《模式识别》

第一章 课程简介与预备知识

马锦华

https://cse.sysu.edu.cn/teacher/MaJinhua

SUN YAT-SEN University

中山大學 计算机学院
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING, SYSU

课程目录（暂定）

第一章	课程简介与预备知识	6学时
第二章	特征提取与表示	6学时
第三章	主成分分析	3学时
第四章	归一化、判别分析、人脸识别	3学时
第五章	EM算法与聚类	3学时
第六章	贝叶斯决策理论	3学时
第七章	线性分类器与感知机	3学时
第八章	支持向量机	3学时
第九章	神经网络、正则项和优化方法	3学时
第十章	卷积神经网络及经典框架	3学时
第十一章	循环神经网络	3学时
第十二章	Transformer	3学时
第十三章	自监督与半监督学习	3学时
第十四章	开放世界模式识别	6学时

第三部分：模式识别系统与评估

目标

- 理解并能熟练运用最近邻方法进行分类
- 了解最近邻方法的限制、缺陷以及可能的解决办法
- 理解并掌握模式识别系统各模块的作用、基本概念和解决方案

- 提高目标
  - 进一步能将最近邻方法应用到实际问题中（研究生、部分本科生）

最近邻规则

Nearest neighbor rule

问题设置problem setup

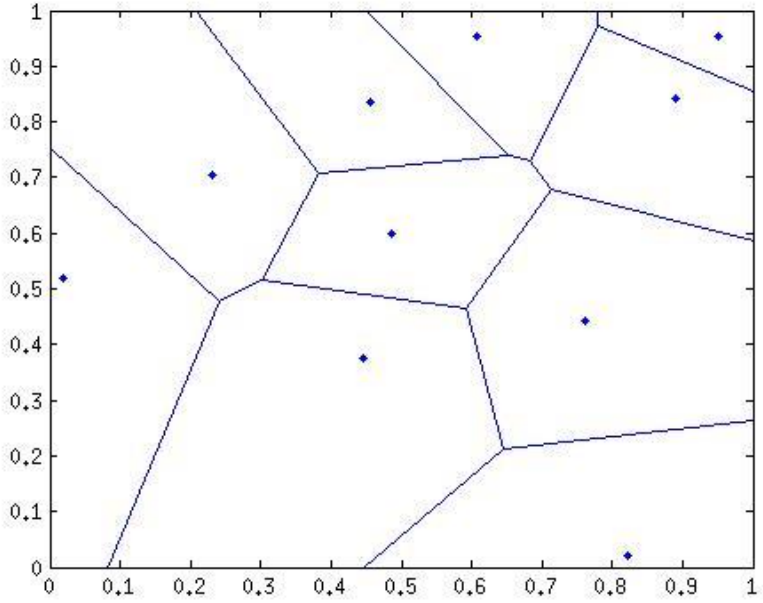
- 分类问题classification
  - 训练集$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)\}$
  - 训练样本(sample)：$\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$
  - 样本的标记(label)：$y_i \in \mathcal{Y} = \{1,2, \dots, C\}$
    - 样本一共被分为C个类别(category)
    - 例如，在性别识别的例子里，$C = 2, y_i = 1$（男）或者$y_i = 0$（女）

- 存在一个距离(distance)函数：$d(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^+$
  - 能够度量$\boldsymbol{x}$和$\boldsymbol{y}$之间的距离，或者不相似程度(level of dissimilarity)

最近邻规则和Voronoi图

给定一个测试样例$\boldsymbol{x}$

1. 发现其最近邻$i^* = \underset{i}{\operatorname{argmin}}\, d(\boldsymbol{x}, \boldsymbol{x}_i)$
2. 输出对$\boldsymbol{x}$的分类的预测：y_{i^*}



Voronoi网格
(Voronoi Diagram)

Nearest Neighbor Classifier

```
def train(images, labels):
  # Machine learning!
  return model
```

Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Predict the label of the most similar training image

Nearest Neighbor Classifier

deer *bird* *plane* *cat* *car*

Training data with labels

?

query data

Distance Metric $\left| \quad , \quad \right| \rightarrow \mathbb{R}$

Distance Metric to compare images

L1 distance: $$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image					training image				pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	46	12	14	1	add → 456
90	23	128	133		8	10	89	100	82	13	39	33	
24	26	178	200		12	16	178	170	12	10	0	30	
2	0	255	220		4	32	233	112	2	32	22	108	

