6大题（三小题/每题）

线性回归、逻辑回归

损失函数，如何通过梯度下降法求解参数，最优参数表达式（如果存在）

$\theta_{MLE} = argmin_\theta \sum_{i=1}^{n}(y_i - \theta^T x_i)^2$

$\theta_{MLE} = (X^T X)^{-1} X^T Y$

防止逆不存在，解$(X^T X)\theta_{MLE} = X^T Y$方程组

梯度下降：$\theta = \theta - \rho \nabla logL(\theta) = \theta + \rho \frac{1}{n} \sum_{i=1}^{n}(y_i - \theta^T x_i)x_i$

随机梯度下降，每次随机选择一个训练数据。

逻辑回归

$P(y=1|x,w) = \frac{1}{1+e^{-wx}}$

$l(w) = \sum_l y^l lnP(y^l = 1|x^l, w) + (1 - y^l) lnP(y^l = 0|x^l, w)$

$= \sum_l y^l (w_0 + \sum_{i=1}^{n} w_i x_i^l) - ln(1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i^l))$

$w_i = w_i + \eta \sum_l x_i^l (y^l - P(y^l = 1|x^l, w))$

过拟合

什么是过拟合，降低过拟合综合风险的方法

模型对训练集拟合的很好，但在测试集上效果很差。可能是因为模型太复杂

把训练样本自身的一些特点当作了所有潜在样本都会具有的一般性质，

调整模型结构减少特征数量，或者减少参数的取值。获得更多更好的数据。早停，正则化，（Dropout，集成学习）

训练方法

训练集-矫正(验证)集-测试集，留出法（Hold-out method）,Repeated hold-out method,k-fold cross-validation, k-fold cross-validation with validation and test sets, Bootstrap method

留出法：直接将数据划分成训练集和测试集。数据集大的时候使用。通常是1/3测试集，2/3训练集。不平衡的数据可以分层，在每类间划分

Repeated hold-out method：每次迭代，按照比例采样训练集，所有迭代错误率的平均作为错误率。不同的测试集会有重复部分。

k-fold cross-validation：将数据集划分成k等份。每个子集都做一次测试集，其他子集做训练集。误差取平均。通常10折

k-fold cross-validation with validation and test sets：每次另外选择一个子集做验证集

Bootstrap method：自助法。对大小为n的数据集有放回采样n次。训练集使用有放回的采样，没被采样的就是测试集。每个样例有0.632的概率被采样为训练集

$$err = 0.632 \cdot e_{\text{test instances}} + 0.368 \cdot e_{\text{training instances}}$$

- Use test sets and the hold-out method for “large” data;
- Use the cross-validation method for “middle-sized” data;
- Use the leave-one-out and bootstrap methods for small data;
- Don’t use test data for parameter tuning - use separate validation data.

决策树

熵的求解，条件熵的求解，求解对应属性的信息增益，决策树的构建

“信息熵”(information entropy)是度量样本集合纯度最常用的一种指标. 假定当前样本集合 $D$ 中第 $k$ 类样本所占的比例为 $p_k$ $(k = 1, 2, \ldots, |\mathcal{Y}|)$, 则 $D$ 的信息熵定义为

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \,. \tag{4.1}$$

# 信息增益 (ID 3: Iterative Dichotomiser)

假定离散属性 $a$ 有 $V$ 个可能的取值 $\{a^1, a^2, \ldots, a^V\}$, 若使用 $a$ 来对样本集 $D$ 进行划分, 则会产生 $V$ 个分支结点, 其中第 $v$ 个分支结点包含了 $D$ 中所有在属性 $a$ 上取值为 $a^v$ 的样本, 记为 $D^v$. 我们可根据式(4.1) 计算出 $D^v$ 的信息熵, 再考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出用属性 $a$ 对样本集 $D$ 进行划分所获得的“信息增益”(information gain)

$$\mathrm{Gain}(D, a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v) \,. \tag{4.2}$$

一般而言，信息增益越大，则意味着使用属性a来进行划分所获得的“纯度提升”越大。

决策树算法第8行选择属性 $a_* = \arg\max_{a \in A} \mathrm{Gain}(D, a)$.

著名的ID3决策树算法

$$\mathrm{Gain\_ratio}(D, a) = \frac{\mathrm{Gain}(D, a)}{\mathrm{IV}(a)} \,, \tag{4.3}$$

其中

$$\mathrm{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|} \tag{4.4}$$

称为属性 $a$ 的“固有值”(intrinsic value) [Quinlan, 1993]. 属性 $a$ 的可能取值数目越多(即 $V$ 越大), 则 $\mathrm{IV}(a)$ 的值通常会越大. 例如, 对表 4.1 的西

## 基尼值

$$\begin{aligned}\mathrm{Gini}(D) &= \sum_{k=1}^{|\mathcal{Y}|}\sum_{k'\neq k} p_k p_{k'} \\ &= 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 \ . \end{aligned} \tag{4.5}$$

直观来说, $\mathrm{Gini}(D)$ 反映了从数据集 $D$ 中随机抽取两个样本, 其类别标记不一致的概率. 因此, $\mathrm{Gini}(D)$ 越小, 则数据集 $D$ 的纯度越高.

