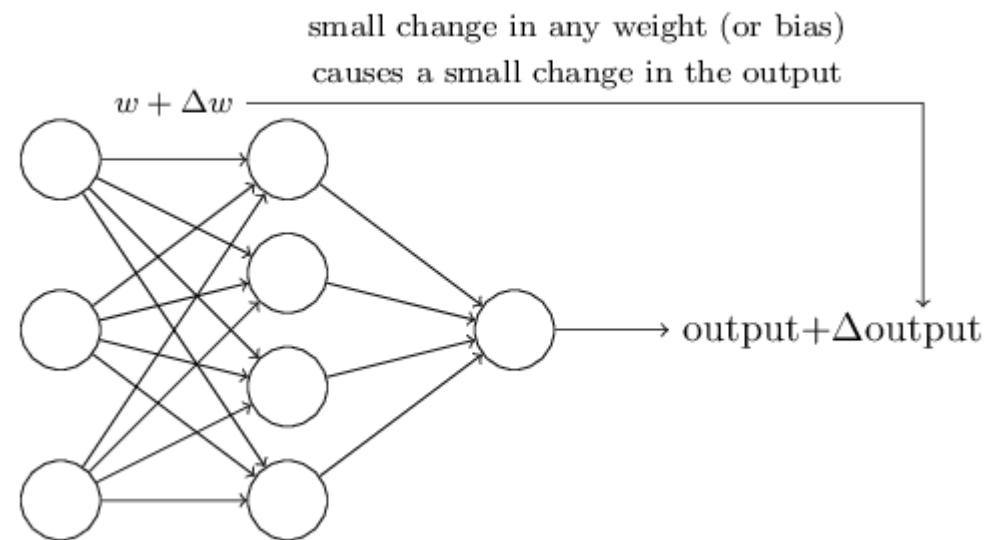
$$\frac{\partial C}{\partial w_j} \approx \frac{C(w + \epsilon e_j) - C(w)}{\epsilon},$$

$$\Delta \text{output} \approx \sum_j \frac{\partial \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \text{output}}{\partial b} \Delta b,$$



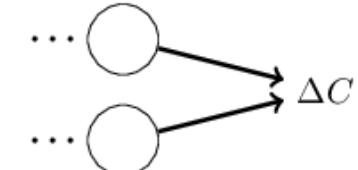
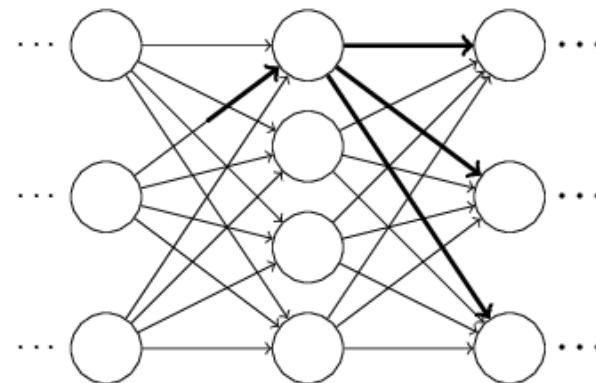
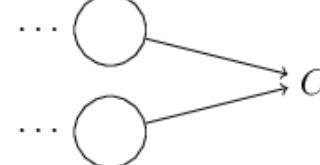
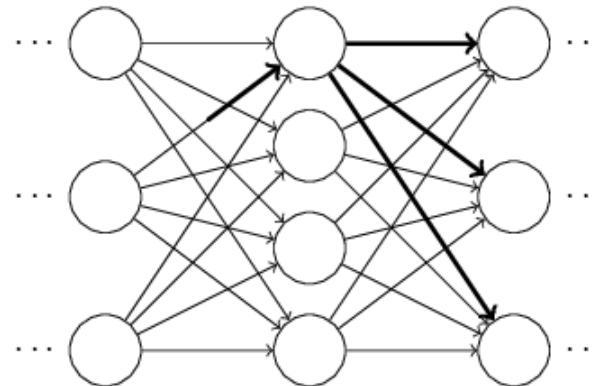
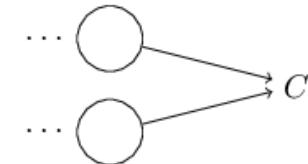
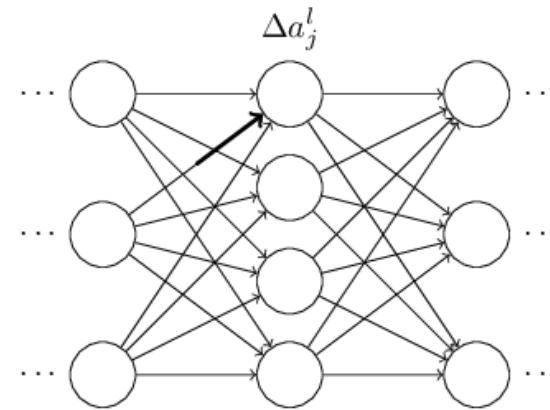
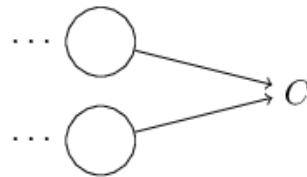
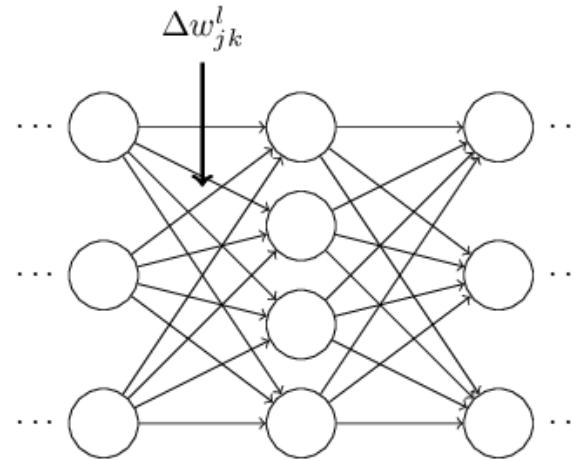
神经网络训练

- 准备足够的训练集
- 训练集的每个样本包含 x 及期望输出 y
- x 经过前向计算得到实际输出 α
- 前向计算就是一些列矩阵乘法
- 根据 y 和 α 计算损失函数关于每个 w 的导数
- 实际训练会一个batch的 x 一起计算求平均
- 根据导数更新每一个 w 及 b
- 重新输入 x , 进行下一次迭代
- 直到整个训练收敛





反向传播算法





反向传播算法

1. **Input x :** Set the corresponding activation a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \dots, L$ compute

$$z^l = w^l a^{l-1} + b^l \text{ and } a^l = \sigma(z^l).$$

3. **Output error δ^L :** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

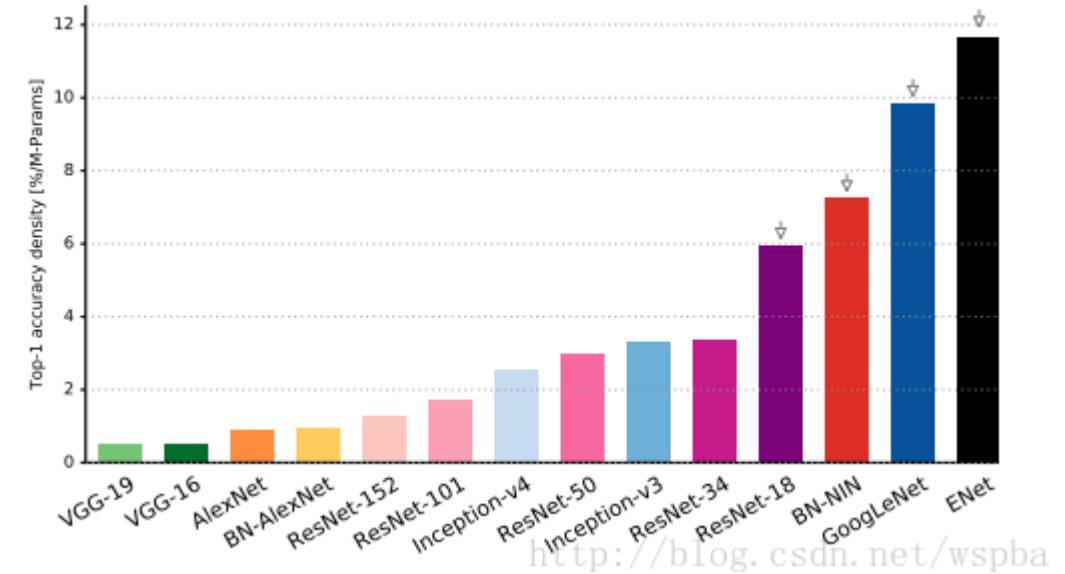
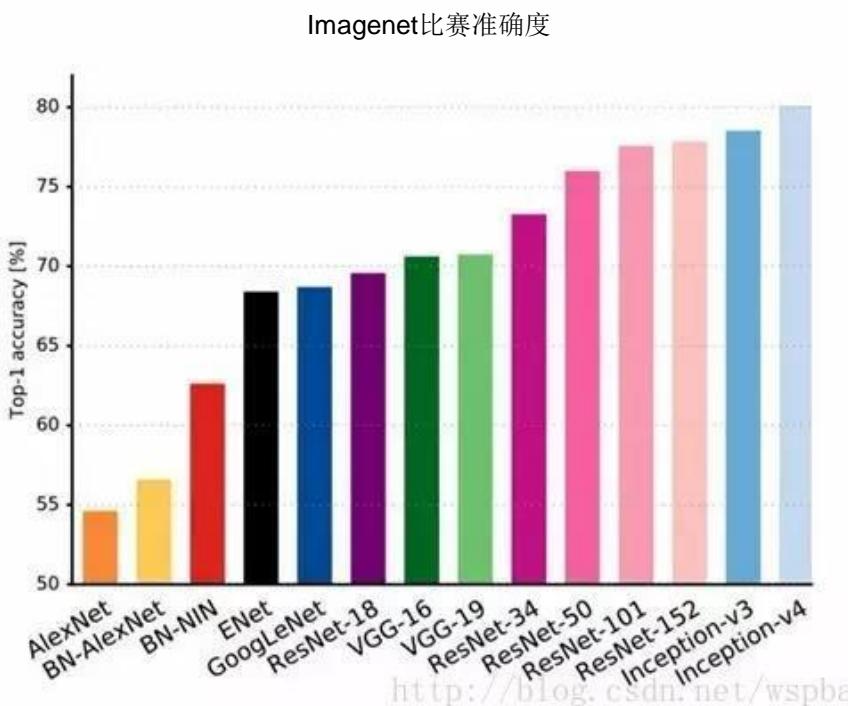
4. **Backpropagate the error:** For each $l = L - 1, L - 2, \dots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

5. **Output:** The gradient of the cost function is given by

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \text{ and } \frac{\partial C}{\partial b_j^l} = \delta_j^l.$$

神经网络训练加速

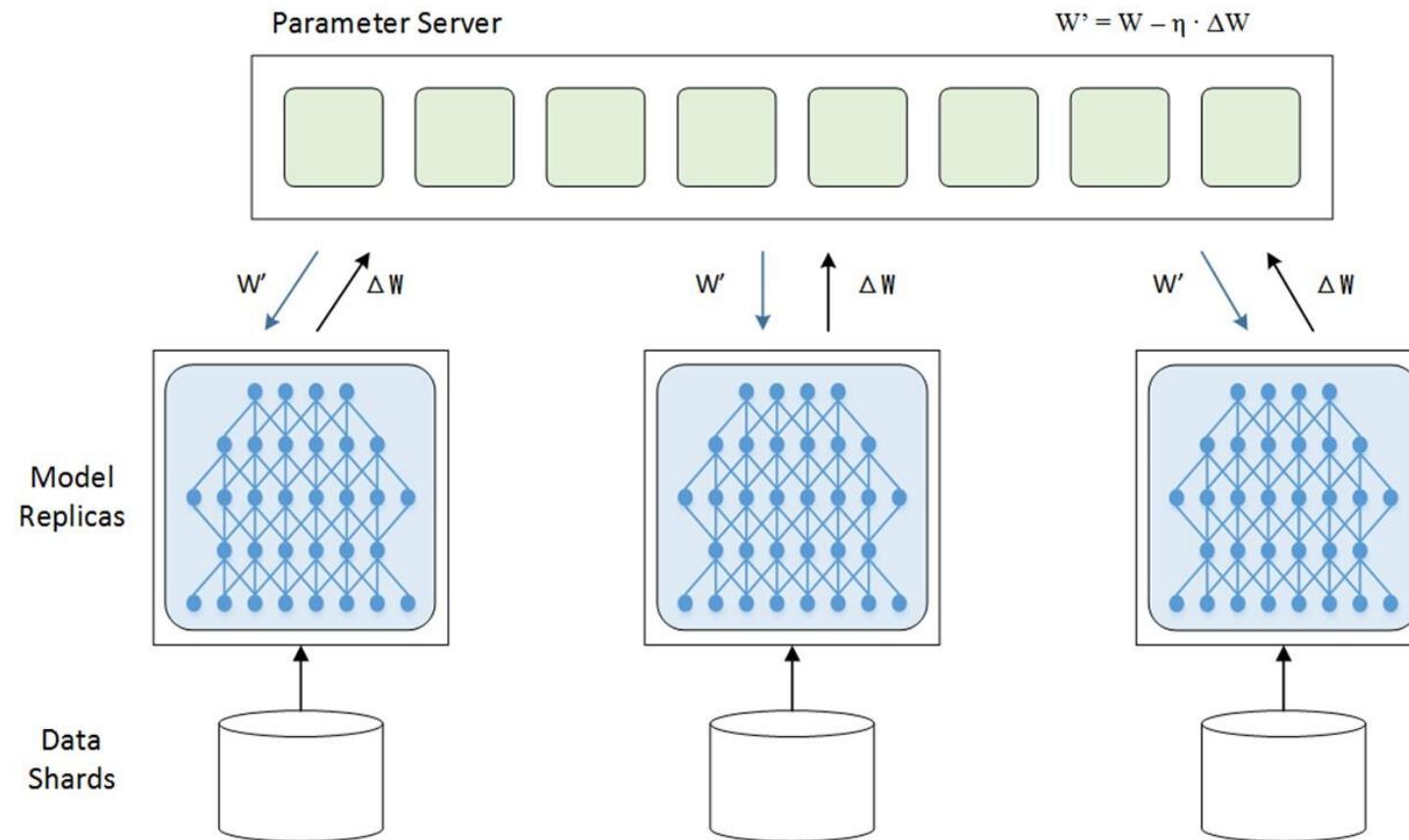
- 复杂的问题需要更深更大的网络
- 训练数据的快速增加也会加大计算量
- 如何加快训练成为深度学习的关键