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

Memorize training data

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

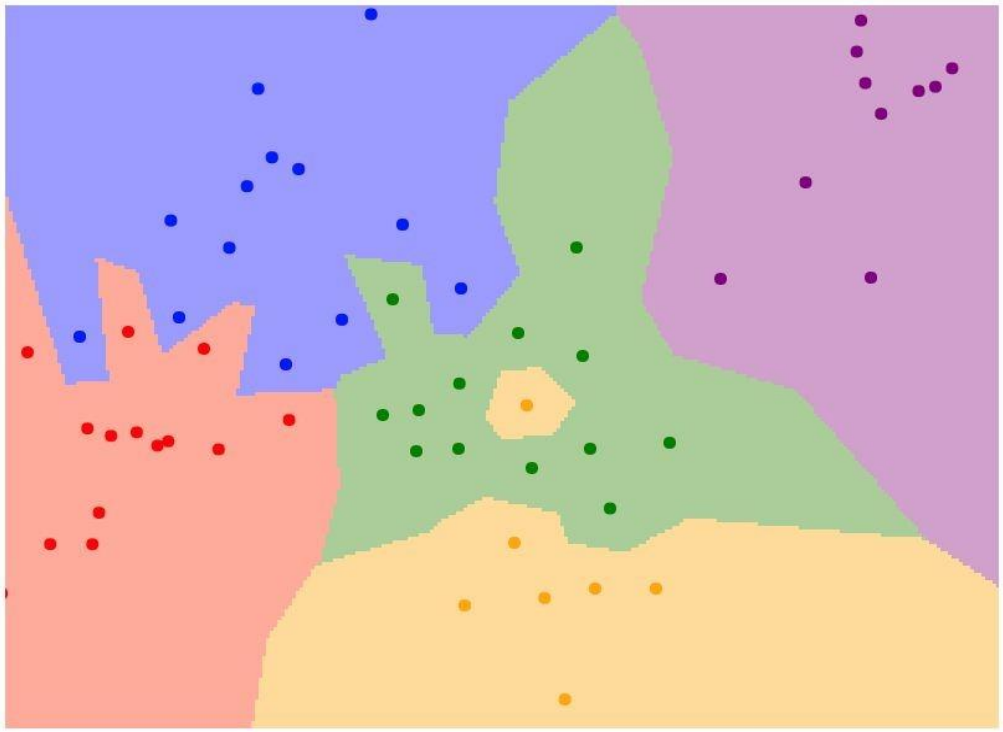
Nearest Neighbor classifier

For each test image:
Find closest train image
Predict label of nearest image

最近邻可能出现的问题

- 如果出现平局(tie)？
  - $d(\boldsymbol{x}, \boldsymbol{x}_i) = d(\boldsymbol{x}, \boldsymbol{x}_j)$
  - $y_i = y_j$？ $y_i \neq y_j$？
- 如果出现离群点(outlier)？
  - k-近邻(kNN，k-nearest neighbor)规则
  - 可能遇到的问题？
- 能做的多好？
  - 当训练样本趋于无穷时($n \to \infty$)，最近邻的错误率R最多是最小（贝叶斯）错误率R^*的两倍，$R^* \leq R \leq 2R^*$
    - 严格证明参考Duda《模式分类》第4章
    - 简化讨论参考周志华《机器学习》第10章
  - 有限样本(finite sample)时的结论尚不清楚

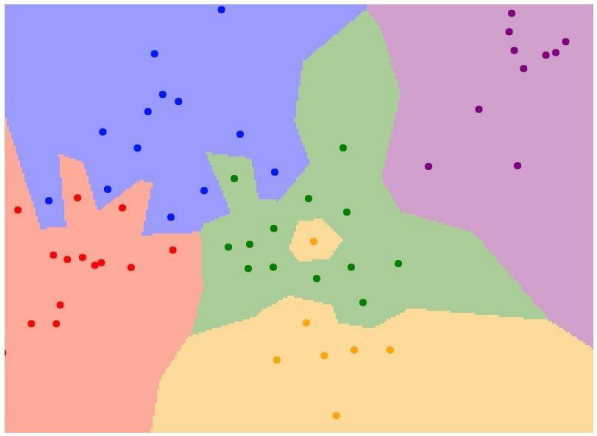
What does this look like?



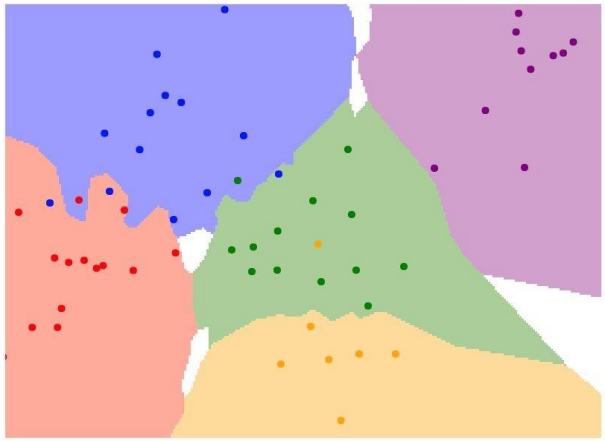
1-nearest neighbor

K-Nearest Neighbors

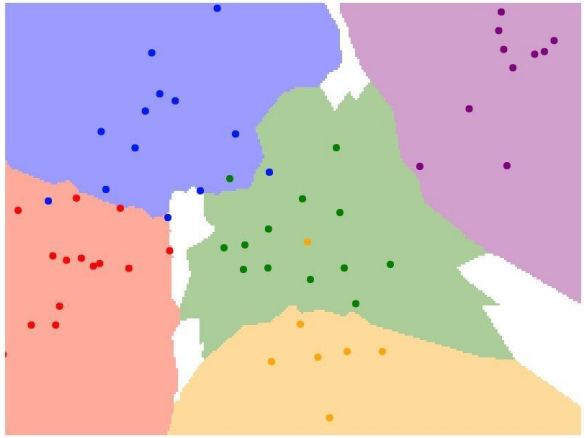
Instead of copying label from nearest neighbor, take **majority vote** *from K closest points*



K = 1



K = 3

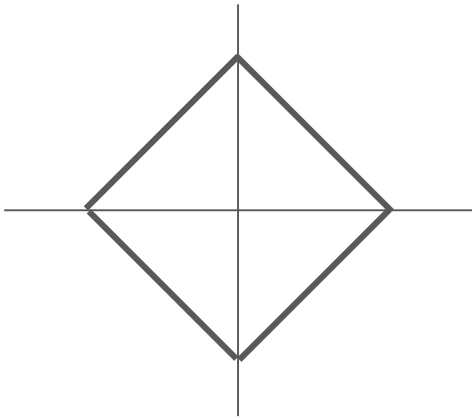


K = 5

K-Nearest Neighbors: Distance Metric

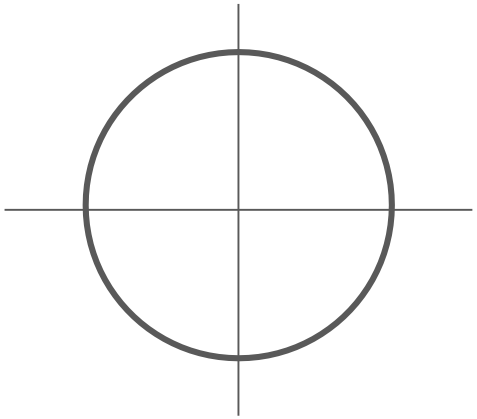
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

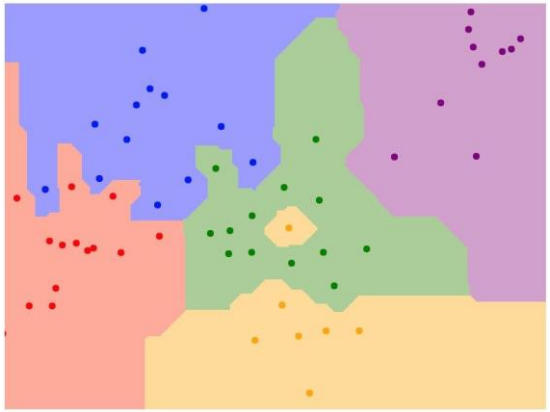
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

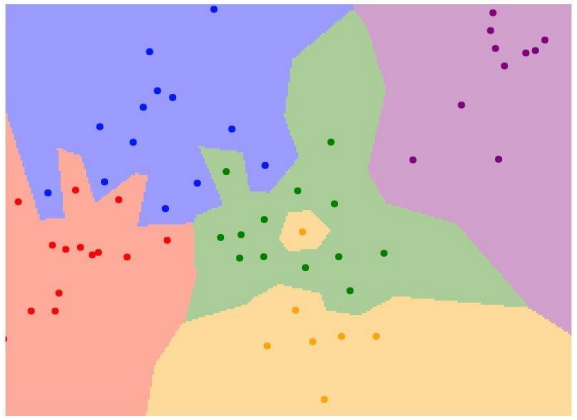
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