## 基尼指数

$$\mathrm{Gini\_index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Gini}(D^v) \ . \tag{4.6}$$

于是, 我们在候选属性集合 $A$ 中, 选择那个使得划分后基尼指数最小的属性作为最优划分属性, 即 $a_* = \arg\min_{a\in A} \mathrm{Gini\_index}(D,a)$.

著名的CART决策树算法

Hunt算法:

**输入:** 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
属性集 $A = \{a_1, a_2, \ldots, a_d\}$.

**过程:** 函数 TreeGenerate($D$, $A$)

1: 生成结点 node;
2: **if** $D$ 中样本全属于同一类别 $C$ **then**
3: 将 node 标记为 $C$ 类叶结点; **return** 无需划分
4: **end if**
5: **if** $A = \varnothing$ **OR** $D$ 中样本在 $A$ 上取值相同 **then**
6: 将 node 标记为叶结点, 其类别标记为 $D$ 中样本数最多的类; **return**
7: **end if** 无法划分
8: 从 $A$ 中选择最优划分属性 $a_*$;
9: **for** $a_*$ 的每一个值 $a_*^v$ **do**
10: 为 node 生成一个分支; 令 $D_v$ 表示 $D$ 中在 $a_*$ 上取值为 $a_*^v$ 的样本子集;
11: **if** $D_v$ 为空 **then**
12: 将分支结点标记为叶结点, 其类别标记为 $D$ 中样本最多的类; **return**
13: **else**
14: 以 TreeGenerate($D_v$, $A \setminus \{a_*\}$)为分支结点 不能划分
15: **end if**
16: **end for**

**输出:** 以 node 为根结点的一棵决策树

**图 4.2** 决策树学习基本算法

SVM的基本思想，SVM的损失函数，SVM中的核函数

选择margin最大的分类器。使得距离分类边界最近的点离分类边界尽可能原，同时保证分类正确。

距离划分数据的超平面最近的点就是支持向量。

目标函数 $max_{w,b}\ min_{1\le i\le n}\frac{|f(x_i)|}{||w||}, s.t.\ y_i f(x_i)>0, 1\le i\le n$

可以写成$max_{w,b} min_{1\le i\le n}\ y_i f(x_i)\ \ s.t. y_i f(x_i)>0 1\le i\le n, w^T w=1$

变成 $max_{w,b}\frac{1}{||w||} s.t.\ y_i f(x_i)\ge 1,\ 1\le i\le n$

最后转化为$min_{w,b}\ \frac{1}{2}w^T w\quad s.t.\ \ y_i f(x_i)\ge 1, 1\le i\le n$

KTT条件$a_i\ge 0\ \ 1-y_i f(x_i)\le 0\quad a_i(1-y_i f(x_i))=0$

所以最终模型仅与支持向量有关,$a_i$非零

$w=\sum_{i=1}^{n} a_i y_i x_i$，解对偶形式获得最优的拉格朗日乘子a，然后获得w

b则将支持向量带入，然后可以直接解出，通常取所有支持向量计算的b的均值

核函数可以将数据点映射到高维空间，注意到对偶问题中训练数据都是以点积的形式成对出现，所以不用直接将数据点映射到高维空间，只需要定义高维空间中的点积形式即可（相似性度量）。使用时将优化问题中的点积替换为核函数即可

$K(x_i, x_j)$半正定,优化问题可以在多项式时间内解决。

线性核、RBF核、多项式核

最近的点可以认为在 wx+b = 1上，因为如果右边是其他数，可以两边同时放缩相同倍数，使得右边变成1.

PCA

PCA的基本思想，PCA的不同理解角度（重构误差最小，方差最大，奇异值分解），PCA的推导过程（ppt）

基本思想：高位数据经常可以使用一个低维表示来近似,即存在一个内在维度，可以保存数据的大部分信息。

最小化重构误差：

$\hat{x}^{=}\sum_{i=1}^{M} a_i u_i$

$E=\frac{1}{N}\sum_{n=1}^{N}|(x^n-\overline{x})-\hat{x}^n|^2$

$a_i=u_i^T(x-\overline{x})$

聚类

K-Means聚类的思想、步骤、目标函数，K的选取问题（损失函数，初始状态），高斯混合聚类算法的思想和步骤（公式要掌握）（EM）

是一种无监督的方法，将距离较近的数据划分到一个集合中。先随机选取一些簇中心，然后所有数据点根据距离簇中心的距离划分为不同类别，然后更新各个类别的簇中心为该类别所有点的中心，重复这个过程。

## Improving a suboptimal configuration...

$$\text{Distortion} = \sum_{i=1}^{R} (\mathbf{x}_i - \mathbf{c}_{\text{ENCODE}(\mathbf{x}_i)})^2$$

What properties can be changed for centers $\boldsymbol{c}_1$ , $\boldsymbol{c}_2$ , … , $\boldsymbol{c}_k$ have when distortion is not minimized?

