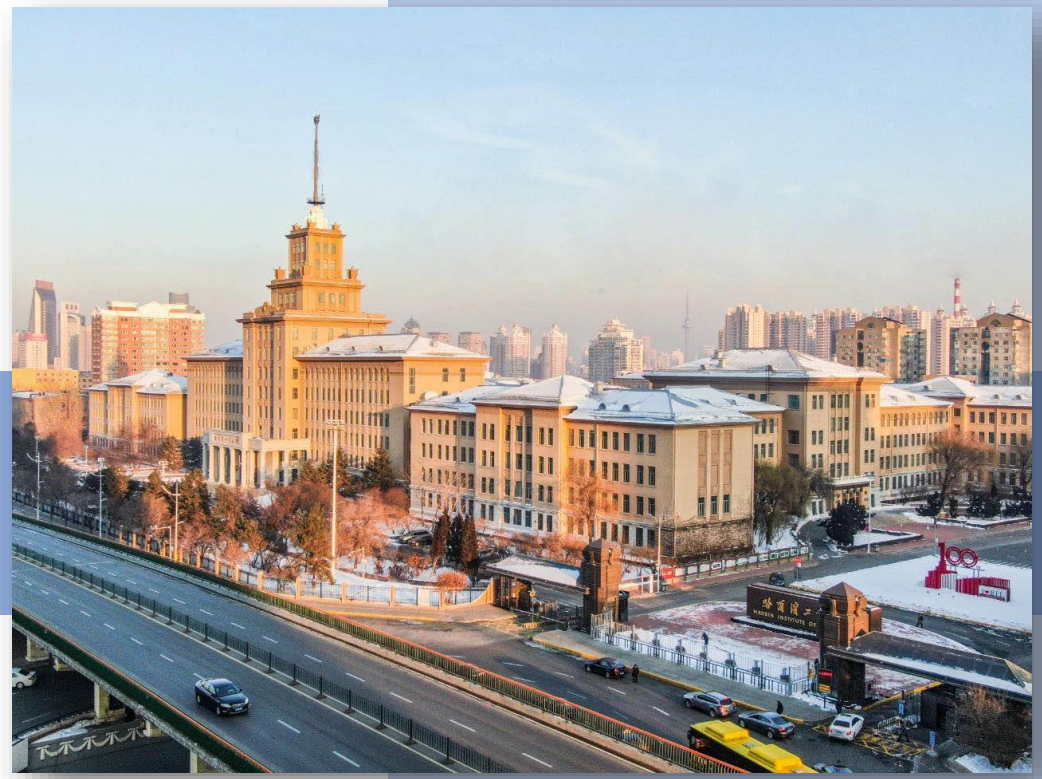
金牌讲师团
模式识别与机器学习A

2023秋期末复习讲座

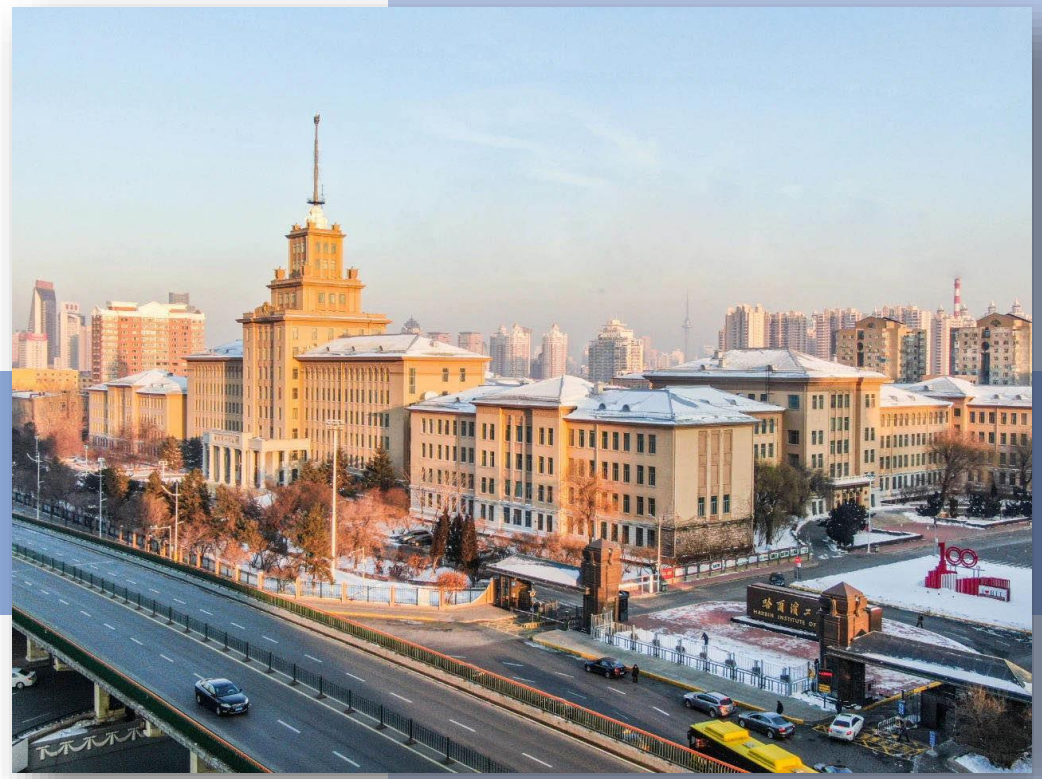
写在前面



前言

- 今年首次将机器学习、模式识别与深度学习两门课程合并为模式识别与机器学习A，学分少任务重。并且本门课不可避免的包含大量概率计算，复习讲座尽可能的覆盖三位老师讲课的重点和难点，同时为帮助大家理解增加了部分拓展内容（考试应该不会考察），希望能够对大家的复习有所帮助。
- 本门课你应该重点掌握：
  - 机器学习部分：信息增益、MLE和MAP、朴素贝叶斯、逻辑回归、聚类算法
  - 深度学习部分：多层感知机MLP、卷积神经网络CNN
- 时间仓促，且水平有限，复习过程中难免有错误之处，欢迎大家批评指正，同时也欢迎大家在QQ群内友好交流。
- 注：在准备过程中，再次感叹机器学习领域的日新月异。如果以后有志于从事人工智能相关职业和研究的同学，本门课应该作为长期学习的基础，而持续保持学习的热情。

机器学习



你应该知道的概率计算

- **事件的独立性：** A, B相互独立$\Leftrightarrow P(AB) = P(A)P(B) \Leftrightarrow P(A|B) = P(A) \Leftrightarrow A, \bar{B}$相互独立
- **条件概率：** $P(B \mid A) = \frac{P(AB)}{P(A)}, P(A) > 0 \text{ and } B \in \mathcal{F}$
- **条件独立：** 给定c的条件下，a条件独立于b $\Leftrightarrow p(a, b \mid c) = p(a \mid b, c)p(b \mid c) = p(a \mid c)p(b \mid c)$
- **全概率公式：** 如果事件 $A_1, A_2, \ldots, A_n$互不相容, 且满足$B \subset \bigcup_{j=1}^{n} A_j$， 则$P(B) = \sum_{j=1}^{n} P(A_j)P(B \mid A_j)$
- **贝叶斯公式：** 如果事件 $A_1, A_2, \ldots, A_n$互不相容, 且满足$B \subset \bigcup_{j=1}^{n} A_j$， 则有

$$P(A_j \mid B) = \frac{P(A_j, B)}{P(B)} = \frac{P(A_j)P(B \mid A_j)}{\sum_{i=1}^{n} P(A_i)P(B \mid A_i)}, \qquad 1 \le j \le n$$

- **随机变量及其分布：**
  - **期望**$E = \sum_i x_i p(x_i) \text{ or } E = \int x_i p(x_i)\, dx$
  - **方差**$V = \sum_i \left(x_i - E(X)\right)^2 p(x_i) \text{ or } V = \int \left(x_i - E(X)\right)^2 p(x_i) dx$

E. g. **高斯分布：** $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, x \in R \Longrightarrow$ 多元$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\}$

MLE和MAP

MLE: 极大似然估计

对于模型θ和样本X，目标即为$\max P(\theta \mid X) = \max \dfrac{P(X \mid \theta)P(\theta)}{P(X)}$

似然函数：$L(\theta) = P(X \mid \theta) = P(x_1 \mid \theta) \cdot P(x_2 \mid \theta) \cdots P(x_n \mid \theta) = \prod_{i=1}^{n} P(x_i \mid \theta)$

后验概率 $= \dfrac{\text{先验概率} \cdot \text{似然}}{\text{归一化项}}$

目标函数：

$$\theta_{MLE} = \text{argmax}_\theta \, L(\theta) = \text{argmax}_\theta \log L(\theta) = \text{argmin}_\theta - \log L(\theta) = \text{argmin}_\theta - \sum_{i=1}^{n} \log P(x_i \mid \theta)$$

频率学派！

MAP: 极大后验估计

在MAP中使用一个高斯分布的先验等价于在MLE中采用L2的正则项！

目标函数：$\theta_{MAP} = \arg\max_\theta P(\theta \mid X) = \arg\max_\theta \dfrac{P(X|\theta)P(\theta)}{P(X)} \propto \arg\max_\theta P(X \mid \theta)P(\theta)$

$$= \arg\max_\theta \log(P(\theta) \prod_{i=1}^{n} P(x_i \mid \theta)) = \arg\max_\theta \left(\log(P(\theta)) + \sum_{i=1}^{n} \log(P(x_i \mid \theta))\right)$$

$$= \arg\min_\theta \left(-\log(P(\theta)) - \sum_{i=1}^{n} \log(P(x_i \mid \theta))\right)$$

$P(\theta)$ 假设：简单→ $Beta$分布；复杂 → 高斯分布

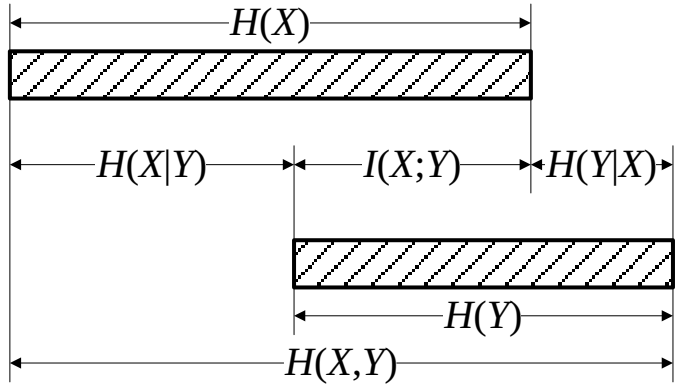
统计学派！

补充：完全贝叶斯估计，后验分布将考虑所有可能的参数值，而不仅仅是先验分布所假设的参数值。

信息熵

- **熵（Entropy）**：从X随机采样值在最短编码情况下的每个值平均（期望）长度（对称、非负）

$$H(x) = -\sum_{i=1}^{n} p(x_i)\log p(x_i)$$



- **条件熵**：$H(X|Y) = \sum_{j\in Val(y)} p(y=j)H(X|y=j)$
- **互信息**：$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y)$
- **信息增益**：在一个条件下，信息复杂度（不确定性）减少的程度（决策树）
- **相对熵（KL散度）**：衡量同一随机变量的两个不同概率分布之间的距离（差异）

$$D_{KL}(p \parallel q) = \sum_{i=1}^{n} p(x_i)\log \frac{p(x_i)}{q(x_i)}$$

  - **非对称性**：$D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$，当P、Q概率分布完全一样时才能相等；
  - **非负性**：$D_{KL}(p \parallel q) \geq 0$恒成立，当P、Q概率分布完全一样时等于0。 证明：$x - 1 \geq lnx$

- JS散度：$\mathrm{JS}(p\|q) = \frac{1}{2}KL\left(p\left\|\frac{p+q}{2}\right.\right) + \frac{1}{2}KL\left(q\left\|\frac{p+q}{2}\right.\right)$，相比于KL散度是对称的

回忆：离散信源最大熵定理？如果信源连续，何种情况熵最大？（高斯分布）

生成式模型与判别式模型

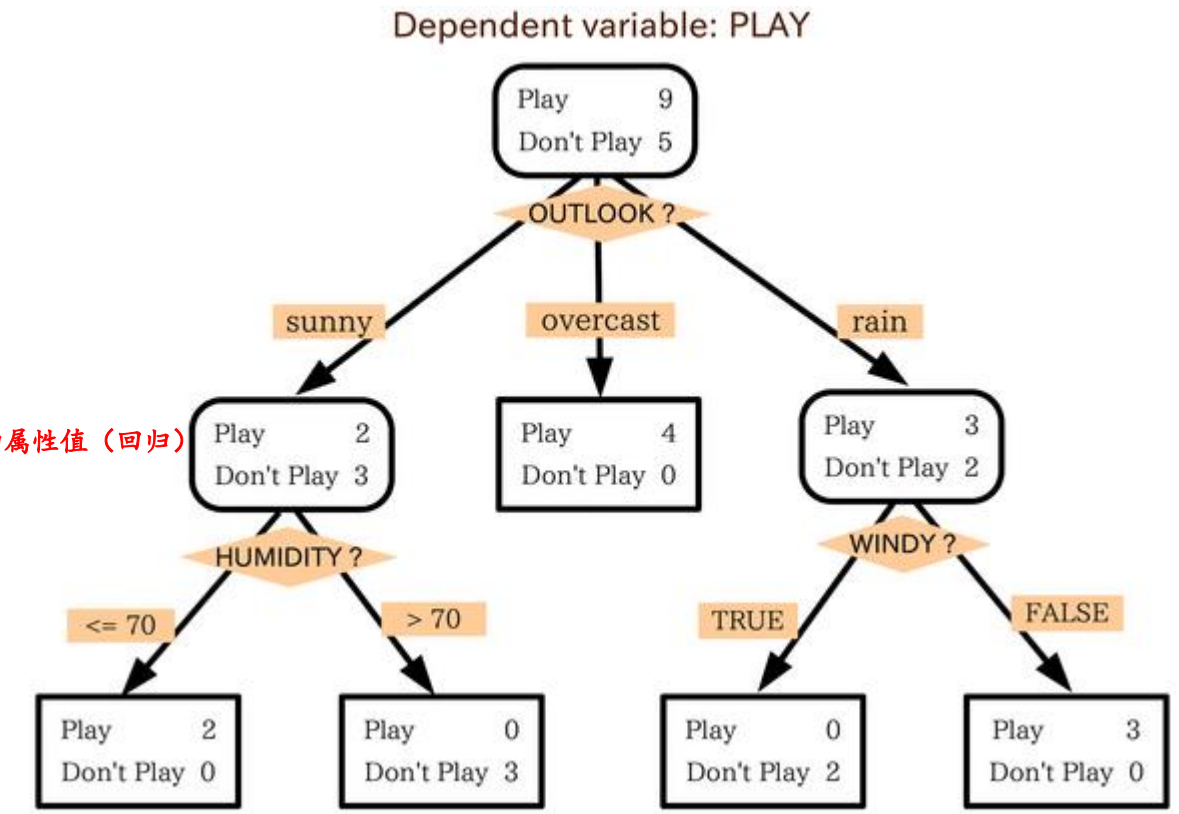
	生成式模型	判别式模型
特点	对后验概率建模，从统计的角度表示数据的分布情况，能够反映同类数据本身的相似度	寻找不同类别之间的最优分类面，反映的是异类数据之间的差异
区别	估计的是联合概率分布：$p(x,y)$	估计的是条件概率分布:$p(y\|x)$
联系	由产生式模型可以得到判别式模型， 但由判别式模型得不到产生式模型。	
常见模型	朴素贝叶斯，马尔可夫随机场等	逻辑回归，支持向量机（SVM）， 神经网络，k近邻算法等
优点	①实际上带的信息更丰富 ②研究单类问题灵活性强 ③模型可以通过增量学习得到， 能用于数据不完整情况。	①分类边界更灵活，能清晰的分辨出类别间差异特征； ②适用于较多类别的识别； ③模型简单，比较容易学习。
缺点	学习和计算过程比较复杂，且容易产生错误信息	不能反映训练数据本身的特性，变量间的关系不可视。
性能	较差	较好

决策树

- 决策树可以表示输入属性的任何函数
  - ✓ *同样一个训练数据集，决策树不唯一*
  - ✓ *具有可解释性*
- 决策树归纳算法
  - Top-Down的决策树归纳算法（贪心策略）
    - 1. A ←下一个结点node的**最好属性**
    - 2. 把A作为决策属性赋给结点node
    - 3. 对A的每一个取值，创建一个新的儿子结点node
    - 4. 把相应的训练样本分到叶结点
    - 5. 如果训练样本被很好的分类，则停止，否则在新的叶结点上重复上述过程
    - **很好的分类：一个节点上所有样本为一个类别（分类） or 一个节点上具有相似的属性值（回归）**
  - 如何确定特测条件？
    - 依赖于属性类型（名词性\离散、有序的、连续）
    - 依赖于切分的分支个数（两路切分 or 多路切分）
    - **二值决策需要考虑可能的切分并选择最好的，计算量较大**
  - C4.5（信息增益—极大似然法）和ID3 （信息增益比）
    - 信息增益$g(D,A) = H(D) - H(D|A)$
    - 信息增益比$g_R(D,A) = \frac{g(D,A)}{H_A(D)}$

Input →属性向量　　*Output* →决策Y（叶节点）

内节点→测试属性X_i　　分支→属性X_i的取值



决策树

- 决策树：基于规则来进行分类，典型的决策树算法ID3

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

- 1. 计算决策属性的熵，买/不买，得$H = 0.9573$
- 2. 计算条件属性的熵，
  - 2.1 计算年龄属性，分为3组，
    - ① 青年：买/不买:128/256，$\mathrm{H}(A_1) = 0.9183$,
    - ② 中年：买/不买:256/0，$H(A_2) = 0$
    - ③ 老年：买/不买:125/127,$H(A_3) = 0.9157$
    - ④ 计算根据年龄分组后熵的期望为$H(A) = 0.6877$

