Machine Learning: from Theory to Practice

Author: TSINGHUA Computer Science & Finance Shiying Zhang 2021011056

Chapter0 Definition and Terms

- **人工智能 / Artificial Intelligence**

  计算机科学领域，专注于开发通常需要人类智能的任务，例如策略博弈、自然语言处理等

- **机器学习 / Machine Learning**

  是人工智能的一个子集，基于训练的算法（training algorithm），即对数据集进行学习和模式识别，从而进行预测或决策，而非对问题进行直接的人为编程

  - **监督学习 / Supervised Learning**：用于训练的数据集是标签的（labeled data）
    - **回归问题 / Regression**：Trees (Random Forests, GBM, DT / Decision Trees 决策树), Linear / GLMS, Ensemble, Neural Networks
    - **分类问题 / Classification**：SVM (支撑向量机 / Support Vector Machines), Discriminant Analysis, GNB (高斯朴素贝叶斯分类 / Gaussian Naïve Bayes) , Nearest Neighbor
  - **无监督学习 / Unsupervised Learning**：用于训练的数据集没有预先标签（unlabeled data）

    聚类 / Clustering：K-Means (K-均值聚类)，Gaussian Mixture Model (高斯混合模型), Hierarchical, Neural Networks
  - **半监督学习 / Semi-Supervised Learning**：同时使用标签的和未标签的数据集进行训练（algorithm 用 labeled data 识别模式，用 unlabeled data 进一步学习和理解模式）

    self-training, generative models, S3VMs, Graph based algorithms, Multiview algorithms

    ---

  - **强化学习 / Reinforcement Learning**：根据环境反馈进行决策，通过trial and error学习

    Markov Decision Process (MDP), Monte-Carlo Simulation (蒙特卡洛模拟)
  - **深度学习 / Deep Learning**：使用Artificial Neural Network的特定领域

    全连接神经网络，卷积神经网络，循环神经网络

  除此之外，机器学习还可以分为基于模型学习和基于实例学习：

  - **基于模型学习 / Model-based Learning**：对数据进行整体归纳提炼（从实例中构建模型，用模型进行预测）
  - **基于实例学习 / Instance-based Learning**：未对数据进行整体归纳提炼（学习示例，然后用相似度度量新的实例和已经学习的实例的关系，从而泛化新实例）

  机器学习过程的两个阶段是训练（train / fit）和推断（inference / predict）

Chapter1 启发式搜索 Search

寻找起点`s`到终点`g`的最短路（最短耗散值）

引入：宽度优先搜索（BFS）+ Dijkstra算法、深度优先搜索（DFS）

1.1 A* 算法

$$\text{评价函数 } \mathbf{f(n)=g(n)+\text{启发函数 } h(n)}$$

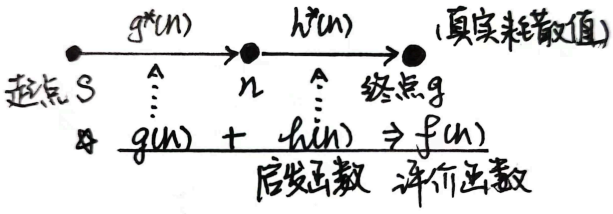
$g^*(n)$：从起点$\mathbf{s}$到$\mathbf{n}$的最短路径的耗散值

$h^*(n)$：从$\mathbf{n}$到终点$\mathbf{g}$的最短路径的耗散值

$f^*(n)$：从起点$\mathbf{s}$经过啊$\mathbf{n}$到终点$\mathbf{g}$的最短路径的耗散值

$f(n),\ g(n),\ h(n)$分别是$f^*(n),\ g^*(n)$和$h^*(n)$的估计值

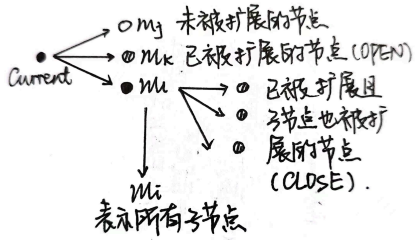
$$\mathbf{Condition:\ h(n)\le\ h^*(n)}$$



1. 伪代码及算法描述

优先扩展$f(n)$值最小的节点，直到$f(\text{终点})$（在当前CLOSED表中）最小

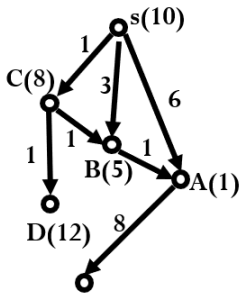
> **Notation**:
> m_j：未被扩展的节点
> m_k：已被扩展的节点（OPEN表中）
> m_l：已被扩展且子节点也被扩展的节点（CLOSED表中）
> m_i：表示所有的子节点



算法：

1 $OPEN := (s),\ CLOSED := (\ )$
2 **while OPEN** 不空 **do**:
3 $n := First(OPEN)$
4 **if** $n == goal$ **then return n**
5 $Expand(n) \to m_i,\ Calc\ f(n,\ m_i) = g(n,\ m_i) + h(m_i)$
6 $Remove(n,\ OPEN),\ Add(n,\ CLOSED)$
7 $switch\ m_i$:
8 $Add(m_j,\ OPEN)$, 标记m_j连接到n的指针
9 **if** $f(n, m_k) < f(m_k)$ **then**
10 $f(m_k) = f(n,\ m_k)$, 标记m_k连接到n的指针
11 **if** $f(n, m_l) < f(m_l)$ **then**
12 $f(m_l) = f(n,\ m_l)$, 标记m_l连接到n的指针
13 $Add(m_l, OPEN)$
14 $OPEN$中的节点按$\mathbf{f}$值从小到大排序

举例：



OPEN表	CLOSED表
s(10)	s(10)
A(7) B(8) C(9)	A(7) s(10)
B(8) C(9) G(14)	B(8) s(10)
A(5) C(9) G(14)	A(5) B(8) s(10)
C(9) G(12)	C(9) A(5) s(10)
B(7) G(12) D(14)	B(7) C(9) s(10)
A(4) G(12) D(14)	A(4) B(7) C(9) s(10)
G(11) D(14)	

当`g(n)`更新时，A^*算法的m_l类节点要重新放回到$OPEN$表中，因此可能会多次重复扩展同一个节点，导致搜索效率下降

2. 算法分析

- **可采纳性定理**：条件 $h(n) \leq h^*(n)$ 保证了A*算法的正确性（只要存在起点`s`到重点`g`的路径）
- **启发信息定理**：如果$h_2(n) > h_1(n)$（目标节点除外），则A_2^*扩展的结点数 $\leq A_1^*$扩展的节点数

对**启发函数** $h(\cdot)$ 的**评价方法**：**平均分叉数b***，b^*越小，说明 $h(\cdot)$ 越好
设共拓展了d层节点，搜索了N个节点，则有：$N = \frac{b^{*(d+1)}-1}{b^*-1}$
实验表明：b^*是一个比较稳定的常数，同一问题基本不随问题规模变化

1.2 改进版A*算法

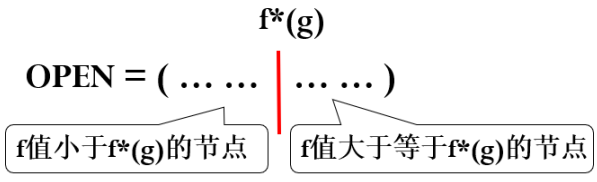
启发函数$h(\cdot)$的单调性：如果所有节点n_i和n_j（其中n_j是n_i的子节点），满足：

$$h(n_i) - h(n_j) \leq c(n_i, n_j)$$
$$h(g) = 0$$

则称启发函数$h(\cdot)$是单调的

1. 伪代码及算法描述

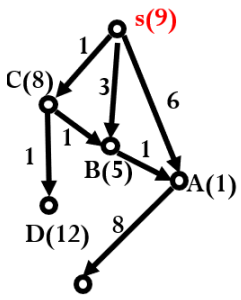
> **Notation**:
> f_m：到目前为止已扩展节点（即$OPEN$表中的节点）的最大f值



算法：

$OPEN := (s),\ CLOSED := (\ ),\ f(s) = g(s) + h(s),\ f_m = 0$
while OPEN 不空 **do** :
  $NEST = \{n_i | f(n_i) < f_m,\ n_i \in OPEN\}$
  if NEST $\neq (\ )$ **then**
    $n = NEST$ 中**g**最小的节点
  else $n = First(OPEN),\ f_m = f(n)$
  if $n == goal$ **then return n**
  $Expand(n) \rightarrow m_i,\ Calc\ f(n,\ m_i) = g(n,\ m_i) + h(m_i)$
  $Remove(n,\ OPEN),\ Add(n,\ CLOSED)$
  $switch\ m_i$:
    $Add(m_j,\ OPEN)$, 标记m_j连接到n的指针
    if $f(n, m_k) < f(m_k)$ **then**
      $f(m_k) = f(n,\ m_k)$, 标记m_k连接到n的指针
    if $f(n, m_l) < f(m_l)$ **then**
      $f(m_l) = f(n,\ m_l)$, 标记m_l连接到n的指针
      $Add(m_l, OPEN)$
    $OPEN$中的节点按**f**值从小到大排序

举例：（注意，下例不满足启发函数单调性条件）



G 目标

OPEN表	CLOSED表	f_m
s(0+9)	s(0+9)	9
A(6+1) B(3+5) C(1+8)	s(0+9) B(3+5)	9
A(4+1) C(1+8)	s(0+9) B(3+5) A(4+1)	9
C(1+8) G(12+0)	s(0+9) A(4+1) C(1+8)	9
B(2+5) G(12+0) D(2+12)	s(0+9)C(1+8)B(2+5)	9
A(3+1)G(12+0) D(2+12)	s(0+9)C(1+8)B(2+5) A(3+1)	9
G(11+0) D(2+12)		

说明：蓝颜色表示在nest中的节点

$NEST$的选取其实相当于 $h(\cdot) := 0$，此时 $h(\cdot)$ 满足单调性

2. 算法分析

- 若$h(\cdot)$是单调的，则A^*扩展了节点n之后，就已经找到了到达节点n的最佳路径，即：当选择n扩展时，有$g(n) = g^*(n)$
  - 可以证明，当$h(\cdot)$满足单调条件时，一定也满足A^*条件
- $OPEN$表上任意具有$f(n) < f^*(g)$的节点一定会被A^*扩展
  A^*选做扩展的任一节点，一定有$f(n) \leq f^*(g)$

1.3 viterbi 算法

如何通俗地讲解 viterbi 算法?