(1) Change encoding so that $\boldsymbol{x}_i$ is encoded by its nearest center

(2) Set each Center to the centroid of points it owns.

There's no point applying either operation twice in succession.

But it can be profitable to alternate.

...And that's K-means!

*Easy to prove this procedure will terminate in a state at which neither (1) or (2) change the configuration. Why?*

Define...

$$\text{Distortion} = \sum_{i=1}^{R} \left(\mathbf{x}_i - \text{DECODE}[\text{ENCODE}(\mathbf{x}_i)]\right)^2$$

We may as well write

$$\text{DECODE}[j] = \mathbf{c}_j$$

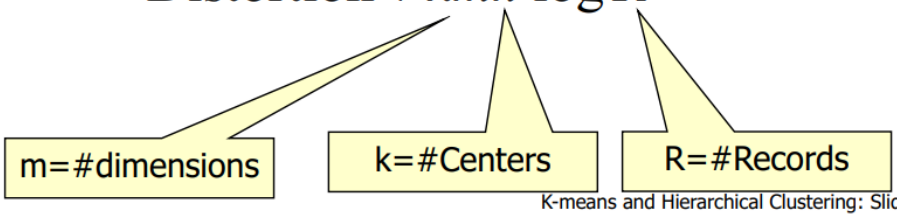
$$\text{so} \quad \text{Distortion} = \sum_{i=1}^{R} (\mathbf{x}_i - \mathbf{c}_{\text{ENCODE}(\mathbf{x}_i)})^2$$

# Choosing the number of Centers

- A difficult problem
- Most common approach is to try to find the solution that minimizes the Schwarz Criterion (also related to the BIC, schwarz's bayesian criterion (bic))

$$\text{Distortion} + \lambda\,(\#\text{parameters})\log R$$

$$= \text{Distortion} + \lambda mk \log R$$






---

EM算法

EM算法中的E步和M步（高斯混合算法GMM）

- Algorithm

E-step: Evaluating the expectation

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{(t)})}[\log p(\boldsymbol{x},\ \boldsymbol{z}; \boldsymbol{\theta})]$$

M-step: Updating the parameter

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$$

推荐系统

基于矩阵分解的推荐系统，基于用户的协同推荐，基于商品的协同推荐，冷启动问题（基于内容的推荐），数据稀疏（递归协同过滤）的问题。

## Collaborative Filtering (CF)

- **The most prominent approach to generate recommendations**
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)

- **Approach**
  - use the "wisdom of the crowd" to recommend items
- **Basic assumption and idea**
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future



## User-based nearest-neighbor collaborative filtering (1)

- **The basic technique:**
  - Given an "active user" (Alice) and an item I not yet seen by Alice
  - The *goal is to estimate Alice's rating for this item*, e.g., by
    - find a set of users (peers) who liked the same items as Alice in the past **and** who have rated item I
    - use, e.g. the average of their ratings to predict, if Alice will like item I
    - do this for all items Alice has not seen and recommend the best-rated

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

## Measuring user similarity

- **A popular similarity measure in user-based CF: Pearson correlation**

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\ \sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

a, b : users

$r_{a,p}$ : rating of user a for item p

P : set of items, rated both by a and b

Possible similarity values between -1 and 1; $\overline{r_a}$, $\overline{r_b}$ = user's average ratings

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

sim = 0,85
sim = 0,70
sim = -0,79



## Making predictions

- A common prediction function:

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

- Calculate, whether the neighbors' ratings for the unseen item ***i*** are higher or lower than their average
- Combine the rating differences – use the similarity as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

## Item-based collaborative filtering

- **Basic idea:**
  - Use the similarity between items (and not users) to make predictions
- **Example:**
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

## The cosine similarity measure

- **Produces better results in item-to-item filtering**
  - for some datasets, no consistent picture in literature
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

- **Adjusted cosine similarity**
  - take average user ratings into account, transform the original ratings

$$sim(a, b) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}$$

## Data sparsity problems

- **Cold start problem**
  - How to recommend new items? What to recommend to new users?
- **Straightforward approaches**
  - Ask/force users to rate a set of items
  - Use another method (e.g., content-based, demographic or simply non-personalized) in the initial phase
- **Alternatives**
  - Use better algorithms (beyond nearest-neighbor approaches)
  - Example:
    - In nearest-neighbor approaches, the set of sufficiently similar neighbors might be to small to make good predictions
    - Assume "transitivity" of neighborhoods