    因此，年龄熵的信息增益为：

    $$0.9573 - 0.6877 = 0.2660$$
  - 2.2 同理，计算收入属性的信息增益为：

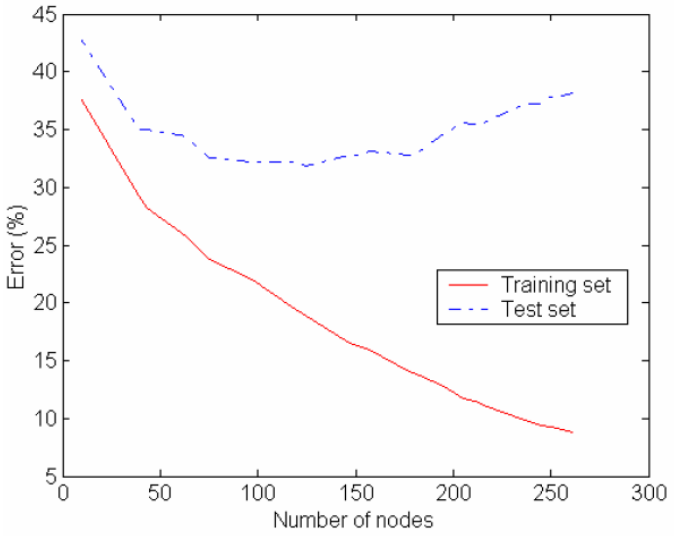
    $$0.9573 - 0.9361 = 0.0176$$
  - 2.3 同理，计算学生属性的信息增益为：

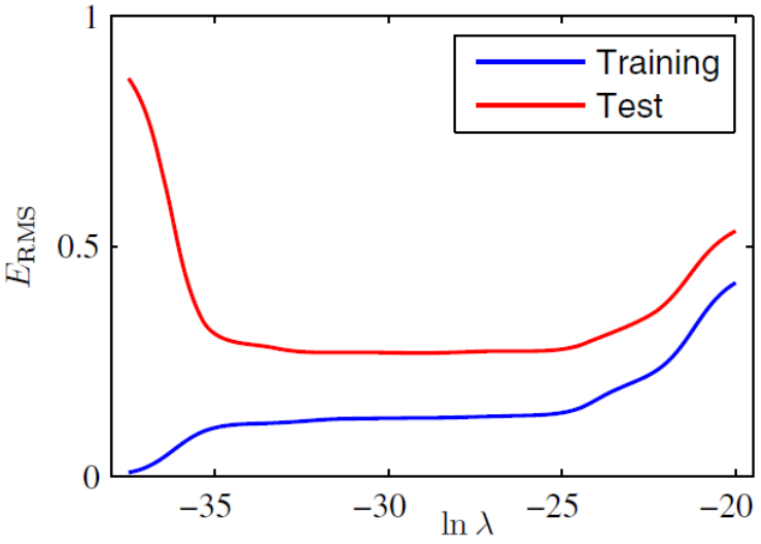
    $$0.9573 - 0.7811 = 0.1726$$
  - 2.4 同理，计算信誉属性的信息增益为：

    $$0.9573 - 0.9048 = 0.0453$$
- 问题：先选哪个属性进行分类？

过拟合与欠拟合

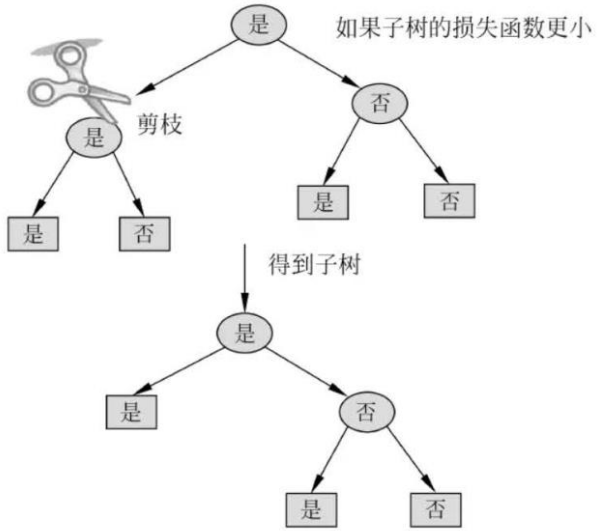
- 决策树的优点
  - 构建过程计算资源开销小
  - 分类未知样本速度极快
  - 对于小规模的树比较容易解释
  - 在许多小的简单数据集合上性能与其它方法相近
- C4.5（ID3）算法的特征
  - 深度优先构建方法
  - 在每一个结点需要对示例依据连续属性排序
  - 数据需要全部装入内存
  - 不适合大规模数据
- 决策树的缺点
  - 只考虑对训练数据的拟合
  - 欠拟合——模型对数据的复杂性或特征之间的关系进行了过于简单的建模，导致模型在训练数据上表现不佳
  - 过拟合——导致决策树比必要的更复杂，训练误差不再能够很好地估计树在以前未见过的记录上的表现





决策树的剪枝

- 早停法（预剪枝），在原来的停止条件基础上增加更多的限制性条件
  - 如果实例数量小于用户指定的阈值，则停止 ——避免在数据样本太少的情况下创建过于复杂的树结构，从而降低了模型的泛化能力
  - 如果实例的类分布独立于可用实例特征(例如，使用χ^2检验)，则停止 ——特征与类别之间没有显著的关联
  - 如果扩展当前节点没有改善不纯度(例如，基尼或信息增益)，则停止 ——防止不必要的树生长
- 后剪枝，完整地生长决策树，而后以自底向上的方式修剪决策树的节点
  - 损失函数： $C_\alpha(T)=\sum_{t=1}^{|T|} N_t H_t(T)+\alpha|T|$ 平衡正确率和模型复杂度（正则的思想）：$Cost(Model,Data)=Cost(Data|Model)+Cost(Model)$
  - 决策树的剪枝往往通过极小化决策树整体的损失函数(loss function)来实现。设树T的叶结点个数为$|T|$，t是树T的叶结点，该叶结点有N_t个样本点，其中k类的样本点有N_{tk}个，$k=1,2,\dots,K$， $H_t(T)$为叶结点t上的熵。
  - 剪枝算法
    - 1. 计算每个节点的熵
    - 2. 递归的从树的叶节点向上回缩
    - 3. 如果损失函数减小，则将父节点变为新的叶节点
    - 4. 重复步骤2，直至不能继续为止



- 处理丢失的属性值
  - 划分属性选择：信息增益$g(D,A)=\rho\cdot\big(H(D)-H(D|A)\big)$，$\rho$为无缺失样本所占比例
  - 缺失样本划分：按权重划分到各子类中（西瓜书87-88页）

分类与回归

Classification: 分类

$Input \rightarrow$ 训练样本的特征$D = \{x_1, x_2, \dots, x_n\}, x_i \in R^d$，对应离散类别$W = \{w_1, w_2, \dots, w_c\}$

$Output \rightarrow$ 分类器$y = f(x)$: $R^d \rightarrow W$

Regression: 回归分析

$Input \rightarrow$ 训练样本的特征$D = \{x_1, x_2, \dots, x_n\}, x_i \in R^d$，对应连续变量$Y = \{y_1, y_2, \dots, y_n\}, y_i \in R$

$Output \rightarrow$ 函数$y = g(x)$: $R^d \rightarrow R$

二者之间的联系：

- **利用回归模型进行分类：**可将回归模型的输出离散化以进行分类，即$y = \text{sigmoid}\big(f(x)\big)$。
- **利用分类模型进行回归：**也可利用分类模型的特点，输出其连续化的数值

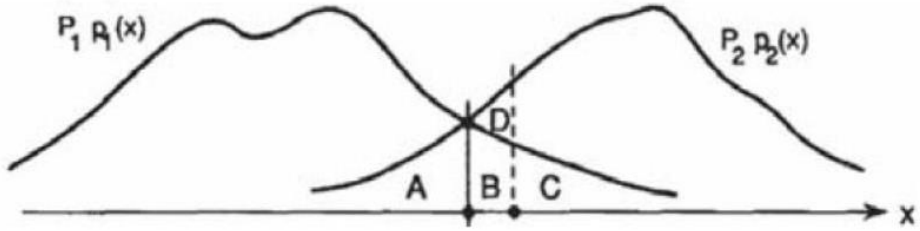
分类问题的评价指标：

- **精确率：**反映了模型判定的正例中真正正例的比重
- **召回率：**反映了总正例中被模型正确判定正例的比重

回忆：检测准则？

贝叶斯决策

- 最小错误率判别准则：
  - 错误率： $P(error|x_i) = \sum_{j\neq i} P(\omega_j|x) = 1 - P(\omega_i|x)$
  - 判别准则$i = \arg\max_{1\le j\le c} P(\omega_j|x) \Longrightarrow$贝叶斯判别准则： $i = \arg\max_{1\le j\le c} P(x|\omega_j)\,P(\omega_j)$
  - 二分类下，
    - 错误率$\varepsilon = \int \min[\pi_1 p_1(x), \pi_2 p_2(x)]\,dx = \pi_1\varepsilon_1 + \pi_2\varepsilon_2$
    - 判别准则$h(x) = -\ln p_1(x) + \ln p_2(x) \gtrless \ln\frac{\pi_1}{\pi_2} \Leftrightarrow h(x) = \frac{p(x|w_1)}{p(x|w_2)} \gtrless \frac{P(w_2)}{P(w_1)}$, 则$x \in \frac{w_1}{w_2}$



- 最小平均风险准则： （考虑错误判别带来的不同风险）
  - 有c个类别$\omega_1, \omega_2, \ldots, \omega_c$，将$\omega_i$类的样本判别为$\omega_j$的代价为$\lambda_{ij}$
  - 判别准则$i = \arg\min_{1\le j\le c} \sum_{i=1}^{c} \lambda_{ij} P(x|\omega_j)\,P(\omega_j)$
    - 二分类下$\lambda_{11}P(w_1|x) + \lambda_{12}P(w_2|x) \gtrless \lambda_{21}P(w_1|x) + \lambda_{22}P(w_2|x) \Leftrightarrow h(x) = \frac{p(x|w_1)}{p(x|w_2)} \gtrless \frac{\lambda_{12}-\lambda_{22}}{\lambda_{21}-\lambda_{11}}\frac{P(w_2)}{P(w_1)}$, 则$x \in \frac{w_1}{w_2}$
- Neyman-Pearson准则： （在限定一类错误率条件下使另一类错误率为最小）
  - 问题可以描述为$\min P_1(e),\ s.t.\, P_2(e) - \epsilon_0 = 0 \Rightarrow \min \gamma = P_1(e) + \lambda(P_2(e) - \epsilon_0)$
  - 决策边界$h(x) = \frac{p(x|w_1)}{p(x|w_2)} \gtrless \lambda$, 则$x \in \frac{w_1}{w_2}$

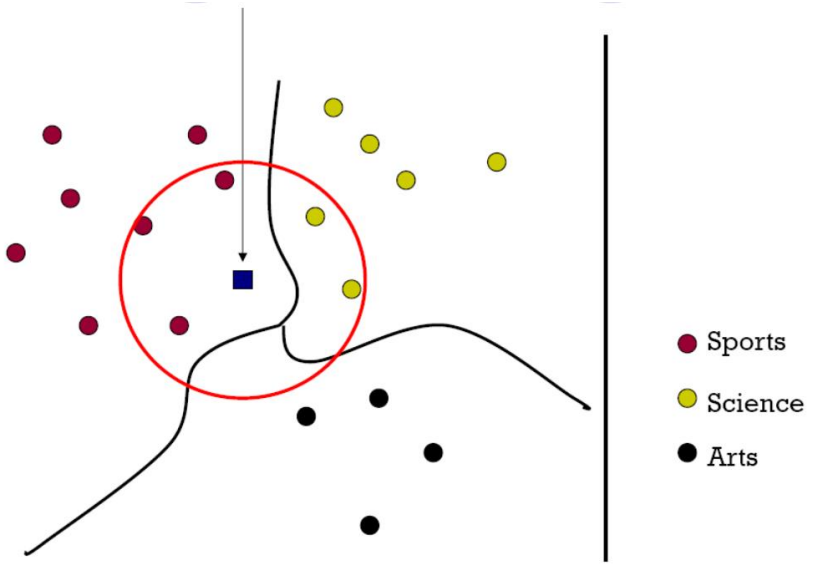
模式识别总结（1）统计决策贝叶斯理论https://zhuanlan.zhihu.com/p/114944507

非参数估计

- 基本思想：
  - 令R是包含样本点x的一个区域，其体积为V，设有n个训练样本，其中有k个落在区域R中的样本，则可对概率密度作出一个估计：$p(x) \approx \frac{k/n}{V}$，即相当于用R区域内的平均性质来作为一点$x$的估计，是一种数据的平滑
  - 有效性：取决于样本数量的多少，以及区域体积选择的合适
  - 收敛性：当$\lim_{n\to\infty} V_n = 0$，$\lim_{n\to\infty} k_n \to \infty$，$\lim_{n\to\infty} \frac{k_n}{n} = 0$时，$p_n(x)$收敛于$p(x)$
- 基本途径
  - Parzen窗法：区域体积V是样本数n的函数，$\hat{p}(X) = \frac{1}{N}\frac{k(X)}{V}$
  - K-近邻法：落在区域内的样本数k是总样本数n的函数，$\hat{p}(X) = \frac{1}{N}\frac{(k-1)}{V(X)}$
- 贝叶斯分类
  - $h(X) = -\ln\frac{p_1(X)}{p_2(X)} = -\ln\frac{(k_1-1)N_2V_2(X)}{(k_2-1)N_1V_1(X)} \gtrless \ln\frac{\pi_1}{\pi_2}$
  - 欧式距离：$D(x,x') = \sqrt{(x-x')^T\Sigma(x-x')}$

KNN算法流程：

1. 准备数据，对数据进行预处理。
2. 计算测试样本点（也就是待分类点）**到其他每个样本点的距离**。
3. **对每个距离进行排序**，然后选择出距离最小的K个点。
4. 对K个点所属的类别进行比较，根据少数服从多数的原则，将测试样本点归入在K个点中占比最高的那一类



朴素贝叶斯

- 前提假设：假设每个输入变量独立$P(X_1 \dots X_n|Y) = \prod_i P(X_i|Y)$
  - 给定Y，当$i \neq j$时，X_i和X_j条件独立$\Leftarrow (\forall i,j,k) P(X = x_i|Y = y_j, Z = z_k) = P(X = x_i|Z = z_k)$
  - $P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y) = P(X_1|Y)P(X_2|Y)$
- 分类规则
  - $P(Y = y_k|X_1 \dots X_n) = \frac{P(Y=y_k)P(X_1 \dots X_n|Y=y_k)}{\sum_j P(Y=y_j)P(X_1 \dots X_n|Y=y_j)} = \frac{P(Y=y_k)\prod_i P(X_i|Y=y_k)}{\sum_j P(Y=y_j)\prod_i P(X_i|Y=y_j)}$
  - $X^{new} = \langle X_1, \dots, X_n \rangle, Y^{new} \leftarrow \arg\max_{y_k} P(Y = y_k) \prod_i P(X_i|Y = y_k)$
- 训练
  - 对每个类别y_k，估计类别先验$\pi_k \equiv P(Y = y_k)$；对每个样本的特征X_i，计算$\theta_{ijk} \equiv P(X_i = x_{ij}|Y = y_k)$
  - MLE:

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

Number of items in dataset D for which $Y=y_k$

  - MAP:

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + \alpha_k}{|D| + \sum_m \alpha_m}$$

Only difference: "imaginary" examples

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\} + \alpha'_k}{\#D\{Y = y_k\} + \sum_m \alpha'_m}$$

E.g.文本分类

aardvark	0
about	2
all	2
Africa	1
apple	0
anxious	0
...	
gas	1
...	
oil	1
...	
Zaire	0

概率必须满足归一性，所以只需要估计$n-1$个参数

Beta分布：$P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$

1.Beta分布的取值范围在[0, 1]之间，符合实际的概率取值范围。

2.α和β参数控制了分布的形状。当α和β都为1时，Beta分布变成均匀分布，所有的取值都是等概率的。

3.Beta分布的期望值$\frac{\alpha}{\alpha+\beta}$，方差为$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$

4.在贝叶斯推断中常用作先验分布

高斯朴素贝叶斯

- 前提假设：特征连续，假设每个输入变量独立$P(X_1 \dots X_n|Y) = \prod_i P(X_i|Y)$
  - 假设特征在给定类别下的条件概率分布是高斯分布
  - $P(X_i = x \mid Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\exp(-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2})$　　进一步假设方差与Y, X无关，$\sigma_{ik} \rightarrow \sigma$
- 分类规则：
  - $X^{new} = \langle X_1, \dots, X_n \rangle, Y^{new} \leftarrow \arg\max_{y_k} P(Y = y_k) \prod_i \mathcal{N}(X_i, \mu_{ik}, \sigma_{ik})$
- 训练
  - 对每个类别y_k，估计类别先验$\pi_k \equiv P(Y = y_k)$；
  - 对每个样本的特征X_i，计算特征均值与方差μ_{ik}, σ_{ik}

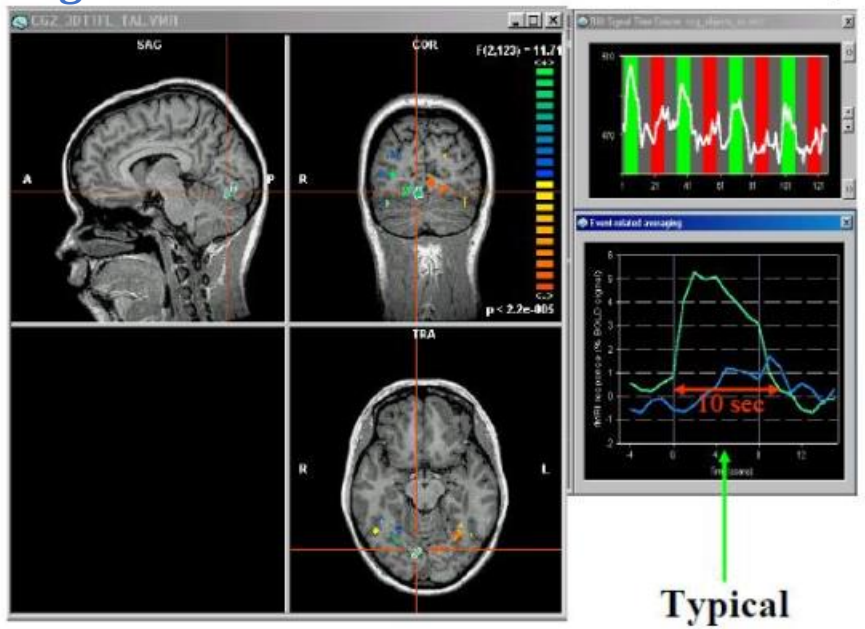
Maximum likelihood estimates:

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

ith feature　kth class　jth training example　δ(z)=1 if z true, else 0

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

E.g.精神状态分类



Typical impulse response

线性分类器—逻辑回归

- **想法：**直接计算$P(Y|X)$，而不必构造$P(Y|X) \propto P(X|Y)P(Y)$
- **模型设计：**
  - 输入：$Y \in R^n$；输出：$X \in \{0,1\}$。
  - 假设：$X|Y$服从伯努利分布
  - 模型：θ^T，$\theta^T Y = \theta_0 + \theta_1 y^{(1)} + \cdots + \theta_n y^{(n)}$