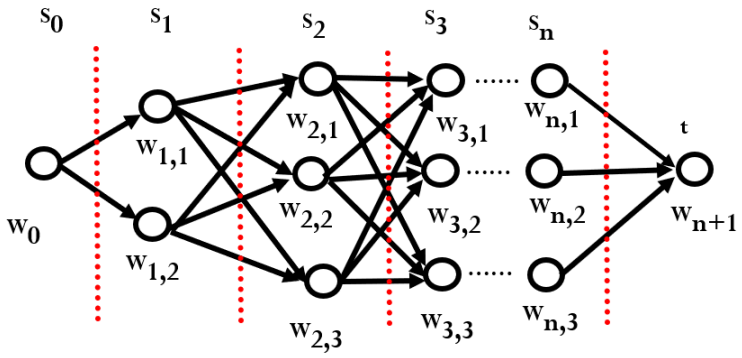
维特比算法（英语：Viterbi algorithm）是一种动态规划算法，它用于寻找最有可能产生观测事件序列的维特比路径——隐含状态序列

其中，动态转移方程如下：

$$\mathbf{Q(W_{i,j})=min_k(Q(W_{i-1,k})+D(W_{i-1,k},W_{i,j}))}$$

其中，$Q(W_{i,j})$表示起点到点$W_{i,j}$的最短路径距离，

$D(W_{i-1,k},W_{i,j})$表示$W_{i-1,k}$到$W_{i,j}$的距离



算法：

```
1   for i in range(0, n)
2       for j in range(0, k)
3           Q(W_{i,j}) = min_k(Q(W_{i-1,k}) + D(W_{i-1,k}, W_{i,j}))
3           Record(path(W_{i-1,k}) + W_{i,j})
4   find Max_k(Q(W_{n,k}) + D(W_{n,k}, W_{n+1}))
```

1. 汉字识别后处理

$Notations:$

S：表示待判定的句子（汉字序列变量）

O：表示输入图像或拼音等信息

w_i：表示句子中的第i个字

$$\mathbf{P(S|O)=\frac{P(S)P(O|S)}{P(O)}}$$

其中，$P(S)=\prod_{i=1}^{n}P(w_i|w_1..w_{i-1})$，$P(O)$为常量，$P(O|S)$用识别信度$CF$代替

二元模型：$P(S)=\prod_{i=1}^{n}P(w_i|w_{i-1})\overset{frequency}{\Rightarrow}\frac{w_{i-1}w_i\text{同现的次数}}{w_{i-1}\text{出现的次数}}$

三元模型：$P(S)=\prod_{i=1}^{n}P(w_i|w_{i-1}w_{i-2})\overset{frequency}{\Rightarrow}\frac{w_{i-2}w_{i-1}w_i\text{同现的次数}}{w_{i-1}\text{出现的次数}}$

平滑处理：解决$P(w_i|w_{i-1})$可能为0的问题

$$P(w_i|w_{i-1})\leftarrow\lambda P(w_i|w_{i-1})+(1-\lambda)P(w_i)$$

$\Rightarrow$ 二元模型下，问题转换为了：求$\prod_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{P(w_i|w_{i-1})CF(w_i)}$最大

2. 拼音输入法

在拼音输入法的例子里，上式有$P(O|S)\approx 1$，即识别信度CF为常数，因此问题转变为：

$$Max(\prod_{i=1}^{n}P(w_i|w_{i-1}))$$

等价于求$min(-\sum_{i=1}^{n}log(P(w_i|w_{i-1})))$

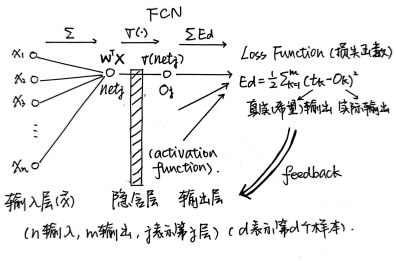
Chapter2 神经网络

神经网络的理论推导 + PyTorch语法实践

- **MLP / MultiLayer Perceptron**（多层感知器神经网络）：由多层全连接神经网络（FCN / Fully-Connected Network）组成的 feed forward neural network，由 input layer，output layer 和 hidden layer 共同组成
- **CNN / Convolutional Neural Network**（卷积神经网络）：包含*卷积*计算且具有深度结构的 feed forward neural network，包括了input layer, convolutional layer, Relu layer, pooling layer and fully-connected / linear layer
- **RNN / Recurrent Neural Network**（循环神经网络）：以序列数据为输入，在序列的演进方向进行 recursion，且所有节点/循环单元按链式连接的 recursive neural network
  - **LSTM / Long Short-Term Memory**（长短期记忆网络）
  - **GRU / Gated Recurrent Unit**（门控循环单元）

2.1 全连接神经网络 MLP / FCN

结构：输入层（input layer）+ 若干隐含层（hidden layer）+ 输出层（output layer）



- **激活函数 Activation Function**

  符号函数：$sgn(x)$，值域为$\{-1,1\}$
  sigmoid函数：$\sigma(x)=\frac{1}{1+e^{-x}}$，值域为$[0,1]$
  双曲正切函数：$tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$，值域为$[-1,1]$，可以由sigmoid线性转换得到
  ReLU / 线性整流函数：$ReLU(x)=max(0,x)$
  Softmax函数：$o_k=\frac{e^{x_k}}{\sum_{i=1}^m e^{x_i}}$，满足$\sum_{k=1}^m o_k=1$

- **损失函数 Loss Function**
  - **“误差平方和”损失函数**

    每次处理单个样本的误差：$E_d(w)=\frac{1}{2}\sum_{k=1}^m(t_{kd}-o_{kd})^2$，对应**随机梯度下降算法**
    每次处理全部样本的误差：$E(w)=\frac{1}{2}\sum_{d=1}^n\sum_{k=1}^m(t_{kd}-o_{kd})^2$，对应**批量梯度下降算法**

    小批量梯度下降算法：每次处理小批量（数量为B）的样本，$E(w)=\frac{1}{2}\sum_{i=k}^{k+B}\sum_{k=1}^m(t_{ki}-o_{ki})^2$
  - **交叉熵损失函数**

$$H_d(w)=-\sum_{k=1}^{M}t_{kd}log(o_{kd})，要求o_{kd}是概率值，假设有M个输出$$

$$H(w)=\sum_{d=1}^{N}H_d(w)=-\sum_{d=1}^{N}\sum_{k=1}^{M}t_{kd}log(o_{kd})，N表示样本数量$$

其中，t_{kd}表示样本d对应的希望输出值，o_{kd}表示对应的实际输出值（概率值）

- **梯度下降法 Gradient Descent**

$$\nabla_{\mathbf{w}} E(\mathbf{w}) = [\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \dots, \frac{\partial E(\mathbf{w})}{\partial w_n}]$$

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \Delta\mathbf{w}，其中：\Delta\mathbf{w} = -\eta\nabla_{\mathbf{w}} E(\mathbf{w})$$

这里的$\mathbf{w}$即为FCN隐含层的权重参数

- **反向传播 Back Propagation**

  算法过程：

  初始化所有权值为小的随机值（如$[-0.05, 0.05]$）
  在满足结束条件前：
  　　每个训练样例 $\rightarrow$ 输入网络，计算每个单元u的输出o_u
  　　　　对于输出层单元k，计算误差项：$\delta_k = (t_k - o_k)o_k(1 - o_k)$
  　　　　对于隐含层单元h，计算误差项：$\delta_h = o_h(1 - o_h)\sum_{k\in后继(h)} \delta_k w_{k,h}$
  　　　　更新每个权值：$w_{ji} += \Delta w_{ji}$，其中$\Delta w_{ji} = \eta\delta_j x_{ji}$

  理论推导：

  > **Notations**:
  > x_{ji}：神经元`j`（即那一层的第`j`个神经元）的第`i`个输入
  > w_{ji}：神经元`j`（即那一层的第`j`个神经元）的第`i`个权重
  > net_j：神经元`j`计算的加权和，即$\mathbf{w_j}^T\mathbf{x_j}$
  > δ_j：即$(t_j - o_j)o_j(1 - o_j)$，过程见以下推导
  > $\sigma(x)$：激活函数，即$\frac{1}{1+e^{-x}}$
  >
  > $$\vec{x}_j, \vec{w}_j \overset{\sum}{\rightarrow} net_j \overset{\sigma}{\rightarrow} o_j \overset{E_d}{\rightarrow} \frac{1}{2}\sum_{k=1}^{m}(t_k - o_k)^2$$

  - 输出层

$$\mathbf{proof}: \frac{\partial E_d}{\partial w_{ji}} = -\delta_j x_{ji}, \quad \delta_j = (t_j - o_j)o_j(1 - o_j)$$

$$\begin{aligned} \frac{\partial E_d}{\partial w_{ji}} &= \frac{\partial E_d}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}} \ (= x_{ji}) \\ 其中，\frac{\partial E_d}{\partial net_j} &= \frac{\partial E_d}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \ (= \frac{\partial \sigma}{\partial net_j}) \\ &= -(t_j - o_j) \cdot o_j(1 - o_j) \\ &= -\delta_j \end{aligned}$$

$$\therefore \frac{\partial E_d}{\partial w_{ji}} = -\delta_j x_{ji}$$

  - 隐含层

    最靠近输出层的隐含层：

$$\textbf{proof}: \quad \frac{\partial E_d}{\partial w_{ji}} = -\delta_j x_{ji}, \quad \delta_j = o_j(1-o_j)\sum_{k\in 后继(j)} \delta_k w_{k,j}$$

$$\begin{aligned}
\frac{\partial E_d}{\partial w_{ji}} &= \frac{\partial E_d}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}} \ (=x_{ji}) \\
其中，\frac{\partial E_d}{\partial net_j} &= \sum_{k\in 后继(j)} \frac{\partial E_d}{\partial net_k}\cdot\frac{\partial net_k}{\partial net_j} \\
&= \sum_{k\in 后继(j)} -\delta_k\cdot(\frac{\partial net_k}{\partial o_j}\cdot\frac{\partial o_j}{\partial net_j}) \\
&= \sum_{k\in 后继(j)} -\delta_k\cdot(w_{kj}\cdot o_j(1-o_j)) \\
&= \delta_j \\
\therefore \ \frac{\partial E_d}{\partial w_{ji}} &= -\delta_j x_{ji}
\end{aligned}$$

其它隐含层同上更新参数，$\Delta w_{ji} = \eta\delta_j x_{ji}$

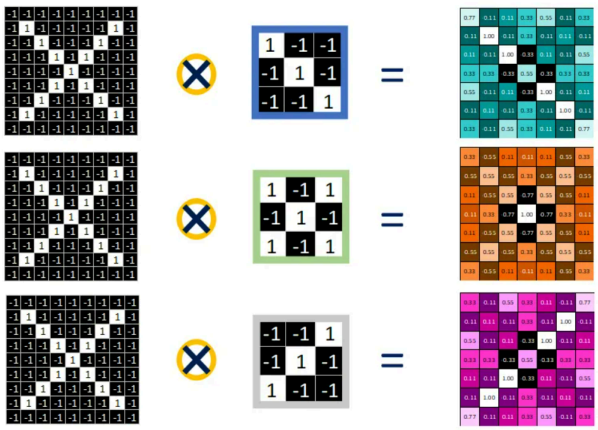
- **Softmax层**：一般在输出层前采用Softmax函数转换为概率，可用于分类问题

2.2 卷积神经网络 CNN

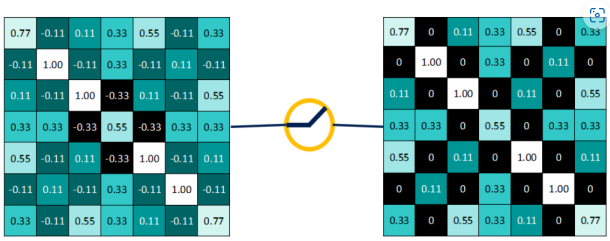
结构：卷积层 + Relu层 + 池化层 + 全连接层等

- **【卷积层】**一个3×3的卷积核（代表一个feature）作用于一张图片，会产生一张feature map
  - 卷积核的**大小**：卷积核的大小决定了其参数量，上例参数量为 $3\times3+1=10$，其中$+1$表示偏置参数
  - **步长**：卷积核每次移动的距离
  - 采用**填充**的方式：使得每层得到的结果和大小和输入一致（这也是为什么大小是奇数）
    比如：3*3填充1，5*5填充2
  - （输出）**通道**数：即卷积核的**个数**
    （输入）**通道**数：即卷积核的**厚度**