K = 1

计算、存储代价(cost)

- 假设$d(\boldsymbol{x}, \boldsymbol{y})$是曼哈顿距离(Manhattan distance, ℓ_1 distance)或欧氏距离(Euclidean distance, ℓ_2 distance)
  - 其复杂度(complexity)是$O(d)$
- NN的训练复杂度和测试复杂度分别是
  - 训练：$O(nd)$，测试：$O(nd)$
- k-NN的复杂度同样是$O(nd)$
  - 或者$O(nd) + O(nk)$，通常k较小，可以忽略
  - 从n个数（距离）中选择k个最小的，复杂度是？
- 考虑一下，如果是ImageNet，需要多长时间，多大的存储空间？这是NN的主要问题
  - $n \approx 1{,}400{,}000$
  - $d = 384 \times 384 \times 3 = 442{,}368$ (e.g. ViT)

降低NN的计算、存储代价

- 近似最近邻, $(1+\epsilon)$ approximate nearest neighbor
  - 不要求一定是距离最短的k个
  - 如第k个NN，其距离是d_k，则近似NN要求其选取的所有k个样例的距离$\hat{d}$满足$\hat{d} \leq (1+\epsilon)d_k$即可
  - 可以将kNN搜索(search、查找)速度提高几个数量级
- 二值哈希(binary hashing)
  - hash函数f_i: 将$\mathbb{R}^d$分为两部分，分别用$f_i = 0$，1表示
  - 设计m个hash函数$f_1, \ldots, f_m$，每个$\boldsymbol{x}$表示为m个bit
  - $m \ll d$，计算和存储大幅简化，需要设计好的hash
    - 参考Rajaraman《Mining of Massive Datasets》
  - 进一步：基于深度学习的hashing，GPU加速 https://github.com/facebookresearch/faiss

k-Nearest Neighbor with pixel distance **never used**

- Distance metrics on pixels are not informative.

Original *Occluded* *Shifted (1 pixel)* *Tinted*

Original image is CC0 public domain

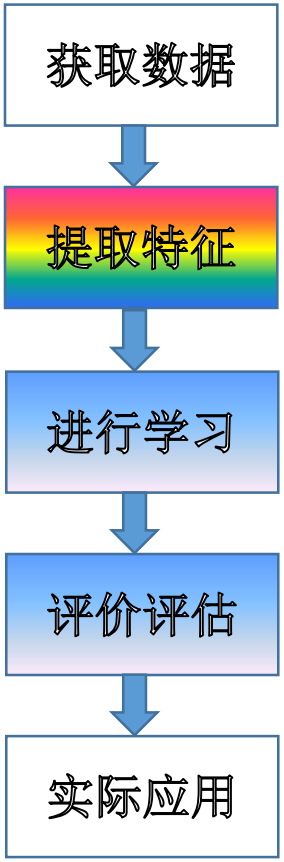
(All three images on the right have the same pixel distances to the one on the left.)

系统各模块(混合)简介

Introducing various components in a mixed order

细化(refined)的框架

- 机器学习$f: \mathcal{X} \mapsto \mathcal{Y}$
  1. 与领域无关的特征变换和特征抽取
     - Normalization, PCA, FLD,…
  2. 针对不同数据特点的不同学习算法
     - SVM, Decision Tree, imbalanced learning, HMM, DTW, graphical model, deep learning, pLSA, …
  3. 机器学习方法常见分类、策略
     - 分类：按范式可分为监督、无监督、半监督、强化学习等；按模型类型可分为生成/判别、参数/非参数等
     - 策略：涵盖损失函数设计、优化算法、正则化、评估方法、集成学习等，需结合任务特点灵活选择
- 针对不同问题的评价准则(evaluation criterion)



评价准则：泛化和测试误差

- 暂时只考虑分类问题的评价
- 假设$(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)$
  - 泛化误差generalization error：$\mathrm{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}[f(\boldsymbol{x}) \neq y]$
    - 通常无法实际计算
  - 基本假设（实际问题中不一定成立）：训练集D_{train}和测试集D_{test}都是服从真实数据分布$p(\boldsymbol{x}, y)$的，或者，他们的样例是从$p(\boldsymbol{x}, y)$中取样（sample）的
  - 独立同分布（i.i.d.）采样假设下的测试误差(testing error)

$$err = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(f(\boldsymbol{x}_i) \neq y_i), \qquad \boldsymbol{x}_i \in D_{test}$$

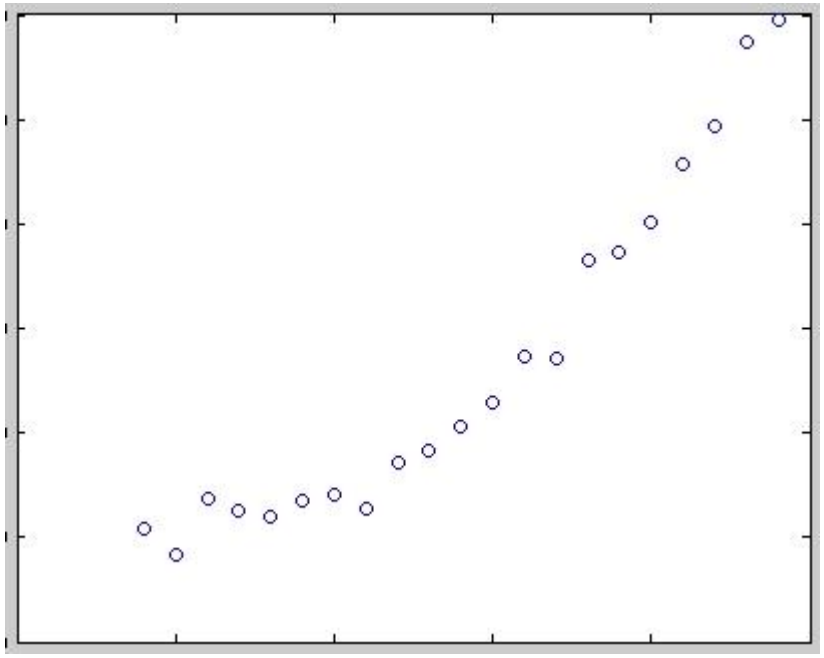
  - 准确率(accuracy)：$acc = 1 - err$

一种常见的学习框架

- 代价最小化（cost minimization）
  - 错误是最常考虑的代价，所以现在我们可以说学习的目标是在训练集上获得最小的代价
- $\min_f \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(f(\boldsymbol{x}_i) \neq y_i)$， $\boldsymbol{x}_i \in D_{train}$
  - 难以优化 – 怎样求解?
  - 一种方法是：把不连续的指示函数(indicator function)换成性质相似，但容易优化的函数
  - 如，$(f(\boldsymbol{x}_i) - y_i)^2$
- 学习这种**思路**：形式化、简化、优化
  - Formalization, simplification, optimization