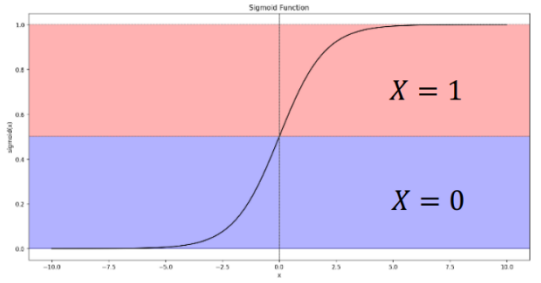
$$P(X=1|Y) = \frac{1}{1+e^{-\theta^T y}};\ P(X=0|Y) = 1 - \frac{1}{1+e^{-\theta^T y}}$$

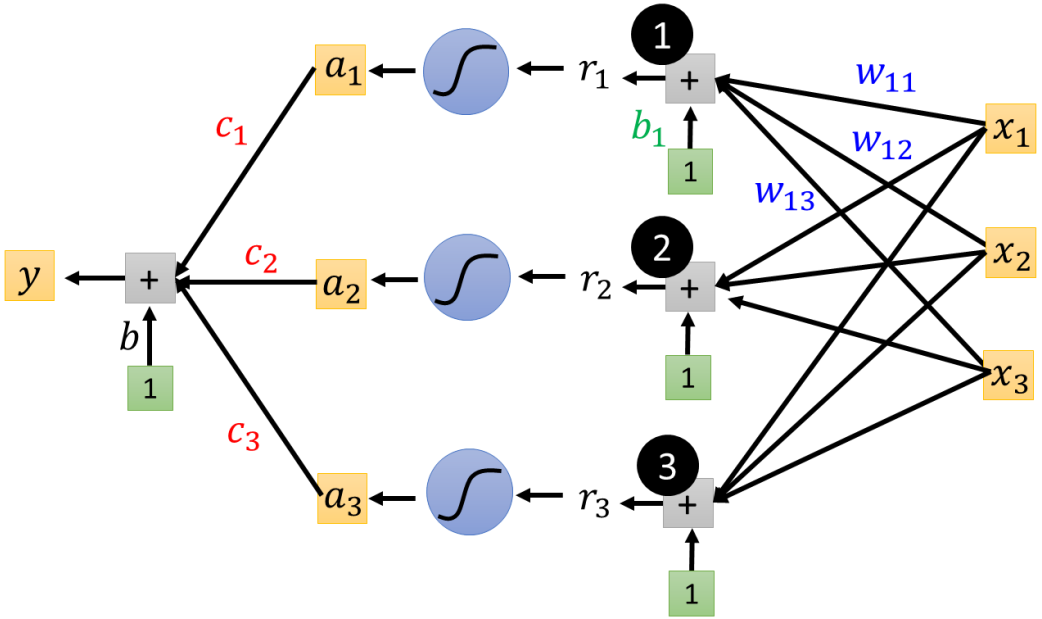
  - 决策判别：根据两个条件概率值的大小将实例分到概率值较大的一类。
- **优化：**
  - 似然函数：$L(\theta) = \prod_{i=1}^{m} P(x_i|z_i;\theta) = (h_\theta(z))^x (1-h_\theta(z))^{1-x}$（**MLE**）
  - 负变换：$L(\theta) = \left(-\frac{1}{m}\right) \prod_{i=1}^{m} \log P(x_i|z_i;\theta)$

$$= -\frac{1}{m} \sum_{i=1}^{m} [x_i \log h_\theta(z_i) + (1-x_i) \log((1-h_\theta(z_i))]$$

  - 优化方法：**梯度下降和牛顿法**

$$Sigmoid(Z) = \frac{1}{1+e^{-z}}$$





$$y = b + c^T \sigma(b + W x)$$

线性分类器—逻辑回归

- **理论推导（选看）：**直接计算$P(Y|X)$，而不必构造$P(Y|X) \propto P(X|Y)P(Y)$
  - $P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1)+P(Y=0)P(X|Y=0)} = \frac{1}{1+\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}} = \frac{1}{1+\exp\left(\ln\frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}\right)}$

$$\sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)$$

高斯朴素贝叶斯→假设方差与Y无关

$$= \frac{1}{1+\exp\left(\left(\ln\frac{1-\pi}{\pi}\right)+\sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)}\right)} \Rightarrow P(Y=1|X) = \frac{1}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)}$$

- 为什么是线性的？
  - 决策面为：$P(Y=1|X) = \frac{1}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)} = P(Y=0|X) = \frac{\exp(w_0+\sum_{i=1}^n w_i X_i)}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)} \Leftrightarrow \boldsymbol{w_0+\sum_{i=1}^n w_i X_i = \ln 1 = 0}$
- **拓展到多分类问题**
  - **Sigmoid → Softmax：** $P(Y=y_k|X) = \frac{\exp(w_{k0}+\sum_{i=1}^n w_{ki}X_i)}{\sum_{j=1}\exp(w_{j0}+\sum_{i=1}^n w_{ji}X_i)}$
- **MAP与正则**
  - $L(\theta)\prod_{i=1}^m P(x_i|z_i;\theta) = (h_\theta(z))^x(1-h_\theta(z))^{1-x} + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$
  - $P(\theta) \propto \exp(-\lambda\|\theta\|_2^2)$

Features, X

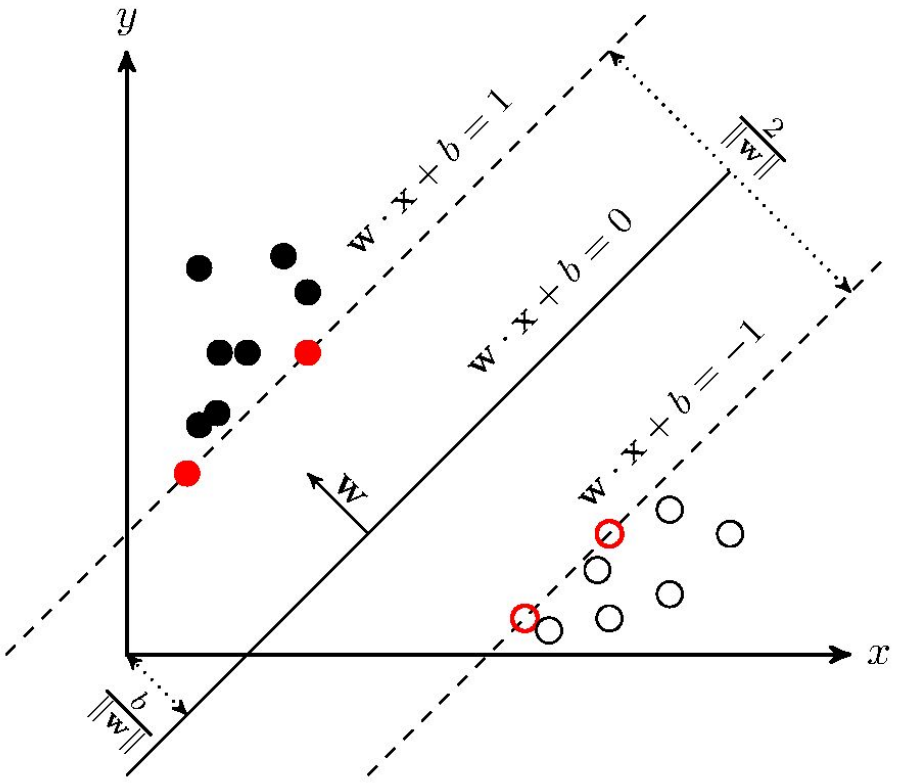
Sports
Science
News

Labels, Y

线性分类器—支持向量机

- 想法：寻找最优分解面（支持向量的引入）
- 模型设计：
  - 输入：$T=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\},\ x_i\in\mathbb{R}^n,\ y_i\in\{+1,-1\},\ i=1,2,\dots N$
  - 输出：分类函数g: $\mathbb{R}^n\to\mathbb{R}\to\{+1,-1\}$
  - 假设：数据是线性可分的—$y_i=1$，有$w^T\cdot x+b\geq\varepsilon$；$y_i=0$，有$w^T\cdot x+b\leq\varepsilon$
- 理论推导：
  - 几何间隔：$r_i=\frac{|w^Tx_i+b|}{\|w\|}=y_i\left(\frac{w^T}{\|w\|}\cdot x_i+\frac{b}{\|w\|}\right),r=\min_{i=1,2,\dots N}r_i$
  - 支持向量：$\text{margin}=\frac{w^T}{\|w\|}(x_1^*-x_2^*)=\frac{2r}{\|w\|}$
  - 目标函数：$\max_{w,b}\frac{2r}{\|w\|},\ s.t.\,y_i\left(\frac{w^T}{\|w\|}\cdot x_i+\frac{b}{\|w\|}\right)\geq r,i=1,2,\dots N$

    等价于$\min_{w,b}\frac{1}{2}\|\hat{w}\|^2,s.t.\,y_i\left(\hat{w}^T\cdot x_i+\hat{b}\right)\geq 1,i=1,2,\dots N,\ \hat{w}=\frac{w}{\|w\|r},\hat{b}=\frac{b}{\|w\|r}$
  - 拉格朗日函数：$L(w,b,\mu)=\frac{1}{2}\|\hat{w}\|^2-\sum_{i=1}^N\mu_i\left[y_i\left(\hat{w}^T\cdot x_i+\hat{b}\right)-1\right],\mu_i\geq 0$



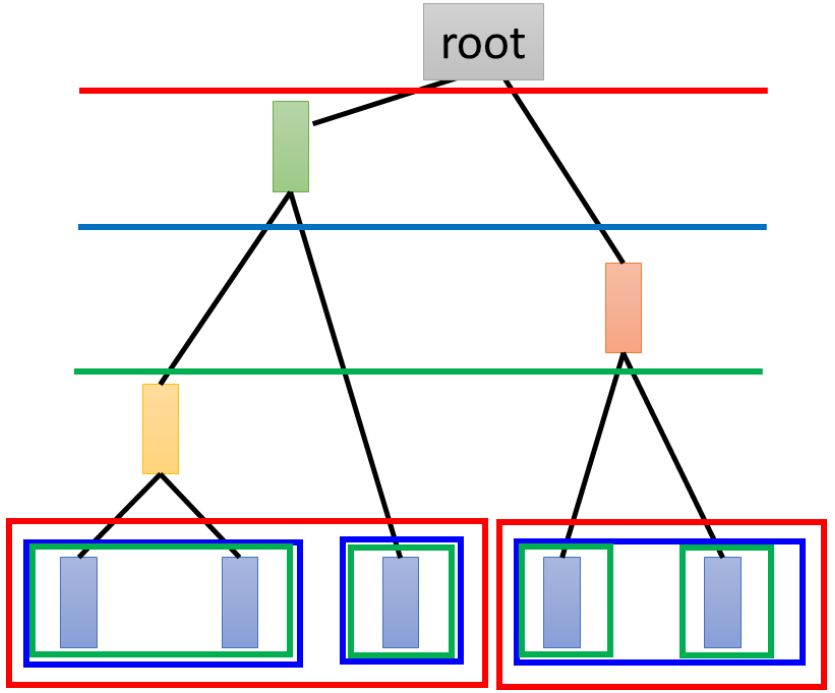
线性分类器—支持向量机

- **优化函数：** $\max_{\mu}\min_{w,b} L(w,b,\mu) = \max_{\mu}\min_{w,b} \frac{1}{2}\|\hat{w}\|^2 - \sum_{i=1}^{N}\mu_i\left[y_i\left(\hat{w}^T\cdot x_i+\hat{b}\right)-1\right], \mu_i \geq 0$
  - 1. 求解$\min_{w,b} L(w,b,\mu)$
    - 对L 进行求导 $\begin{cases}\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N}\mu_i y_i x_i = 0\\ \frac{\partial L}{\partial b} = -\sum_{i=1}^{N}\mu_i y_i = -\mu^T y = 0\end{cases}$
    - 解得$w = \sum_{i=1}^{N}\mu_i y_i x_i$

    ⬅ KKT条件 $\begin{cases}\mu_i & \geq 0\\ y_i(\boldsymbol{w}_i\cdot\boldsymbol{x}_i+b)-1 & \geq 0\\ \mu_i(y_i(\boldsymbol{w}_i\cdot\boldsymbol{x}_i+b)-1) & = 0\end{cases}$
  - 2. 带回$L(w,b,\mu)$
    - $\max L(w,b,\mu) = \sum_{i=1}^{m}\mu_i - \frac{1}{2}\sum_{i,j=1}^{m}\mu_i\mu_j y_i y_j\left(x_i^T x_j\right),\ s.t\sum_{i=1}^{N}\mu_i y_i = 0,\ \mu_i \geq 0$
    - 解得最优解$\mu^* = [\mu_1^*, \mu_2^*, .., \mu_n^*]^T$（过程略），$w^* = \sum_{i=1}^{N}\mu_i^* y_i x_i$，$b^* = y_i - \sum_{i=1}^{n}\mu_i^* y_i x_i^T x_i$
    - w^*和b^*仅由对应$\mu_i^* > 0$的样本点(x_i, y_i)决定，这样的x_i即为**支持向量**
- **加入非线性因素**
  - **核函数，**非线性映射$\varphi: \mathbb{R}^n \to \mathbb{R}^m (m \gg n)$将低维的输入解耦至高维空间
  - **软阈值soft margin：引入松弛变量ξ_i进行修正：** $\min_{w,b}\frac{1}{2}\|\hat{w}\|^2 + C\sum_{i=1}^{N}\xi_i, s.t.\begin{cases}y_i\left(\hat{w}^T\cdot x_i+\hat{b}\right) \geq 1-\xi_i\\ \xi_i \geq 0\end{cases}, i = 1,2,\ldots N, C > 0$

无监督学习—聚类

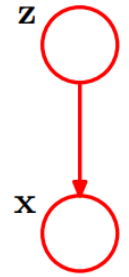
- K-means
  - 将数据集 $X = \{x_1, \dots, x_N\}, x_i \in \mathbb{R}^D$ 划分为K个类别
  - 初始化类别中心$\mu_k, k = 1,2, \dots K$ (K random x^n from X)
  - 重复迭代
    - For all x_n in X: $r_{nk} = \begin{cases} 1, & k = argmin_j \|x_n - \mu_j\|^2 \\ 0, & \text{其他情况} \end{cases}$
    - Updating all μ_k: $\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$
- 层次聚类
  - 建立划分树
  - 类别之间的划分
    - 自顶向下：一开始所有个体都属于一个簇，找出簇中距离最远的两个簇进行**分裂**，不断重复到预期簇或者其他终止条件
    - 自下而上：每个个体都是一个簇，然后根据距离寻找同类合并，最后形成一个簇
  - 距离选择
    - **Single Linkage**-两个组合数据点中距离**最近**的两个数据点间的距离
    - **Complete Linkage**-两个组合数据点中距离**最远**的两个数据点间的距离
    - **Average Linkage**-两个组合数据点中的每个数据点与其他所有数据点的距离的**均值**
    - **Centroid**−两个组合间的**中心**间的距离



无监督学习—聚类

- **想法：** 任意连续概率密度都能用多个高斯分布的线性组合叠加的高斯混合概率分布$p(x)=\sum_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)$来描述。

回忆：K-means聚类相当于假设$P(X|Y)$服从多元高斯分布（特征之间相互独立，协方差矩阵$\Sigma=\lambda I$），且$P(Y)$为等概率均匀分布。



- **模型细节：**
  - 引入隐变量构造条件概率$p(x|z_k=1)=\mathcal{N}(x|\mu_k,\Sigma_k)\Leftrightarrow p(x|z)=\prod_{k=1}^{K}\mathcal{N}(x|\mu_k,\Sigma_k)^{z_k}$
  - 边缘概率$p(x)=\sum_z p(z)\cdot p(x|z)=\sum_{k=1}^{K}\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)$
  - 后验概率$\gamma(z_k)\equiv p(z_k=1|x)=\frac{p(z_k=1)p(x|z_k=1)}{\sum_{j=1}^{K}p(z_j=1)p(x|z_j=1)}=\frac{\pi_k\mathcal{N}(x|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x|\mu_j,\Sigma_k)}$
  - 对数似然函数为$\ln p(X\mid\pi,\mu,\Sigma)=\sum_{n=1}^{N}\ln\{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_n\mid\mu_k,\Sigma_k)\}$
- **优化：EM算法**
  - 计算后验概率$\gamma(z_{nk})$
  - $\ln p(X\mid\pi,\mu,\Sigma)$分别关于$\mu_k$、 Σ_k、 π_k求导
  - **注意：**关于π_k求导时需满足π_k需要满足和为1的条件**（拉格朗日乘子法）**

ALGORITHM EM for Gaussian Mixture Models

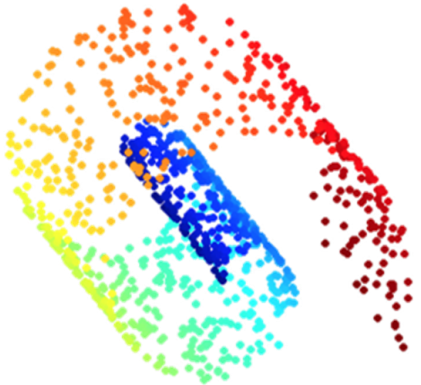
1：**input** X←数据集，K←类别数目，$iter$←迭代次数；
2：初始化均值μ_k、协方差Σ_k和混合系数π_k
3：计算对数似然$\ln p(X\mid\pi,\mu,\Sigma)\leftarrow\sum_{n=1}^{N}\ln\{\sum_{k=1}^{K}\pi_k\mathcal{N}(x_n\mid\mu_k,\Sigma_k)\}$
4：**while** $i<iter$ **do**
5：　　$\gamma(z_{nk})\leftarrow\frac{\pi_k\mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j,\Sigma_j)}$；　　（E 步）
6：　　$\mu_k^{new}\leftarrow\frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_n$；
7：　　$\Sigma_k^{new}\leftarrow\frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(x_n-\mu_k^{new})(x-\mu_k^{new})^T$；　　（M 步）
8：　　$\pi_k^{new}\leftarrow\frac{N_k}{N}$；
9：**end while**
10：**return** μ_k,Σ_k,π_k //返回最优参数；