  运算逻辑：矩阵对应相乘，各元素之和取平均
  结果直观理解：越接近1表示与卷积核 feature 越匹配，接近0表示没什么关联

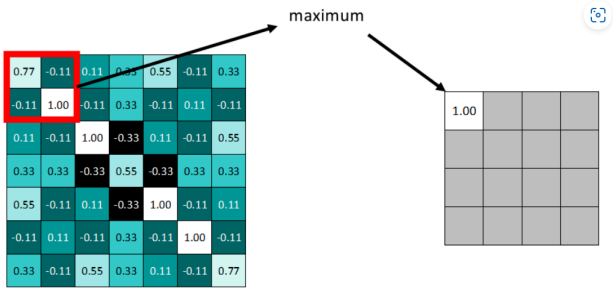


- **【Relu层】**非线性激活函数`Relu()`：保留 feature map 中大于等于0的值，其余所有小于0的数值直接改写为0



- **【池化层】**pooling：减少数据量的方式，一般分为最大池化（Max pooling）和平均池化（Average pooling）两种

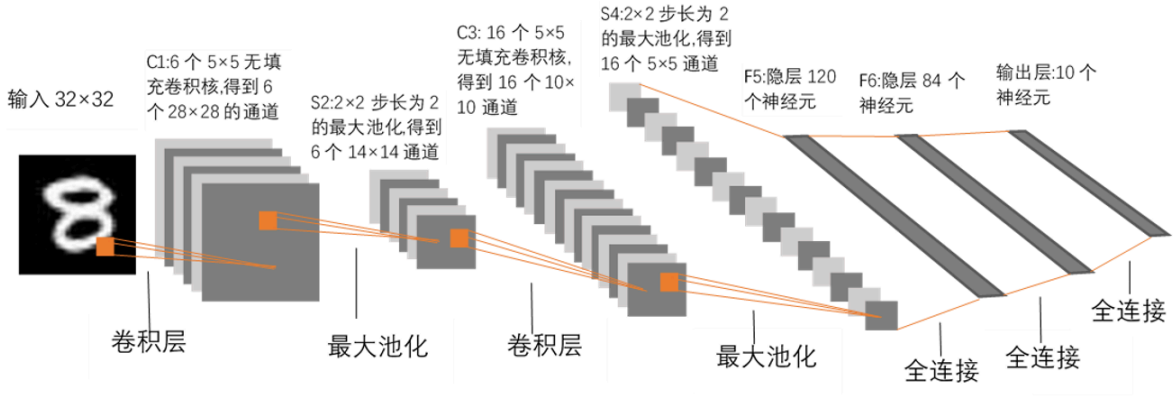
  例如，选择池化尺寸为2×2，选出最大值更新进新的 feature map



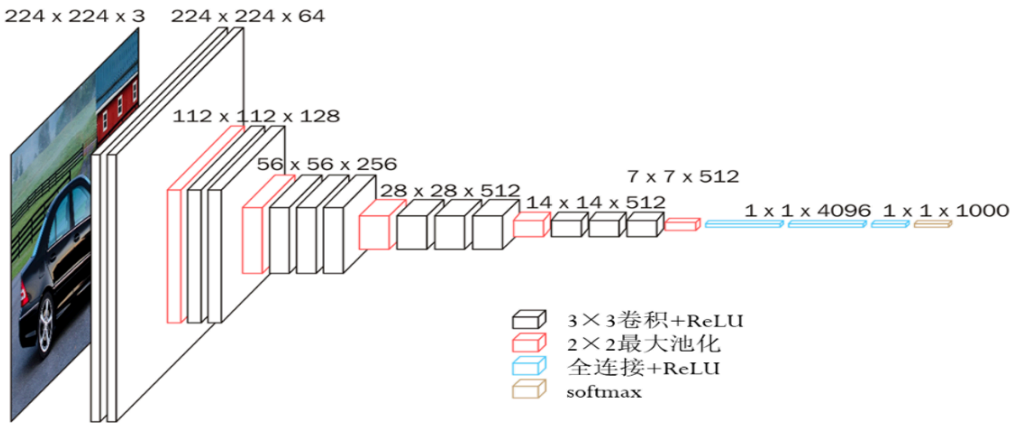
- **【全连接层】**Fully-Connected / Linear：相邻层的所有节点全部连接

举例：

1. LeNet神经网络（数字识别）



2. VGG-16神经网络（彩色图像）



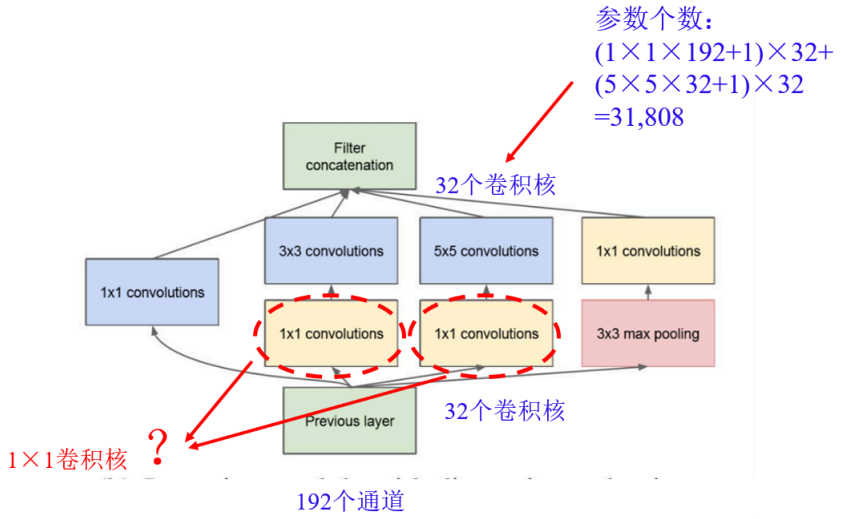
神经网络遇到的两大问题：① 梯度消失问题，② 过拟合问题

- **梯度消失问题**：当神经网络层数增多之后，$\delta_h \to 0$，无法计算$\Delta w_{ji} = \eta \delta_j x_{ji}$

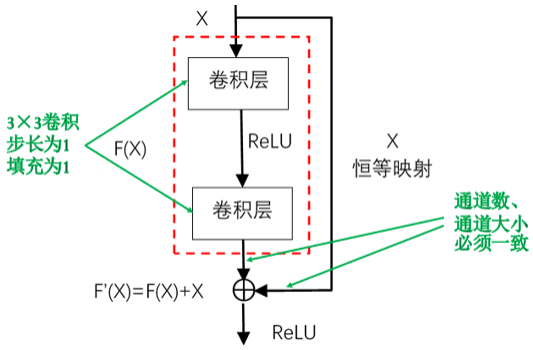
  比如说，$\delta_h = o_h(1-o_h)\sum_{k\in 后继(h)} \delta_k w_{kh}$，而使用sigmoid激活函数时，$\sum$前的最大值不超过$0.25$

  改进方式：

  1. 使用ReLU激活函数
     此时，导数为1，而不是sigmoid的导数$o_h(1-o_h) \leq 0.25$
  2. GoogLeNet & Inception模块
     分布输出，使得往下输入的梯度消失得到缓解：
     损失函数同时考虑多个输出，求解多个输出的总和最小；
     训练的时候多个输出一起输出，真正输出的时候只有一个



  3. 残差网络（ResNet）
     起因：在讨论结构“K层神经网络 + 1层恒等映射”神经网络的退化现象（更深的网络在训练过程中的难度——training error 和 test error 更大）
     解决思路：将第“K+1层恒等映射”变为残差模块，按位相加



- **过拟合问题**的解决方法

  1. 使用验证集：使用验证集确定迭代轮次停止点
  2. 正则化项法
     在原误差平方和损失函数的基础上加入正则化项（以下以**2-范数**为例）：

$$E_d(w) = \sum_{k=1}^{m}(t_{kd} - o_{kd})^2 + ||w||_2^2$$

$$其中，||w||_2^2 = w_1^2 + w_2^2 + \ldots + w_N^2$$

原理：降低模型复杂性（过拟合的很大概率源自于参数过大）

- **2-范数**：很多参数值很小，但基本不会为0；抗干扰能力强
- **1-范数**：一些参数为0，起到特征选择的作用

3. 舍弃发（Dropout）：每次训练随机地临时舍弃一些神经元
4. 数据增强法：数据越多，过拟合的风险就越小
   人为增加一些数据（例如图像的缩放、旋转、局部截取、改变颜色）

TextCNN：自然语言处理 NLP

1. 词向量

- **词向量的表示**
  - **独热（one-hot）编码**：用与词表等长的向量表示一个词，向量只有一个元素为1（第`i`个元素为1的向量用于表示此表中的第`i`个词），其余为0

    优点：编码简单，适用于**稀疏向量**
    缺点：编码太长；无法度量词之间的相似性
  - **分布式表示**：可以度量词之间的相似性，适用于**稠密向量**
- **词嵌入 / Word embedding**

  每个词对应一个训练得到的词向量，是把词向量从高维空间嵌入到低维空间中的一个方法
  语义相近的词，对应词向量的“距离”也越近

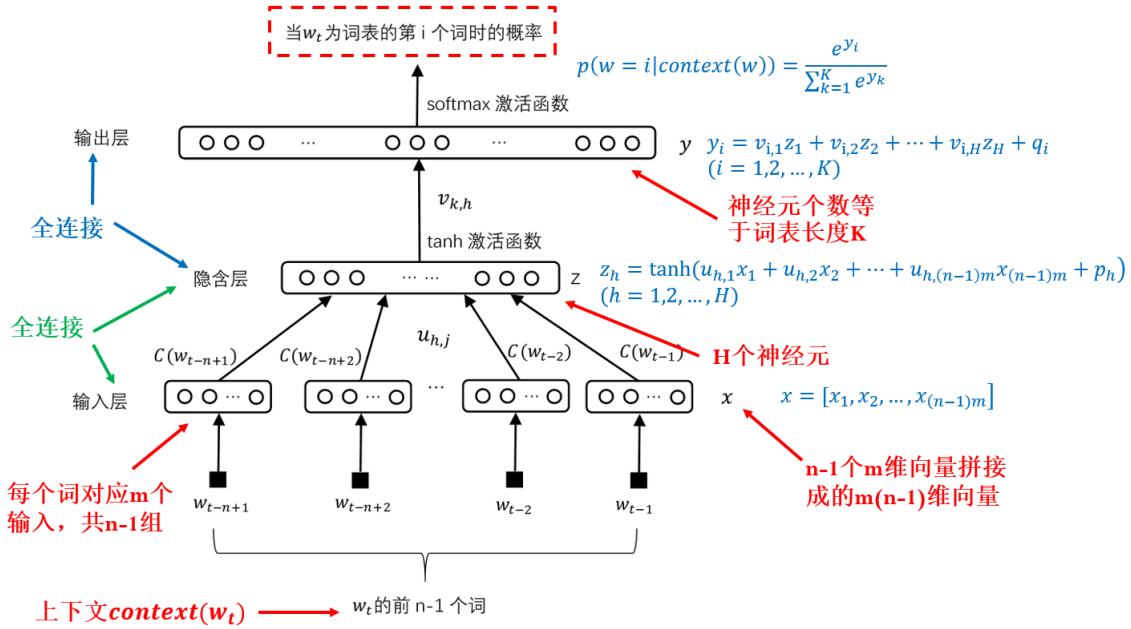
2. 语言模型 & 词向量的训练

用神经网络（**NNLM / Neural Network Language Model**）实现语言模型

- **模型结构**：计算一个句子概率的模型 => 训练得到的词向量

> **Notation**：
> w_i：其中$i \in [t-1, t-n+1]$，表示w_t之前的$n-1$个词
> $C(w_k)$：其中$k \in [t-1, t-n+1]$，表示与w_k对应的m维词向量
> $u_{h,j}$：第j个词向量的边权（用于隐含层第h个神经元的计算）
> z_h：隐含层的第h个神经元，其使用$tanh()$激活函数中的参数p_h表示偏置
> $v_{k,h}$：隐含层第h个神经元和输出层第k个神经元之间的边权
> y_i：输出层的第i个神经元，其中加权和后的q_i参数表示偏置，输出层共K个神经元（对应词表长度）

$$\text{后一个词的概率 } p_i = p\left(w_i \middle| w_{i-(n-1)} w_{i-(n-2)} \dots w_{i-2} w_{i-1}\right)$$