Imagenet比赛网络参数（单位百万）

神经网络并行训练

- 数据并行



神经网络并行训练

- 使用GPU来加速运算

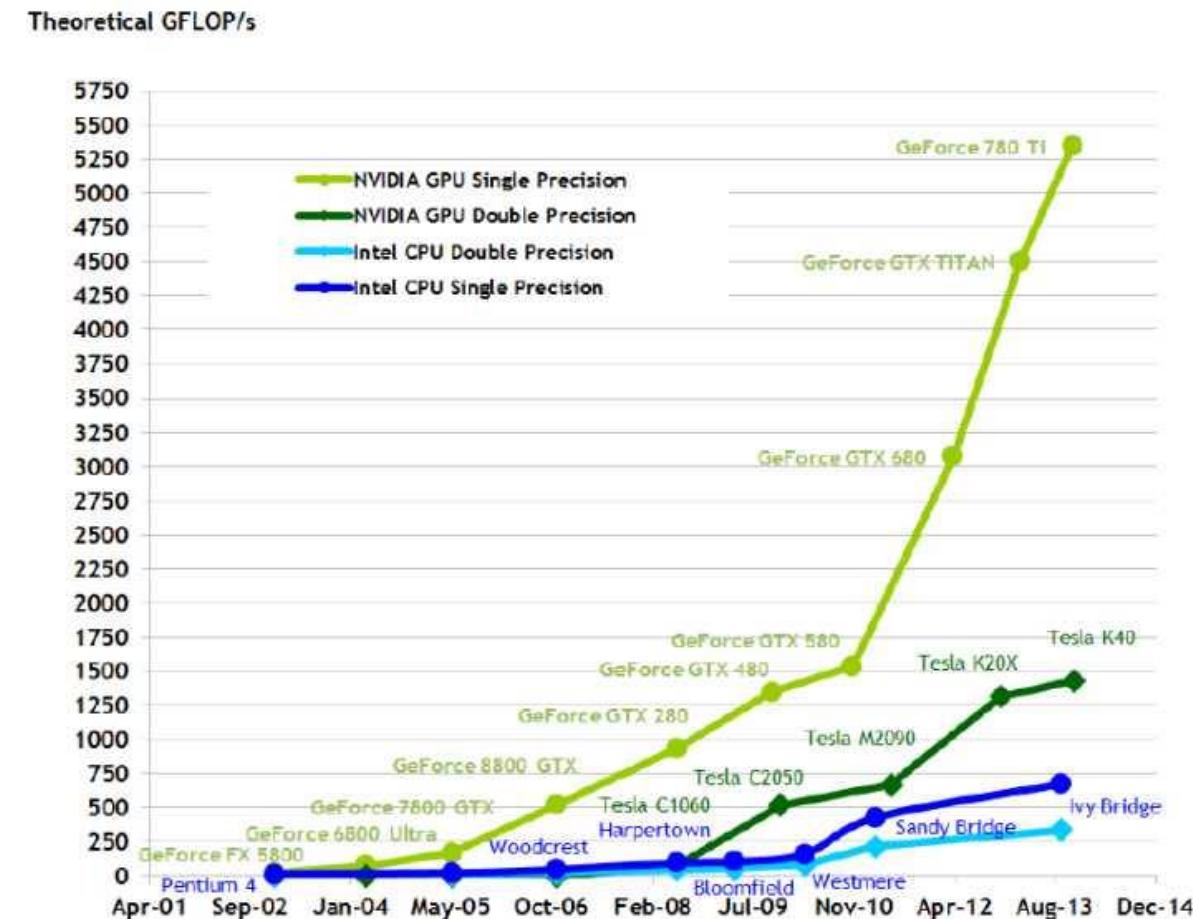
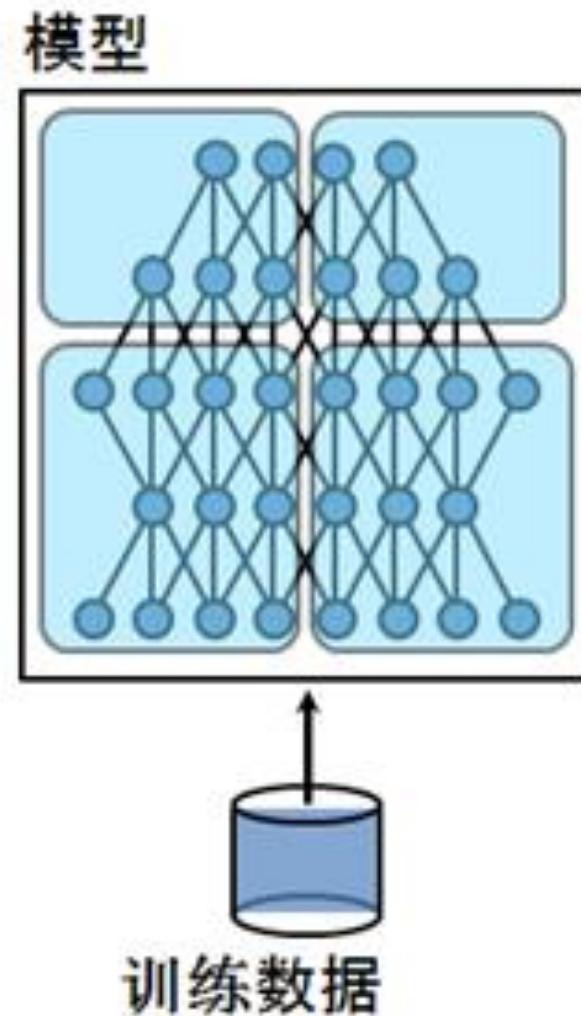


Figure 1 Floating-Point Operations per Second for the CPU and GPU

神经网络并行训练

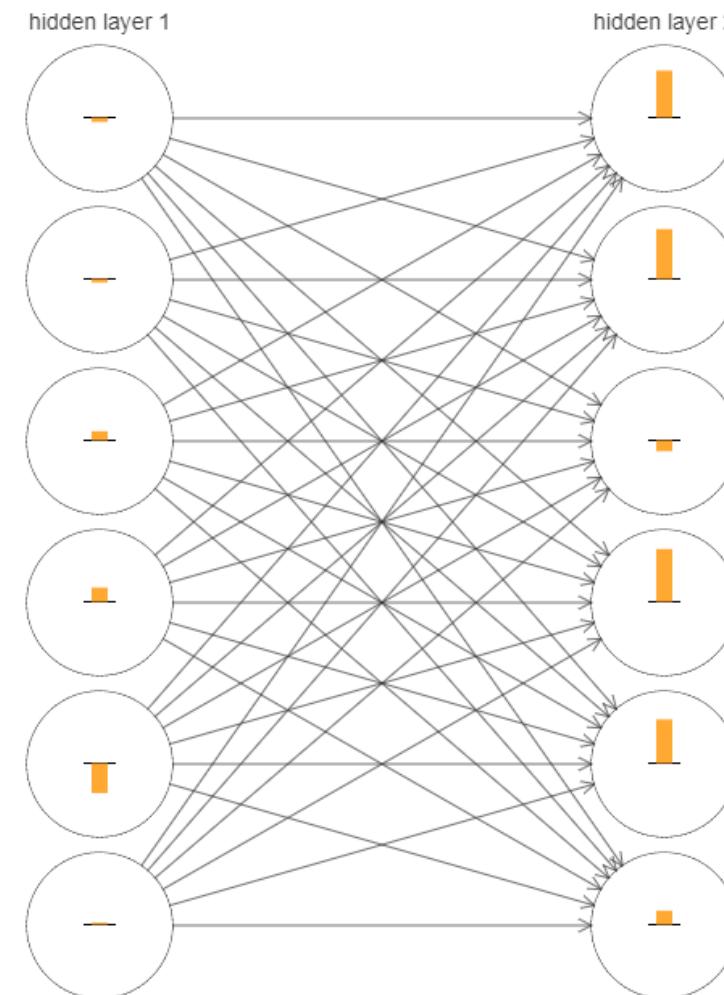
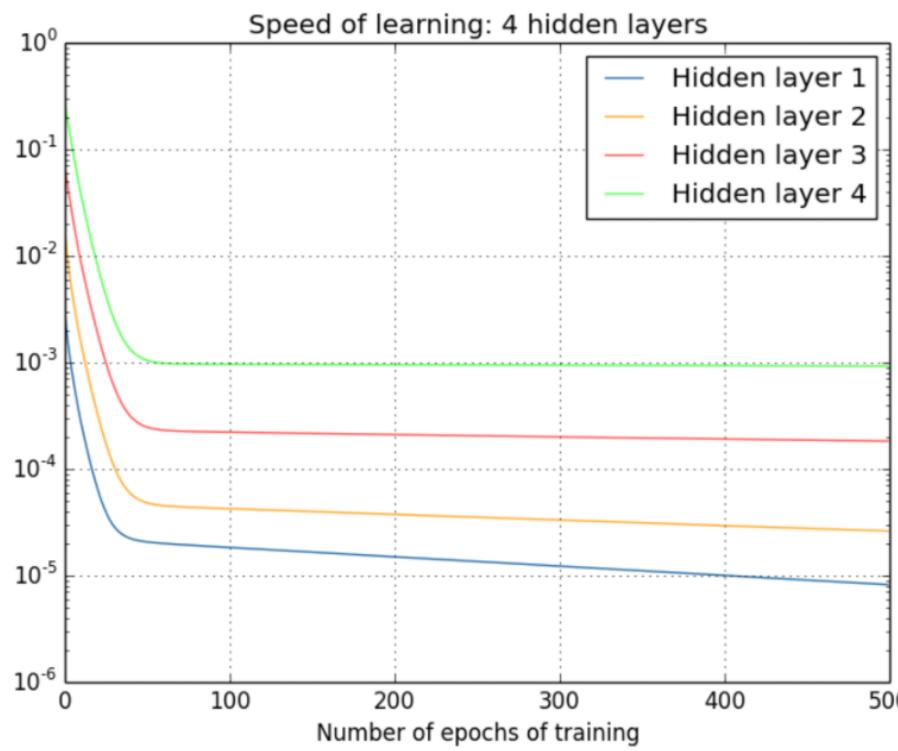
- 模型并行





神经网络并行训练

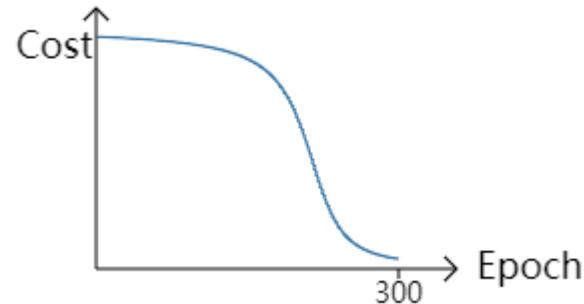
- 神经网络容易出现梯度消失或者爆炸
- 随着网络层数增加、神经网络更难训练



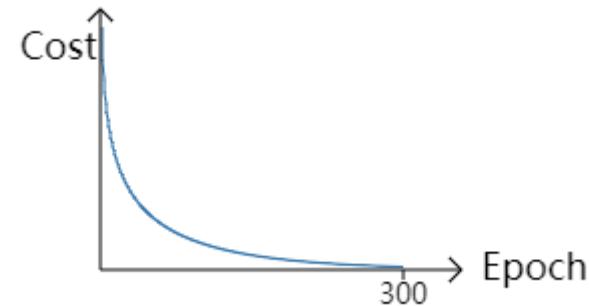
选择更合适的LOSS函数

- 梯度希望与LOSS成正比
- 可以使用交叉熵LOSS函数

$$C = \frac{1}{2} \|y - a^L\|^2 = \frac{1}{2} \sum_j (y_j - a_j^L)^2,$$

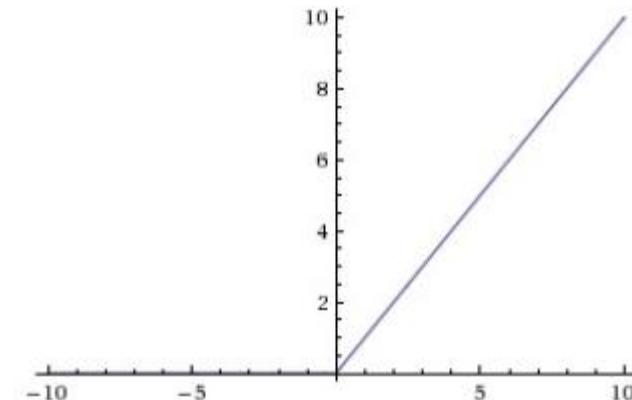
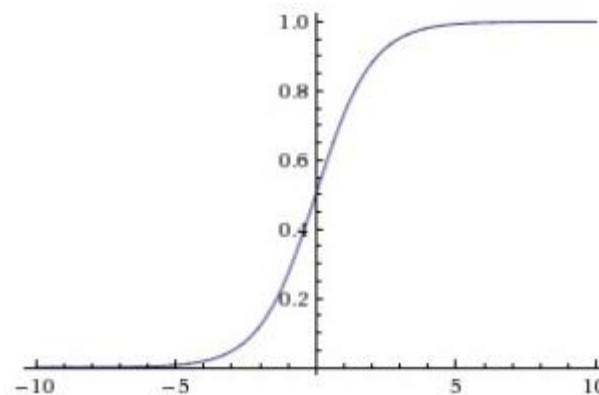