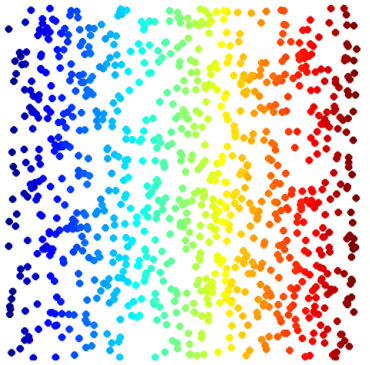
无监督学习—降维

- 想法：将高维数据投影到低维空间，同时保留他们的多样性。
- 模型细节：
  - 样本集$X=\{x^1,x^2,\dots,x^m\}\in\mathbb{R}^{n\times m},x^i\in\mathbb{R}^n$，表示$m$个包含$n$个特征的样本
  - 投影矩阵$W=[w^1,w^2,\dots,w^k]^T\in\mathbb{R}^{k\times n},w^i\in\mathbb{R}^n$
  - 特征提取$W\cdot X=Z\in\mathbb{R}^{k\times m}\rightarrow\{z^1,z^2,\dots,z^m\}$
- 理论推导：最大化方差←熵
  - $z=Wx$—样本x，z_i表示第i维新的特征，z_i^j表示第j个样本x^j经降维投影后的第i维特征
  - $\max\ Var(z_i)=\frac{1}{m}\sum_{j=1}^{m}\left(z_i^j-\bar{z_i}\right)^2$，$\left\|w^i\right\|_2=1$且$(w^i)^T\cdot w^j=0$—方差最大
  - 化简得$Var(z_i)=\left(w^i\right)^T Cov(x)w^i=(w^i)^T Sw^i$
  - 根据正交条件构造拉格朗日方程组如下：

$$g(w^1)=(w^1)^TSw^1-\alpha((w^1)^Tw^1-1)$$
$$g(w^2)=(w^2)^TSw^2-\alpha((w^2)^Tw^2-1)-\beta((w^2)^Tw^1-0)$$
$$\dots\dots$$
$$g(w^k)=(w^2)^TSw^2-\alpha((w^2)^Tw^2-1)-\sum_{j=1}^{k-1}\beta_j\left((w^k)^Tw^j-0\right)$$

  - 求导解得$w^1,\dots,w^k$分别对应协方差矩阵S的前k大特征值$\lambda_1,\dots,\lambda_k$的特征向量


Looks like 3-D

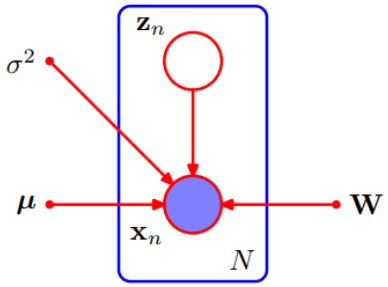

Actually, 2-D

ALGORITHM Principal Component Analysis (主成分分析)

1：**input** X←高维数据矩阵$(m\times n)$，k←降维目标；
2：$X\leftarrow(X-\bar{X})$ //数据中心化；
3：协方差矩阵$C\leftarrow 1/m(X\cdot X^T)$；
4：求出C的特征值和特征矩阵，$W\leftarrow C$前k大的特征值对应特征向量构成；
5：**return** $W\cdot X$ //返回降维后的矩阵；

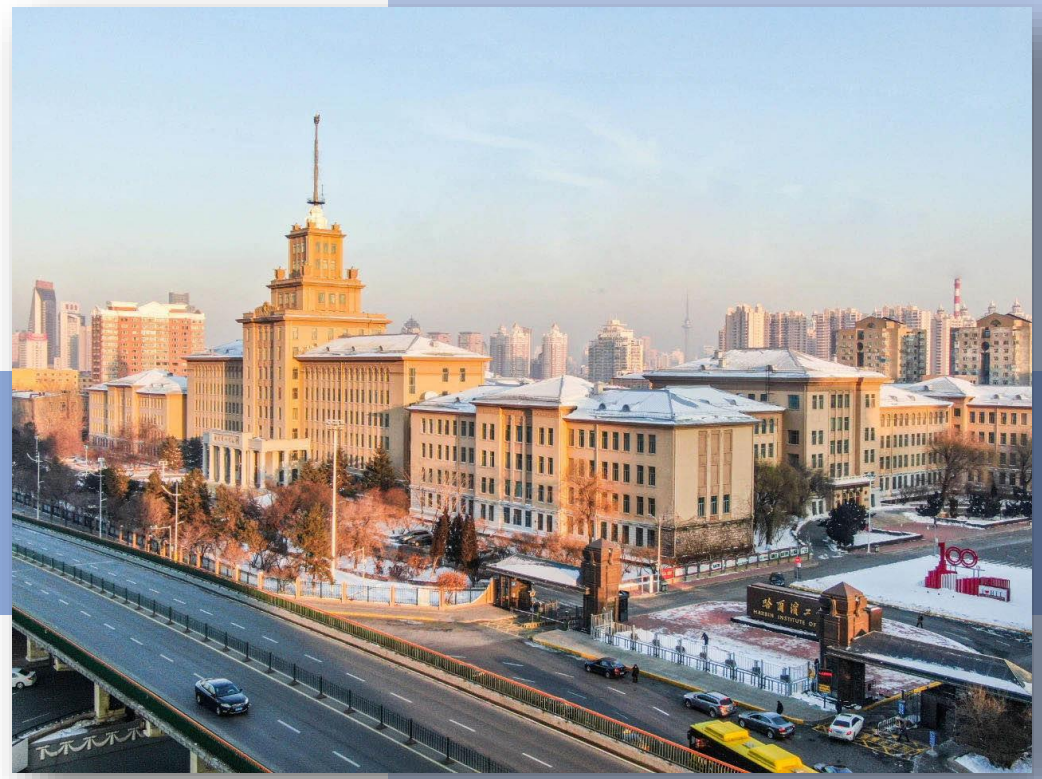
拓展： 概率PCA

- 想法：线性投影→ 概率潜在变量模型
- 模型细节：
  - 潜在变量$p(z)=\mathcal{N}(z\mid 0,I)$， $z\in\mathbb{R}^k$
  - 观测变量$p(x\mid z)=\mathcal{N}(x\mid Wz+\mu,\sigma^2 I)$， $W\in\mathbb{R}^{n\times k}$
  - n维观测变量x由k维潜在变量z的一个线性变换附加一个高斯“噪声”定义：$x=Wz+\mu+\varepsilon$



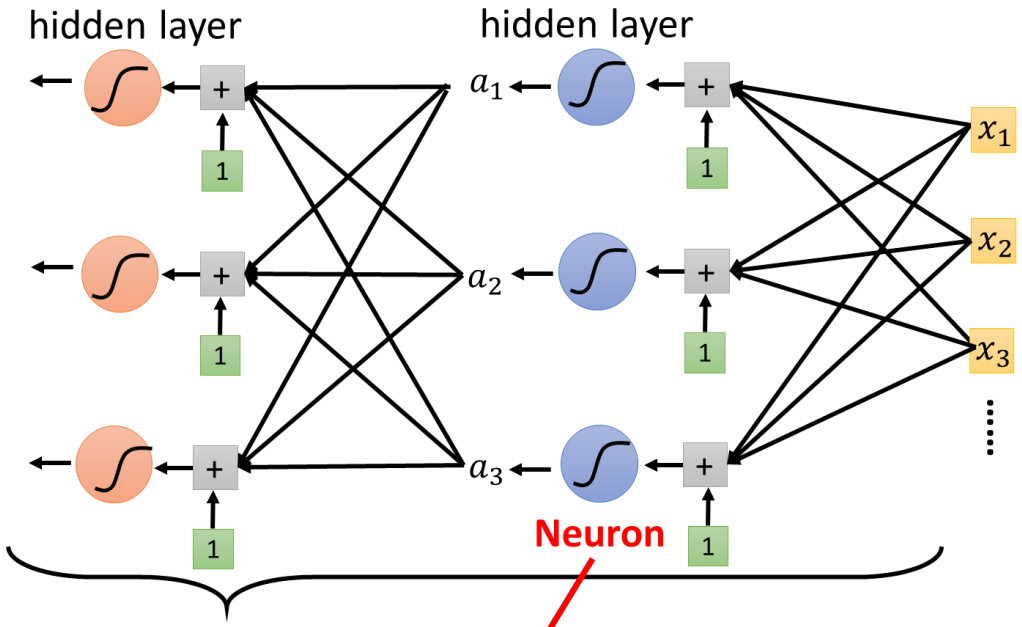
- 理论推导：
  - 边缘概率分布$p(x)=\int p(x\mid z)p(z)\mathrm{d}z$仍为正态分布$\mathcal{N}(x\mid \mu,\ WW^T+\sigma^2 I)$ ← **z和ε是独立的随机变量**
  - $(WW^T+\sigma^2 I)^{-1}=\sigma^{-2}I-\sigma^{-2}W\underbrace{(W^TW+\sigma^2 I)^{-1}}_{M}W^T$
  - 后验概率为$p(z\mid x)=\mathcal{N}(z\mid M^{-1}W^T(x-\mu),\sigma^2M^{-1})$
- 极大似然：（证明略）
  - $\ln p(X\mid \mu,W,\sigma^2)=\sum_{n=1}^{N}\ln p(x_n\mid W,\mu,\sigma^2)$
    - 令似然函数关于μ的导数等于零得到$\mu=\bar{x}$，代回得$\ln p(X\mid W,\mu,\sigma^2)=-\frac{N}{2}\{D\ln(2\pi)+\ln|C|+\mathrm{Tr}(C^{-1}S)\}+\mathrm{const}$
    - $W_{ML}=U_M(L_M-\sigma^2 I)^{1/2}R$（$U_M$—特征向量的组合， L_M—对应的特征值， R—正交矩阵）
    - $\sigma^2_{ML}=\frac{1}{n-k}\sum_{i=k+1}^{D}\lambda_i$（与丢弃的维度相关联的平均方差）

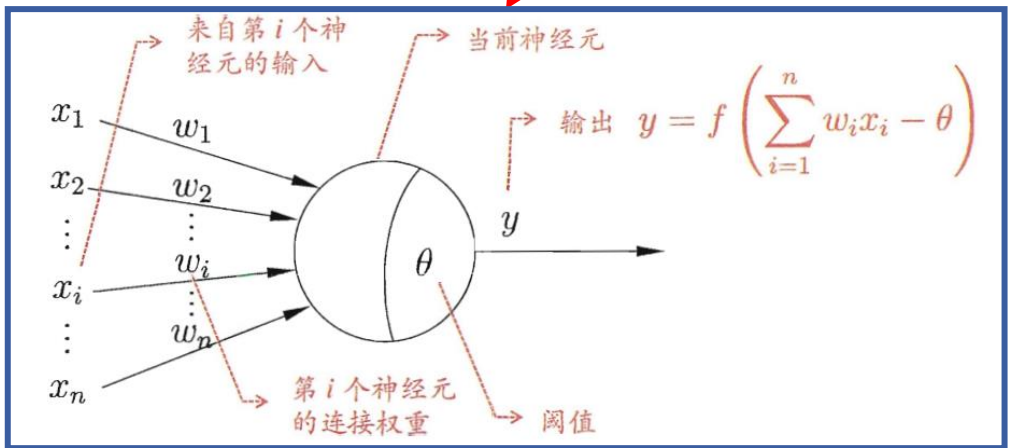
深度学习



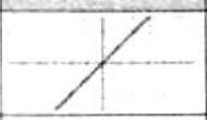
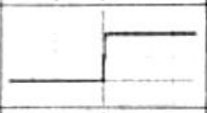
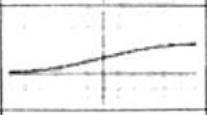
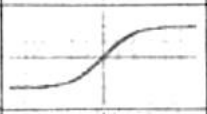
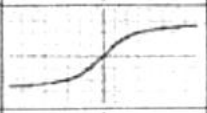
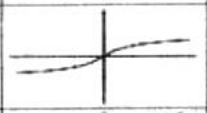
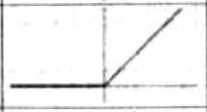
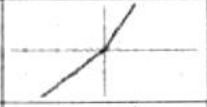
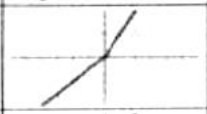
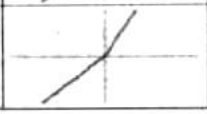
多层感知机MLP

- **想法：引入非线性因素→非线性映射**
  - 有向图，由多个的节点层所组成，每一层都全连接到下一层，除了输入节点，每个节点都是一个带有**非线性激活函数**的神经元。
- **模型：**
  - $y_j = \sigma(\sum_i w_i b_i - \theta_i)$相互(嵌套)代入
  - 训练集$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}, x \in \mathbb{R}^d, y \in \mathbb{R}^l$
  - 隐藏层第h个神经元接收到的输入为$\alpha_h = \sum_{i=1}^{d} v_{ih} x_i$
  - 输出层第j个神经元接收到的输入为$\alpha_h = \sum_{h=1}^{q} w_{hj} b_h$
  - 神经元激活函数为$\sigma(\cdot)$
- **前向推理：**
  - 从输入层开始，沿着网络的层级顺序将数据传递到输出层，从而计算模型的预测值
  - $y_k = g\left\{\sum_{j=1}^{m} w_{kj}^{(s)} \cdots \left[a\left(\sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)}\right)\right] \cdots + b_k^{(s)}\right\}, k = 1,2,\cdots,l$
  - 不涉及权重和偏差的更新

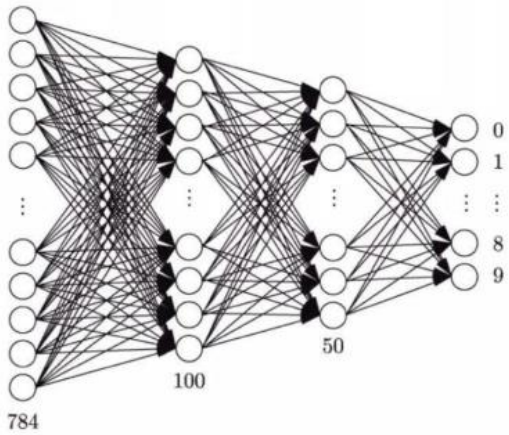




常见激活函数

Name	Plot	Equation	Derivative (with respect to x)	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geqslant 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
Softsign [7][8]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$	$(-1, 1)$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geqslant 0 \end{cases}$	$[0, \infty)$
Leaky rectified linear unit (Leaky ReLU)[10]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geqslant 0 \end{cases}$	$(-\infty, \infty)$
Parameteric rectified linear unit (PReLU)[11]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geqslant 0 \end{cases}$	$(-\infty, \infty)$
Randomized leaky rectified linear unit (RReLU)[12]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$ [1]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geqslant 0 \end{cases}$	$(-\infty, \infty)$

多层感知机MLP



- 反向传播：基于链式法则计算损失对每个权重和偏差的梯度
- 推导过程：

① 考虑一个s层神经网络，其中第t层的神经元定义为：$h_j^{(t)} = a\left(z_j^{(t)}\right), z_j^{(t)} = \sum_{i=1}^{n} w_{ji}^{(t)} h_i^{(t-1)} + b_j^{(t)}, j = 1,2,\cdots,m$

② 损失函数对第t层的权重和偏置的梯度分别为$\frac{\partial L}{\partial w_{ji}^{(t)}}$和$\frac{\partial L}{\partial b_j^{(t)}}$。根据**链式求导规则**，可以展开为：$\frac{\partial L}{\partial w_{ji}^{(t)}} = \frac{\partial L}{\partial z_j^{(t)}} \frac{\partial z_j^{(t)}}{\partial w_{ji}^{(t)}}$，$\frac{\partial L}{\partial b_j^{(t)}} = \frac{\partial L}{\partial z_j^{(t)}} \frac{\partial z_j^{(t)}}{\partial b_j^{(t)}}$

③ 令$\delta_j^{(t)} = \frac{\partial L}{\partial z_j^{(t)}}, j = 1,2,\cdots,m$，则上式可写成：$\frac{\partial L}{\partial w_{ji}^{(t)}} = \delta_j^{(t)} h_i^{(t-1)}$，$\frac{\partial L}{\partial b_j^{(t)}} = \delta_j^{(t)}$

④ 而对于第t层的$\delta_j^{(t)}$，可展开为$\delta_j^{(t)} = \frac{\partial L}{\partial z_j^{(t)}} = \sum_{k=1}^{l} \frac{\partial L}{\partial z_k^{(t+1)}} \frac{\partial z_k^{(t+1)}}{\partial z_j^{(t)}}, j = 1,2,\cdots,m$

⑤ 求解得到$\delta_j^{(t)} = \frac{da}{dz_j^{(t)}} \sum_{k=1}^{l} w_{kj}^{(t+1)} \delta_k^{(t+1)}$。

⑥ 其中，$\frac{da}{dz_j^{(t)}}$为第t层激活函数关于$z_j^{(t)}$的导数。

⑦ 也就是说可以根据$t+1$层的$\delta_k^{(t+1)}$计算$\delta_j^{(t)}$。

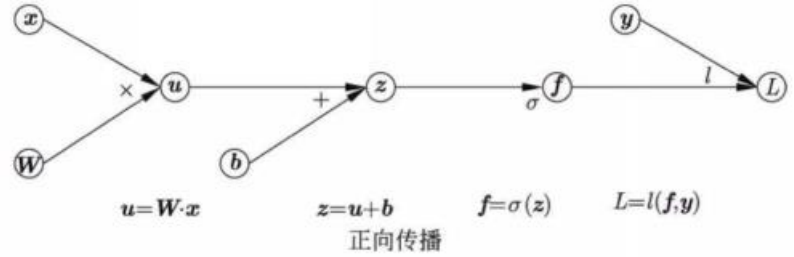
ALGORITHM Backpropagation(反向传播算法)

1: **input** $f(x;\theta)$ ←神经网络，θ ←参数向量，(x', y') ←样本，η ←学习率；
2: **for** $t = 1,2,\ldots,s$ **do** ***# 正向传播***
3: $h^{(0)} = x'; h^{(t)} = a\left(W^{(t)} h^{(t-1)} + b^{(t)}\right)$（$a(\cdot)$为激活函数）
4: **for** $t = s, s-1, \ldots, 1$ **do** ***# 反向传播***
5: $\delta^{(s)} = h^{(s)} - y'; \nabla_{W^{(t)}} L = \delta^{(t)} \cdot h^{(t-1)^T}, \nabla_{b^{(t)}} L = \delta^{(t)}$
6: $W_{\text{new}}^{(t)} \leftarrow W^{(t)} - \eta \nabla_{W^{(t)}} L, b_{\text{new}}^{(t)} \leftarrow b^{(t)} - \eta \nabla_{b^{(t)}} L$
7: **if** $t > 1$ **do**
8: $\delta^{(t-1)} = \frac{\partial a}{\partial z_j^{(t-1)}} \odot \left(W^{(t)^T} \cdot \delta^{(t)}\right)$
9: **return** θ