- **模型分析**：实际上是用最大似然估计法在确定词向量（以及模型中的权重函数$\mathbf{w}$）

  举例：

  > 假定语料库有3句话，窗口为2：“计算机|科学”，“计算机|科学”，“计算机|工程”
  >
  > 则联合概率：$p(\text{计算机科学，计算机科学，计算机工程})=p(\text{科学}|\text{计算机})^2\times(1-p(\text{科学}|\text{计算机}))$，是关于$p(\text{科学}|\text{计算机})$的函数
  >
  > 希望上面的式子概率最大，得到$p(\text{科学}|\text{计算机})=2/3$

  估计神经网络语言模型的参数：

  $$max_\theta \prod_{w\in C} p(w=k|context(w),\theta) \quad \text{（似然函数）}$$
  $$max_\theta \sum_{w\in C} log\, p(w=k|context(w),\theta) \quad \text{（对数似然函数）}$$
  $$-\sum_{w\in C} log\, p(w=k|context(w),\theta) \quad \text{（负对数似然函数）}$$

  表示：[给定神经网络参数θ和上下文窗口$context$，语料库C中的词w为词表中第k个词的概率]最大时的θ

- **存在问题**

  输出层神经元个数等于词表长度K（很大）

  $m(n-1)$个输入全连接参数多

word2vec模型

一种简化的神经网络语言模型

两种实现方式：① 连续词袋模型（CBOW），② 跳词模型（Skip-Gram Model）

CBOW模型的特点：

① 词表的数字是连续的

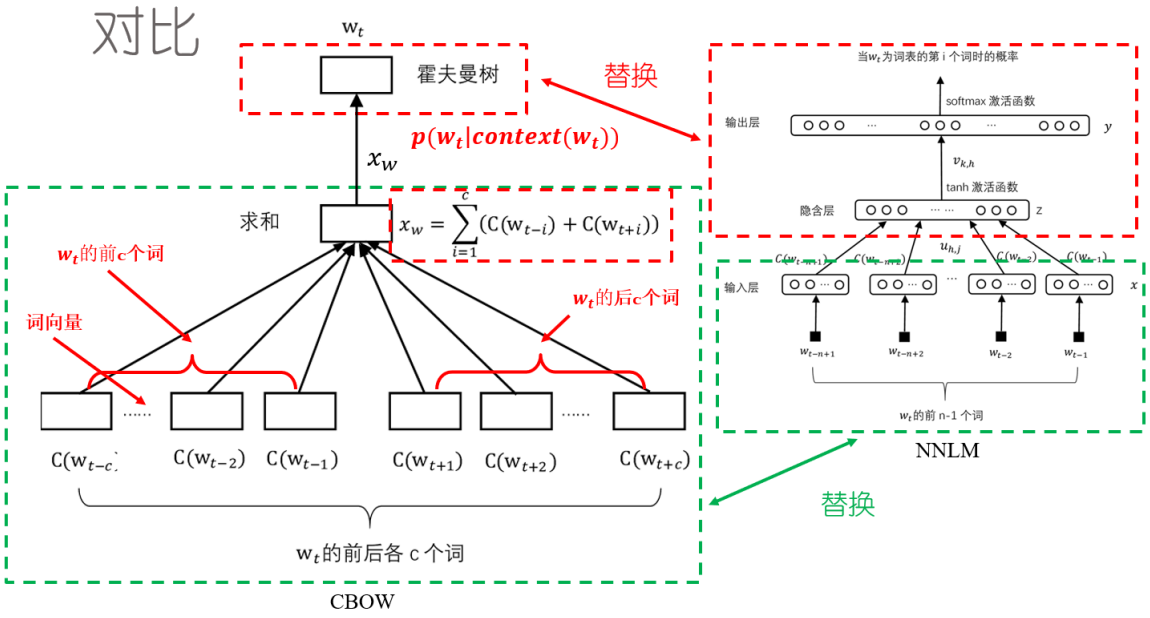
② 词袋中不考虑词语的顺序（x_w是求和运算，无关乎顺序）

③ 霍夫曼树（Hierarchical Softmax）相当于NNLM中的隐含层 + 输出层

下面详细介绍CBOW模型：

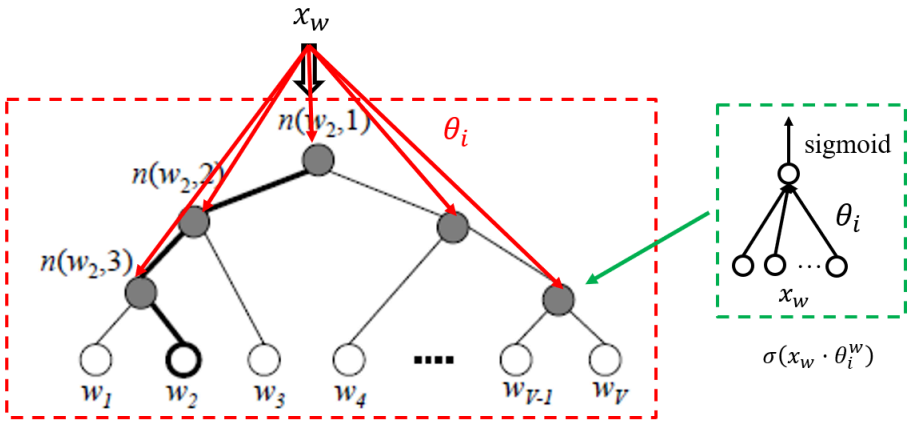
[Word2vec如何得到词向量-CSDN博客]

CBOW模型的训练**输入**是某一个特征词的上下文相关的词对应的词向量（one-hot），而**输出**是一棵哈夫曼树。



- **霍夫曼树与霍夫曼编码**

  1. 词表内的词根据出现频率编码为霍夫曼树（左边为`1`，右边为`0`）
  2. 叶节点w_i为词，$n(w_i, j)$为w_i经过的第j个节点所选择的方向
  3. 非叶节点**输入**为$\mathbf{x_w}$（当前内部节点的词向量）、权重为θ_i（待求的逻辑回归模型参数）的神经元，**输出**为选择左右的概率
     选择右边的概率：$p(R) = \sigma(\mathbf{x_w} \cdot \theta_i) = \frac{1}{1+e^{-x_w^T \cdot \theta_i}}$
     选择左边的概率：$p(L) = 1 - p(R)$
  4. 以w_2为例，即最大化 $L = \Pi_{i=1}^{3} p(L, \theta_1) \cdot p(L, \theta_2) \cdot p(R, \theta_3)$



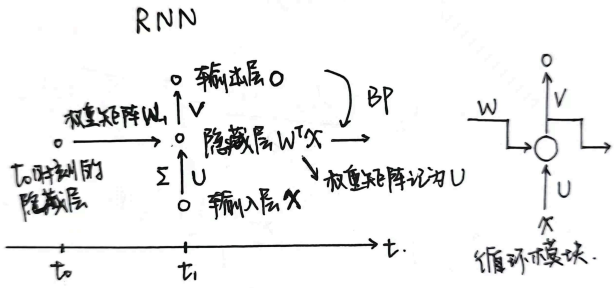
  5. 由此，可以定义词的最大似然函数以及损失函数（负对数函数），根据梯度上升（$\eta\frac{\partial L}{\partial \theta}$）求出参数为$\theta$的更新量，同时将$\mathbf{x_w}$的更新量（$\eta\frac{\partial L}{\partial \mathbf{x_w}}$）作用到每一个输入的词向量上——从而不断更新词向量

优势：每次只计算与该词有关的参数；越是常用词涉及的参数越少

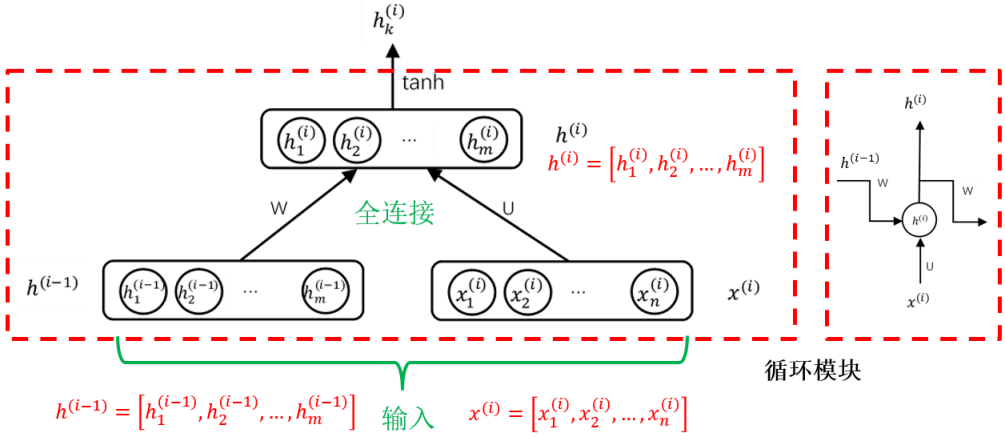
2.3 循环神经网络 RNN

RNN采用反馈网络机制，擅长处理序列数据（数据先后有所联系）

结构：全连接神经网络CNN的基础上增加上一时刻隐藏层反馈算法

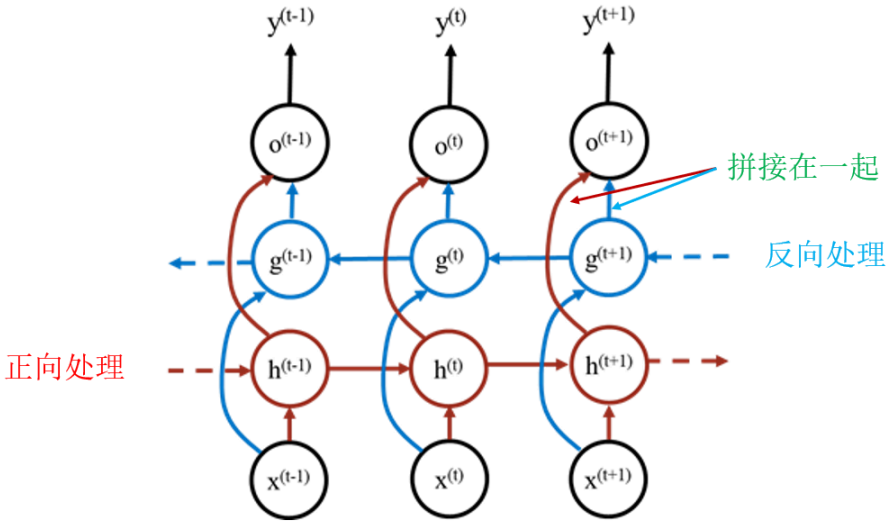


$$h_k^i = \tanh(w_{k,1} \cdot h_1^{(i-1)} + w_{k,2} \cdot h_2^{(i-1)} + \cdots + w_{k,m} \cdot h_m^{(i-1)} + u_{k,1} \cdot x_1^{(i)} + u_{k,2} \cdot x_2^{(i)} + \cdots + u_{k,n} \cdot x_n^{(i)})$$