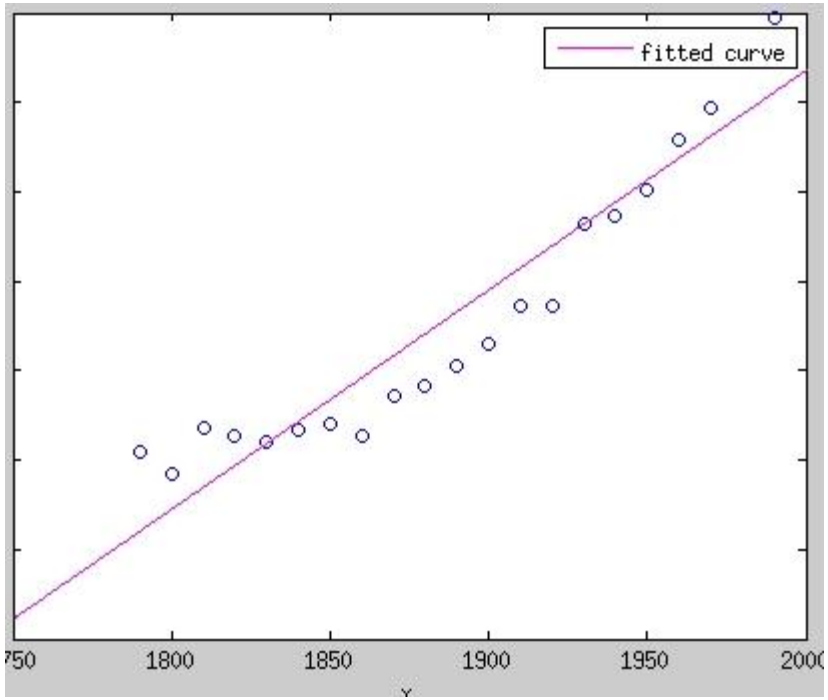
过拟合和欠拟合

- Overfitting & underfitting，以回归(regression)为例



过拟合和欠拟合

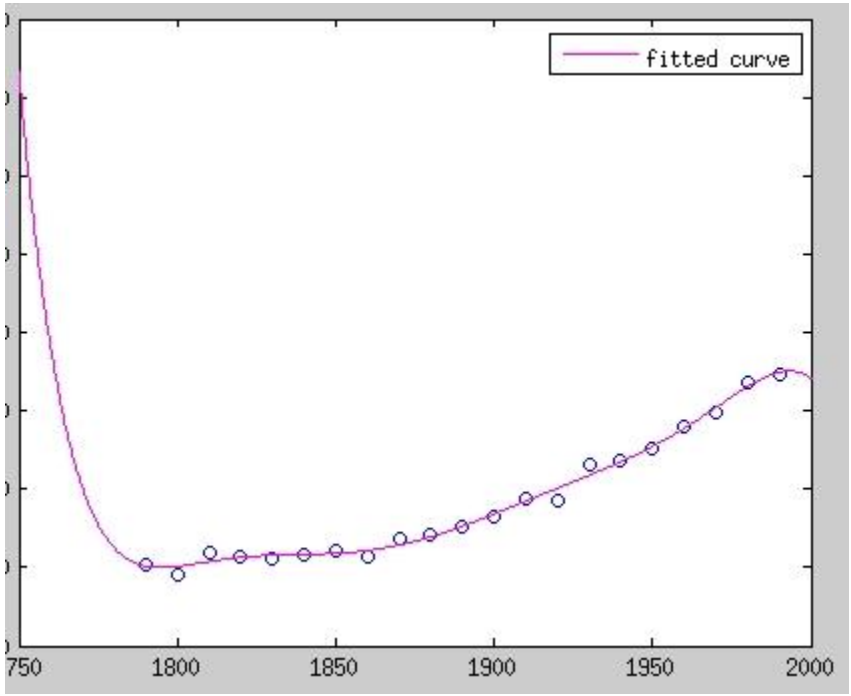
- Overfitting & underfitting，以回归(regression)为例



- 以一阶多项式拟合（直线）
- 学习模型的复杂性小于数据的复杂性
- 称为欠拟合 underfitting

过拟合和欠拟合

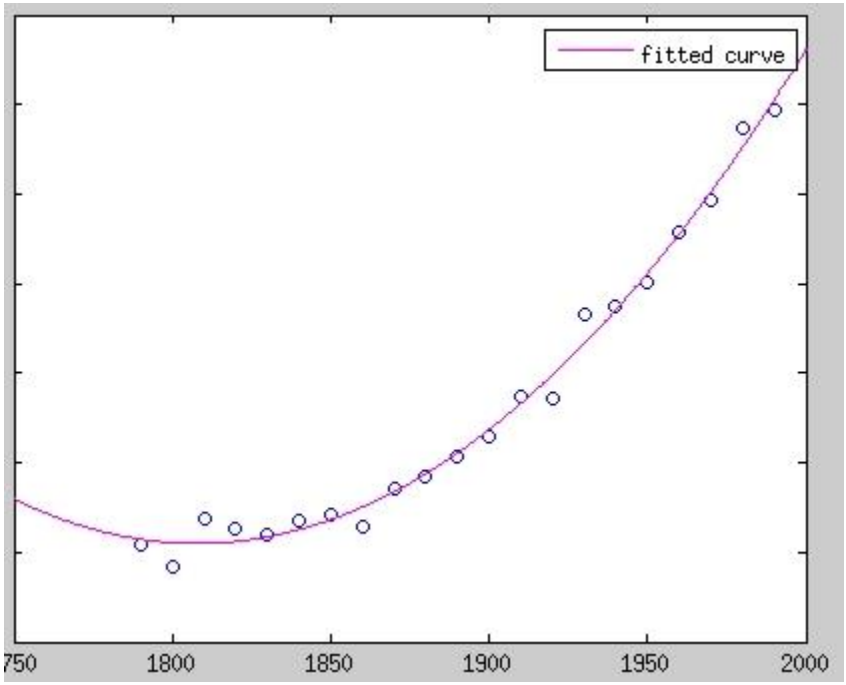
- Overfitting & underfitting，以回归(regression)为例