- 问题：梯度消失和梯度爆炸——由于层数过多、前面几层的梯度有时会接近于0（消失）或者接近无穷（爆炸），无法使程序运行下去

多层感知机MLP

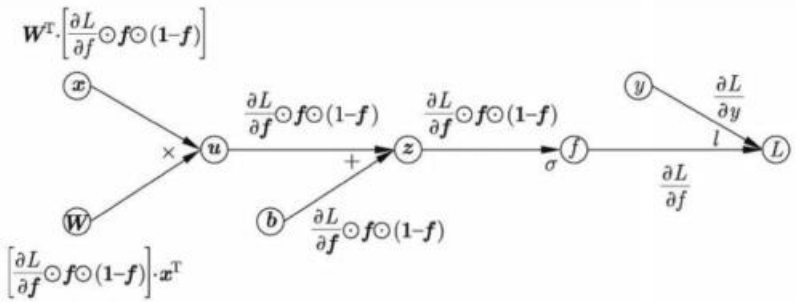
- 损失函数：（一般是极大似然）
  - 回归：平方损失 $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2 + N \log \sigma + \frac{N}{2} \log 2\pi \right] \longrightarrow \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} (y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta}))^2$
  - 二分类：交叉熵损失 $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ -\sum_{i=1}^{N} [y_i \log f(\boldsymbol{x}; \boldsymbol{\theta}) + (1 - y_i) \log(1 - f(\boldsymbol{x}; \boldsymbol{\theta}))] \right\}$
  - 多分类：交叉熵损失 $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ -\sum_{i=1}^{N} \left[\sum_{k=1}^{l} y_{ik} \log f(\boldsymbol{x}; \boldsymbol{\theta}) \right] \right\}$



正向传播

  - 上述损失函数对应最后一层误差均为：$\boldsymbol{\delta}_k^{(s)} = \boldsymbol{h}_k^{(s)} - \boldsymbol{y}_k, \boldsymbol{k} = \mathbf{1}, \mathbf{2}, \cdots, \boldsymbol{l}$

- 优化算法：
  - **GD** $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$
  - **SGD** $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{1}{n} \sum_{j=1}^{n} \frac{\partial L_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$

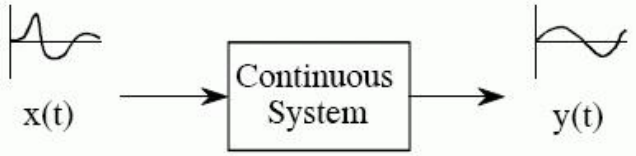


反向传播

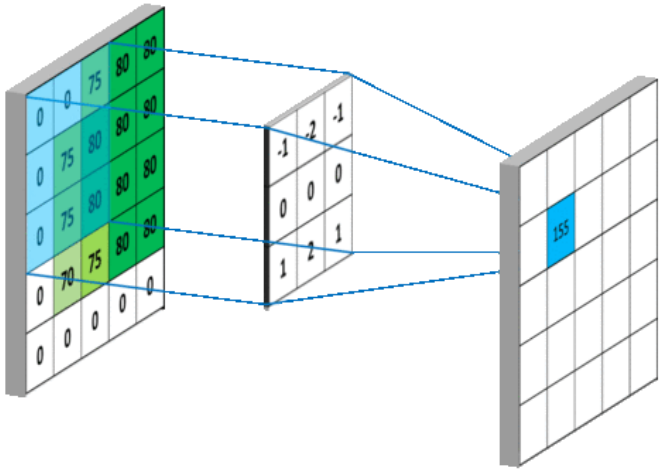
- 实现方法：计算图、批量归一化
- 正则化：早停法、 暂退法（dropout）

卷积神经网络CNN

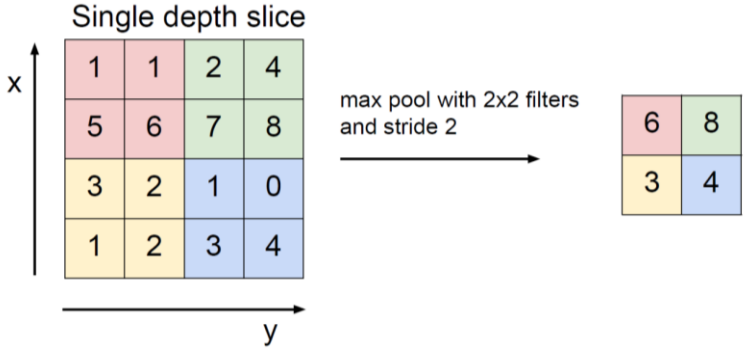
- 卷积实质上是对信号进行滤波
  - 一维连续形式：$(f*g)(t)=\int_{-\infty}^{\infty} f(a)*g(t-a)da$
  - 表示为一个函数对另一个函数的调整



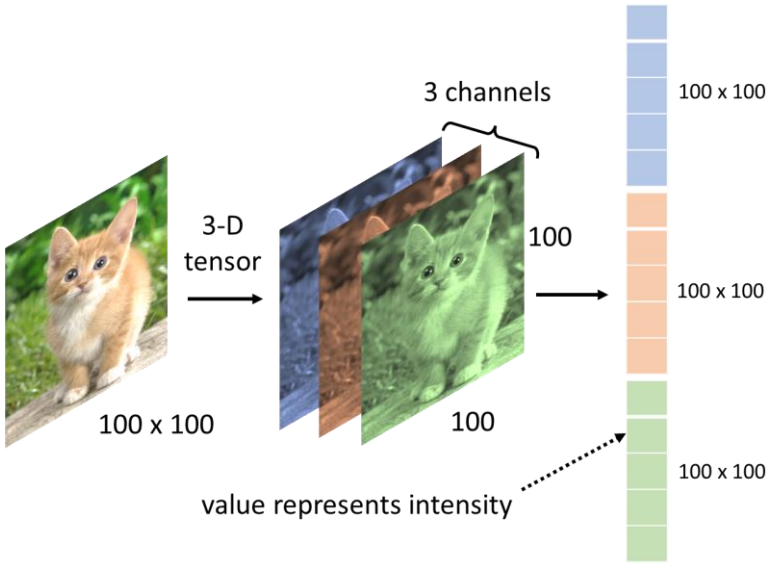
- 离散二维卷积：（实质上是数学上的互相关）
  - 线性滤波，用每个像素的邻域的线性组合来代替这个像素。
  - 平移不变性，共享参数
  - 卷积核滑动计算内积：$y_{kl}=\sum_{m=1}^{M}\sum_{n=1}^{N} w_{m,n}x_{k+m-1,l+n-1}$
  - 填充和步幅（维持信号大小不变）



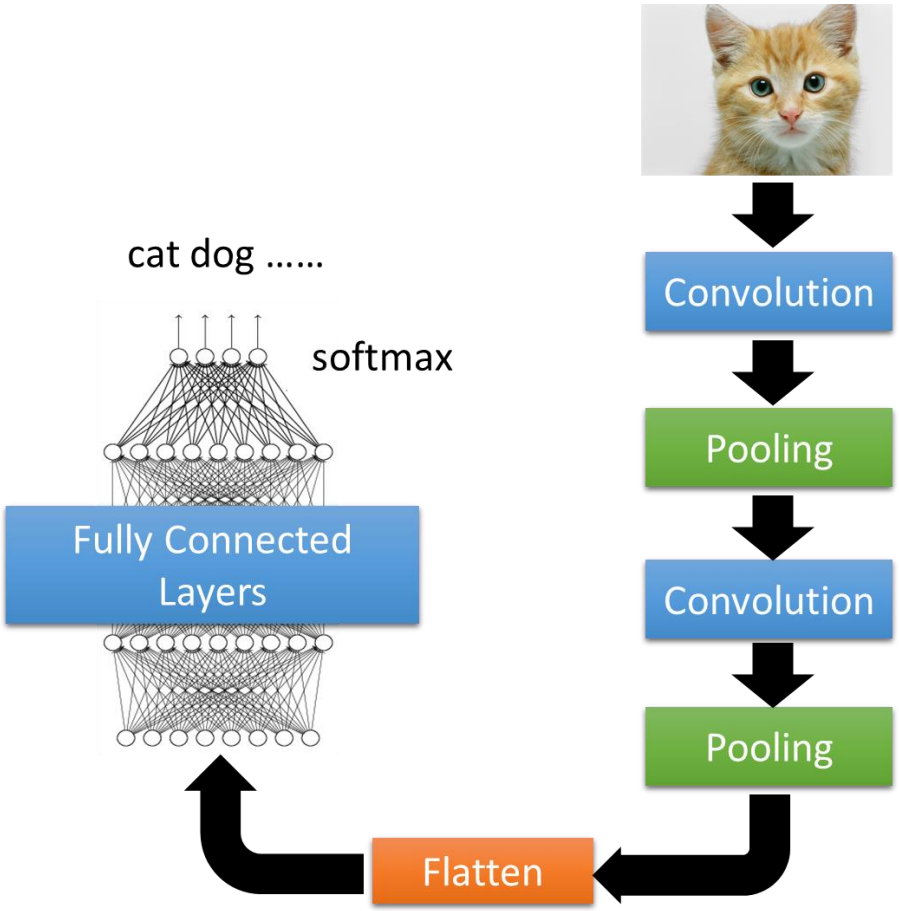
- 池化（pooling）：也叫下采样
  - 压缩信号表示，减少计算
  - 对每一个子矩阵取最大或平均