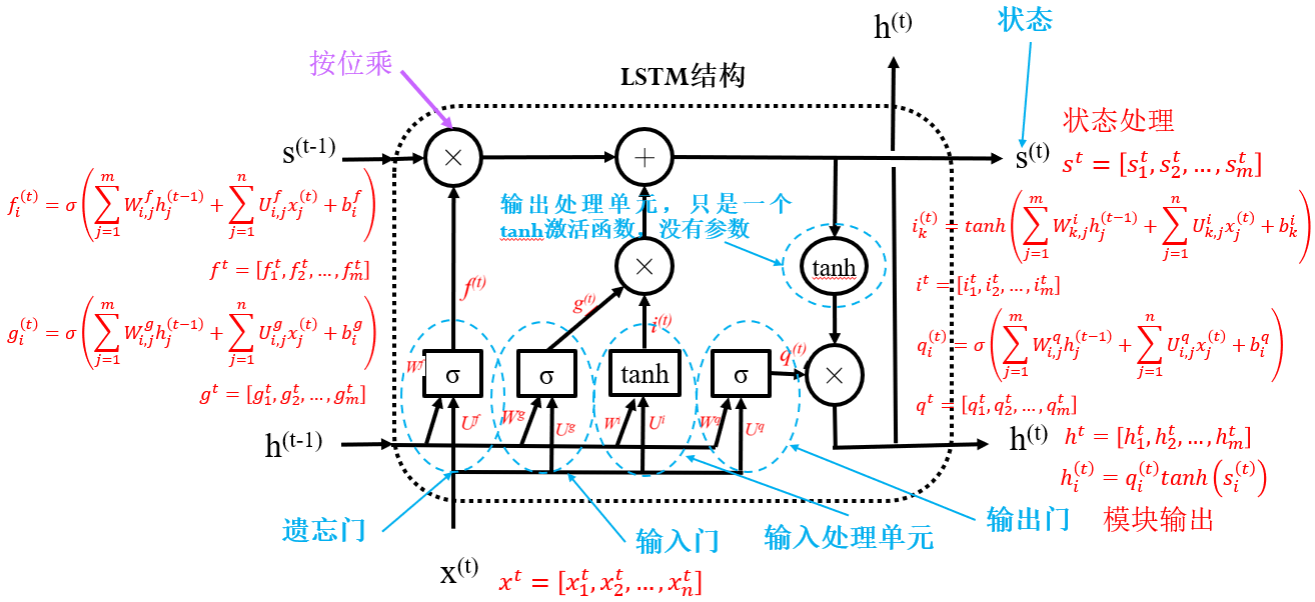
单向RNN存在的问题：序列前面的内容被后面的内容淹没

- **双向循环神经网络**



2.3.1 长短期记忆网络 LSTM

简单RNN存在的问题：
① 长期依赖问题（如：北京**是**一个美丽的（城市）vs 北京**市**一个美丽的（**姑娘**））
② 重点选择问题（不同任务词的重要性不同）



2.3.2 GRU

相比LSTM，使用GRU能够达到相当的效果，并且相比之下更容易进行训练，能够很大程度上提高训练效率

结构：LSTM的基础上，使用更少的门达到同样的处理效果

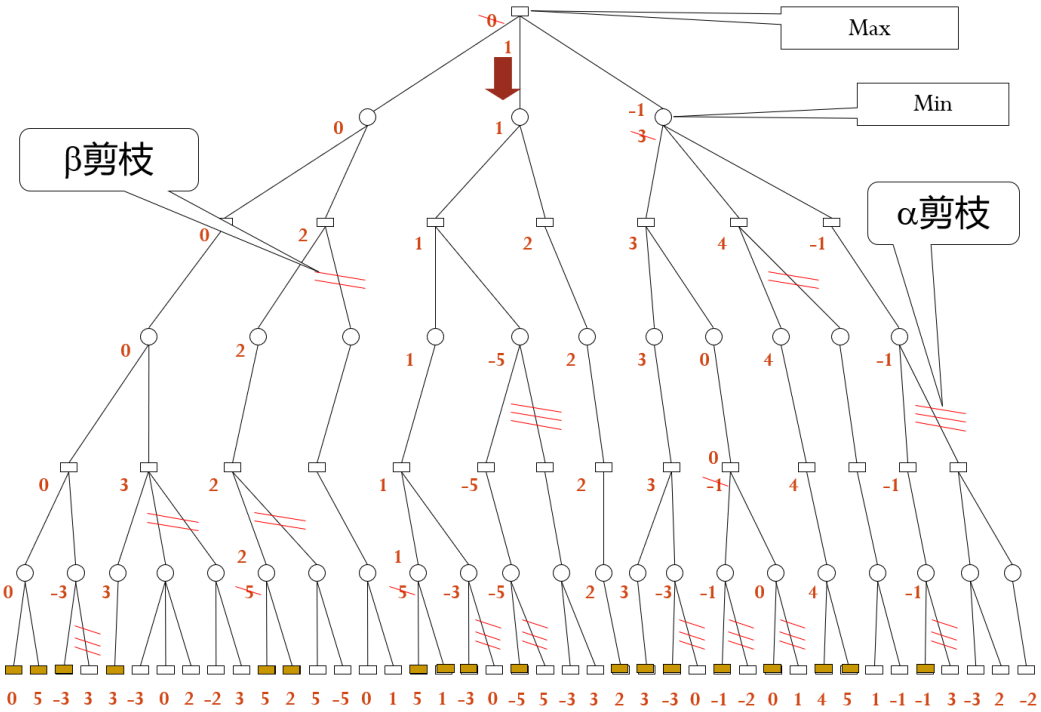
Chapter3 对抗搜索

博弈问题：双人 + 一人一步交替进行 + 双方信息完备 + 零和博弈
场景（**极小-极大模型**）：A和B对抗博弈，一方A以评分（score）大为优，另一方B以评分小为优

3.1 α-β剪枝算法

极大节点 `[]` 的下界为α，极小节点 `()` 的上界为β

剪枝条件：（注意，这里的“祖先”**不只**包括“父节点”）
后辈节点的β值≤祖先节点的α值（极小≤极大）时，α剪枝
后辈节点的α值≥祖先节点的β值时（极大≥极小）时，β剪枝



注意：一次剪枝过程只得到**一次**走步

方法问题：需要大量的专家知识

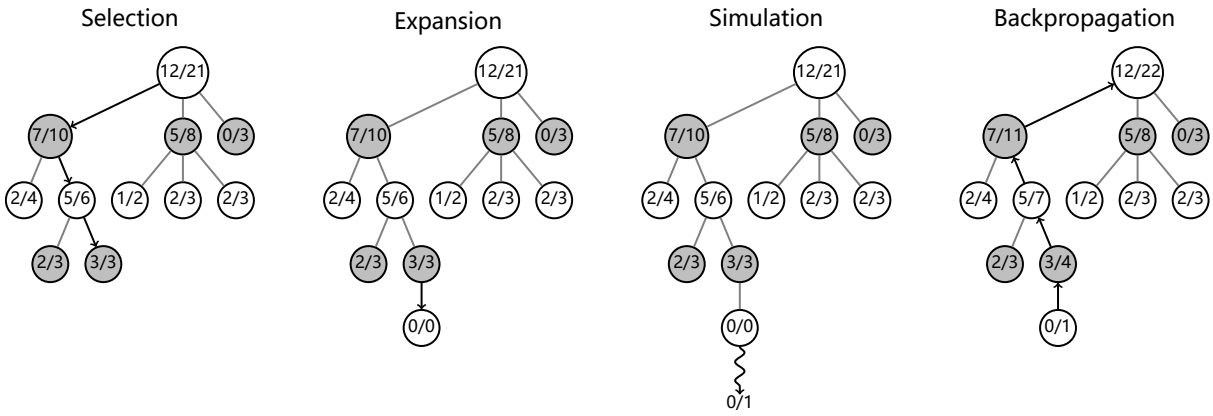
3.2 蒙特卡洛方法 MCM

蒙特卡洛方法（Monte Carlo methods）通过随机抽样、基于大数定律近似计算出问题的解或者评估问题的概率分布。即当样本量足够大时，样本的统计特征会趋近于总体的真实特征。

优势：应用于各种复杂的数值计算和概率统计问题，不需要事先对问题进行严格的数学推导，只需要进行大量的随机抽样和分析

劣势：计算效率通常较低，需要大量的计算资源和时间

接下来介绍**蒙特卡洛树搜索 MCTS / Monte Carlo Tree Search**：



> *一种简介的理解方式：*
>
> 步骤：选择（Select）⇒ 扩展（Expand）⇒ 模拟（Stimulate）⇒ 回溯（BackPropagate）⇒ 决策（Decide）

$$
\begin{aligned}
&function\ MCTS():\\
&\quad while\ within\ computational\ budegt\ \mathbf{do}\\
&\quad\quad node \leftarrow select(root)\\
&\quad\quad expand(node)\\
&\quad\quad stimulate(node)\\
&\quad\quad fathers'\ statistics \leftarrow back_propagate(node)\\
&\quad return\ decide()
\end{aligned}
$$

之后介绍课本上的MCTS步骤

蒙特卡洛树搜索通常包括以下三个阶段：

$$
\begin{aligned}
&function\ SEARCH():\\
&\quad while\ within\ computational\ budget\ \mathbf{do}\\
&\quad\quad nextNode \leftarrow treePolicy(root)\\
&\quad\quad reward \leftarrow defaultPolicy(nextNode)\\
&\quad\quad backpropagation(nextNode)\\
&\quad return\ defaultChild()
\end{aligned}
$$

1. **选择（Selection）**：从当前状态开始，根据一定的策略选择一个子节点进行扩展，即上述代码中的$treePolicy(\cdot)$

   > 不停的遍历、扩展节点
   >
   > - 对于尚未完全被拓展的节点，随机选取未被扩展的子节点进行扩展（`expand()`）
   > - 对于完全被扩展的节点，选取**最有希望**的子节点（`bestChild()`）重复上述步骤

   扩展（Expansion）：对选定的子节点进行扩展，生成新的子节点

   $$
   \begin{aligned}
   &function\ expand(v):\\
   &\quad choose\ \mathbf{a} \in untried\ actions\ from\ v's\ expandables\\
   &\quad add\ the\ new\ child\ v'\ to\ v\\
   &return\ v'
   \end{aligned}
   $$

2. **模拟（Simulation）**：通过随机模拟的方式，在新生成的子节点上进行多次完整的随机模拟或者游戏对局，即上述代码中的$defaultPolicy(\cdot)$

   回溯（Backpropagation）：根据模拟的结果，更新从根节点到当前节点路径上的统计信息，如胜率、访问次数等即上述代码中的$defaultPolicy(\cdot)$

   注意：更新的过程每一层都要“转换身份 / 正负号”！

   $$
   \begin{aligned}
   &function\ defaultPolicy(v):\\
   &\quad while\ v\ is\ non-terminal\ \mathbf{do}\\
   &\quad\quad choose\ next\ step\ v'\ from\ v\ randomly\\
   &\quad\quad until\ there\ is\ a\ winner\\
   &return\ (reward\ for\ v')
   \end{aligned}
   $$

3. **决策（Decision）**：当到了一定的迭代次数或者时间之后结束，选择根节点下最好的子节点作为本次决策的结果，即上述代码中的$defaultChild(\cdot)$

以上叙述中，“节点”表示某个需要决策的局面，或者当前状态（status）

- **UCT**：“最有希望”的子节点用什么准则描述？

  UCB算法（Upper Confidence Bound，信心上限算法）：

$$\mathbf{UCB_i = win_rate_i + C\sqrt{\frac{lnN}{n_i}}}$$

其中：

win_rate_i：当前节点i的胜率（ $= \frac{win}{total}$ ）

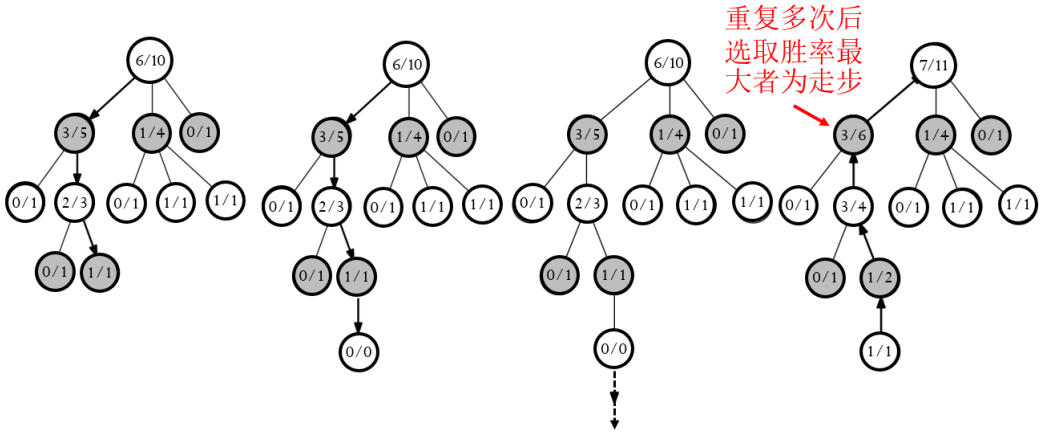
N：当前节点的父节点被访问的次数

n_i：当前节点i被访问的次数

C：探索参数; C越大，越照顾访问次数较少的子节点，即探索性越大。

UCT算法（Upper Confidence Bound Apply to Tree，上限置信区间算法）：UCB算法 + MCTS算法

注意：UCT和上述MCTS的叙述不同——UCT中节点标注“获胜次数 / 模拟总次数”中的“获胜次数”是从本节点角度说的（下图中黑色节点代表己方状态，白色节点代表对手状态）