$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)],$$



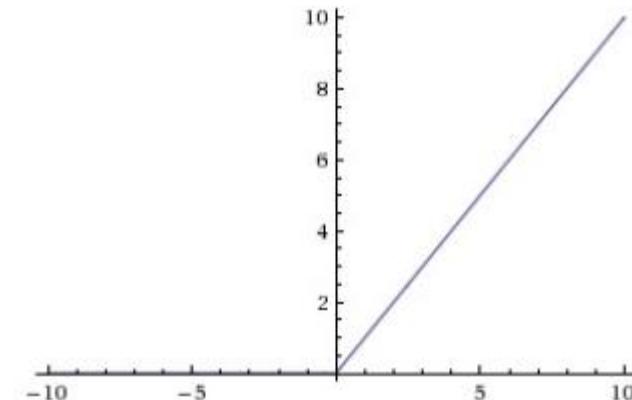
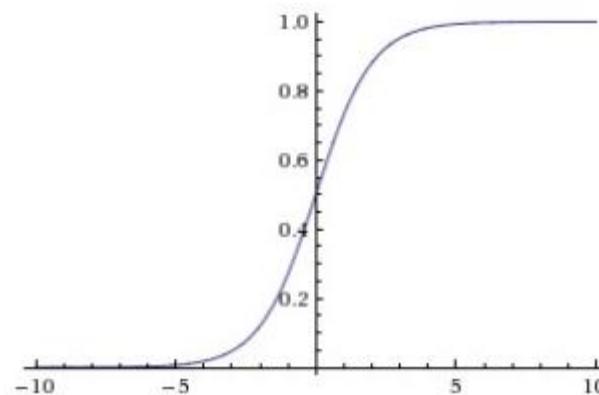
选择更合适激活函数

- 可以改进sigmod激活函数
- 避免位于两端的饱和现象



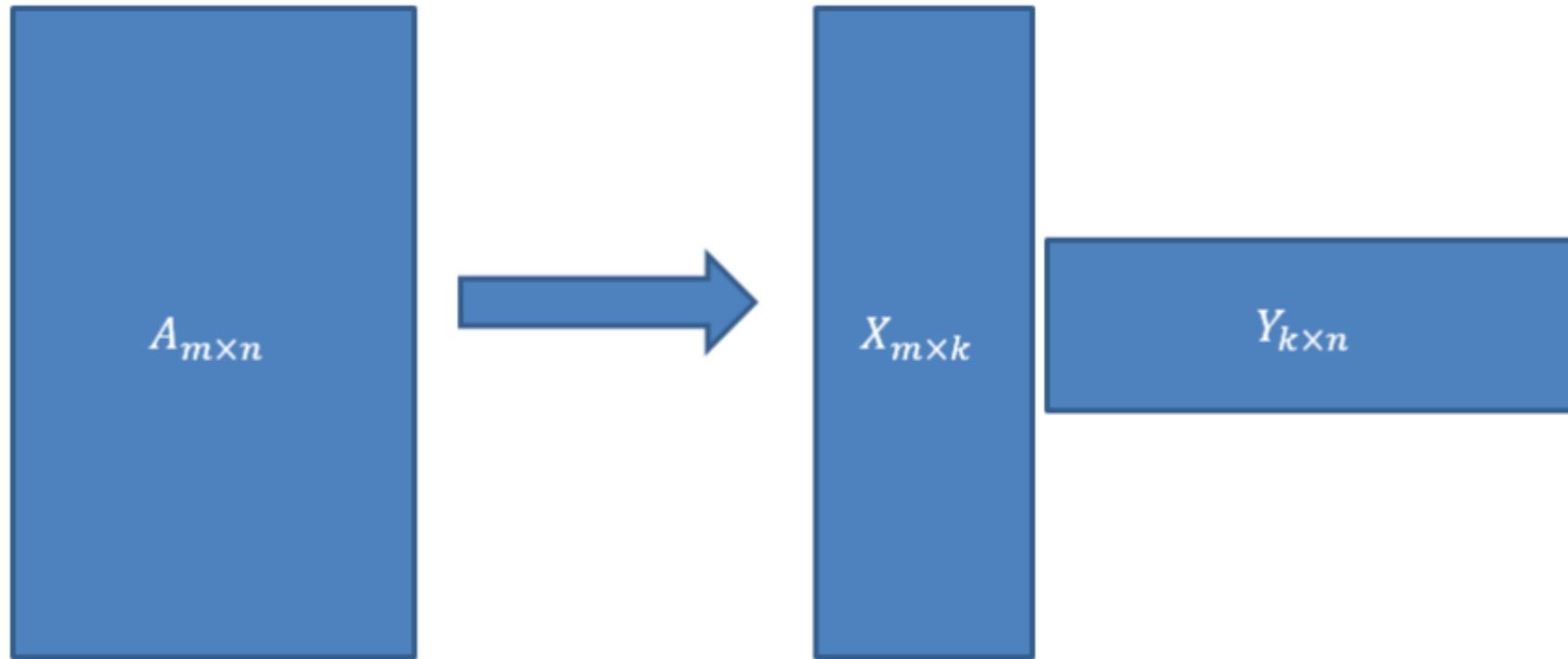
选择更合适激活函数

- 可以改进sigmod激活函数
- 避免位于两端的饱和现象

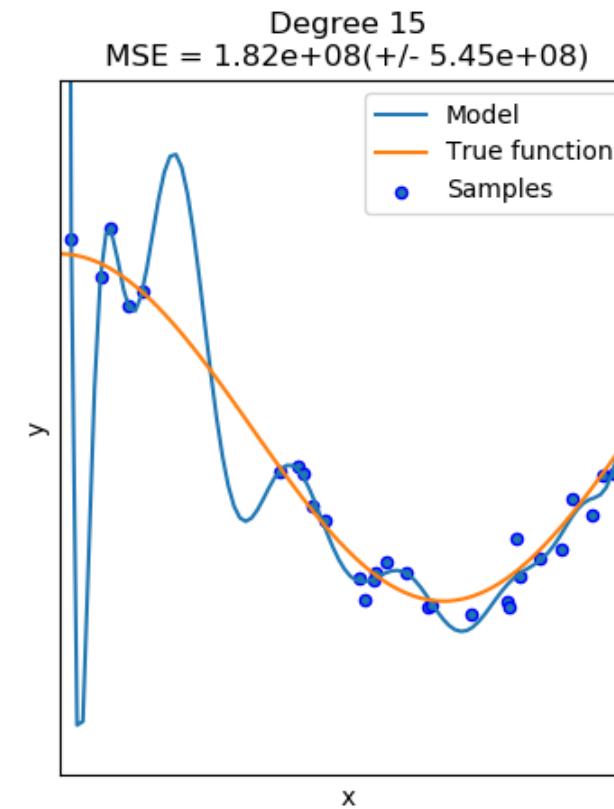
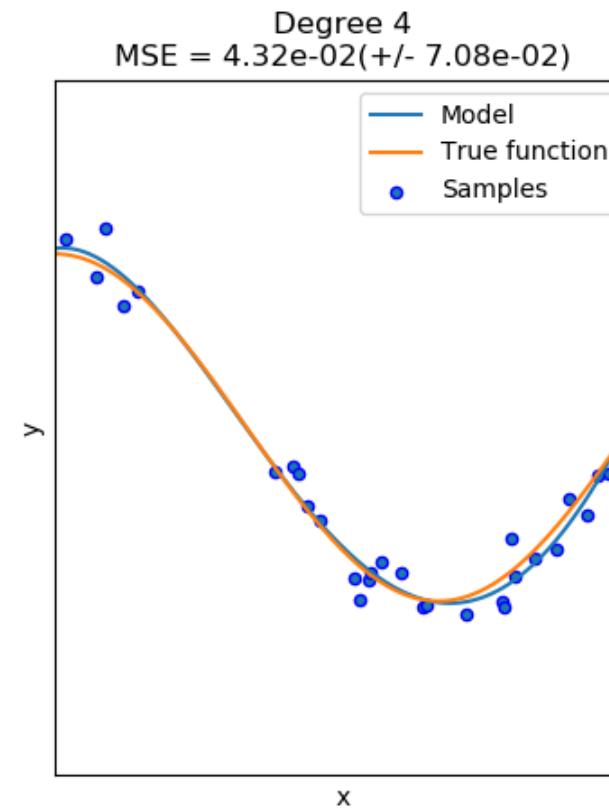
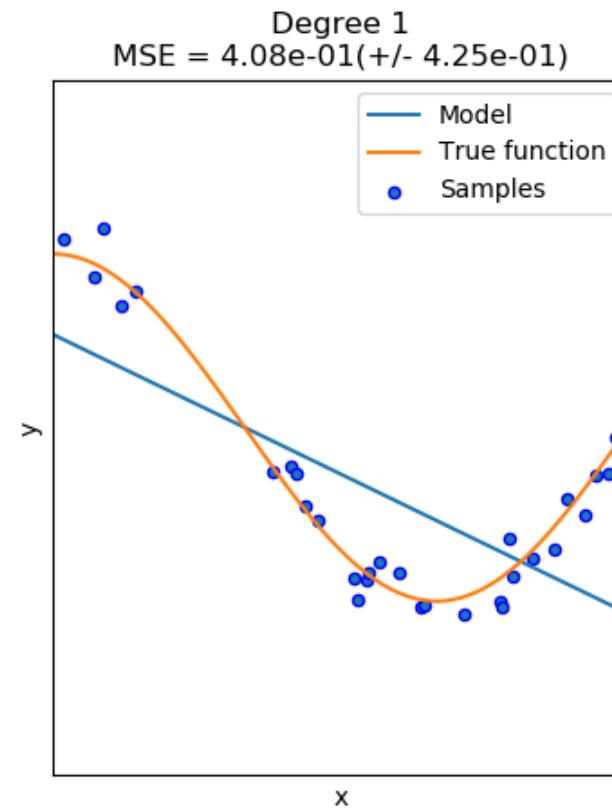


奇异值分解

- 通过减少计算量来加快计算

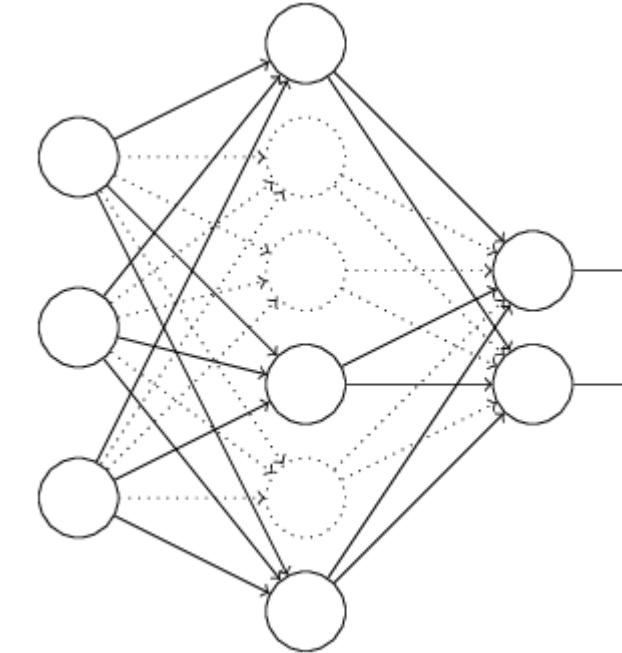
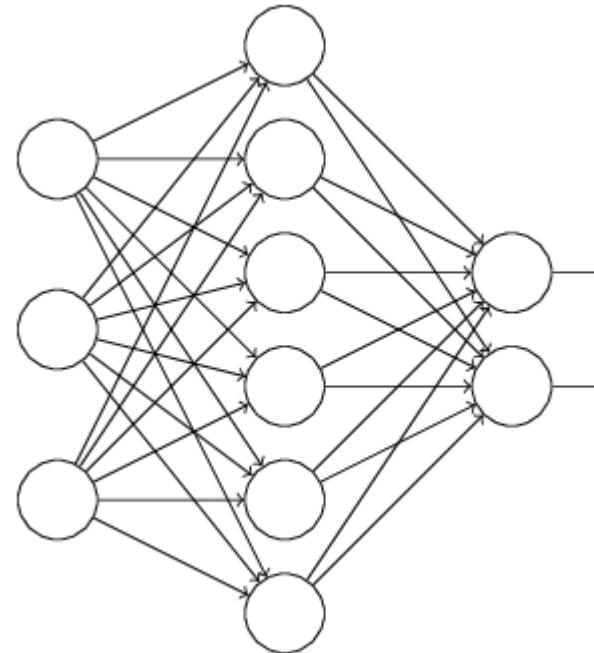