- 以七阶多项式拟合（直线）
- 学习模型的复杂性大于数据的复杂性
- 称为过拟合 overfitting

过拟合和欠拟合

- Overfitting & underfitting，以回归(regression)为例



- 以二阶多项式拟合（直线）
- 学习模型的复杂性 适合数据的复杂性
- 效果最佳

正则化regularization

- 通常难以精确估计学习模型、数据的复杂性
  - 往往选用较复杂的学习模型
  - 训练集误差通常小于测试集误差(需要两者不相交)
- 那么如何降低overfitting的可能性呢？
  - 正则化regularization，会在SVM部分看到例子

- 进一步阅读：
  - 正则化如何能降低模型的复杂性？PRML以及ESL(The Elements of Statistical Learning)

如果没有测试集

- 例如，总的数据量比较小（如医学图像）
  - 如何评估？
- 交叉验证cross validation
  1. 将训练集分为大小大致相等的N部分
  2. for $i = 1{:}\,N$
     1. 取第i部分的数据为测试集
     2. 取所有其余（一共$N-1$个部分）的数据为训练集
     3. 学习模型并评估/测试得到错误率为err_i
  3. 交叉验证得到的错误率为$\frac{1}{N}\sum_{i=1}^{N} err_i$
  - 称为N重交叉验证N-fold CV （常用N=5 or 10)
- 训练集的划分存在随机性
  - 可能需要进行多次试验、或使用留一法(leave one out)

如何选择超参数(Hyperparameters)？

- 以kNN为例：

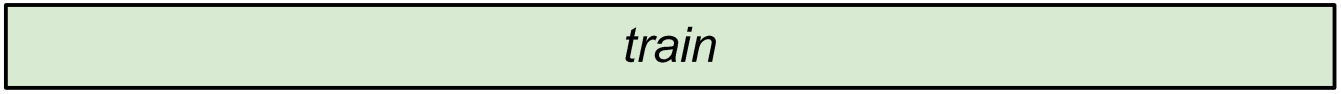
  What is the best value of **k** *to use?*
  What is the best **distance** *to use?*

  These are **hyperparameters***: choices about the algorithms themselves.*

  Very problem/dataset-dependent.
  Must try them all out and see what works best.

Setting Hyperparameters

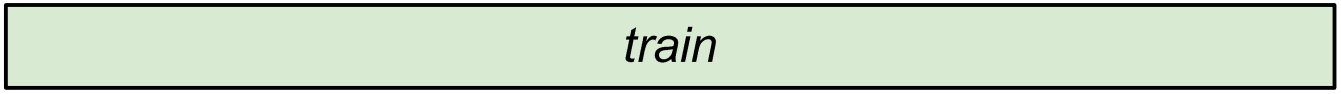
Idea #1: *Choose hyperparameters that work best on the* **training data**



Setting Hyperparameters

Idea #1: *Choose hyperparameters that work best on the* **training data**

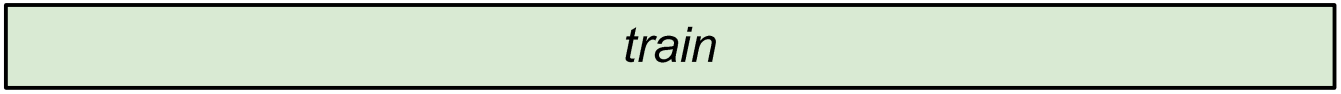
BAD: *K = 1 always works perfectly on training data*



Setting Hyperparameters

Idea #1: *Choose hyperparameters that work best on the* **training data**

BAD: *K = 1 always works perfectly on training data*

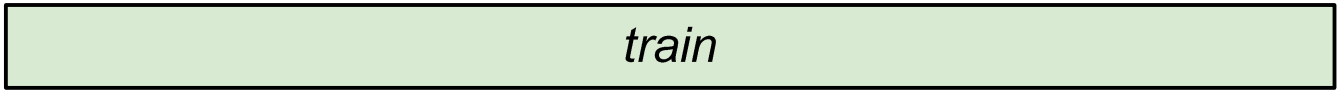


Idea #2: *choose hyperparameters that work best on* **test** *data*

Setting Hyperparameters

Idea #1: *Choose hyperparameters that work best on the* **training data**

BAD: *K = 1 always works perfectly on training data*



Idea #2: *choose hyperparameters that work best on* **test** *data*

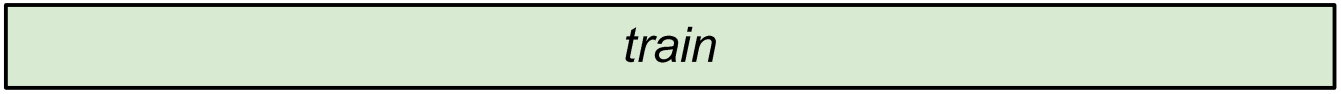
BAD: *No idea how algorithm will perform on new data*

Never do this!

Setting Hyperparameters

Idea #1: *Choose hyperparameters that work best on the* **training data**

BAD: *K = 1 always works perfectly on training data*



Idea #2: *choose hyperparameters that work best on* **test** *data*

BAD: *No idea how algorithm will perform on new data*

Idea #3: *Split data into* **train**, **val**; *choose hyperparameters on val and evaluate on test*

Better!

Setting Hyperparameters

train

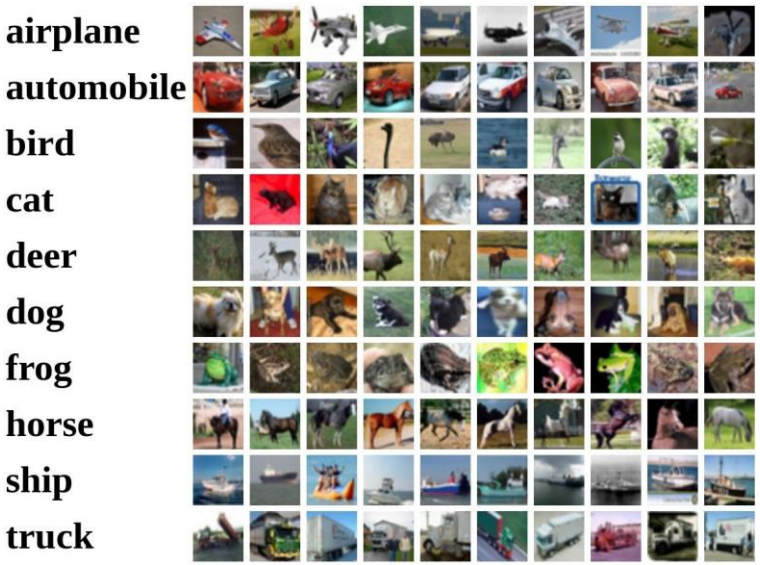
Idea #4: **Cross-Validation**: *Split data into* **folds**, *try each fold as validation and average the results*

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but not used too frequently in deep learning