AlphaGo原理

待补充

3.3 深度强化学习方法（围棋）

强化学习：学习“做什么才能使得收益最大化”的方法；学习者不会被告知如何做，必须自己通过尝试发现哪些动作会产生最大的收益

特征：试错和延迟收益（区别于监督学习）

深度强化学习：用深度学习（神经网络）实现的强化学习；关键在于损失函数的定义

三种实现方式：基于策略梯度的强化学习、基于价值评估的强化学习、基于演员-评价方法的强化学习

- **基于策略梯度的强化学习**：学习到的是每个落子点获胜的概率

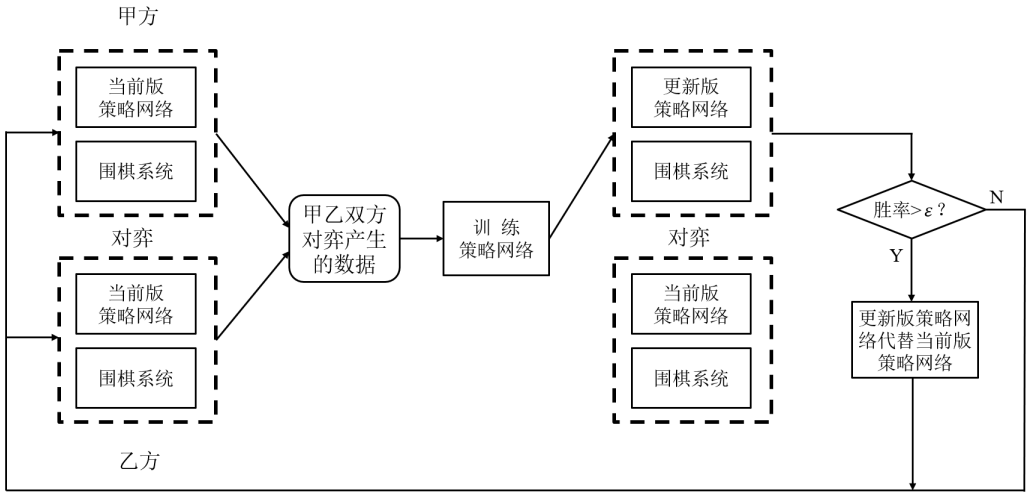
  数据：自我博弈产生，(s, a, p_a, t_a)

  > **Notations**:
  > s：当前棋局
  > a：s棋局下在a处行棋
  > p_a：s棋局下在a处行棋的获胜概率
  > t_a：胜负值，胜为1，负为-1

损失函数：$L(w) = -t_a log(p_a)$

假设获胜者的行为都是正确的，负者行为都是不正确的

假设获负时对权重的修改量大小与获胜时一样，方向相反



注意：

1. 在强化学习过程中，每个样本只使用一次
2. 基于策略梯度的强化学习方法学到的是在**每个**可落子点行棋的**获胜**概率（监督学习策略网络学到的是在**某个**可落子点行棋的概率）

- **基于价值评估的强化学习**：学习到的是每个落子点获取最大收益的概率

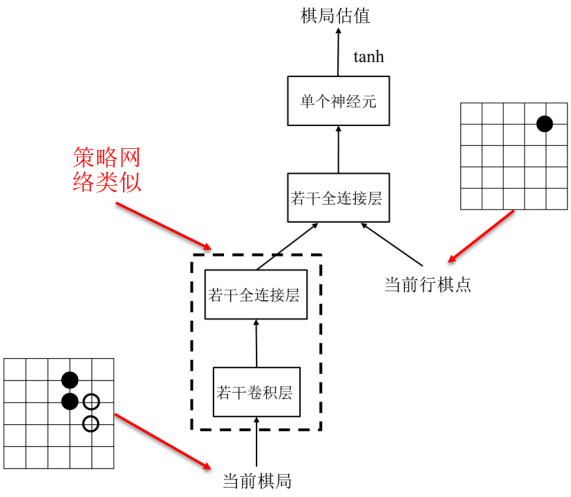
  对一个行棋点的价值，也就是收益进行评估

  - 输入：当前棋局和行棋点
  - 输出：取值在[-1,1]之间的估值

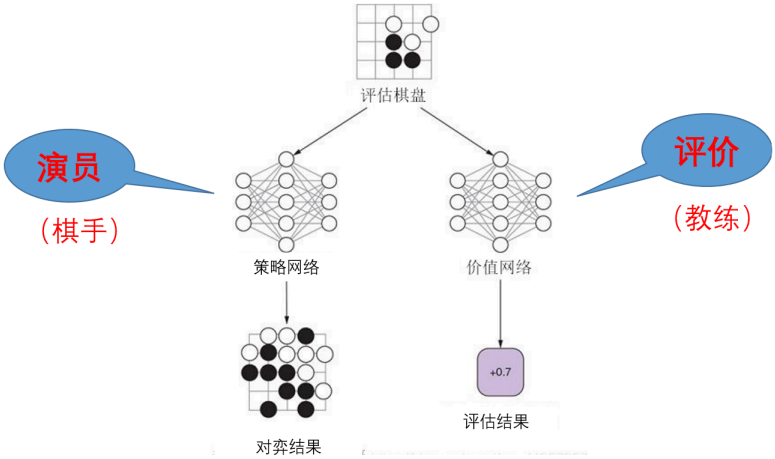
  数据：自我博弈产生，$(s, a, V(s,a), R)$

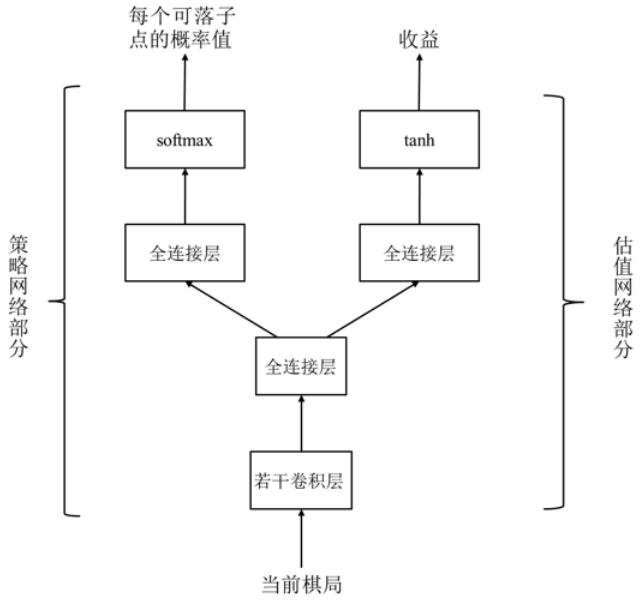
  > **Notations**:
  > s：当前棋局
  > a：s棋局下在a处行棋
  > $V(s,a)$：s棋局下在a处行棋时网络的输出
  > R：胜负值，胜为1，负为-1

  损失函数：$L(w) = (R - V(s,a))^2$



- **基于演员-评价方法的强化学习**：学习到的是每个落子点获得最大收益增量的概率





收益增量评价一步棋的好坏：$A = Q(s,a) - V(s) = R - V(s)$

> **Notations**:
> $V(s)$：当前棋局s的预期收益，取值范围为$[-1,1]$
> $Q(s,a)$：s棋局下在a处行棋后的收益，取值范围为$[-1,1]$
> A：收益增量，取值范围为$[-2,2]$，A越大越说明走了一步妙招

损失函数：

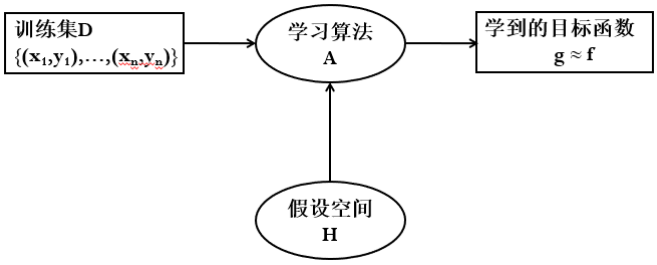
- 评价部分：$L_1(w) = (R - V(s))^2$
- 演员部分：$L_2(w) = -Alog(p_a)$，这里p_a为s棋局下在a处行棋的~~获胜~~概率
- **综合损失函数**：$L(w) = L_1(w) + \lambda L_2(w)$

AlphaGo Zero 强化学习

待补充

Chapter4 统计机器学习

统计机器学习方法：学习算法A根据训练集D，从假设空间H中选择一个最好的$g \approx f$



三要素：模型、策略、算法
模型：学习什么样的模型（条件概率分布、决策函数）
策略：模型选择的准则（经验风险最小化、结构风险最小化）
算法：模型学习的算法（一般归结为一个最优化问题）

4.1 支撑向量机 SVM

SVM：Support Vector Machines，是一个二分类器——特征空间上的间隔最大化线性分类器，通过核技巧可以实现非线性分类

$$
\begin{gathered}
\text{给定线性可分训练集 } \mathbf{T}=\{(\mathbf{x_1}, y_1), \ldots,(\mathbf{x_N}, y_N)\}, \\
\mathbf{x_i} \in X=\mathbb{R}^n,\ y \in Y=\{+1,-1\},\ i=1,2, \ldots, N \\
\text{这里，}\mathbf{x_i}\text{为第}i\text{个特征向量，}\ y_i\text{为}\mathbf{x_i}\text{的类标记，}+1\text{表示正类，}-1\text{表示负类}
\end{gathered}
$$

$$
\begin{gathered}
\text{通过间隔最大化得到分类超平面：} \\
\mathbf{w^{*T}x}+\ b^*=0 \\
\text{相对应的决策函数为：} \\
f(x)=sign(\mathbf{w^{*T}x}+b^*) \\
\text{称为线性可分支持向量机}
\end{gathered}
$$

- 函数间隔：

$$
\begin{gathered}
\text{设训练集}T\text{和超平面}(\mathbf{w}, b) \\
\text{定义超平面}(\mathbf{w}, b)\text{关于样本点}(\mathbf{x_i}, y_i)\text{的函数间隔为：}\hat{\gamma}_i=y_i(\mathbf{w^T x_i}+b) \\
\text{定义超平面关于训练集}\mathbf{T}\text{的函数间隔为：}\hat{\gamma}=min_i\ \hat{\gamma}_i
\end{gathered}
$$

- 几何间隔：

$$
\begin{gathered}
\gamma_i=y_i(\frac{\mathbf{w^T x_i}}{||\mathbf{w}||}+\frac{b}{||\mathbf{w}||}) \\
\gamma=min_i\ \gamma_i \\
\text{期中，}||\mathbf{w}||\text{为}\mathbf{w}\text{的}L_2\text{范数}
\end{gathered}
$$