防止过拟合



一些防止过拟合的方法

- 使用更多的数据来训练
- L1及L2正则法
- Early Stopping
- Dropout方法
- 多个不同的Cost函数

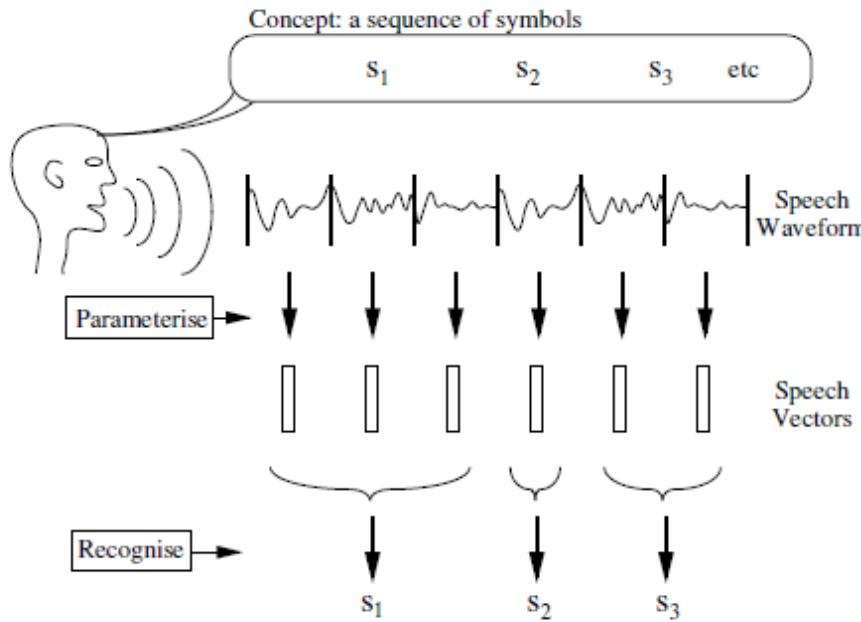


第三部分



语音识别原理

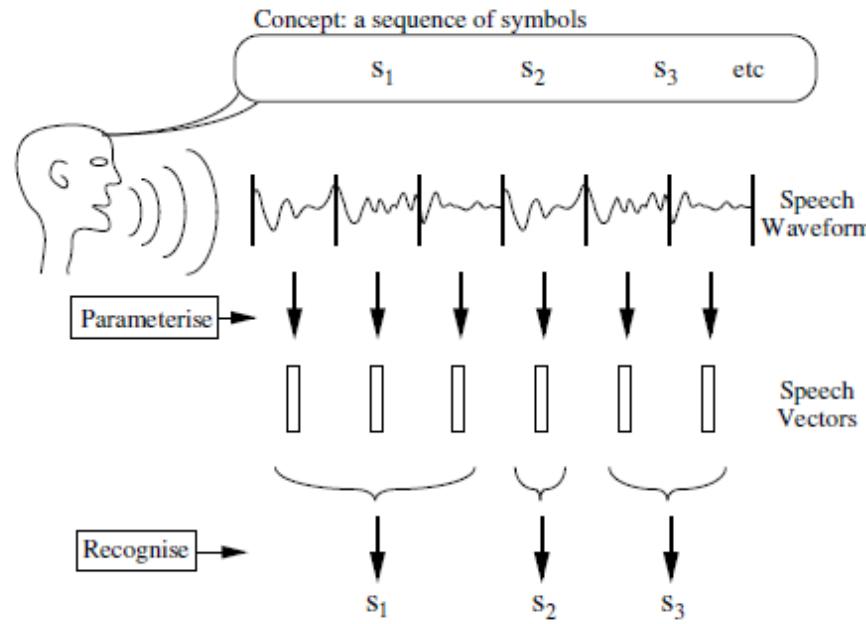
语音识别的难点



- 从音频推断用户说话的内容
- 不同用户语速不一样、同一个音节在音频上对应的时长也不一样
- 同一个音节、不同用户或者同一个用户多次发音都会不一样
- 连续的多个音节之间的起始也不确定
- 用户口音的干扰
- 环境噪声的干扰



单字识别



- 原始音频会先转成观察序列

$$O = o_1, o_2, \dots, o_T$$

- 已知观察序列情况下求生成词的最大概率

$$\arg \max_i \{P(w_i|O)\}$$

- 使用Bayes公式简化计算

$$P(w_i|O) = \frac{P(O|w_i)P(w_i)}{P(O)}$$

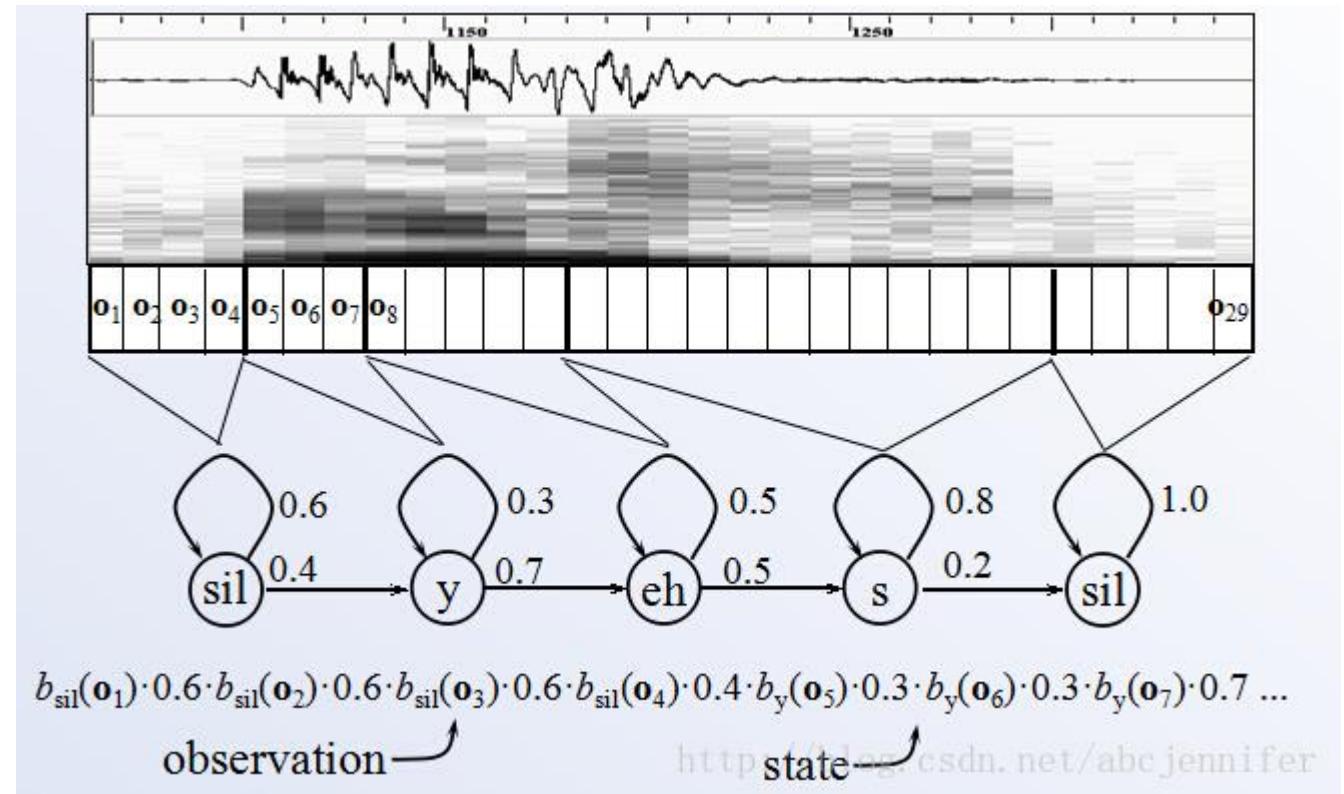
- 单字识别简化成计算如下概率

$$P(o_1, o_2, \dots | w_i)$$



单字识别

- 单字识别简化成计算如下概率
 $P(o_1, o_2, \dots | w_i)$
- 观察序列长度不定，不好计算
- HMM模型完美解决上述问题

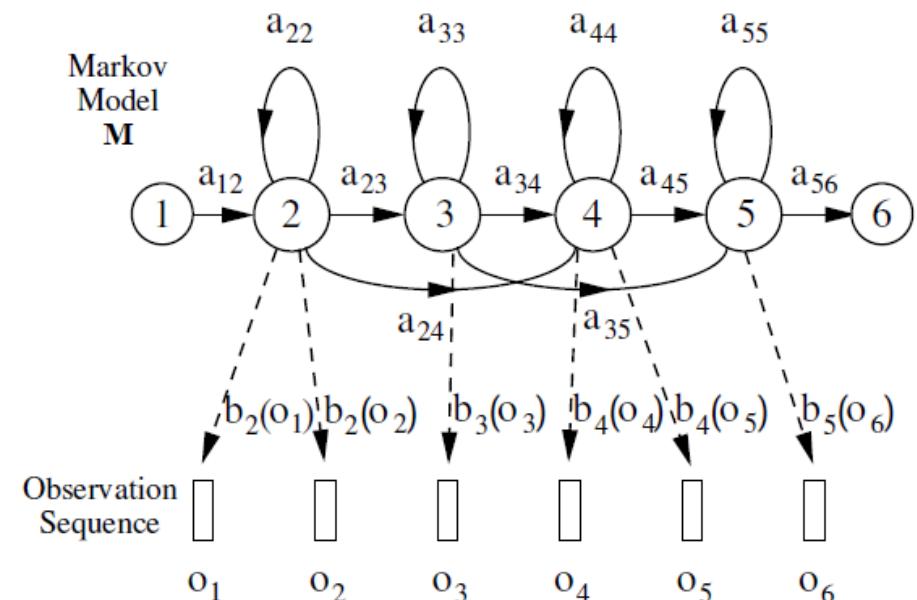




HMM模型

- 有限状态机
- 每个时刻会按概率 a_{ij} 改变一次状态
- 状态改变包含自循环或者跳转到另一状态
- 在每个状态上，会按概率 $b_j(o_t)$ 生成观察向量 o_t
- 已知HMM模型参数，对任一个观察序列，可以计算HMM模型产生观察序列的概率

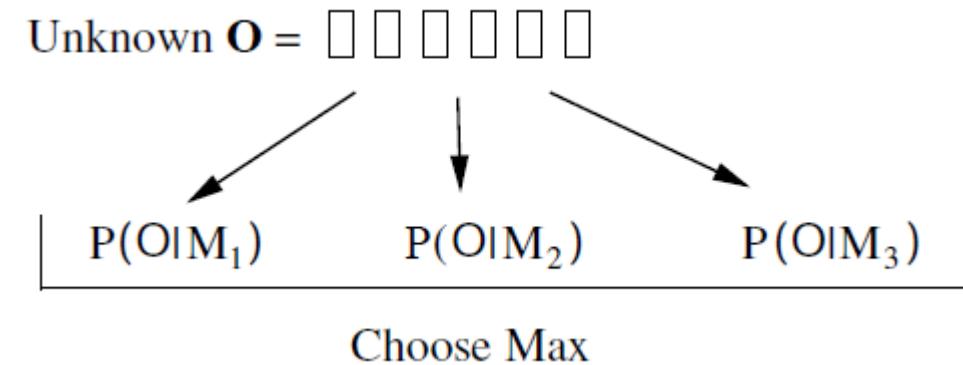
$$P(O, X | M) = a_{12} b_2(o_1) a_{22} b_2(o_2) a_{23} b_3(o_3) \dots$$





单字识别

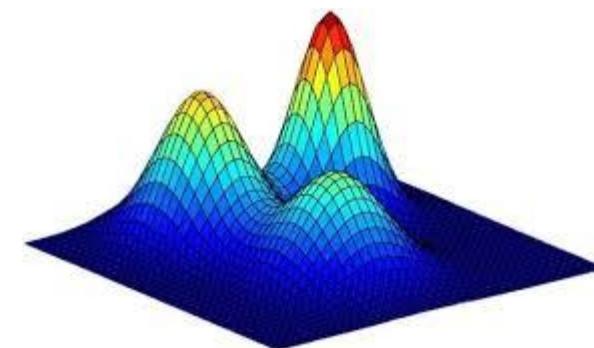
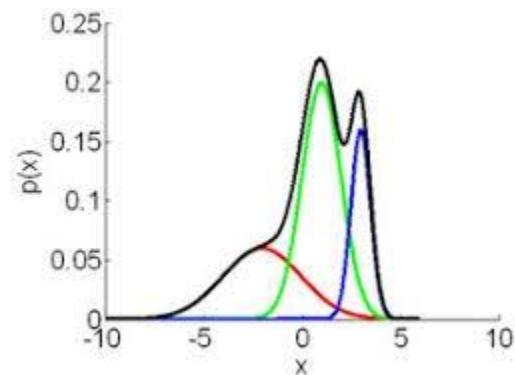
- 简化为选择让观察序列概率最大的模型





高斯模型

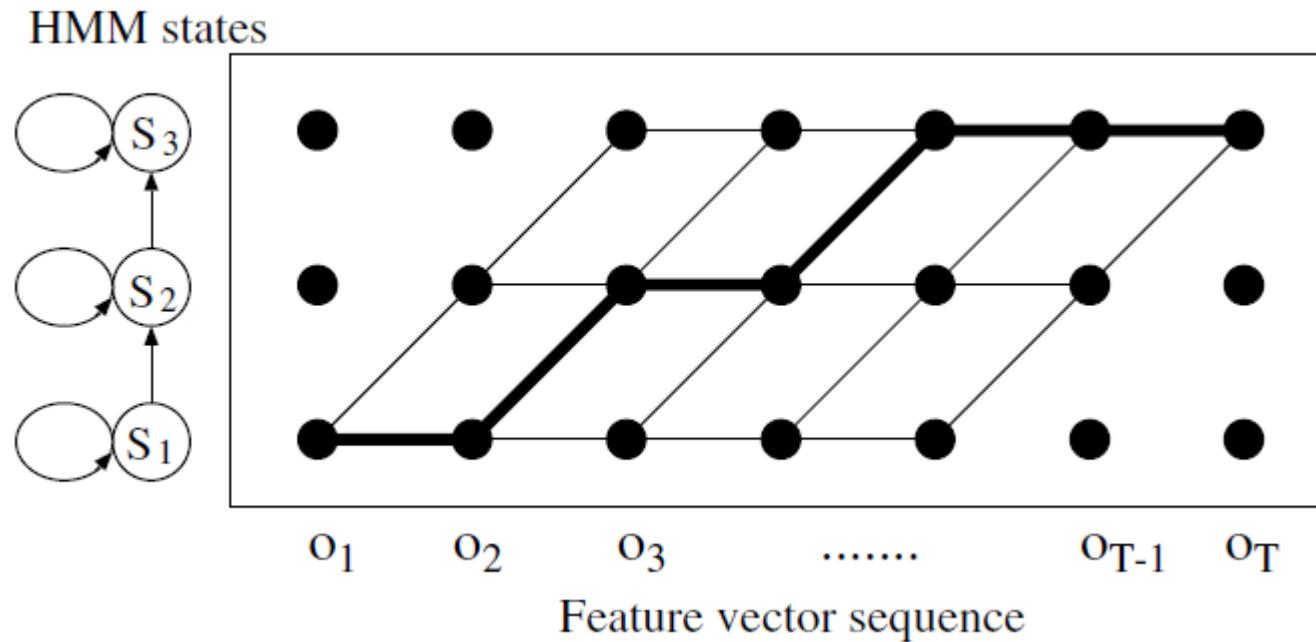
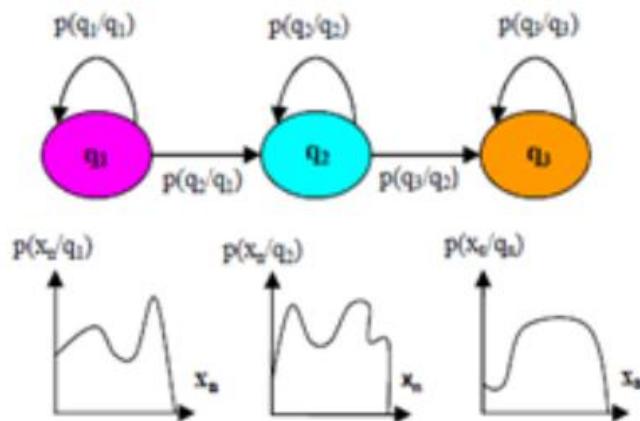
- DNN广泛应用到语音识别前、高斯模型/混合高斯模型被用来建模观察向量
- HMM模型的每个状态，可以建立一个高斯模型
- 给定一个向量，可以求出每个高斯模型产生它的概率
- 1维高斯模型
- 2维高斯模型
- 4维高斯模型