Example Dataset: **CIFAR10**

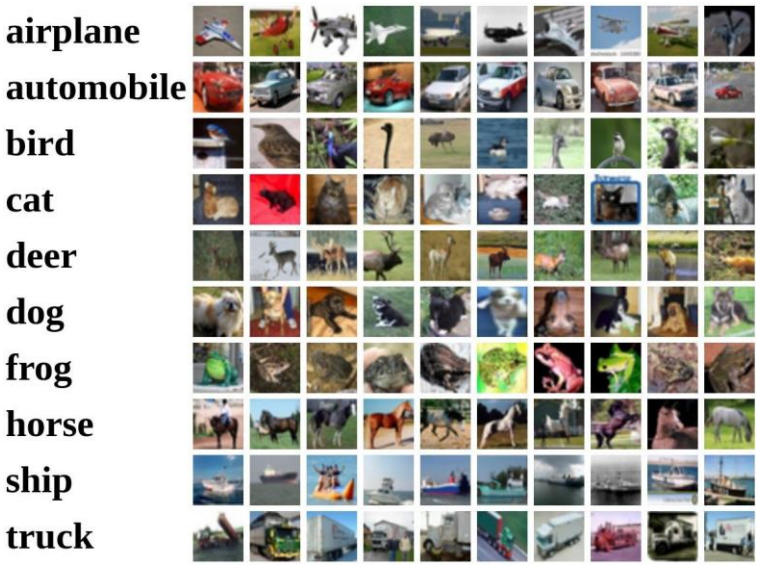
10 *classes*
50,000 *training images*
10,000 *testing images*



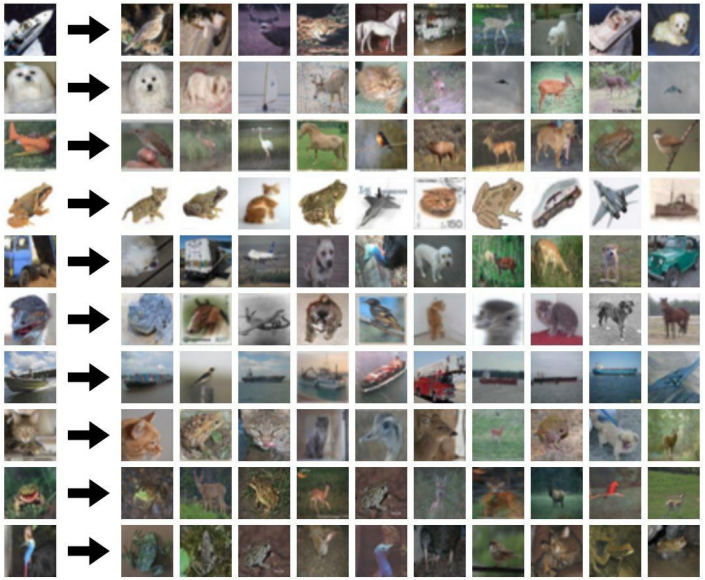
Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Example Dataset: **CIFAR10**