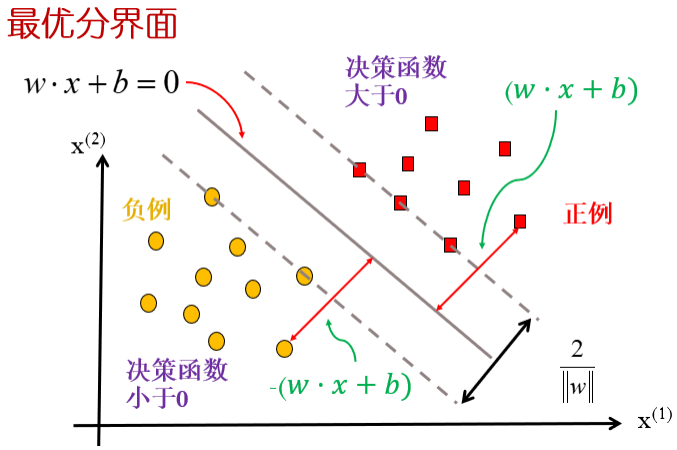
最优分界面：即间隔最大的超平面，满足

用函数间隔表示：

$$max_{\mathbf{w},b}(\frac{\hat{\gamma}}{||\mathbf{w}||})\ \ s.t.\ \ y_i(\mathbf{w^T x_i}+b)\geq\hat{\gamma},\ i=1,2,\dots,N$$

用几何间隔表示：

$$max_{\mathbf{w},b}\gamma\ \ s.t.\ \ y_i(\frac{\mathbf{w^T x_i}}{||\mathbf{w}||}+\frac{b}{||\mathbf{w}||})\geq\gamma,\ i=1,2,\dots,N$$



由于函数间隔是可缩放的，成比例变化不影响最优化问题，所以可取$\hat{\gamma}=1$

转化为如下的凸二次规划问题：

$$min_{\mathbf{w},b}\frac{1}{2}||\mathbf{w}||^2\ s.t.\ \ y_i(\mathbf{w^T x_i}+b)\geq 1,\ i=1,2,\dots,N$$

1. SVM与对偶算法

原始问题：$min_{\mathbf{w},b}\frac{1}{2}||w||^2\quad s.t.\quad y_i(\mathbf{w^T x_i}+b)\geq 1,\quad i=1,2,\dots,N$

定义拉格朗日函数：

$$L(\mathbf{w},b,\alpha)=\frac{1}{2}||\mathbf{w}||^2+\sum_{j=1}^{N}\alpha_j[1-y_i(\mathbf{w^T x_i}+b)]$$

其中，$\alpha_j\geq 0,\alpha=(\alpha_1,\alpha_2,\dots,\alpha_N)^T$为拉格朗日乘子向量

拉格朗日函数与原始优化问题的关系：

$$max_\alpha(L(w,b,\alpha))=\begin{cases}\frac{1}{2}||\mathbf{w}||^2, & \text{满足约束条件时}\\ \infty, & \text{其它}\end{cases}$$

$\therefore\ \mathbf{min_{w,b}\ max_\alpha(L(w,b,\alpha))}$与原始问题等价

满足KKT条件时，$max_\alpha min_{\mathbf{w},b}(L(\mathbf{w},b,\alpha))\leq min_{\mathbf{w},b}max_\alpha(L(\mathbf{w},b,\alpha))$ 等号成立

$\therefore$ 原始问题的对偶问题为$\mathbf{max_\alpha min_{w,b}(L(w,b,\alpha))}$，

且原始问题与对偶问题等价

其中，KKT：

1 $\nabla_{\mathbf{w},b}L(\mathbf{w},b,\alpha)=0$

2 $\alpha_i[1-y_i(\mathbf{w^T x_i}+b)]=0$

3 $[1-y_i(\mathbf{w^T x_i}+b)]\leq 0$

4 $\alpha_i\geq 0$

5 $i=1,2,\dots N$

求解对偶问题：对$\mathbf{w}, b$求偏导令其为0并代入（下式为二维的情况）

$$\mathbf{max}_{\alpha}\mathbf{min}_{\mathbf{w},b}: \ L(\mathbf{w},b,\alpha)=\frac{1}{2}||\mathbf{w}||^2+\sum_{i=1}^{N}\alpha_i[1-y_i(\mathbf{w^T x_i}+b)]$$

$$max_{\alpha}(-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x_i^T x_j})+\sum_{i=1}^{N}\alpha_i) \quad s.t. \ \sum_{i=1}^{N}\alpha_i y_i=0$$

由此可以解得α

利用KKT条件，得到：

$$\mathbf{w}^*=\sum_{i=1}^{N}\alpha_i^* y_i \mathbf{x_i}$$

$$b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i \mathbf{x_i^T x_j} \ (\text{选择一个}\alpha_j\neq 0)$$

其中，$\alpha_i\neq 0$对应的样本$\mathbf{x_i}$称为**支持向量**

例题：

- **例：设正例：$\mathbf{x_1}=(3,3)^T, \mathbf{x_2}=(4,3)^T$，负例：$\mathbf{x_3}=(1,1)^T$，**
- **用对偶问题求线性可分支持向量机。**

$$\min_{\alpha}\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i$$

$$=\min_{\alpha}\frac{1}{2}(18\alpha_1^2+25\alpha_2^2+2\alpha_3^2+42\alpha_1\alpha_2-12\alpha_1\alpha_3$$

$$-14\alpha_2\alpha_3)-\alpha_1-\alpha_2-\alpha_3$$

$$s.t. \ \alpha_1+\alpha_2-\alpha_3=0$$

$$\alpha_i\geq 0, i=1,2,3$$

将$\alpha_3=\alpha_1+\alpha_2$代入，并记为：

$$s(\alpha_1,\alpha_2)=4\alpha_1^2+\frac{13}{2}\alpha_2^2+10\alpha_1\alpha_2-2\alpha_1-2\alpha_2$$

通过求偏导并令其为0，易知$s(\alpha_1,\alpha_2)$在点$\left(\frac{3}{2},-1\right)^T$取极值，但该点不满足约束$\alpha_2\geq 0$

所以最小值应该在边界上。

当$\alpha_1=0$时，最小值$s\left(0,\frac{2}{13}\right)=-\frac{2}{13}$，

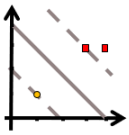
当$\alpha_2=0$时，最小值$s\left(\frac{1}{4},0\right)=-\frac{1}{4}$，

于是$s(\alpha_1,\alpha_2)$在$\alpha_1=\frac{1}{4},\alpha_2=0$时达到最小，

此时$\alpha_3=\alpha_1+\alpha_2=\frac{1}{4}$

这样α_1,α_3对应的实例点x_1,x_3是支持向量

根据前面的公式得到：



$w_1^*=w_2^*=\frac{1}{2}$

$b^*=-2$

分离超平面为：$\frac{1}{2}x^{(1)}+\frac{1}{2}x^{(2)}-2=0$

分类决策函数为：$f(x)=sign\left(\frac{1}{2}x^{(1)}+\frac{1}{2}x^{(2)}-2\right)$

2. “线性不可分”支持向量机

某些点线性不可分，意味着这些点不满足函数间隔大于等于1的条件。为此引入**松弛变量**ξ_i，使得：

$$min_{\mathbf{w},b,\xi}\ (\frac{1}{2}||\mathbf{w}||^2+C\sum_{i=1}^{N}\xi_i)$$

$$s.t. \ y_i(\mathbf{w^T x_i}+b)\geq 1-\xi_i, \ i=1,2,\ldots,N, \ \xi_i\geq 0$$

称为**软间隔最大化**。其中$C>0$为**惩罚参数**，代表目标函数惩罚项的权重。

此时，Lagrange对偶函数为：

$$\mathbf{max}_{\alpha}\mathbf{min}_{\mathbf{w},b}:\ L(\mathbf{w},b,\xi,\alpha)=\frac{1}{2}||\mathbf{w}||^2+C\sum_{i=1}^{N}\xi_i+\sum_{i=1}^{N}\alpha_i[1-\xi_i-y_i(\mathbf{w^T x_i}+b)]-\sum_{i=1}^{N}\gamma_i\xi_i,\ \gamma_i\geq 0$$

$$\frac{\partial L}{\partial \xi_i}=C-\alpha_i-\gamma_i=0\ \Rightarrow\ \gamma_i=C-\alpha_i$$

$$max_{\alpha}(-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x_i^T x_j})+\sum_{i=1}^{N}\alpha_i)\quad s.t.\ \sum_{i=1}^{N}\alpha_i y_i=0,\ 0\leq\alpha_i\leq C$$

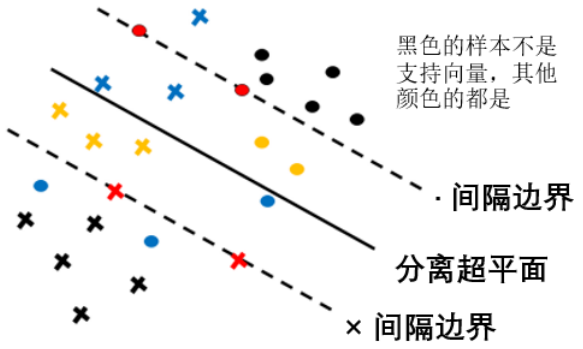
求解思路同前，但是b的计算从$\alpha_j\neq 0$变为$0<\alpha_j<C$:

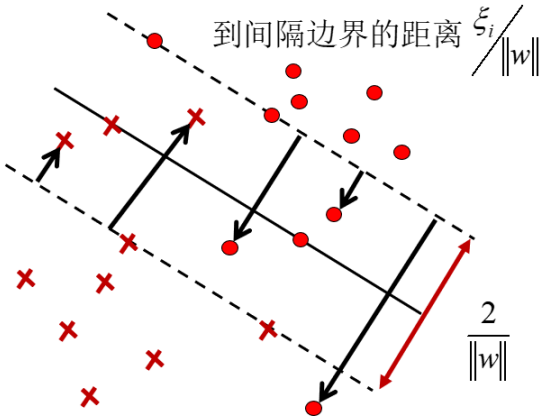
$$\mathbf{w^*}=\sum_{i=1}^{N}\alpha_i^* y_i\mathbf{x_i}$$

$$b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i\mathbf{x_i^T x_j}\ (\text{选择一个}0<\alpha_j<C)$$

其中，$\alpha_i>0$对应的样本$\mathbf{x_i}$称为**（软间隔的）支持向量**:

- 若$0<\alpha_i<C$，则$\xi_i=0$，x_i在间隔边界上
- 若$\alpha_i=C$，$0<\xi_i<1$，则分类正确，x_i在间隔边界与分离超平面之间
- 若$\alpha_i=C$，$\xi_i=1$，则x_i在超平面上
- 若$\alpha_i=C$，$\xi_i>1$，则x_i位于误分一侧



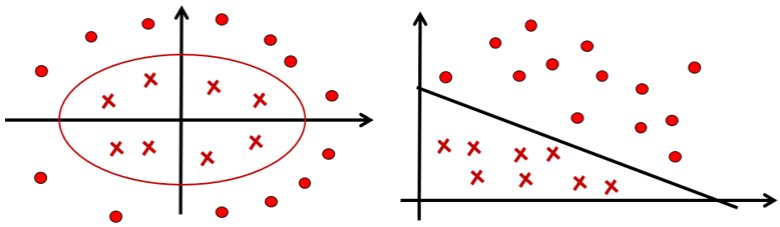


线性可分时，最大间隔是取函数间隔为1，软划分时，是$1-\xi$

注意：即使是线性可分问题，如果C选取不合适，也**有可能**存在软支持向量

3. 非线性支持向量机

设变换：$z=\phi(x)=((x^{(1)})^2,(x^{(1)})^2)^T$，原空间的椭圆：$w_1(x^{(1)})^2+w_2(x^{(1)})^2+b=0$，可以变换为新空间中的直线：$w_1z^{(1)}+w_2z^{(2)}+b=0$。这样原空间的非线性可分问题，变为了新空间线性可分问题。