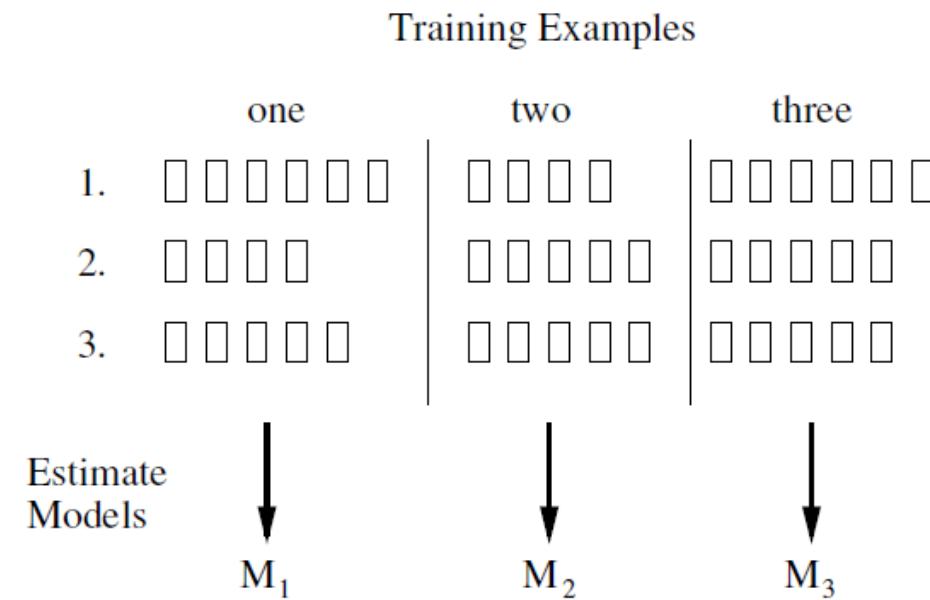
单字识别

- 给每个字建立一个HMM模型以及pdf
- 给定观察序列，计算每个HMM模型生成观察序列的概率或者最优路径概率
- 选取概率最大的HMM模型，对应的字就是解码结果





HMM模型参数如何求

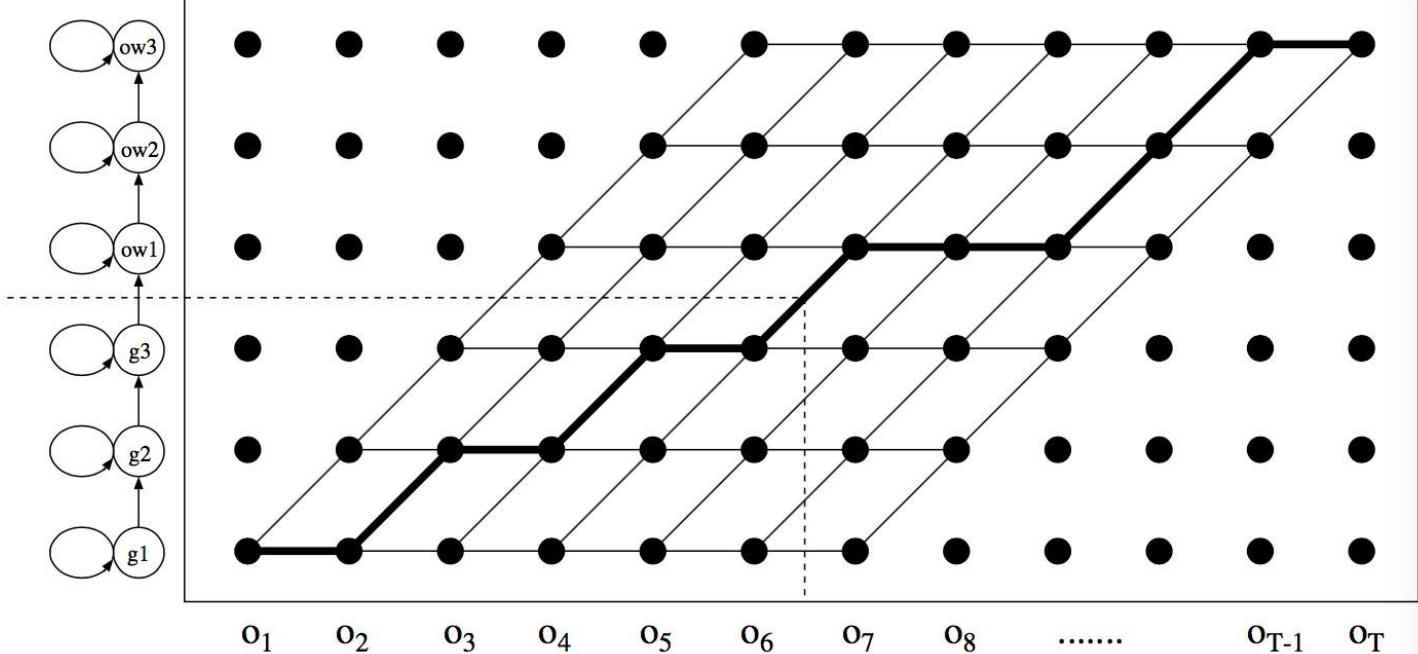




连续语句识别

- 给每个可能的句子建立HMM模型？
- 对于语法受限的任务可以这么做
- 对于自然语言识别不可行
- 可以将单字的HMM模型拼接起来

HMM states

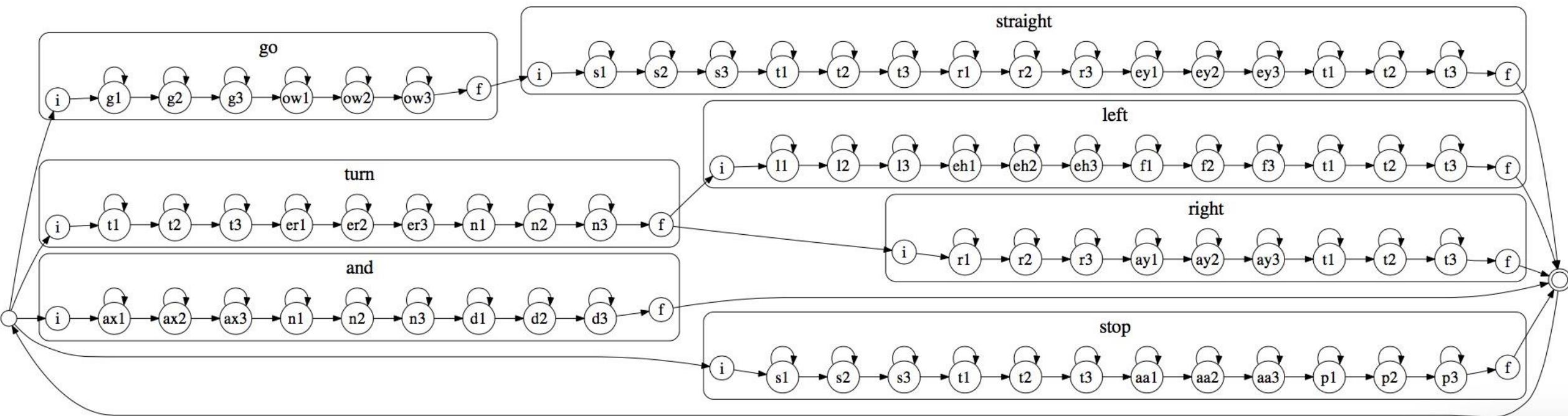




实现中的一些问题

- 有了单个声韵母HMM模型后及特征序列，如何找可能性最大的模型
- 将所有的句子及对应的HMM模型罗列出来？
- 工程实践中有两种方法，动态解码以及WFST解码
- WFST方法目前使用更为普遍

实现中的一些问题





N-Gram语言模型

- 评价一个中文字符串出现在汉语中的可能性大小
- 语言学派 VS 统计学派
- “在海滩上晒太阳” 可能性远高于 “在海滩上被太阳晒”
- 与领域有关系, “给我\$” VS “给我多了”

$$W = w_1, w_2, w_3, \dots, w_n$$

$$p(W) ?$$



N-Gram语言模型

- 通过条件概率可以计算

$$p(w_1, w_2, w_3, \dots, w_n) = p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) \dots p(w_n|w_1, w_2, \dots, w_{n-1})$$

- 通过马尔可夫假设简化前面的计算

$$p(w_1, w_2, w_3, \dots, w_n) \simeq p(w_1) p(w_2|w_1) p(w_3|w_2) \dots p(w_n|w_{n-1})$$