10 *classes*
50,000 *training images*
10,000 *testing images*

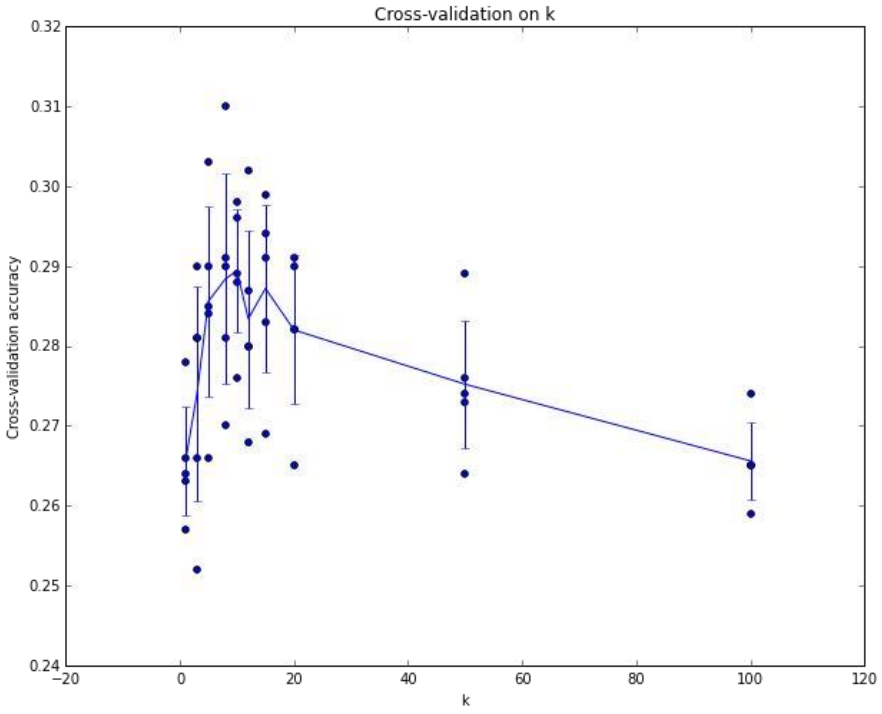


Test images and nearest neighbors



Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Setting Hyperparameters



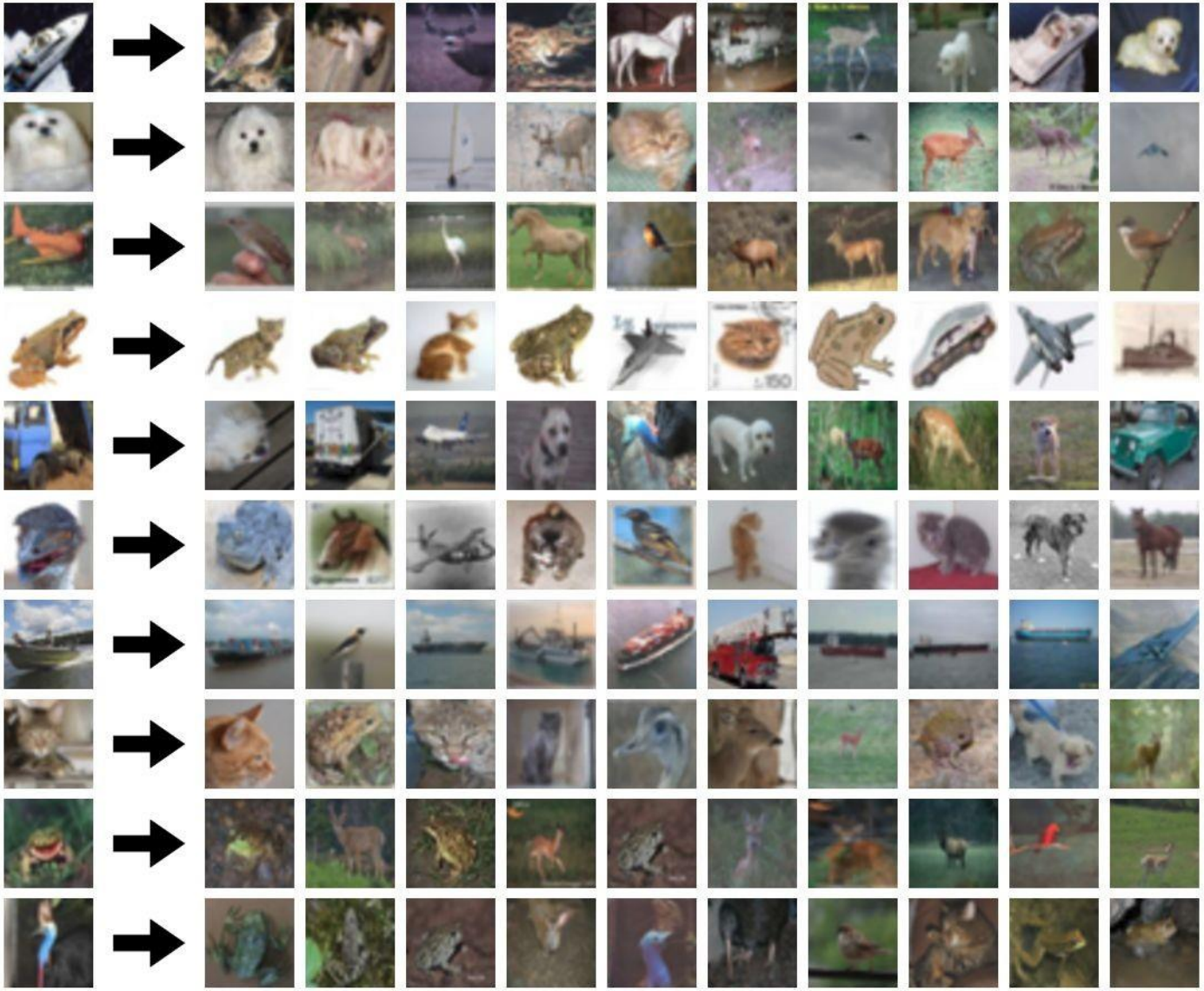
Example of 5-fold cross-validation for the value of k.

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation.

(Seems that k ~= 7 works best for this data)

What does this look like?



What does this look like?

数据、代价的不平衡性

- 例如，两类问题中，一类数据远比另一类数据多
  - 如，体检测试结果中阴性和阳性
  - 或在一类犯错的代价远高于另一类

  - 不平衡学习(imbalance learning)
  - 代价敏感学习(cost-sensitive learning)

  - 进一步阅读：周志华教授主页和论文 http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/publication.htm

不平衡问题的评价准则(1)

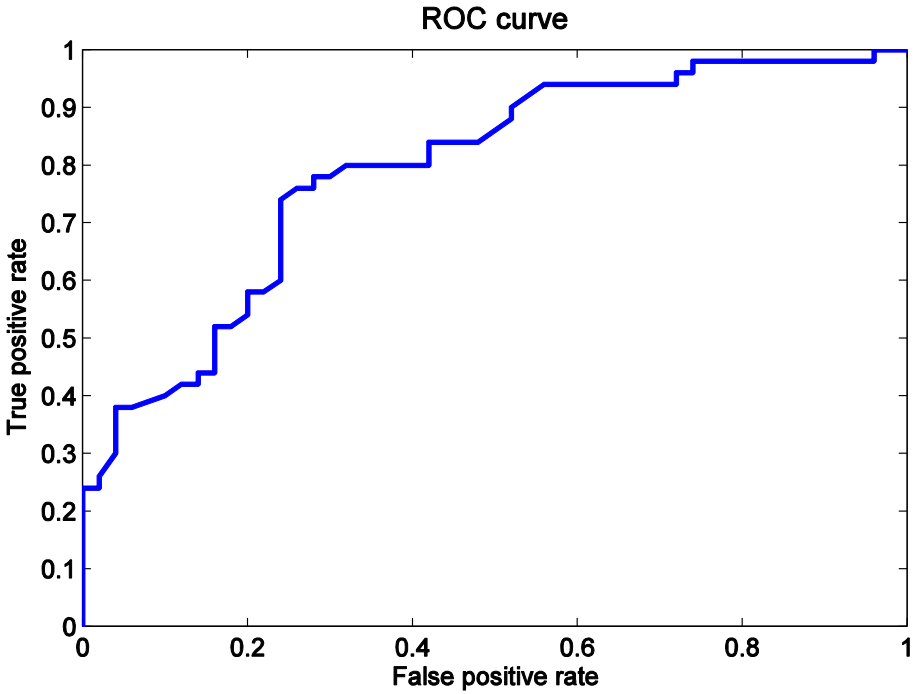
confusion matrix

	预测为positive	预测为negative
真实值为positive	True positive (真阳性)	False negative(伪阴性)
真实值为negative	False positive(伪阳性)	True negative(真阴性)

- ✓ TP、TN、FP、FN：标记四种情况的样例数目
- ✓ TOTAL：总数 TP+TN+FP+FN
  - 正样本数目：P = TP+FN，负样本数目：N = FP+TN
- ✓ False positive rate (type I error): FPR = FP / N
- ✓ False negative rate (miss rate, type II error): FNR = FN / P
- ✓ True positive rate (sensitivity, recall): TPR = TP / P
- ✓ True negative rate (specificity): TNR = TN / N
- ✓ Positive predictive value (precision): PPV = TP / (TP + FP)
- ✓ Accuracy： ACC = (TP+TN) / TOTAL

不平衡问题的评价准则(2)