核技巧：通过一个非线性变换将**输入空间X**（欧式空间或者离散集合）对应于一个**特征空间H**（希尔伯特空间），使得在输入空间X的超曲面模型对应于特征空间H中的超平面模型（支持向量机）。分类问题的学习就可以通过在**H空间中求解线性支持向量机**完成。

非线性支持向量机的对偶问题：

$$max_\alpha(-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_jy_iy_j(\phi(\mathbf{x_i})^{\mathbf{T}}\phi(\mathbf{x_j}))+\sum_{i=1}^{N}\alpha_i)$$

$$s.t.\ \sum_{i=1}^{N}\alpha_iy_i=0,\ 0\le\alpha_i\le C$$

核函数：设X是输入空间，H特征空间。如果存在映射函数

$$\phi(x):\ \mathbf{X}\to\mathbf{H}$$

使得对所有$x,z\in\mathbf{X}$，函数$K(x,z)$满足：

$$K(x,z)=\langle\phi(x),\phi(z)\rangle\ \text{（内积）}$$

则称$K(x,z)$为**核函数**，$\phi(x)$为**映射函数**。
注意：映射函数不唯一（甚至映射到的特征空间H的维度也不唯一）

因此，非线性支持向量机的对偶问题化简为：

$$max_\alpha(-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\phi(\mathbf{x_i})^{\mathbf{T}}\phi(\mathbf{x_j}))+\sum_{i=1}^{N}\alpha_i)$$

$$s.t.\ \sum_{i=1}^{N}\alpha_i y_i=0,\ 0\le\alpha_i\le C$$

选择一个$0<\alpha_j<C$分量：

$$\mathbf{w^{*T}x}=\sum_{i=1}^{N}\alpha_i^* y_i K(\mathbf{x_i},\mathbf{x})$$

$$b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i K(\mathbf{x_i},\mathbf{x_j})$$

分解超平面：$\mathbf{w^*}\cdot\phi(\mathbf{x})+b^*=\sum_{i=1}^{N}\alpha_i^* y_i\phi(\mathbf{x_i})+b^*=0$

或：$\mathbf{w^T x}=\sum_{i=1}^{N}\alpha_i y_i K(\mathbf{x_i},\mathbf{x})$

决策函数：$f(x)=sign(\sum_{i=1}^{N}\alpha_i^* y_i\phi(\mathbf{x_i})+b^*)$

- **正定核的充要条件**

  设$\mathbf{X}\in\mathbb{R}^n$，$K(x,z)$是定义在$\mathbf{X}\times\mathbf{X}$上的对称函数，则$K(x,z)$为正定核的充要条件是：对$\forall\mathbf{x_i}\in\mathbf{X}$，$i=1,\dots,m$，$K(x,z)$对应的Gram矩阵$K=[K(x_i,x_j)]_{m\times m}$是半正定矩阵

- **常用的核函数**
  - **多项式核函数**：$K(x,z)=(x\cdot z+1)^p$
    p越大，函数分界超平面越复杂
  - **高斯核函数**：$K(x,z)=e^{-\frac{||x-z||^2}{2\sigma^2}}$
    可以认为是万能的核函数
    超参σ过大可能造成**欠拟合**，σ过小可能造成**过拟合**
- **序列最小最优化算法 SMO** / Sequential Minimal Optimization
  抽象来说，每次取两个α求最值，检验KKT条件，迭代求解

SVM用于求解多分类问题：

1. 一对多：某类为正例，其余为负例；分类时将未知样本分类为具有最大分类函数值的那类（共构造N个分类器）；问题：样本不平衡
2. 一对一：任意两类构造一个SVM（共构造C_N^2个分类器），分类时采取投票法决定类别（效果最好，一般为默认情况）
3. 层次法：所有类先分成两类

- **文本分类的特征抽取**

  文本表达为一个向量：$(w_{1,j},w_{2,j},\dots,w_{n,j})^T$，$w_{ij}$表示词项$i$在文档$j$中的权重
  - 词频tf_{ij}（表示第i个词项在第j个文档中的词频）权重作为w_{ij}
    问题：只用词频不考虑具体内容的问题，如若想要抽取“清华”相关的新闻，则那些讨论“大学”（与分类无关的

词语）的文章，“大学”一次反复出现会减弱“清华”的权重
- $tf-idf$权重
  文档频率$df_i=\frac{出现词项i的文档数}{训练集的文档总数N}$
  逆文档频率$idf_i=log(\frac{1}{df_i})$，认为$df_i$比较少的词汇才是对分类真正有用的
  权重：$w_{ij}=tf_{ij}\cdot idf_i$，用逆文档频率加权（通常取$log(idf_i)$）

4.2 决策树

给定训练集：$D=\{(\mathbf{x_1},y_1),(\mathbf{x_2},y_2),\dots(\mathbf{x_N},y_N)\}$，其中$\mathbf{x_i}=(x_i^{(1)},\dots,x_i^{(n)})$为输入实例，$n$为**特征**个数，$y_i\in\{1,2,\dots K\}$为类标记，$i=1,2,\dots N$，$N$为样本容量

决策树学习就是从训练集中归纳出一组分类规则，得到一个与训练集矛盾较小的决策树的过程——是一个NPC问题，所以一般采用**启发式方法**得到一个近似解（损失函数最小作为优化目标）

- **特征选择**：用信息增益选择特征

  随机变量X的**熵**：$H(X)=-\sum_{i=1}^K p_i log p_i$，其中$p_i=P(X=x_i)$，也记作$H(p)$，衡量一个样本集的“混乱程度”（如50-50男女的班级熵最高，20-80男女比例的班级熵比较低），其中K表示K个类别（二分类有$K=2$）；当概率由数据集D估计得到时，记作$H(D)$

  条件熵：$H(Y|X)=\sum_{i=1}^n p_i H(Y|X=x_i)$，表示已知$X$时$Y$的不确定性，其中$n$表示特征$X$有$n$个取值

  特征A对数据集D的**信息增益**定义为：$g(D,A)=H(D)-H(D|A)$，即某个特征A对数据集D进行分类的不确定性减少的程度，衡量特征分类能力的强弱

  举例：设训练集D，K个类C_k，特征A有n个不同的取值$\{a_i,\dots,a_n\}$，A的不同取值将D划分为n个子集$D_1,\dots D_n$，其中D_i中属于类C_k的样本的集合为D_{ik}，$|\cdot|$表示样本个数，则信息增益计算如下：

$$H(D)=-\sum_{k=1}^K\frac{|C_k|}{|D|}\cdot log\frac{|C_k|}{D}$$

$$H(D|A)=\sum_{i=1}^n\frac{|D_i|}{|D|}\cdot H(D_i)=-\sum_{i=1}^n\frac{|D_i|}{|D|}\cdot(\sum_{k=1}^K\frac{|D_{ik}|}{|D_i|}log\frac{|D_{ik}|}{|D_i|})$$

$$信息增益\ g(D,A)=H(D)-H(D|A)$$

1. ID3算法

选择一个最大信息增益的特征，如果它足够好（信息增益比较大），那么就按这个特征先分一次类，然后递归建树即可

输入：训练集D，特征集A，阈值$\epsilon>0$

输出：决策树T

1. 若D中所有实例属于同一类C_k，则T为单节点树，将C_k作为该节点的类标记，返回T
2. 若A为空，则T为单节点树，将D中实例数最大的类C_k作为作为该节点的类标记，返回T
3. 否则计算A中各特征对D的信息增益，选择信息增益最大的特征A_g
4. 如果A_g的信息增益小于阈值ϵ，则置T为单节点树，将D中实例数最大的类C_k作为该节点的类标记，返回T
5. 否则，对A_g的每一可能值a_i，依$A_g=a_i$将D分割为若干子集D_i，作为D的子节点
6. 对于D的每个子节点D_i，若D_i为空，则将D中实例最大的类作为标记，构建子节点（比如特征为头发长度，男女分类，没有中发，则该节点返回男女中较多的类别）
7. 否则，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用步1~步6，得到子树T_i，返回T_i

存在的问题：倾向于选择**分支比较多**的属性

ID	年龄 A1	有工作 A2	有房子 A3	信贷情况 A4	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

$$H(D)=-\frac{9}{15}\log_2\frac{9}{15}-\frac{6}{15}\log_2\frac{6}{15}=0.971$$

$$g(D,A_1)=H(D)-\left[\frac{5}{15}H(D_1)+\frac{5}{15}H(D_2)+\frac{5}{15}H(D_3)\right]$$

$$=0.971-\frac{5}{15}\left[\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)+\left(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}\right)+\left(-\frac{4}{5}\log_2\frac{4}{5}-\frac{1}{5}\log_2\frac{1}{5}\right)\right]=0.083$$

$g(D,A_2)=0.324$

$g(D,A_3)=0.420$　该信息增益最大

$g(D,A_4)=0.363$

A_3作为根节点，将D划分为$D_1(A_3=$是$)$和$D_2(A_3=$否$)$,D_1成为叶结点

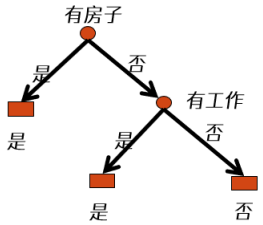
对D_2从特征A_1、A_2、A_4中选择特征

$g(D_2,A_1)=0.251$

$g(D_2,A_2)=0.918$　信息增益最大

$g(D_2,A_4)=0.474$

选取A_2作为节点的特征，根据 其两个取值，可以得到两个子节点，一个对应 “有工作”，并且是一 个叶节点，标记类别为“是”。另 一个节点对应“无工作 ”，且样本属于同一类，也是 一个叶节点，标记类别 为“否”



2. C4.5算法

信息增益比：$g_R(D,A)=\frac{g(D,A)}{H_A(D)}$，其中$H_A(D)=-\sum_{k=1}^{n}\frac{|D_k|}{|D|}log\frac{|D_k|}{|D|}$

C4.5增加了对**连续值**属性的处理，对于连续值属性A，找到一个属性值a_0，将$\leq a_0$的划分到左子树，$>a_0$的划分到右子树

存在的问题：倾向于选择分割**不均匀**的特征
解决办法：先选择n个信息增益大的特征，再从这n个特征中选择信息增益比最大的特征

后来发展到了C5.0

3. 决策树的剪枝

后剪枝（先生成树再剪枝）：为了防止出现过拟合，从已经生成的树上裁掉一些子树或者叶节点，将其父节点作为新的叶节点，用其实例数最大的类别作为标记。

- 当数据量小时，直接利用训练集进行剪枝
- 从下向上逐步剪枝；再验证集上测试性能，直到性能下降位置
- 剪枝的数学模型：当α确定时，选择损失函数最小的模型
  树T的叶节点个数为$|T|$，t是树T的叶节点，该节点有N_t个样本，其中k类的样本点有N_{tk}个（$k=1,\ldots,K$），$H_t(T)$为叶节点t上的经验熵，$a\geq 0$为参数

  $C(T)$表示模型对训练数据的预测误差，$|T|$表示模型的复杂程度

$$\text{损失函数：}\ C_\alpha(T)=\sum_{t=1}^{|T|}N_tH_t(T)+\alpha|T|$$

$$\text{经验熵：}\ H_t(T)=-\sum_k\frac{N_{tk}}{N_t}log\frac{N_{tk}}{N_t}$$

$$\text{记：}\ C(T)=\sum_{t=1}^{|T|}N_tH_t(T)=-\sum_{t=1}^{|T|}\sum_{k=1}^{K}N_{tk}log\frac{N_{tk}}{N_t}$$

$$\text{有：}\ C_\alpha(T)=C(T)+\alpha|T|$$

通俗理解：每个节点的经验熵×该节点样本个个数+整棵树规模的权重值
方法就是贪心，通过从树叶开始向上回溯，如果说损失函数减小，那么剪枝

4. 随机森林

随机森林是由多个决策树组成的分类器；通过投票机制改善决策树
单个决策树的生成：有放回的数据采样，属性（特征）的采样
集外数据的使用：单个决策树的未用到的数据