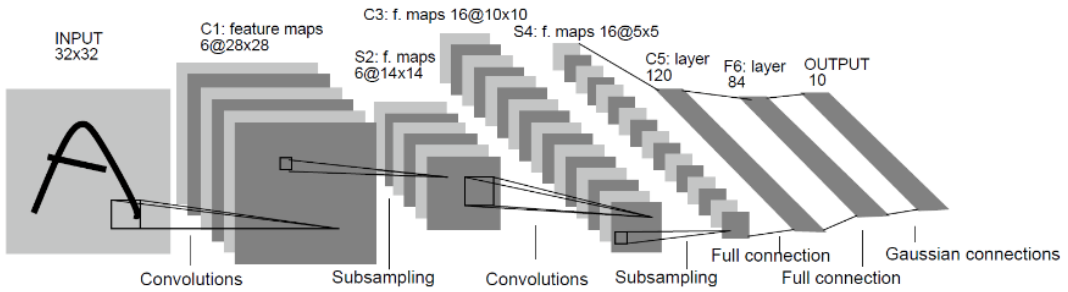
- 全连接层



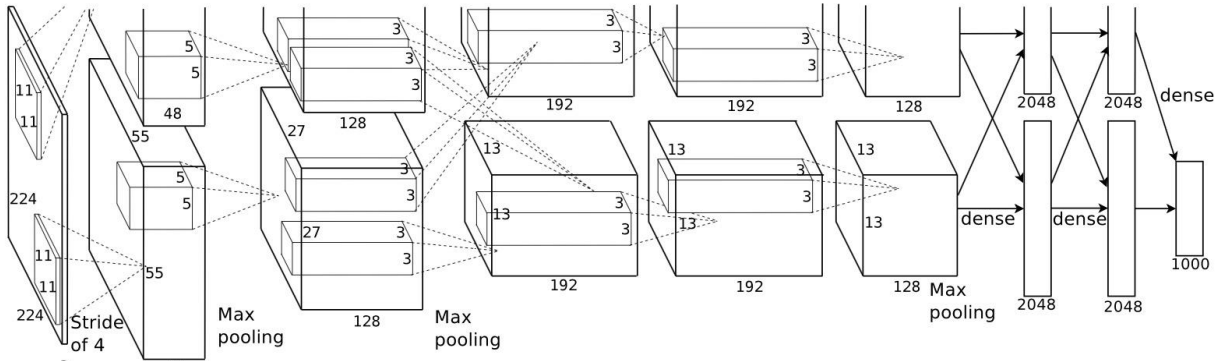
卷积神经网络CNN



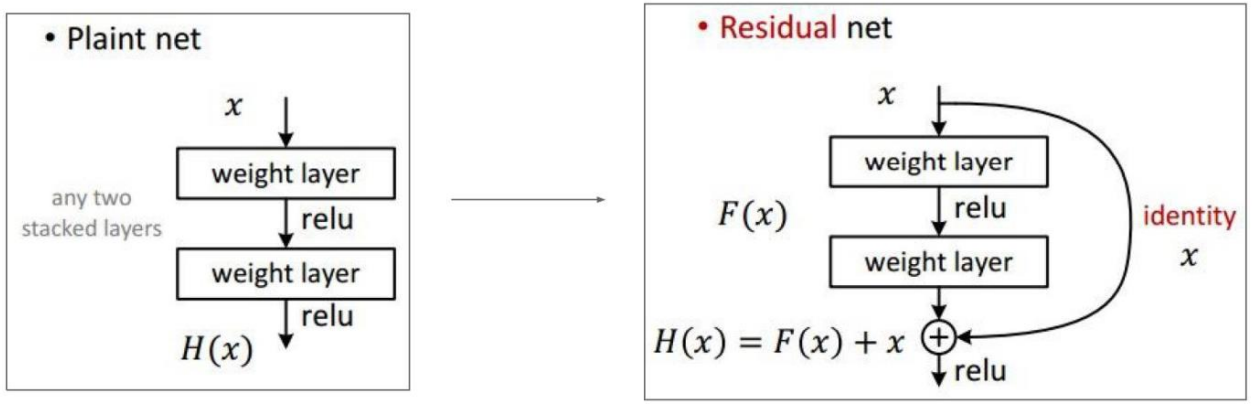
LeNet-5



AlexNet

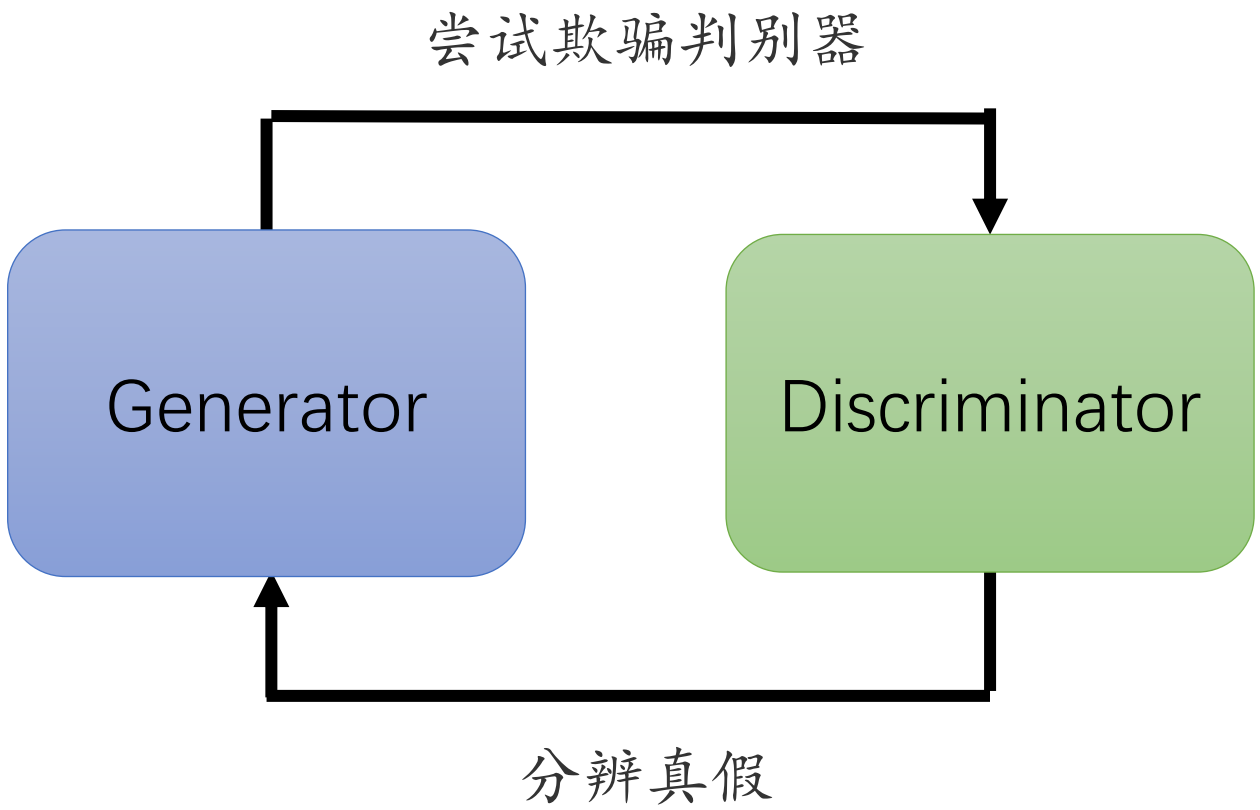


ResNet



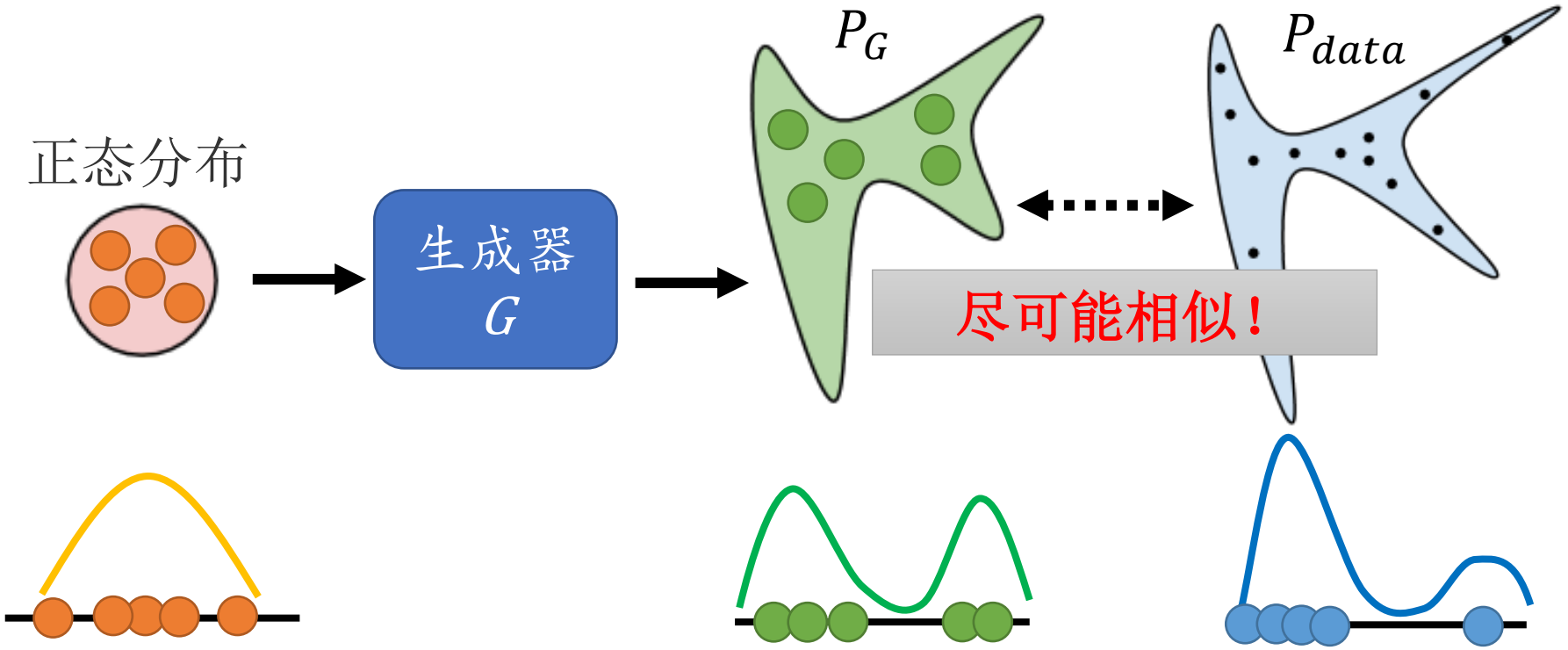
生成对抗网络GAN

- **生成器Generator：** 期望能够产生真实的图片进而骗过判别器。
- **判别器Discriminator：** 期望能够准确的区分真假图片



GAN的数学原理

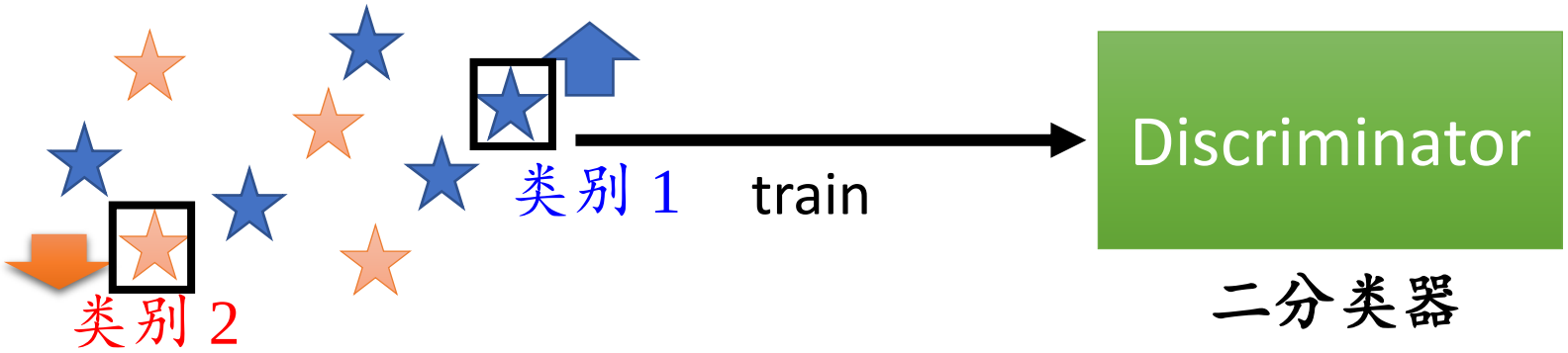
对于Generator：　　目标函数$G^* = arg\min_G Div(P_G, P_{data})$



GAN的数学原理

对于Discriminator：　　目标函数$D^* = arg\max_D V(D,G)$

★：从P_{data}抽样的数据　　★：从P_G抽样的数据



目标函数：

$$V(G,D) = E_{x\sim P_{data}}[logD(x)] + E_{z\sim P_G}\left[log\left(1-D\left(G(z)\right)\right)\right]$$

$D(x)=1$表示x来自真实分布；　$D(x)=0$表示来自生成器拟合分布

GAN的数学原理

目标函数： $\min_G \max_D V(G,D) = E_{x\sim P_{data}}[\log D(x)] + E_{z\sim P_G}\left[\log\left(1-D\left(G(z)\right)\right)\right]$

Step 1: 固定G，更新D $\quad D^* = arg \max_D V(D,G)$

$$V(G,D) = \int_x \left[P_{data}(x)\, log\, D(x) + P_G(x) log\left(1-D(x)\right)\right] dx = \int_x \left[a\, log\, D(x) + b log\left(1-D(x)\right)\right] dx$$

对被积函数求导，取得

$$D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

Step 2: 固定D，更新G $\quad G^* = arg \min_G \max_D V(D,G)$

将D代回$V(G,D)$，

$$V(G,D) = \int_x \left[P_{data}(x)\, log \frac{\frac{1}{2}P_{data}(x)}{\frac{1}{2}\left(P_{data}(x)+P_G(x)\right)} + P_G(x) \frac{\frac{1}{2}P_G(x)}{\frac{1}{2}\left(P_{data}(x)+P_G(x)\right)}\right] dx$$

GAN的数学原理

目标函数： $\min_G \max_D V(G,D) = E_{x\sim P_{data}}[\log D(x)] + E_{z\sim P_G}\left[\log\left(1-D\left(G(z)\right)\right)\right]$

Step 2: 固定D，更新G $\quad G^* = arg \min_G \max_D V(D,G)$

$$V(G,D) = -2\log 2 + KL\left(P_{data}\left\|\frac{P_{data}(x)+P_G(x)}{2}\right.\right) + KL\left(P_G\left\|\frac{P_{data}(x)+P_G(x)}{2}\right.\right)$$

$$= -2\log 2 + 2\mathrm{JS}\left(P_{data}\|P_G\right)$$

当$V(G,D)$最小时，当且仅当$P_{data}(x) = P_G$时，此时$D(x) = 1/2$!

重复这样的步骤！

一个概率分布相对于另一个概率分布的不确定性和信息损失

相对熵（KL散度）两个概率分布间差异的**非对称性度量**；而JS散度是**对称的**

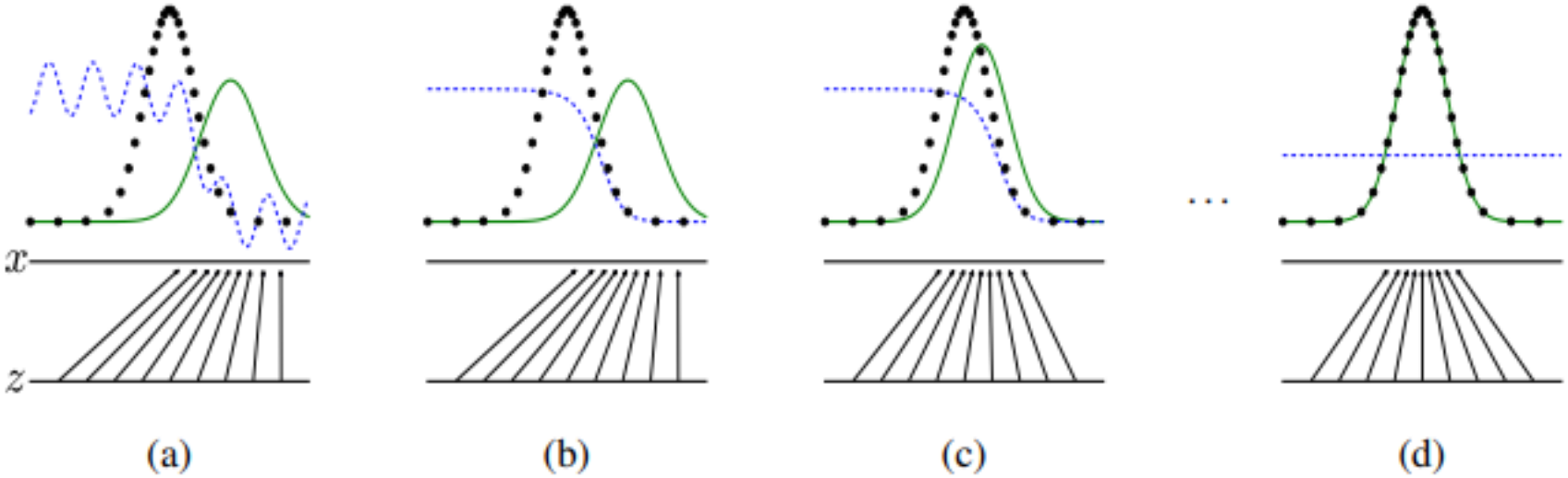
两个概率分布之间的平均信息损失和相似性

KL散度： $KL(p\|q) = \int_x p(x)\log\frac{p(x)}{q(x)}dx$

JS散度： $JS(p\|q) = \frac{1}{2}KL\left(p\left\|\frac{p+q}{2}\right.\right) + \frac{1}{2}KL\left(q\left\|\frac{p+q}{2}\right.\right)$

GAN的数学原理

*D*和*G*训练过程：

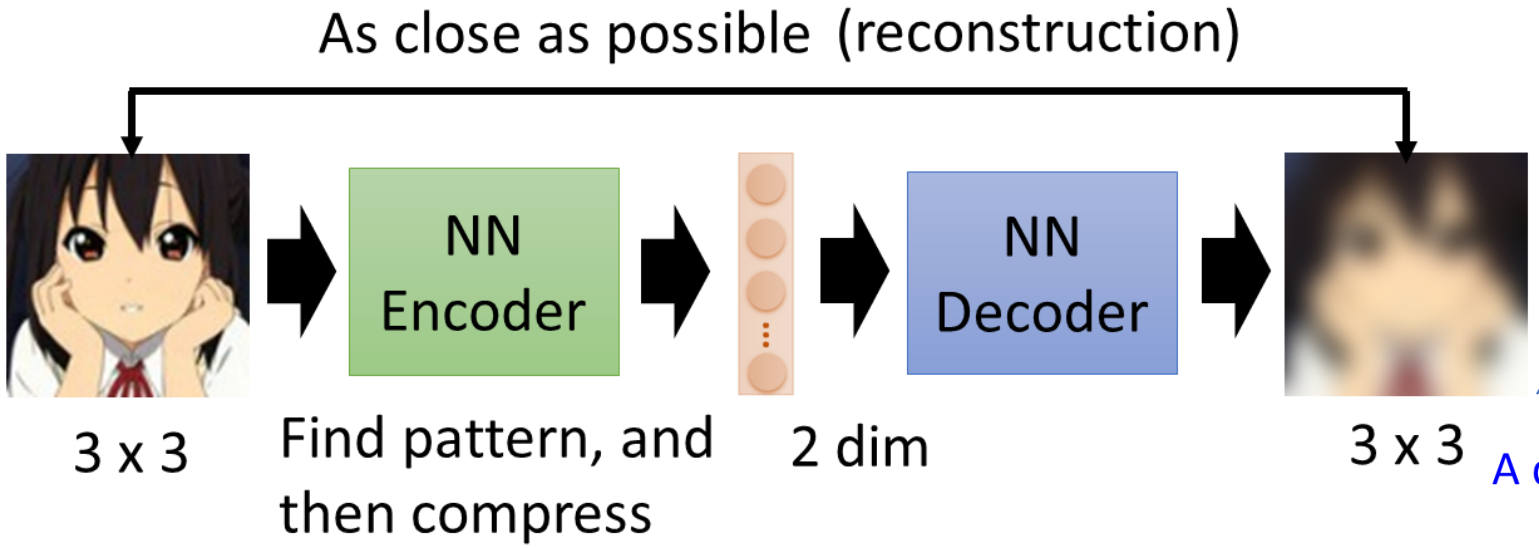


*GAN*的缺陷：

传统的GAN并不能生成高分辨率的图像——数据量过大，训练困难！

控制生成图像的特定特征的能力极其有限——输入是一个随机的正态分布！

自编码器Auto-Encoder



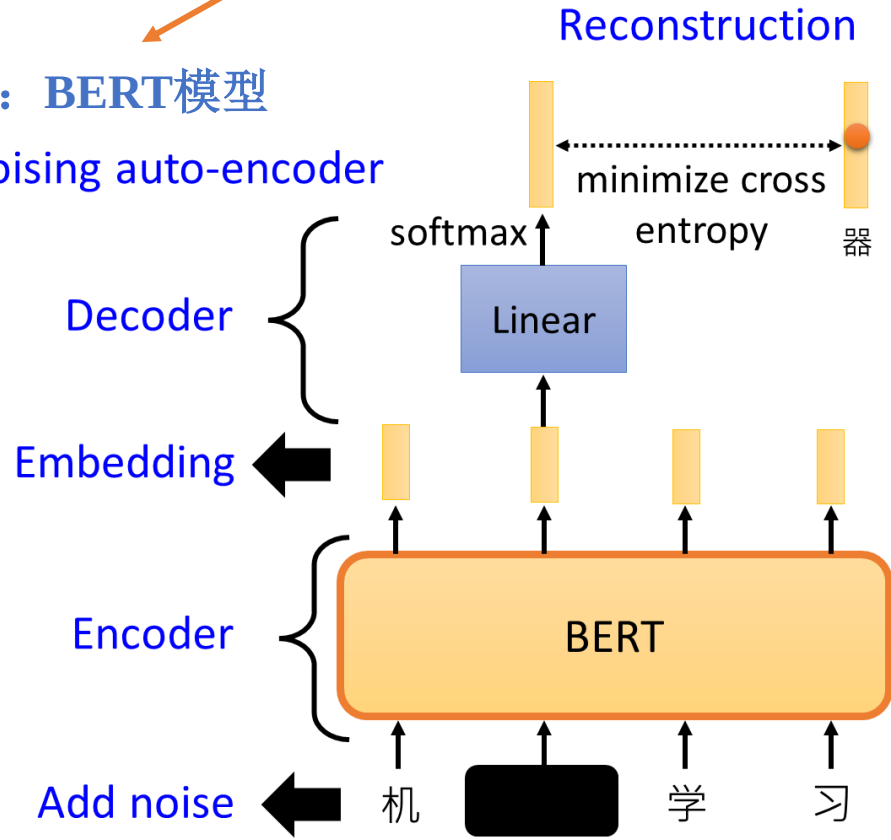
- 想法：PCA可以表示为$\phi \cdot X = Z$，将原始信息压缩表示为低维数据
  - 如何衡量压缩的质量？
  - 通过重建图像与原图的差异来比较性能！
- 损失函数：$\phi, \psi = \mathrm{argmin}_{\phi,\psi} L(X, (\psi \circ \phi)X)$
  - ϕ-压缩矩阵，ψ-重建矩阵
- 应用：特征解耦、图像去噪、压缩…

GPT模型-Autoregressive

Decoder-only

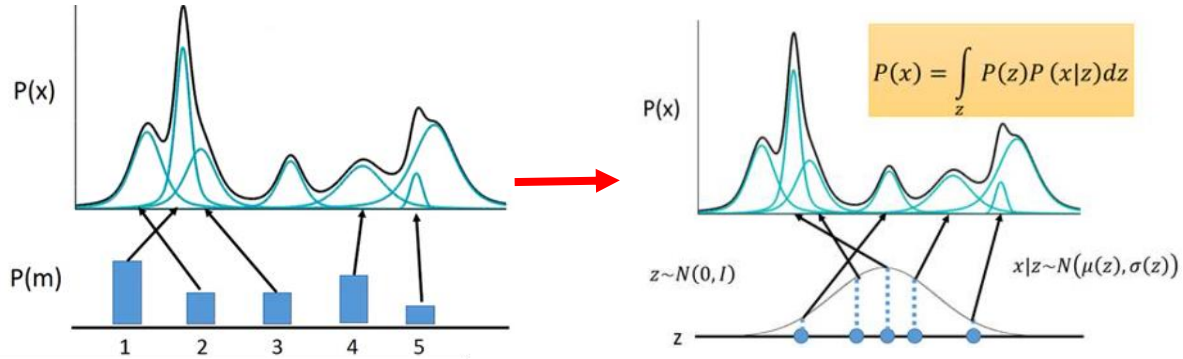
例子：BERT模型

A de-noising auto-encoder

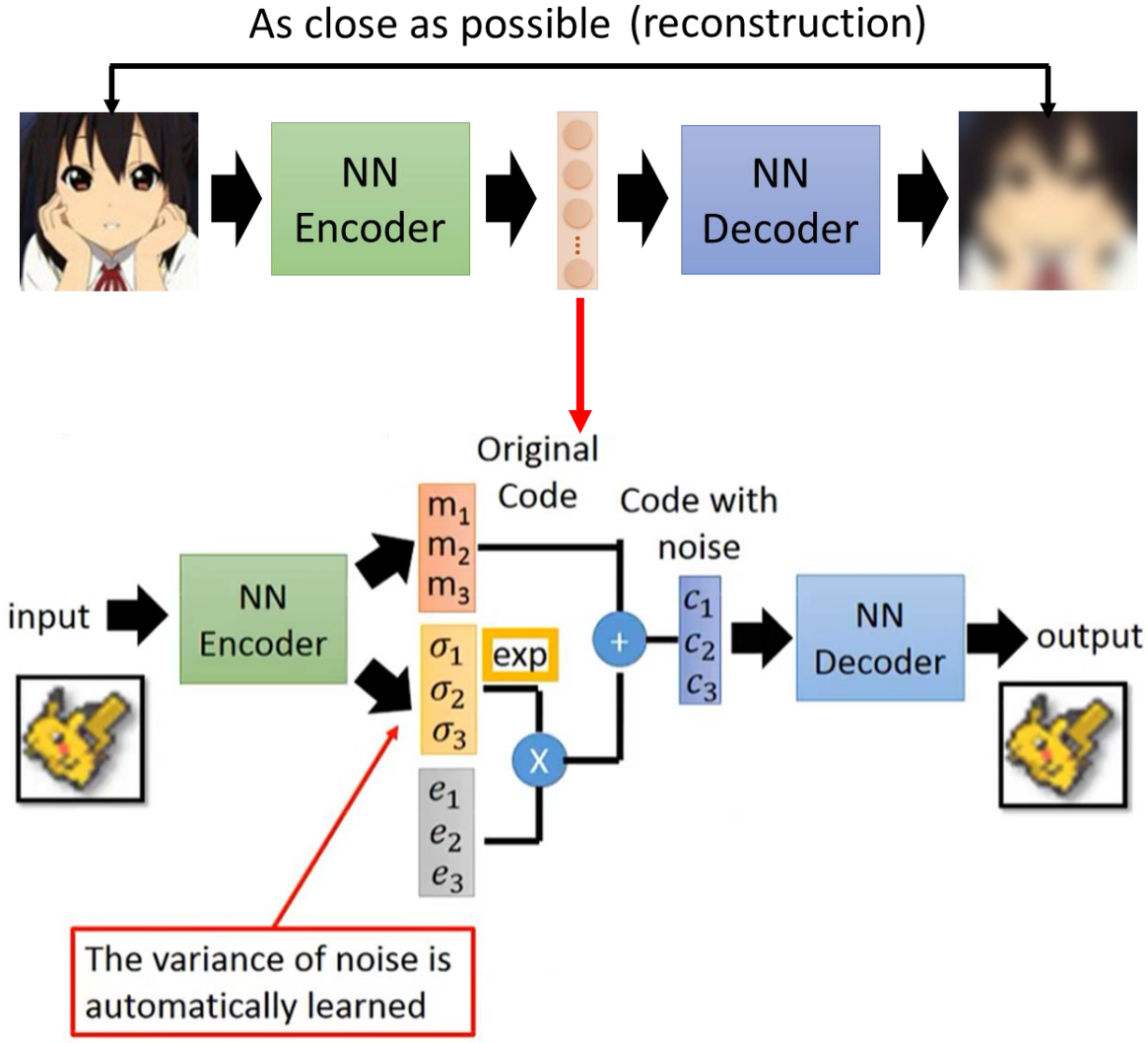


变分自编码器VAE

- 想法：学习离散数据点→数据分布
- 原理：GMM潜在变量z离散→连续



- 目标：$\max P(x) = \int_z P(z)P(x|z)dz$
  - 潜在变量$p(z) = \mathcal{N}(z \mid 0, I), z \in \mathbb{R}^k$
  - 解码Decoder过程$p(x|z) \sim \mathcal{N}\left(\mu(z), \sigma(z)\right)$（神经网络近似）
  - 编码Encoder过程$q_\theta(z|x)$（神经网络近似）