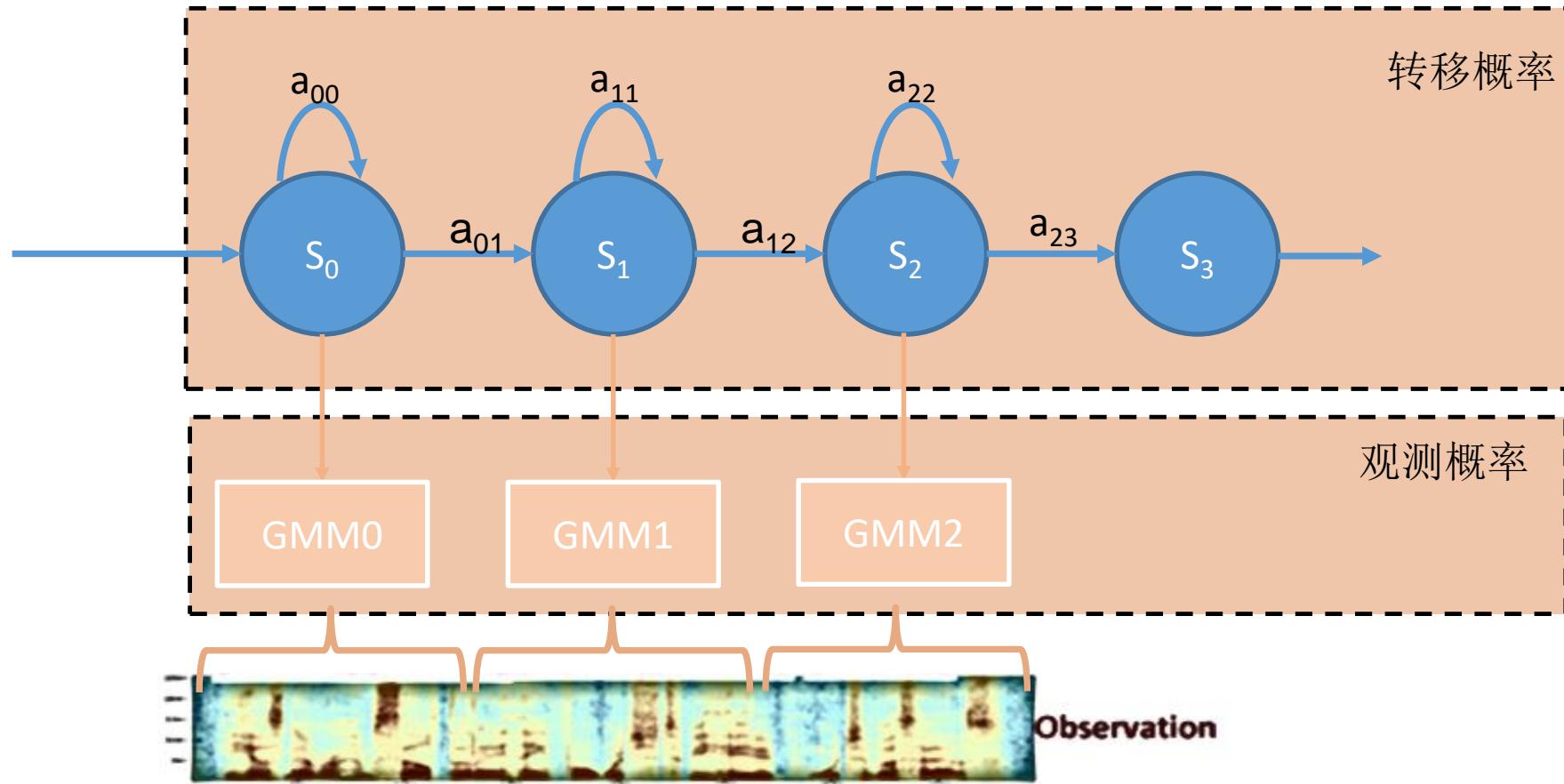
第四部分



神经网络在语音识别中的应用



传统声学模型



谢谢聆听 请多指教



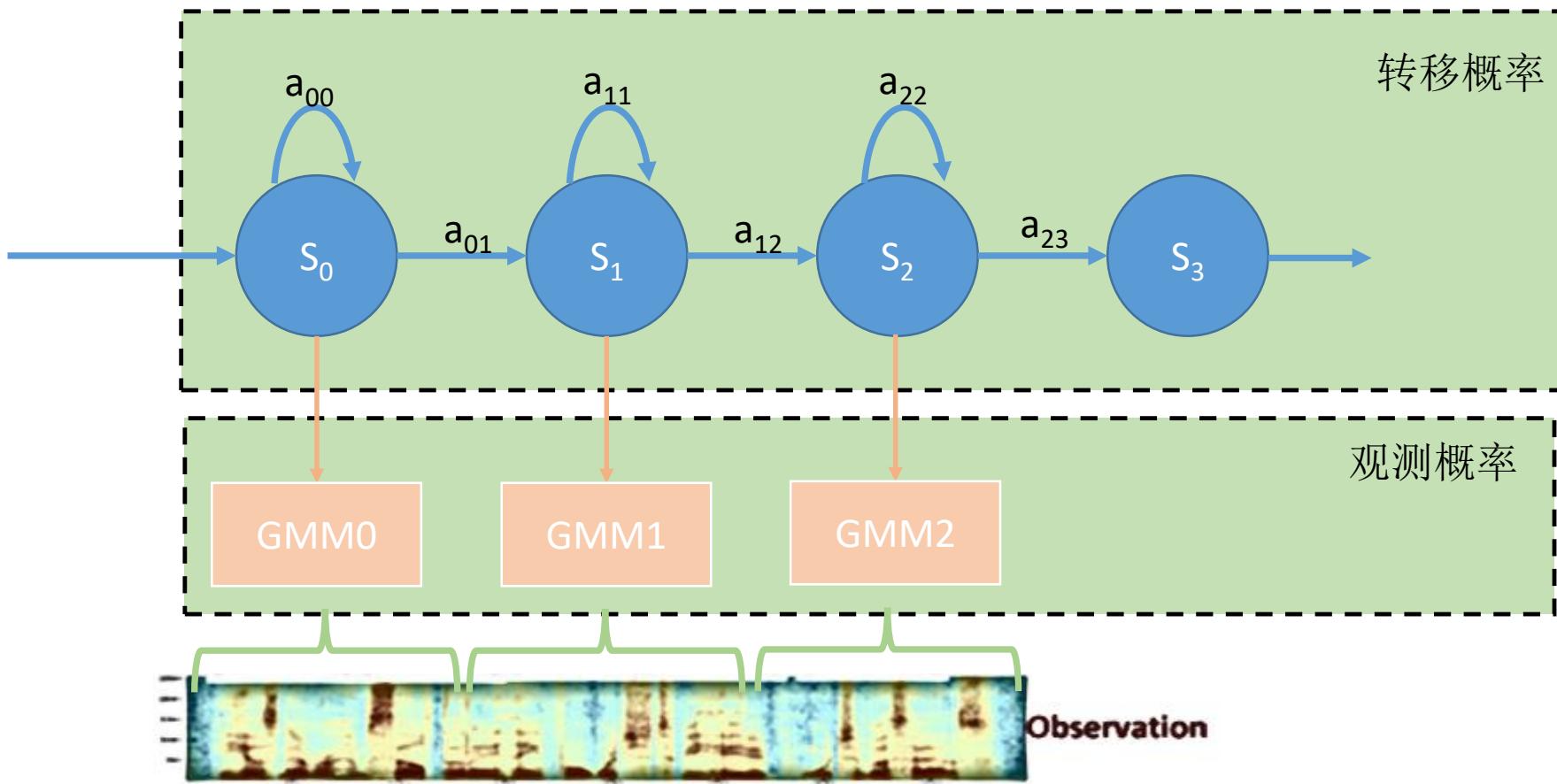


神经网络在ASR中的应用

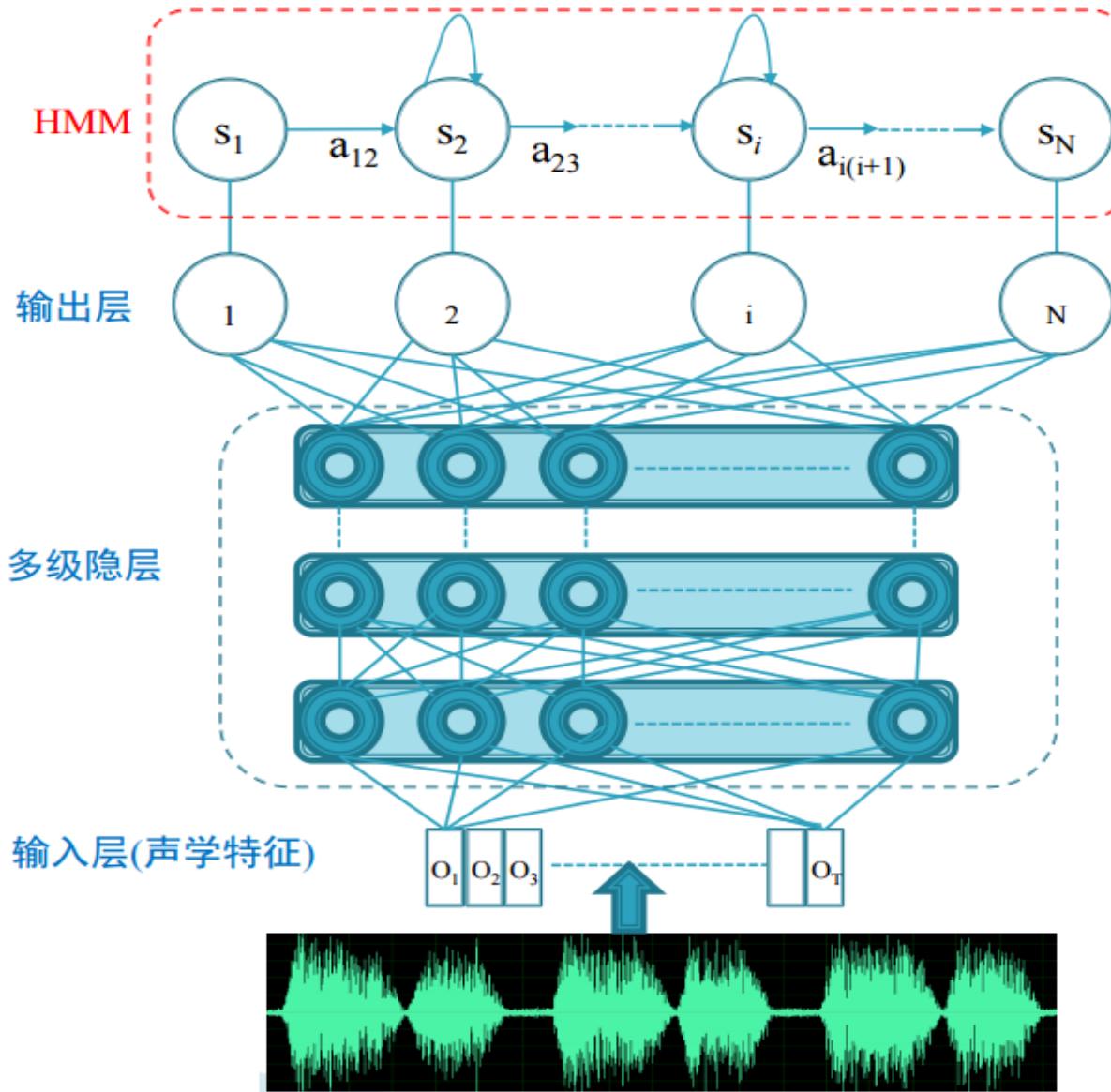
汉大学

- 传统HMM-GMM模型
- HMM-DNN混合模型
- 语言模型中的神经网络

传统HMM-GMM模型



HMM-DNN混合模型



HMM-DNN混合模型

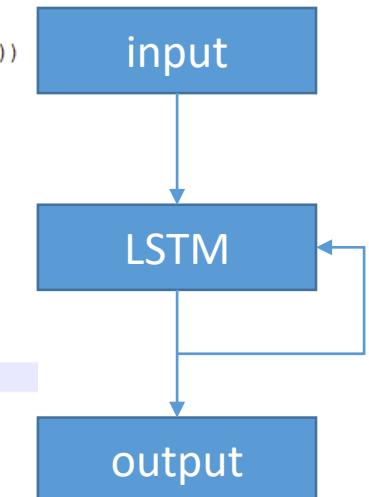
- 通过HMM-GMM的训练结果生成数据帧与状态的对齐

```
show-alignments run/train_lstm/data/lang/phones.txt run/train_lstm/exp/tri4_ali/final.mdl "ark:gunzip -c run/train_lstm/exp/tri4_ali/ali.1.gz""
```

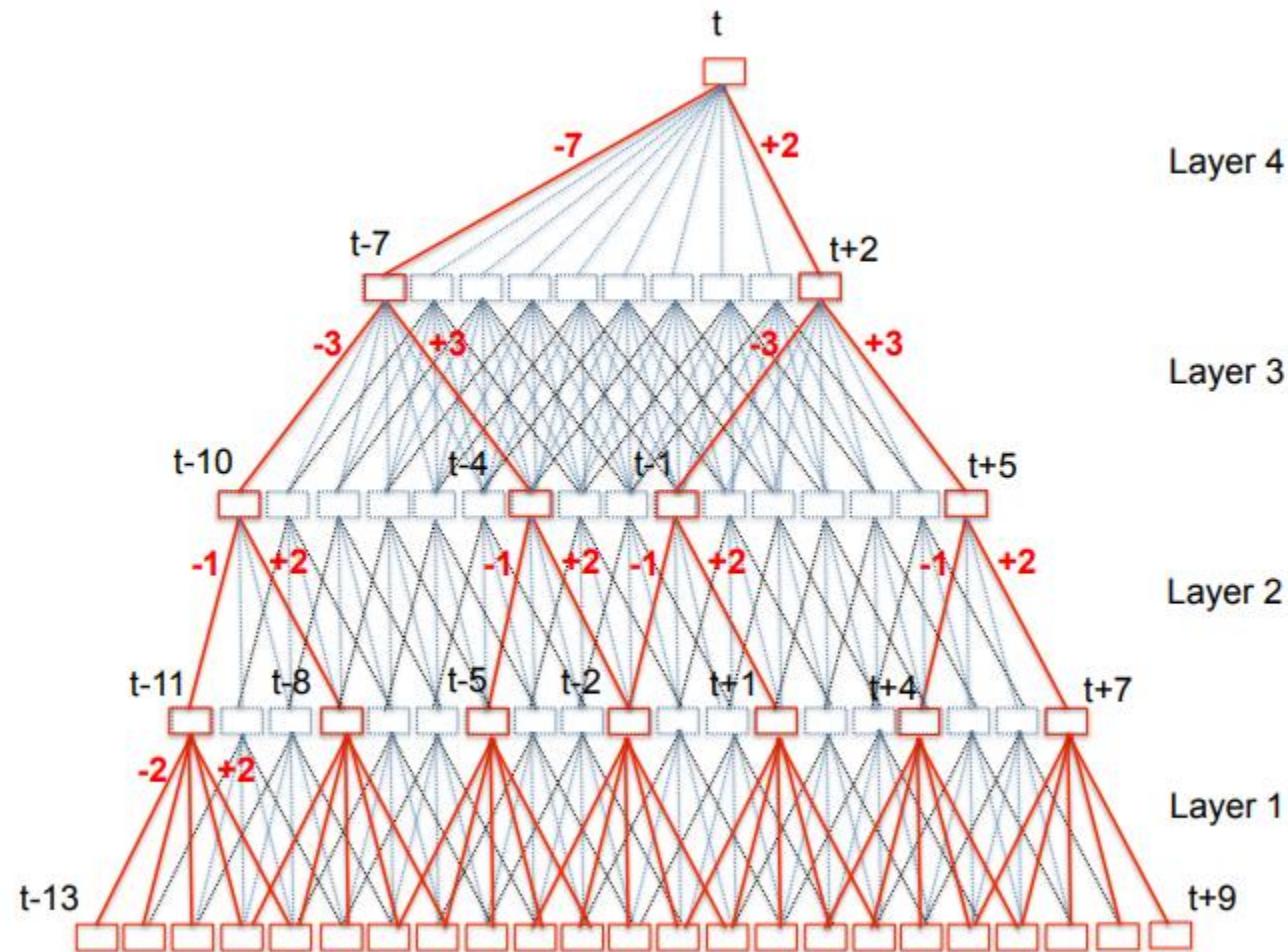
网络的配置

初始化:

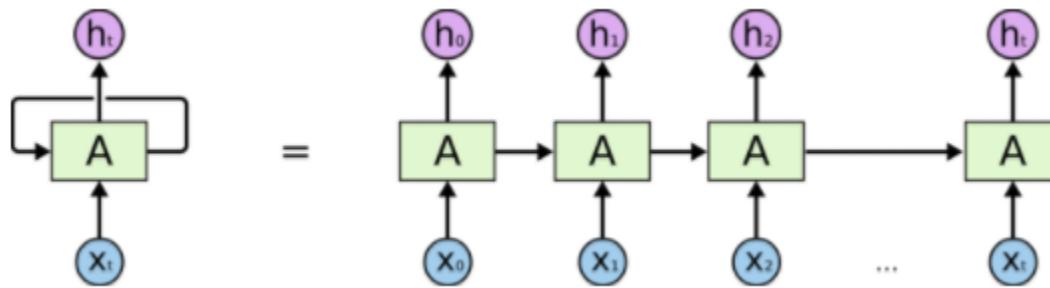
```
<Nnet3>
input-node name=input dim=40
input-node name=ivector dim=100
component-node name=L0_fixaffine component=L0_fixaffine input=Append(Offset(input, -2), Offset(input, -1), input, Offset(input, 1), Offset(input, 2), ReplaceIndex(ivector, t, 0))
component-node name=Lstm1_c_t component=Lstm1_c input=Sum(Lstm1_c1_t, Lstm1_c2_t)
component-node name=Lstm1_i1 component=Lstm1_W_i-xr input=Append(L0_fixaffine, IfDefined(Offset(Lstm1_r_t, -1)))
component-node name=Lstm1_i2 component=Lstm1_w_ic input=IfDefined(Offset(Lstm1_c_t, -1))
component-node name=Lstm1_i_t component=Lstm1_i input=Sum(Lstm1_i1, Lstm1_i2)
component-node name=Lstm1_f1 component=Lstm1_W_f-xr input=Append(L0_fixaffine, IfDefined(Offset(Lstm1_r_t, -1)))
component-node name=Lstm1_f2 component=Lstm1_w_fc input=IfDefined(Offset(Lstm1_c_t, -1))
component-node name=Lstm1_f_t component=Lstm1_f input=Sum(Lstm1_f1, Lstm1_f2)
component-node name=Lstm1_o1 component=Lstm1_W_o-xr input=Append(L0_fixaffine, IfDefined(Offset(Lstm1_r_t, -1)))
component-node name=Lstm1_o2 component=Lstm1_w_oc input=Lstm1_c_t
component-node name=Lstm1_o_t component=Lstm1_o input=Sum(Lstm1_o1, Lstm1_o2)
component-node name=Lstm1_h_t component=Lstm1_h input=Lstm1_c_t
component-node name=Lstm1_g1 component=Lstm1_W_c-xr input=Append(L0_fixaffine, IfDefined(Offset(Lstm1_r_t, -1)))
component-node name=Lstm1_g_t component=Lstm1_g input=Lstm1_g1
component-node name=Lstm1_c1_t component=Lstm1_c1 input=Append(Lstm1_f_t, IfDefined(Offset(Lstm1_c_t, -1)))
component-node name=Lstm1_c2_t component=Lstm1_c2 input=Append(Lstm1_i_t, Lstm1_g_t)
component-node name=Lstm1_m_t component=Lstm1_m input=Append(Lstm1_o_t, Lstm1_h_t)
component-node name=Lstm1_rp_t component=Lstm1_W-m input=Lstm1_m_t
dim-range-node name=Lstm1_r_t_preclip input-node=Lstm1_rp_t dim-offset=0 dim=256
component-node name=Lstm1_r_t component=Lstm1_r input=Lstm1_r_t_preclip
component-node name=Final_affine component=Final_affine input=Lstm1_rp_t
component-node name=Final_log_softmax component=Final_log_softmax input=Final_affine
output-node name=output input=Offset(Final_log_softmax, 5) objective=linear
```



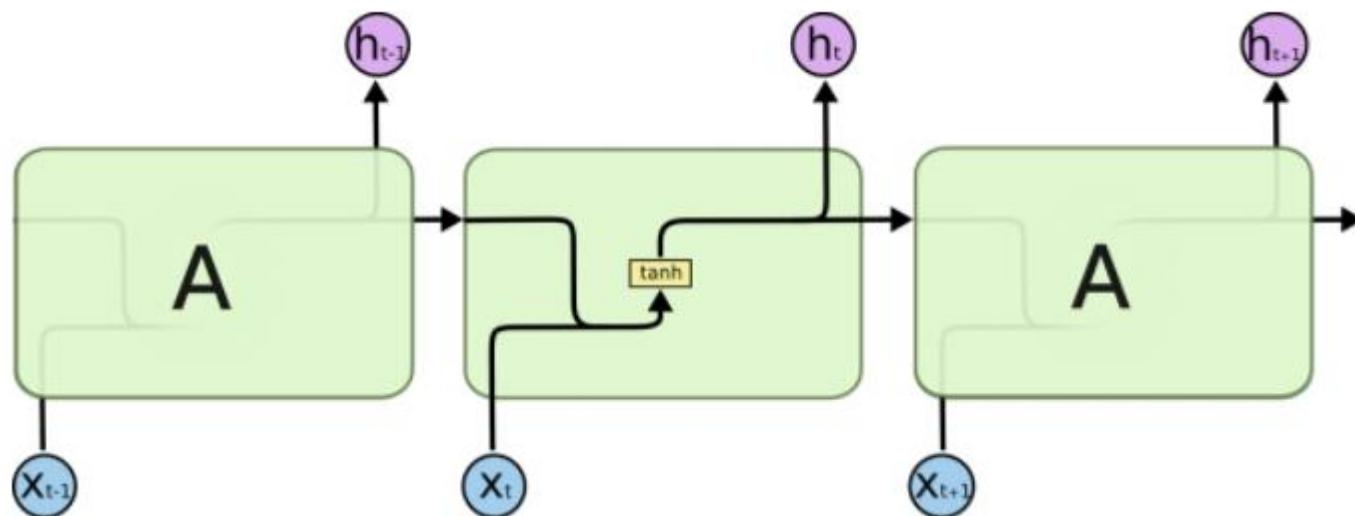
TDNN



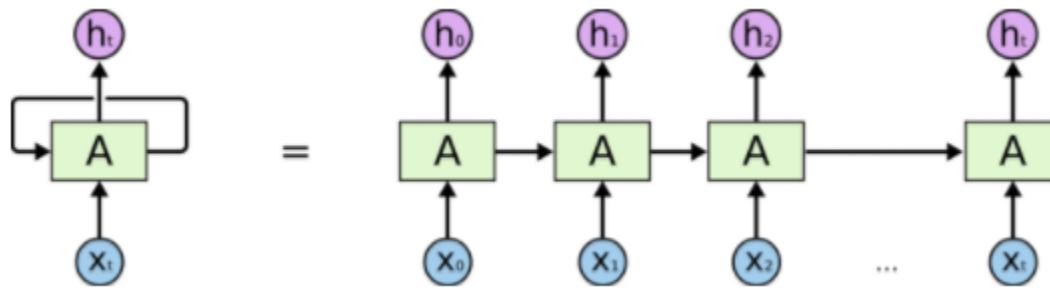
RNN



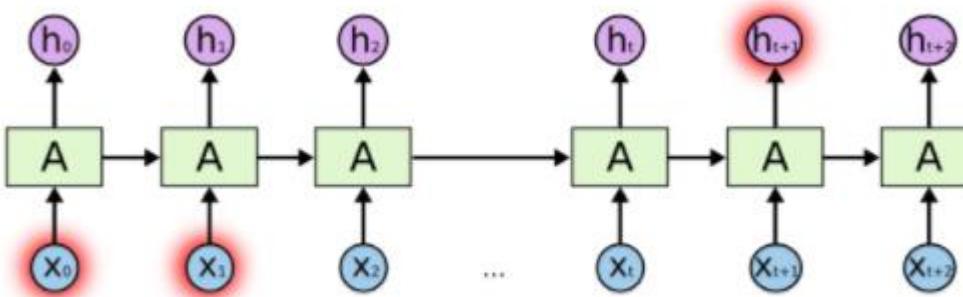
RNN



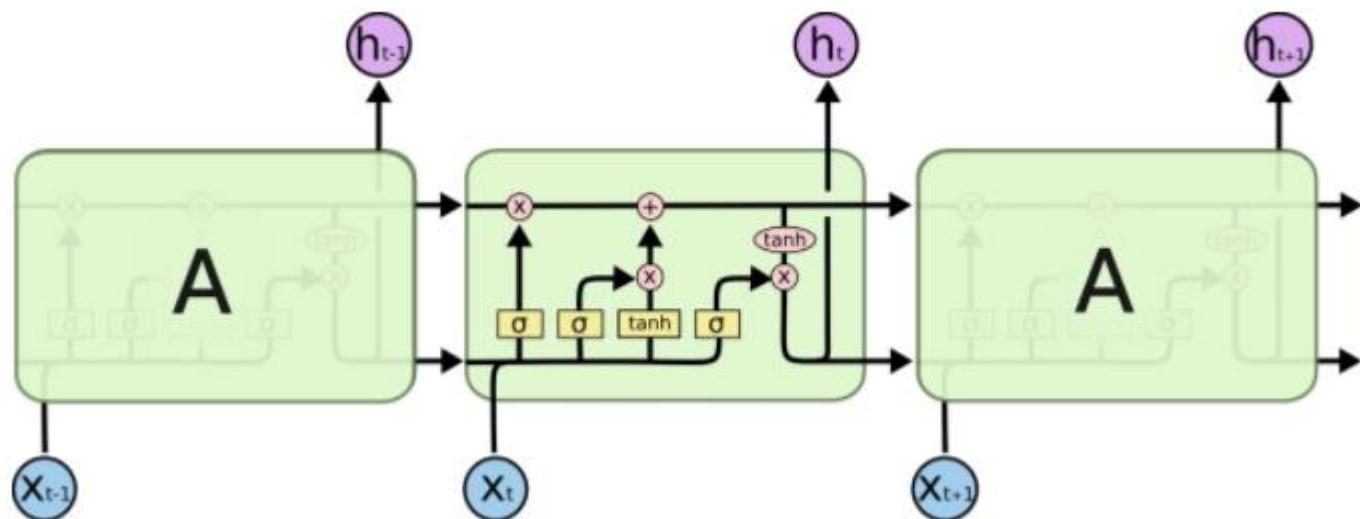
RNN



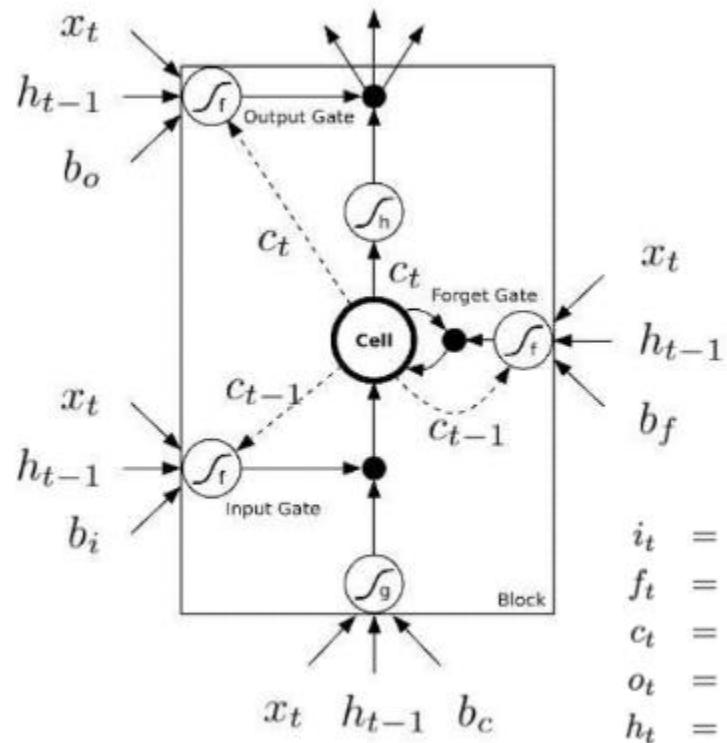
RNN



LSTM

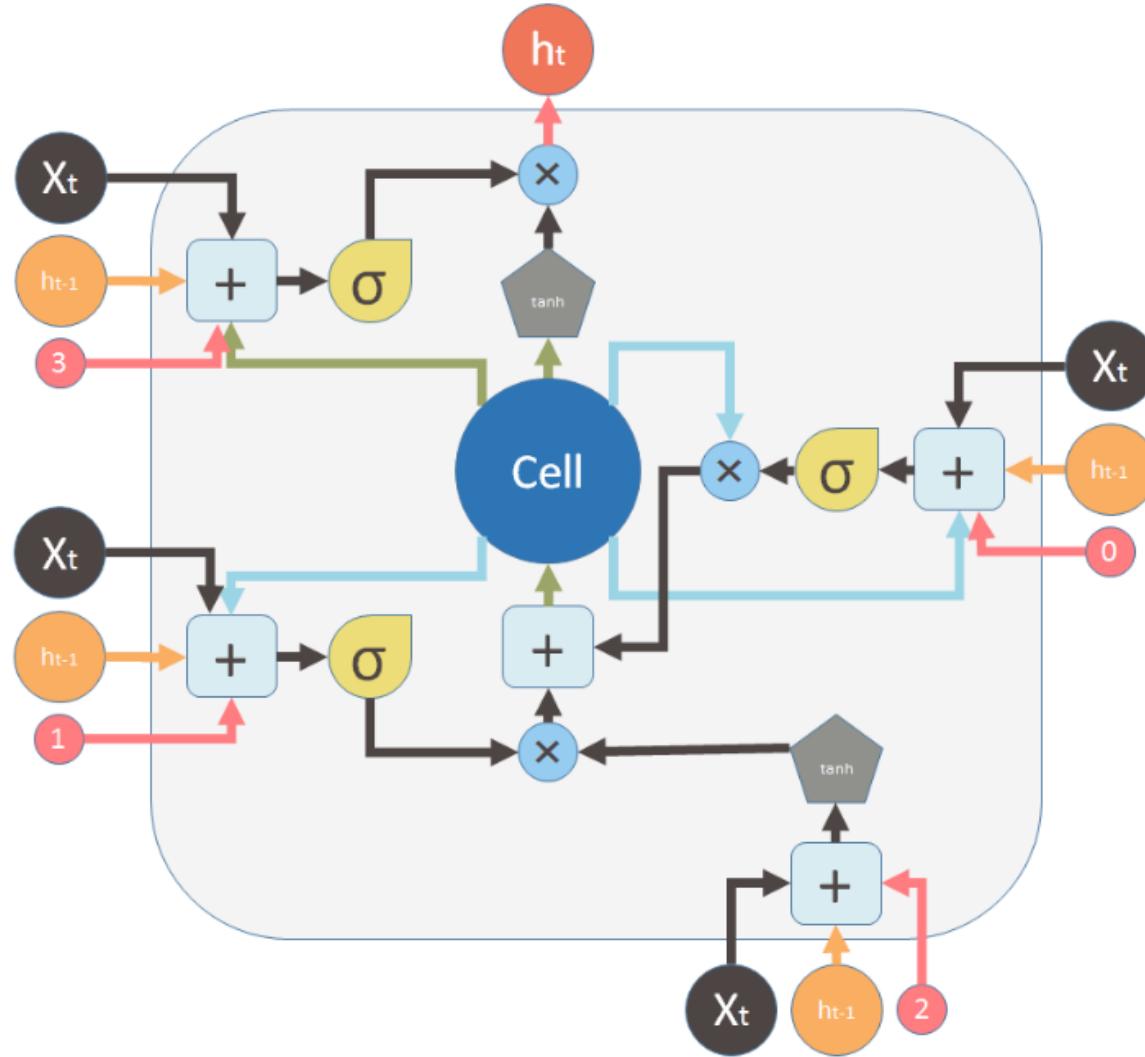


LSTM



$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\ h_t &= o_t \tanh(c_t) \end{aligned}$$