拓展：变分下界优化

- 变分下界：

$$\log P(x) = \int_z q(z|x)\log P(x)dz \quad (q(z|x)\text{可以是任何分布})$$

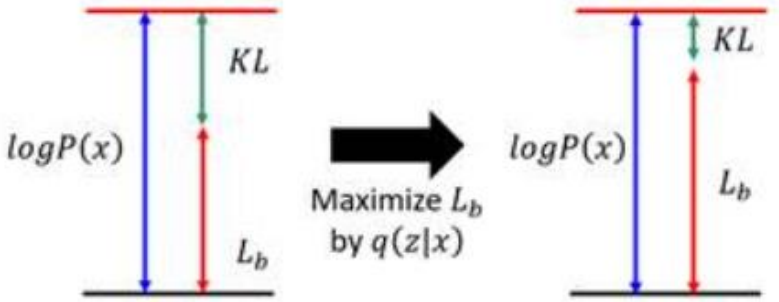
$$= \int_z q(z|x)\log\left(\frac{P(z,x)}{P(z|x)}\right)dz$$

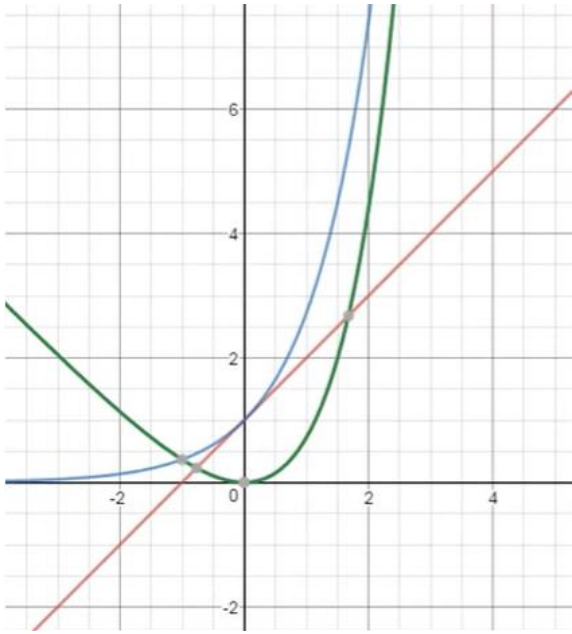
$$= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\frac{q(z|x)}{P(z|x)}\right)dz$$

$$= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\right)dz + \int_z q(z|x)\log\left(\frac{q(z|x)}{P(z|x)}\right)dz$$

$$= \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\right)dz + KL(q(z|x)||P(z|x))$$

$$\geq \int_z q(z|x)log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz = L_b$$





- 训练：提高下界极大化L_b

$$L_b = \int_z q(z|x)\log\left(\frac{P(z,x)}{q(z|x)}\right)dz$$

$$= \int_z q(z|x)\log\left(\frac{P(x|z)P(z)}{q(z|x)}\right)dz$$

$$= \int_z q(z|x)\log\left(\frac{P(z)}{q(z|x)}\right)dz + \int_z q(z|x)\log P(x|z)dz$$

$$= -KL(q(z|x)||P(z)) + \int_z q(z|x)\log P(x|z)dz$$

Loss function

$$\min KL(q(z|x)||P(z)) = \sum_{i=1}^{J}(\exp(\sigma_i) - (1+\sigma_i) + (m_i)^2) \quad \text{多样性损失}$$

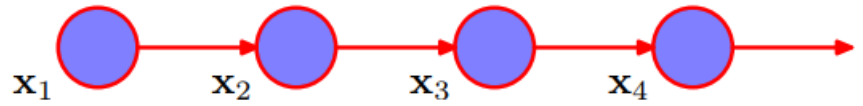
$$\max \int_z q(z|x)\log P(x|z)dz = \mathbb{E}_{q(z|x)}[logP(x|z)] \quad \text{重建损失}$$

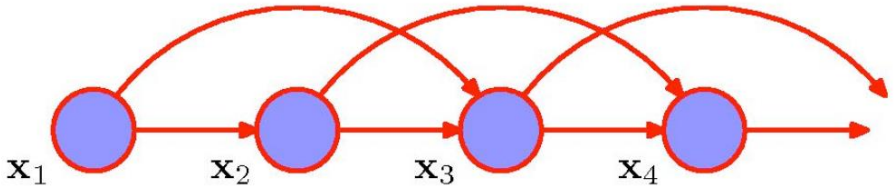
隐马尔科夫HMM

- 处理序列数据（语音、文本、生物信息...）
- 马尔科夫链——当前状态只与上一状态相关
  - 联合分布： $p(\mathbf{x}_1,\ldots,\mathbf{x}_N)=p(\mathbf{x}_1)\prod_{n=2}^{N}p(\mathbf{x}_n|\mathbf{x}_{n-1})$
  - 某一位置的条件概率：

$$p(\mathbf{x}_n|\mathbf{x}_1,\ldots,\mathbf{x}_{n-1})=p(\mathbf{x}_n|\mathbf{x}_{n-1})$$

  - 二维情形：当前状态与上两个状态相关
  - 维数越高，复杂度越高（遗忘机制）

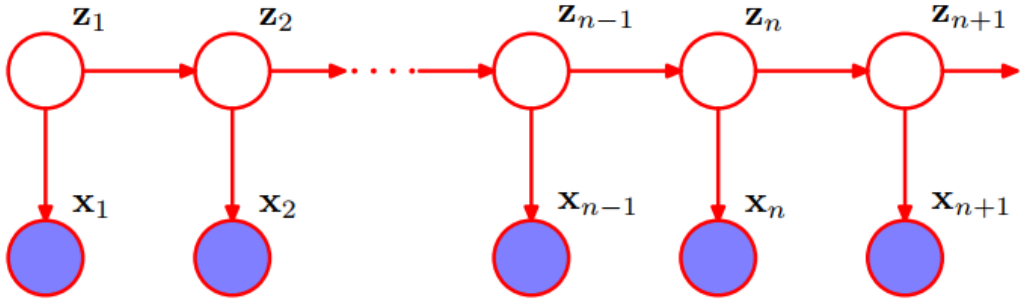




- 隐马尔科夫模型
  - 想法：引入额外的潜在变量来使得更丰富的一类模型能够从简单的成分中构建
  - 联合概率分布：

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_N,\mathbf{z}_1,\ldots,\mathbf{z}_N)=\mathrm{p}(\mathbf{z}_1)\underbrace{\left[\prod_{n=2}^{N}p(\mathbf{z}_n\mid\mathbf{z}_{n-1})\right]}_{\text{状态转移}}\overbrace{\prod_{n=1}^{N}\mathrm{p}(\mathbf{x}_n\mid\mathbf{z}_n)}^{\text{观测生成模型}}$$

  - 潜在变量离散→隐马尔科夫
  - 潜在变量连续→线性动态系统



隐马尔科夫HMM

- 隐马尔科夫模型
  - 潜在变量是离散的服从多项式分布的变量$\mathbf{z}_n$，描述观测变量$\mathbf{x}_n$
  - $\mathbf{z}_n$采用独热编码（1 of K）
  - 转移概率： $p(z_{nk}=1|z_{n-1,j}=1)=A_{jk}, \sum_k A_{jk}=1$
  - 条件概率：权重共享$\boldsymbol{A}$

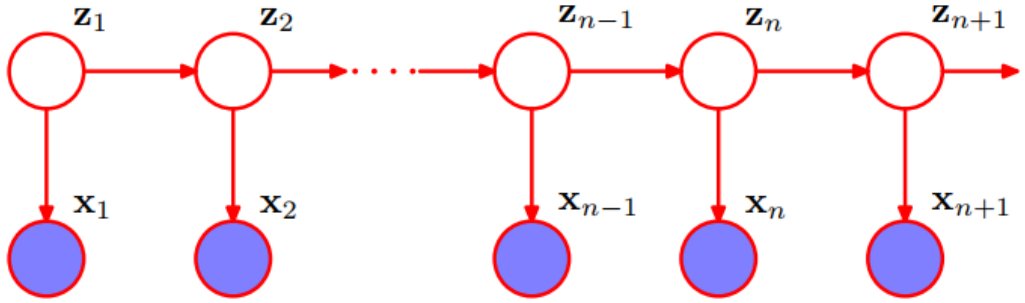
$$p(\mathbf{z}_n|\mathbf{z}_{n-1},A)=\prod_{k=1}^{K}\prod_{j=1}^{K}A_{jk}^{z_{n-1,j},z_{nk}}, \quad p(\mathbf{z}_1|\pi)=\prod_{k=1}^{K}\pi_k^{z_{1k}}$$

  - 发射概率：$\boldsymbol{\phi}$是控制概率分布的参数集合

$$p(\mathbf{x}_n \mid \mathbf{z}_n,\boldsymbol{\phi})=\prod_{k=1}^{K}p(\mathbf{x}_n \mid \boldsymbol{\phi}_k)^{z_{nk}}$$

  - 从而观测变量和潜在变量上的联合概率分布为

$$p(\boldsymbol{X},\boldsymbol{Z} \mid \boldsymbol{\theta})=p(\mathbf{z}_1 \mid \boldsymbol{\pi})\left[\prod_{n=2}^{N}p(\mathbf{z}_n \mid \mathbf{z}_{n-1},\boldsymbol{A})\right]\prod_{m=1}^{N}p(\mathbf{x}_m \mid \mathbf{z}_m,\boldsymbol{\phi})$$



- For example, for a continuous **x**, we have

$$p(\mathbf{x}_n|\mathbf{z}_n,\phi)=\prod_{k=1}^{K}\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)^{z_{nk}}.$$

- For the discrete, multinomial observed variable **x**, using 1-of-K encoding, the conditional distribution takes form:

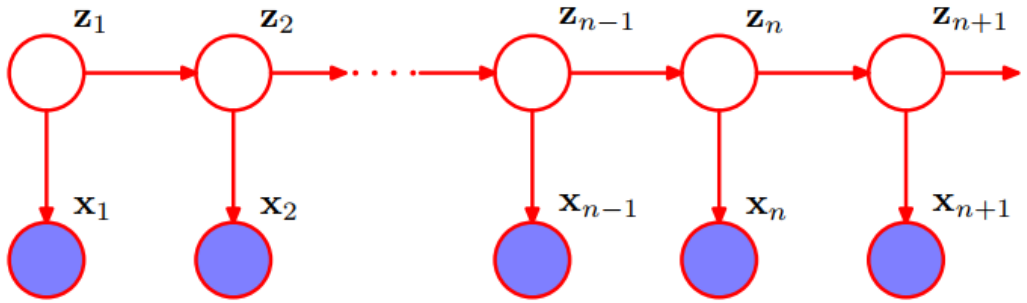
$$p(\mathbf{x}_n|\mathbf{z}_n,\phi)=\prod_{i=1}^{D}\prod_{k=1}^{K}\mu_{ik}^{x_{ni}z_{nk}}.$$

隐马尔科夫HMM

- 隐马尔科夫模型三类问题
  - **评估（概率计算）**：即给定模型$\theta = (A, \phi)$和观测序列X，计算在模型θ下观测序列出现的最大概率$P(X|\theta)$；
    - 前向算法α，观测在此之前的影响
    - 后向算法β，观测在此时候的特征

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) = \frac{p(\mathbf{x}_1, .., \mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})} = \frac{p(\mathbf{x}_1, .., \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}.$$

  - **学习**：即给定观测序列X，估计模型的参数θ, 使得在该参数下观测序列出现的概率最大，即$P(X|\theta)$最大；
    - E步：计算隐变量
    - M步：更新模型参数（极大似然思想）
  - **解码（预测）**：给定模型$\theta = (A, \phi)$和观测序列X，计算最有可能产生这个观测序列的隐含序列Z，即使得概率$P(Z|X, \theta)$最大的隐含序列Z
    - 维特比Viterbi算法



参考讲解：https://www.bilibili.com/video/BV14R4y1N7iH/?spm_id_from=333.337.search-card.all.click&vd_source=6c2f75f31d7201c3507c01a98d35677c

拓展： EM for HMM

EM algorithm

- We cannot perform direct maximization (no closed form solution):

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta).$$

- EM algorithm: we will derive efficient algorithm for maximizing the likelihood function in HMMs (and later for linear state-space models).

- E-step: Compute the posterior distribution over latent variables:

$$p(\mathbf{Z}|\mathbf{X}, \theta^{old}).$$

- M-step: Maximize the expected complete data log-likelihood:

$$Q(\theta, \theta^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \log p(\mathbf{X}, \mathbf{Z}|\theta).$$

- If we knew the true state path, then ML parameter estimation would be trivial.

- We will first look at the E-step: Computing the true posterior distribution over the state paths.

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) = \frac{p(\mathbf{x}_1, .., \mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})}$$

$$= \frac{p(\mathbf{x}_1, .., \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}.$$

- The forward recursion:

$$\begin{aligned}
\alpha(\mathbf{z}_n) &= p(\mathbf{x}_1, .., \mathbf{x}_n, \mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_1, .., \mathbf{x}_{n-1}, \mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, .., \mathbf{x}_{n-1}, \mathbf{z}_{n-1}, \mathbf{z}_n) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, .., \mathbf{x}_{n-1}, \mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1}) \\
&= p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})
\end{aligned}$$

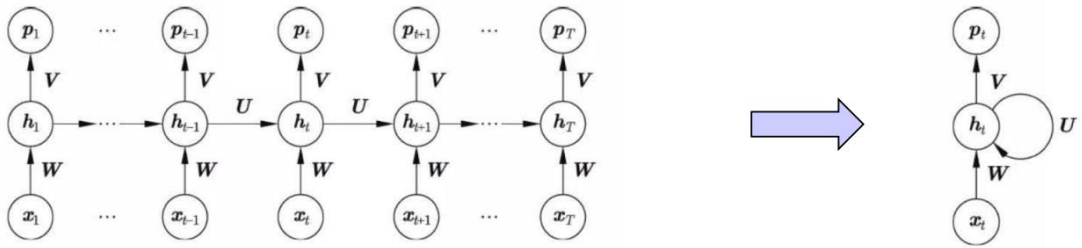
Computational cost scales like $O(K^2)$.

- There is also a simple recursion for $\beta(\mathbf{z}_n)$:

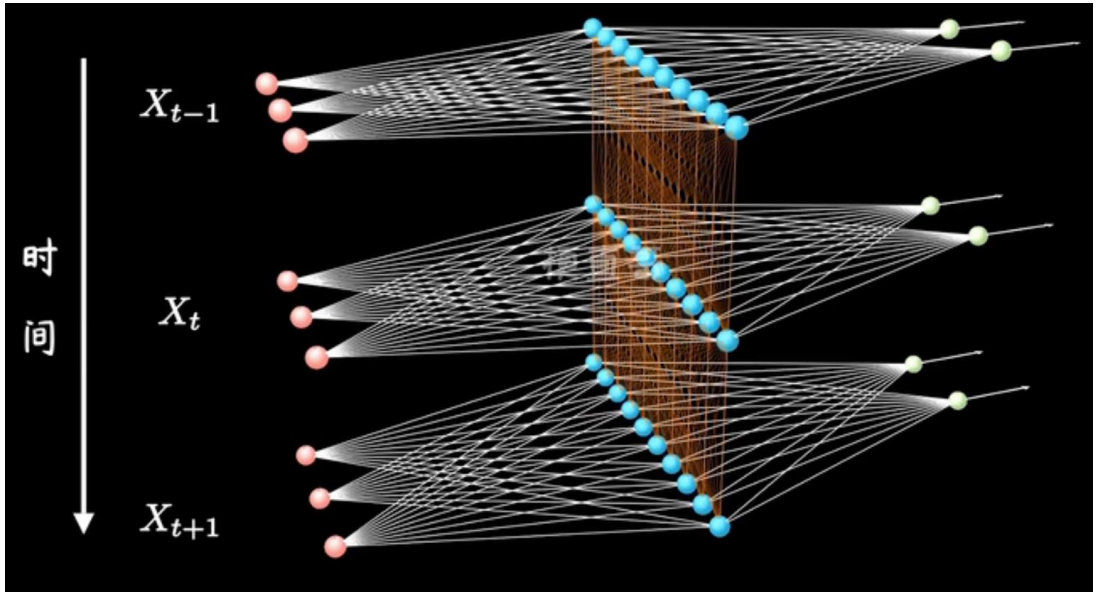
$$\begin{aligned}
\beta(\mathbf{z}_n) &= p(\mathbf{x}_{n+1}, .., \mathbf{x}_N|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N, \mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_{n+1}, \mathbf{z}_n)p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+2}, ..., \mathbf{x}_N|\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n)
\end{aligned}$$

循环神经网络RNN

- 简单循环神经网络
  - 想法
    - 每个时序位置建立相同的网络结构（减少参数）
    - 引入隐状态、并将相邻的神经网络连接（引入条件独立）
  - 当前位置输出依赖
    - 当前输入（短距离依赖）
    - 前一位置隐状态（实质是以前所有输入信息， 长距离依赖）
  - 是一种自回归模型（**auto-regressive model**）
- 模型：
  - 计算：$p_t = \boldsymbol{V} \cdot \sigma(\boldsymbol{U} \cdot h_{t-1} + \boldsymbol{W} \cdot x_t + b) + c$
  - 激活函数$\sigma(\cdot) =$ **Tanh**
- 训练：误差反向传播 + 梯度下降



$$\boldsymbol{h}_t = \tanh\left(\boldsymbol{U} \cdot \boldsymbol{h}_{t-1} + \boldsymbol{W} \cdot \boldsymbol{x}_t + \boldsymbol{b}\right) \qquad \boldsymbol{p}_t = \mathrm{softmax}(\boldsymbol{V} \cdot \boldsymbol{h}_t + \boldsymbol{c})$$