LSTM



ASR中常用的LOSS函数

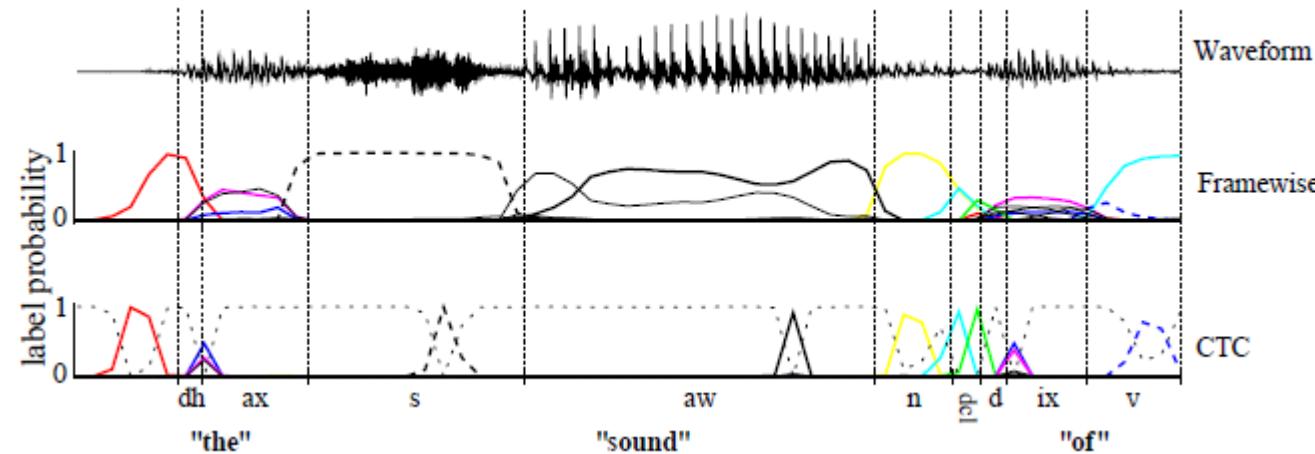
$$\mathcal{F}_{CE} = - \sum_{u=1}^U \sum_{t=1}^{T_u} \log y_{ut}(s_{ut}),$$

$$\mathcal{F}_{MBR} = \sum_u \frac{\sum_W p(\mathbf{O}_u|S)^\kappa P(W) A(W, W_u)}{\sum_{W'} p(\mathbf{O}_u|S)^\kappa P(W')},$$

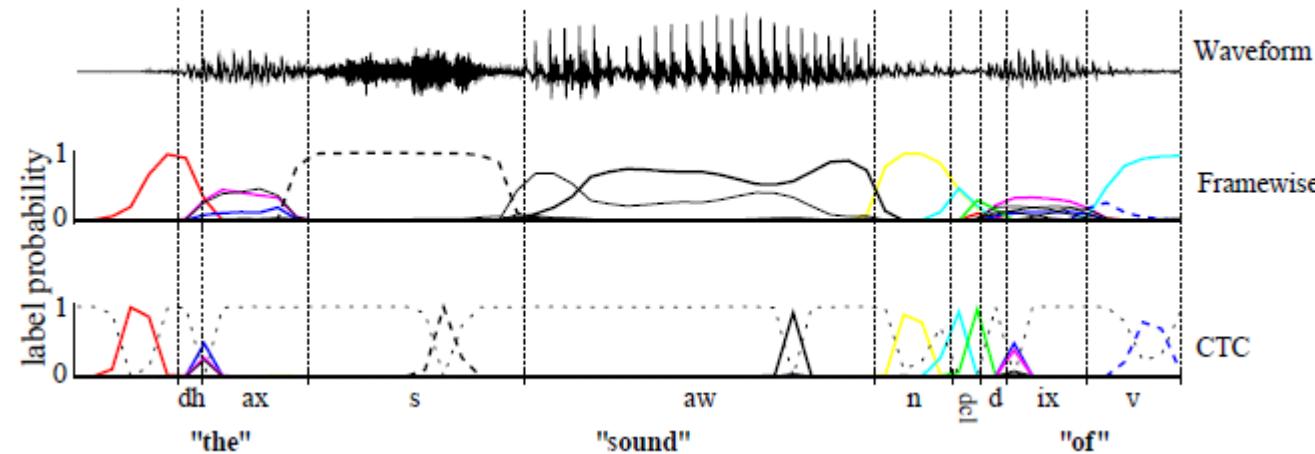
$$\mathcal{F}_{MMI} = \sum_u \log \frac{p(\mathbf{O}_u|S_u)^\kappa P(W_u)}{\sum_W p(\mathbf{O}_u|S)^\kappa P(W)},$$

$$\mathcal{F}_{BMMI} = \sum_u \log \frac{p(\mathbf{O}_u|S_u)^\kappa P(W_u)}{\sum_W p(\mathbf{O}_u|S)^\kappa P(W) e^{-b A(W, W_u)}},$$

CTC模型



CTC模型



CTC模型

- 中国
- zh ong g uo
- zh zh zh ong ong ong g g g g uo uo uo
- zh _ _ _ ong _ _ _ g _ _ _ uo _
- zh _ ong _ _ _ _ _ g _ _ _ uo _
- zh ong _ _ _ _ _ g _ _ _ uo _
- zh _ _ _ _ _ ong g _ _ _ uo _
- zh zh zh zh zh _ _ ong g _ _ _ uo _
- z ong g uo g uo g uo

语言模型在ASR中的使用

- 根据输入的特征序列 $Y=\{y_1, y_2, \dots, y_r, \dots, y_T\}$ ， 找出最有可能的输出单词序列 $\hat{w}=\{w_1, w_2, \dots, w_u, \dots, w_U\}$

$$\begin{aligned}\hat{w} &= \arg \max_w \{P(w|Y)\} \\ &= \arg \max_w \left\{ \frac{P(Y|w)P(w)}{P(Y)} \right\} \quad P(Y) \text{对所有 } w \text{ 都一样所以解码时不用考虑} \\ &= \arg \max_w \{P(Y|w)P(w)\}\end{aligned}$$

- $P(w|Y)$: 句子 w 对特征 Y 的后验概率 (posterior)。
- $P(Y|w)$: 特征 Y 对句子 w 的似然 (likelihood)，通过声学模型得到。
- $P(w)$: 句子 w 的先验概率，通过语言模型得到。

语音识别问题

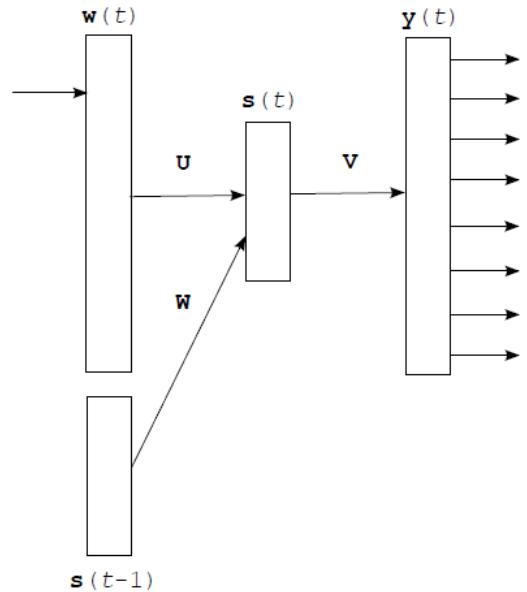
- 根据输入的特征序列 $Y=\{y_1, y_2, \dots, y_t, \dots, y_T\}$ ， 找出最有可能的输出单词序列 $\hat{w}=\{w_1, w_2, \dots, w_u, \dots, w_U\}$

$$\begin{aligned}\hat{w} &= \arg \max_w \{P(w|Y)\} \\ &= \arg \max_w \left\{ \frac{P(Y|w)P(w)}{P(Y)} \right\} \quad P(Y) \text{对所有 } w \text{ 都一样所以解码时不用考虑} \\ &= \arg \max_w \{P(Y|w)P(w)\}\end{aligned}$$

- $P(w|Y)$: 句子 w 对特征 Y 的后验概率 (posterior)。
- $P(Y|w)$: 特征 Y 对句子 w 的似然 (likelihood)，通过声学模型得到。
- $P(w)$: 句子 w 的先验概率，通过语言模型得到。

RNNLM的基本原理

- 最基本的RNNLM如下所示:



$$s(t) = f(Uw(t) + Ws(t-1))$$

$$y(t) = g(Vs(t))$$

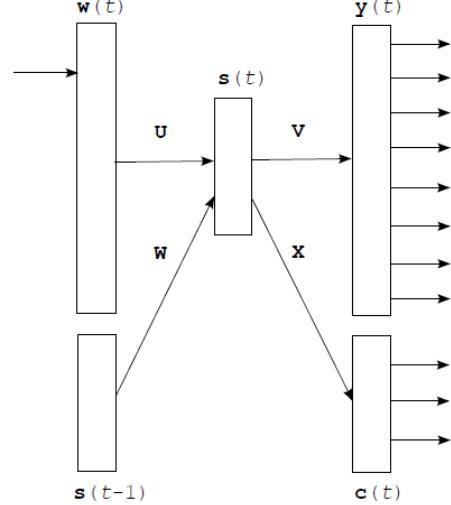
$f(\cdot)$: sigmoid or tanh or ReLU or other

$g(\cdot)$: softmax

- $w(t)$: 维度等于词汇数量，当前输入单词的对应的元素取1，其他元素为取0。(one-hot编码)
- $y(t)$: 维度等于词汇数量，每个元素代表其所对应的单词当前一时刻出现的概率(在给定前文的情况下)，也就是 $P(w_t | w_1, \dots, w_{t-1})$ 。
- $s(t)$: RNN的内部状态，因为它不仅和当前的输入有关，还与 $s(t-1)$ 相关(而 $s(t-1)$ 又与 $s(t-2)$ 相关,...)，所以它起到了对前文的记忆作用。
- 由于 $s(t)$ 的循环记忆的作用，使得RNN在预测当前单词的概率时有更长的历史输入可依据，相比之下N-gram只依据前N-1个历史输入，所以RNNLM有更精准的预测能力。

Class based RNNLM

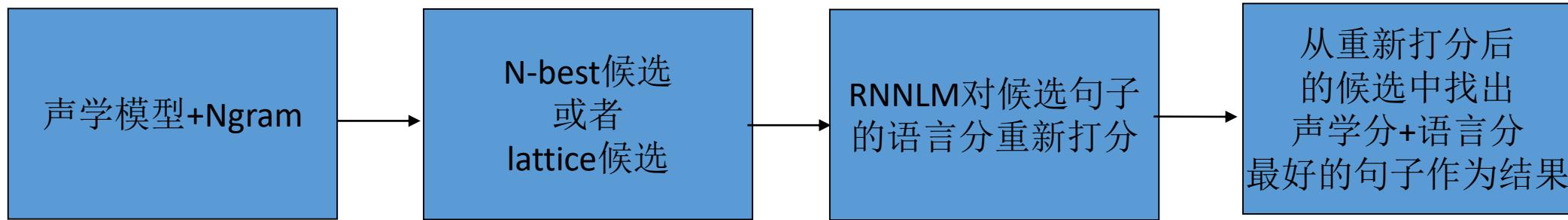
- 把每个单词对应到一个类，RNN的输出变为两项，某个类的概率，以及类中单词的概率：



$$\begin{aligned} P(w_i | s(t)) &= P(c_i | s(t))P(w_i | c_i, s(t)) \\ P(c_i | s(t)) &= g(Xs(t)) \\ P(w_i | c_i, s(t)) &= g(V_{c_i}s(t)) \quad (V_{c_i} \text{ 有多个, 每一个 } c_i \text{ 对应一个 } V_{c_i}) \\ g(\cdot) &: \text{softmax} \end{aligned}$$

- 所以 $H \times V$ 的计算量被拆分成了 $H \times C + H \times V_c$ 。
 - 假设 $H=200$, $V=90000$, 则 $H \times V = 1.8 \times 10^7$ 。
 - 如果 V 拆成 300 个类, 每个类 300 个词, 则 $C=300$, $V_c=300$, $H \times C + H \times V_c = 1.2 \times 10^5$ 。
 - softmax 的计算也是基于 C 和 V_c 的, 所以计算量也大大减小。
- 扩展——Hierarchical Softmax:
 - 把类再分类, 然后把类的类再分类, ..., 最后类的继承关系变成一棵树(比如Huffman树), 词就是树的叶子节点。
 - 运算速度的提升更大 $O(\log V)$ 。
- 缺点:
 - 不利于GPU训练。(因为训练 V_c 矩阵一直在随着输出词的变化而变化)
 - 牺牲精度。(因为RNN不但要学习词出现的概率, 还要学习词属于哪个类) (hierarchical softmax的精度就不如单层的分类)

RNNLM rescore的过程



RNNLM rescore的过程