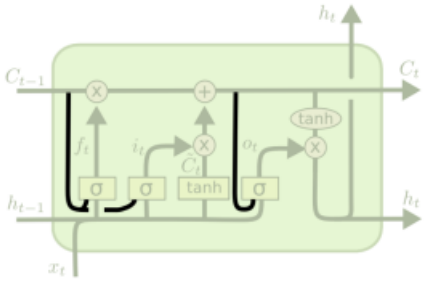


$$P\left(\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_T | \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T\right) = \prod_{t=1}^{T} P\left(\boldsymbol{y}_t | \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t\right)$$

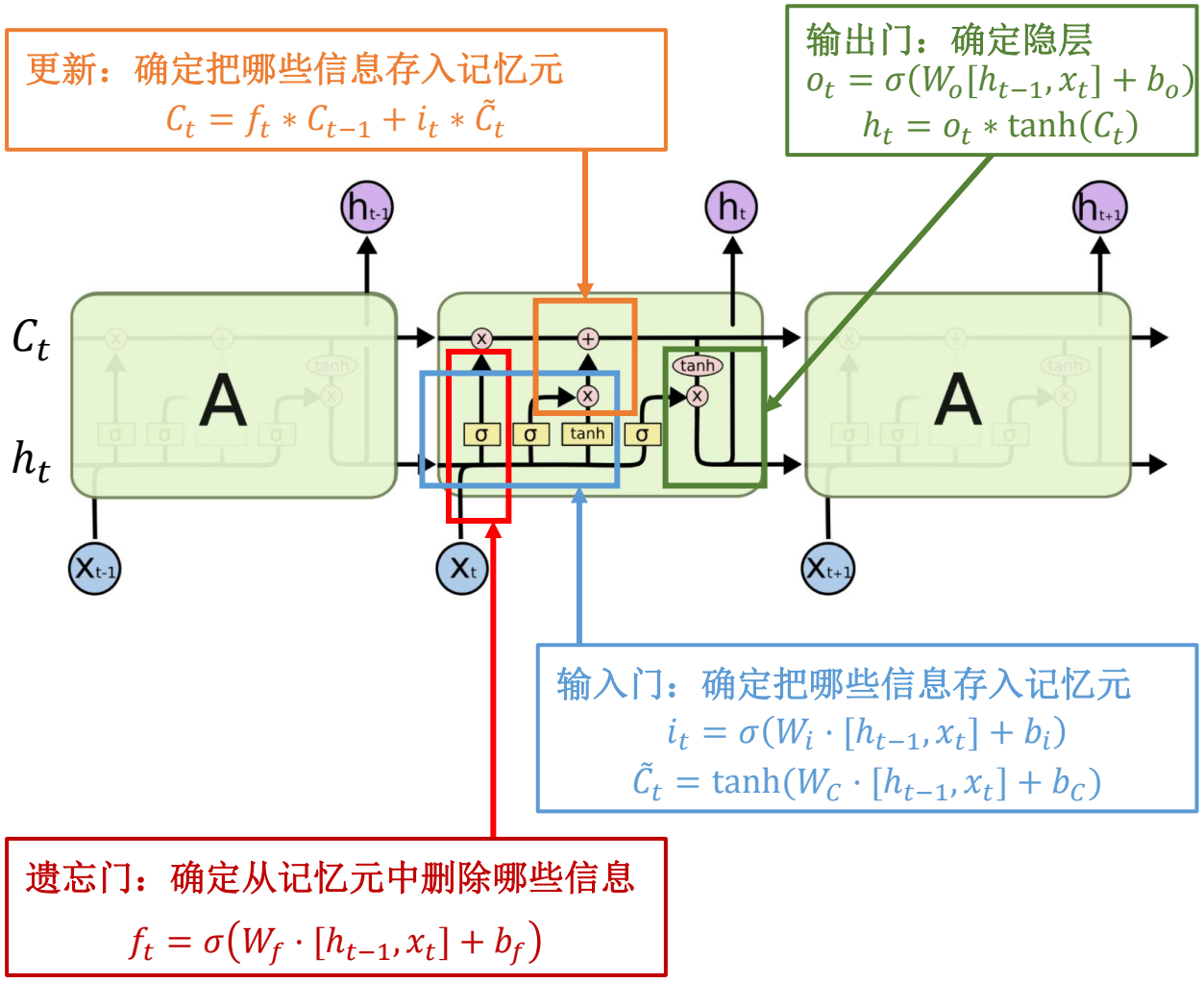
$$L = \sum_{t=1}^{T} L_t = -\sum_{t=1}^{T} \log P\left(\boldsymbol{y}_t | \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t\right) \qquad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \frac{\partial L}{\partial \boldsymbol{\theta}}$$

长短期记忆神经网络LSTM

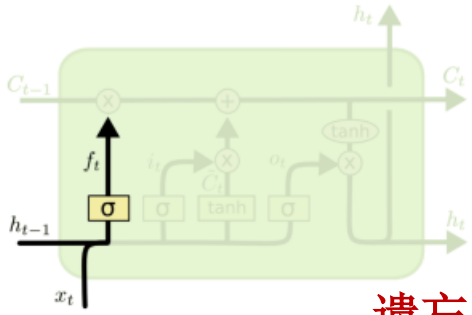
- **RNN的问题：**
  - 梯度爆炸
  - 长距离依赖
- **LSTM：显式地避免长距离依赖**
  - 引入记忆单元和门控
  - C_t—长期记忆链
  - h_t—短期记忆链（和RNN类似）
- 变形：门状态额外考虑记忆元信息



$$f_t = \sigma\left(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [C_t, h_{t-1}, x_t] + b_o\right)$$

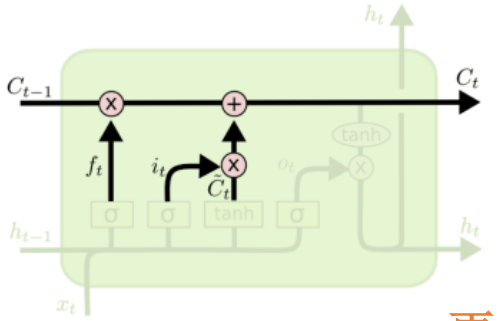


长短期记忆神经网络LSTM



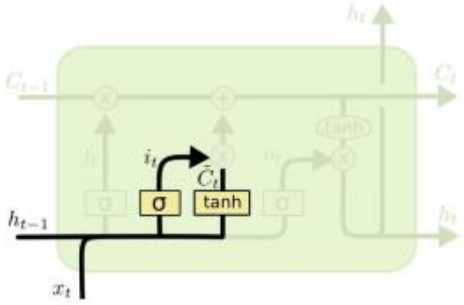
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

遗忘门



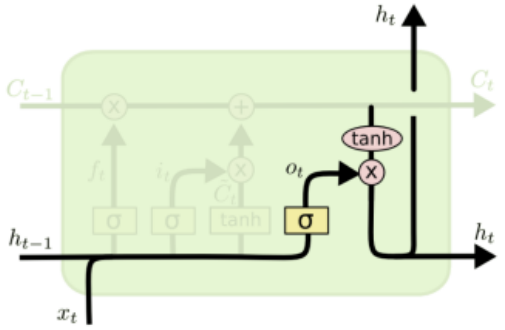
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

更新



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

输入门



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

输出门

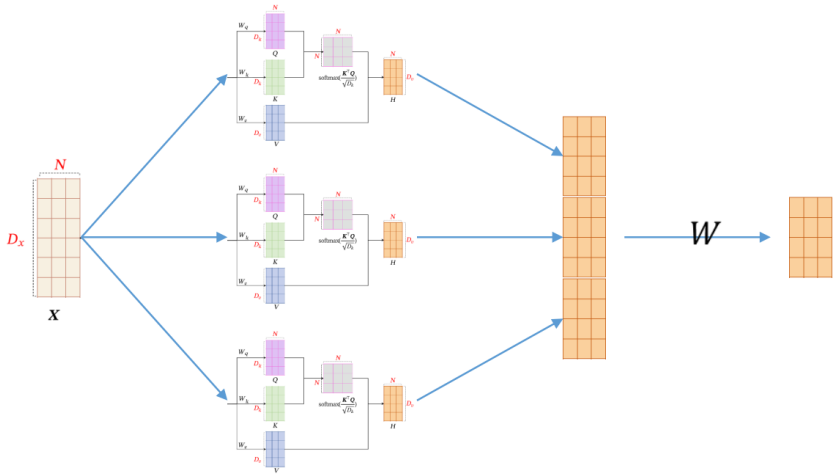
自注意力Self-Attention

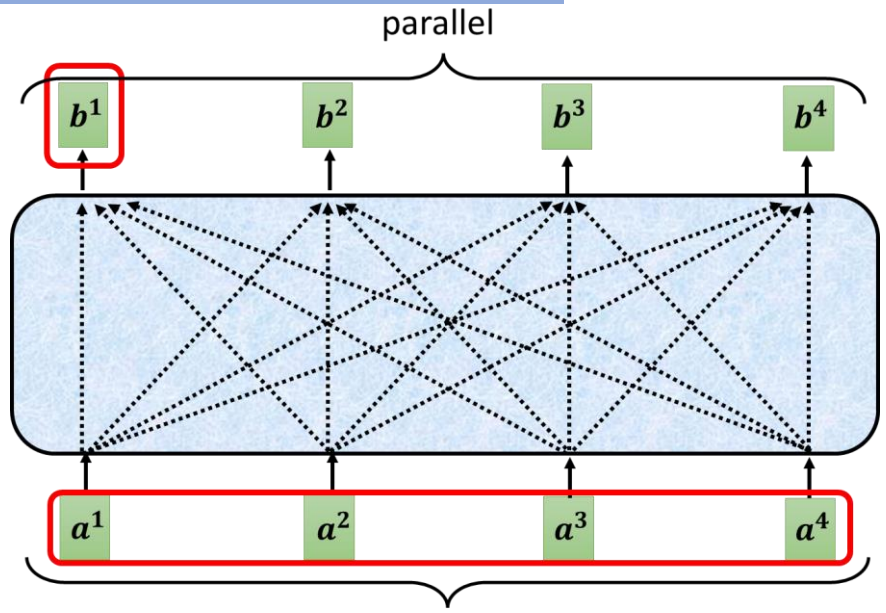
- 自注意力模型
  - 输入序列为$X = [x_1, \dots, x_N] \in \mathbb{R}^{D_x \times N}$
  - 生成三个向量序列

$$\left.\begin{aligned} Q &= W_q X \in \mathbb{R}^{D_k \times N}, \\ K &= W_k X \in \mathbb{R}^{D_k \times N}, \\ V &= W_v X \in \mathbb{R}^{D_e \times N}. \end{aligned}\right\} \quad \text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

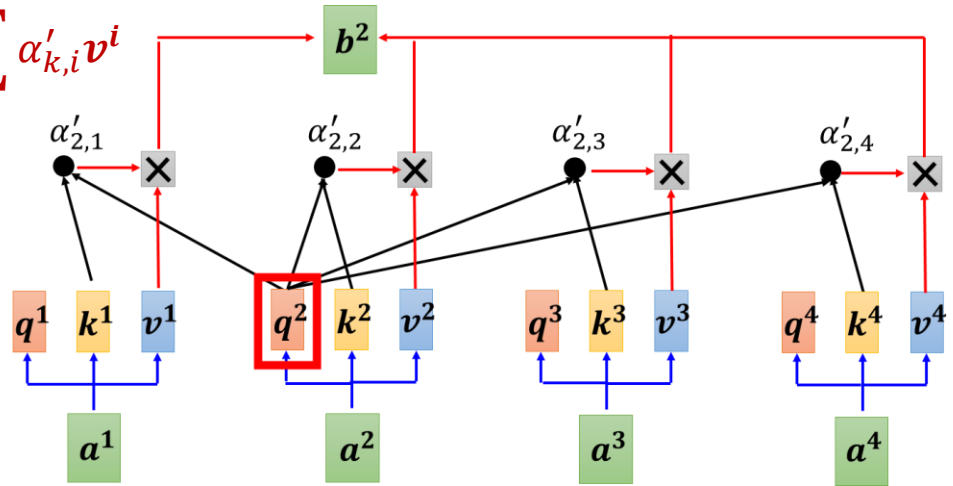
  - 计算$\boldsymbol{b_n = att((K, V), q_n)}$—使用Query和Key计算相似性！

- 多头自注意力（Multi-Head）





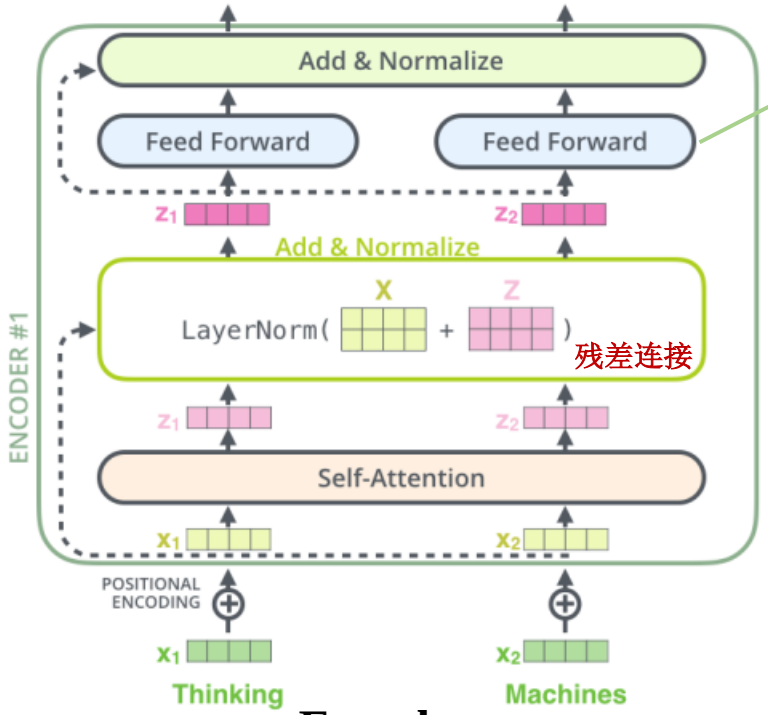
$$\boldsymbol{b^k = \sum_i \alpha'_{k,i} v^i}$$



Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

Transformer

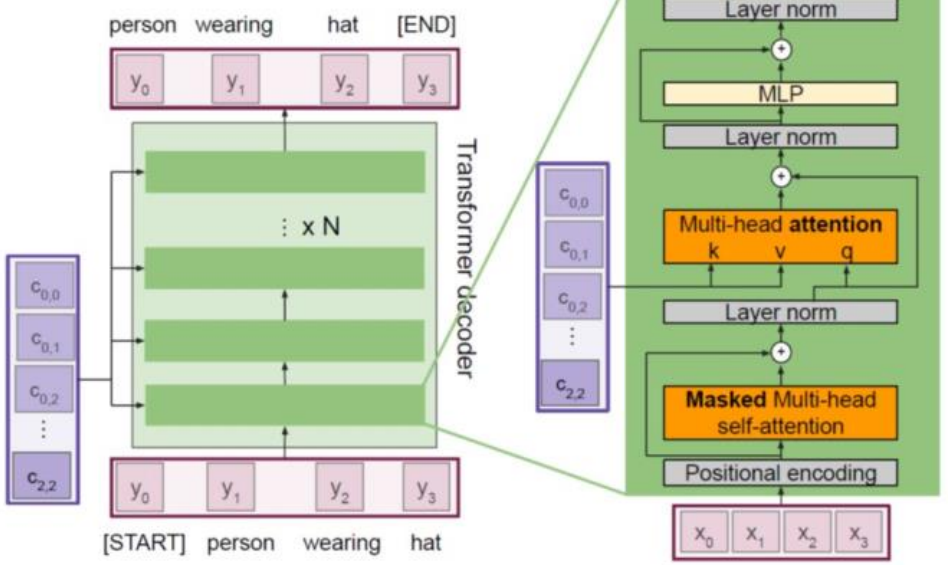
- 一个序列变为另一个序列
- **Encoder-Decoder架构**



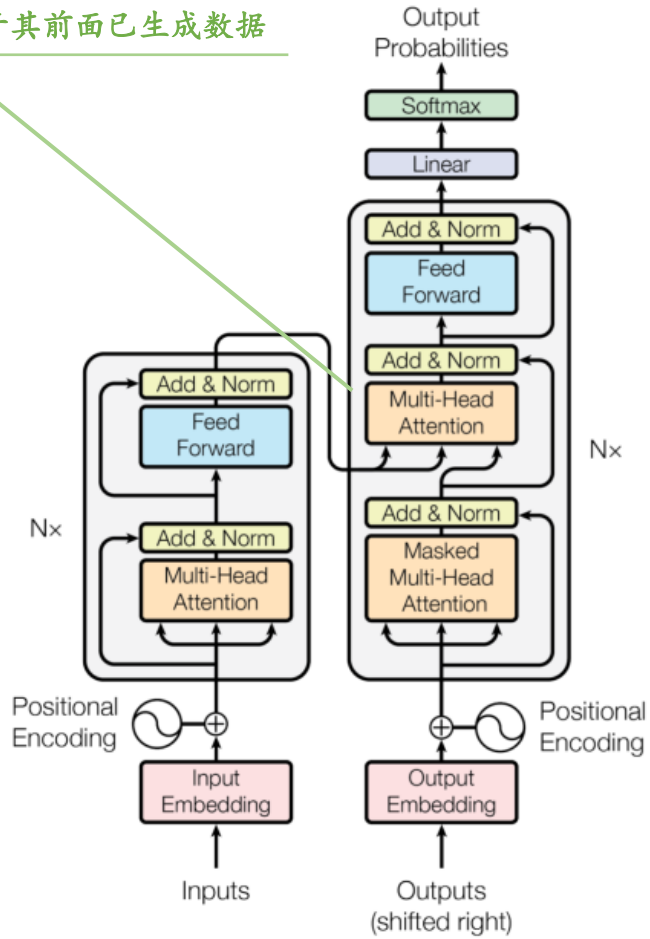
Encoder

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

输出当前位置结果时只依赖于其前面已生成数据



Decoder



https://blog.research.google/2017/08/transformer-novel-neural-network.html

Transformer

谢谢观看！
预祝大家取得满意的成绩

PRML-2023秋期末复习讲座

By 张智雄