- AUC-ROC (Area Under the ROC Curve)
  - ROC – Receiver operating characteristic



- Y轴： TPR
- X轴： FPR
- 其值为面积
- 为什么有效？
- 对角线是？
- 非减

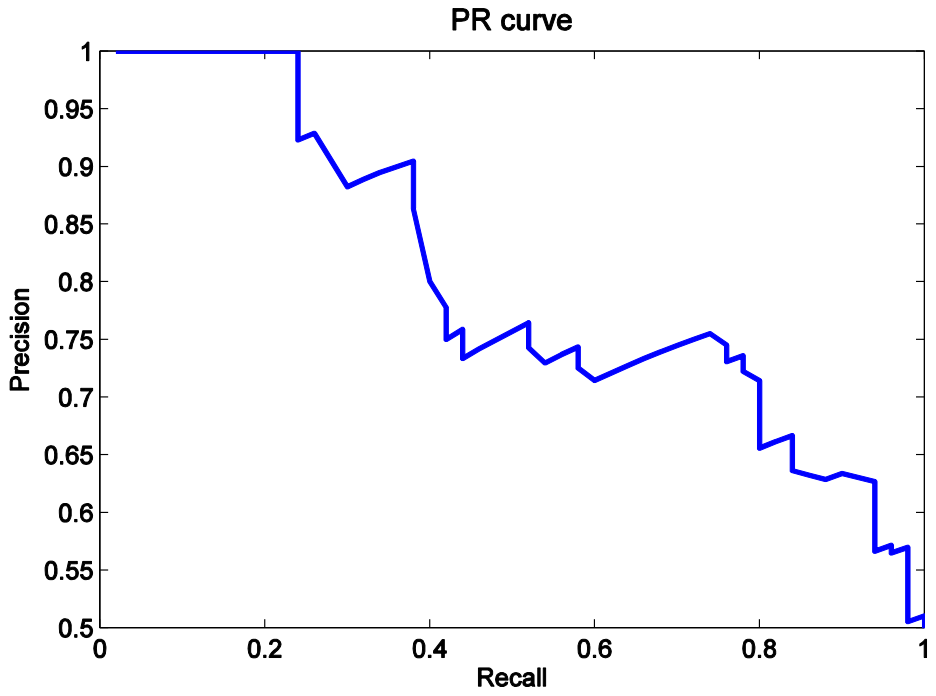
假设分类器$f(\boldsymbol{x}) = 2(\mathbb{I}(g(\boldsymbol{x}) > b) - 0.5)$

不平衡问题的评价准则(3)

- Precision（查准率）：TP / (TP + FP)
- Recall（查全率）：TP / P (和TPR一样）
- F1 score：Precision和Recall的调和平均(harmonic mean)
  - 调和平均：$\left(\frac{x^{-1}+y^{-1}}{2}\right)^{-1} = \frac{2xy}{x+y}$
  - F1 = 2TP / (2TP+FP+FN)
    - 推导一下
  - 为什么用调和平均而不是算术平均？
    - 当precision（recall）很高但是recall（precision）很低的时候，F1 score会很低

不平衡问题的评价准则(4)

- AUC-PR (Area Under the Precision-Recall Curve)



- Y轴： Precision
- X轴： Recall
- 其值为面积
- 为什么有效？
- 单调吗？
- 与AP的区别？

进一步阅读: The Relationship Between Precision-Recall and ROC Curves

代价矩阵

- 目前常见的为$\begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
  - λ_{ij}: 当真实值为i、模型预测为j时付的代价
  - 0-1代价：即分类正确代价为0，分类错误代价为1
  - 但是，根据实际情况，可以给λ_{ij}设置任何值
    - 代价不平衡学习

- 对代价的计算：$\mathrm{E}_{(\boldsymbol{x},y)}\left[\lambda_{y,f(\boldsymbol{x})}\right]$
  - 当使用0-1代价时，和错误率一致

真实值Groundtruth

- 大多数时候，是手工标注的(manual annotation)
  - 或者人也不知道确切的答案
  - 有时候疲劳或其他因素会导致标注的错误
  - 很耗时、昂贵

- 真实值的形式
  - 分类：一个离散的类别
  - 回归regression：一个连续的值
  - 结构structured output：例如，输出一个句子的分词结果“一个/句子/的/分词/结果”

Bayes误差率(1)

- 分类$f: \mathcal{X} \mapsto \mathcal{Y}$、样本对(sample pair): $(\boldsymbol{x}, y)$
  - 假设注重于分类： $\mathcal{Y} = \{1,2,\ldots,m\}$
  - 先验概率prior probability： $p(y = i)$
    - 在没有看到任何数据时，怎么分类？
  - 后验概率posterior probability: $p(y = i|\boldsymbol{x})$
    - 看到数据$\boldsymbol{x}$后，得到更多的信息，可以对分类有更好的估计
  - 类别条件概率class conditional probability: $p(\boldsymbol{x}|y = i)$
    - 数据的整体分布$p(\boldsymbol{x})$和每个类别内部的分布$p(\boldsymbol{x}|y = i)$不一样
- 贝叶斯定理Bayes' theorem

$$p(y = i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = i)p(y = i)}{p(\boldsymbol{x})} = \frac{\text{条件} \times \text{先验}}{\text{数据}}$$

Bayes误差率(2)

- 贝叶斯决策规则（Bayes decision rule）：
  - 对某一$\boldsymbol{x}$选择代价最小的类别输出

$$y^* = \underset{f(\boldsymbol{x})}{\operatorname{argmin}} \mathrm{E}_{p(y|\boldsymbol{x})}\left[\lambda_{y,f(\boldsymbol{x})}\right]$$

  - 贝叶斯风险(Bayes risk)：使用贝叶斯决策规则的风险
  - 其是理论上我们能得到的最好的结果，记为R^*
- 在使用0-1风险时，风险和错误率等价
  - 所以，R^*是我们理论上能得到的最小误差率
  - $1-R^*$是理论上最高的准确率！

- 进一步阅读：Duda《模式分类》Ch2.1、Ch2.2（包括似然比准则likelihood ratio rule）

错误从哪里来—以回归为例？

- 真实（但未知）的分类函数$F(\boldsymbol{x})$
  - 用由F产生的数据集D来学习，即$y = F(\boldsymbol{x})$没有误差
  - 平方误差（欧几里得距离）作为代价函数
- $\mathrm{E}_D[(f(\boldsymbol{x};D) - y)^2] = (\mathrm{E}_D[f(\boldsymbol{x};D)] - y)^2 + \mathrm{E}_D[(f(\boldsymbol{x};D) - \mathrm{E}_D[f(\boldsymbol{x};D)])^2]$
  - $\boldsymbol{x}$和y不包含随机性，只有D出现时才取期望
  - 简写为$E[(f - F)^2] = (F - E[f])^2 + E[(f - E[f])^2]$

偏置-方差分解

- Bias-variance decomposition
  - $E[(f-F)^2]=(F-E[f])^2+E[(f-E[f])^2]$
  - $E[F-E(f)]=F-E(f)$ -- 偏置（偏差，bias）
    - 取决于模型，当训练集取样有差异时，其值不变
  - $E[(f-E[f])^2]=\mathrm{Var}_D(f(\boldsymbol{x};D))$ --方差
    - 取决于模型和训练数据，当训练集取样有差异时，会带来预测的差异（误差不同）
- 误差=偏置^2+方差
- 考虑到样本标签y和真实标签$F(\boldsymbol{x})$可能有误差（白噪声）
  - 误差=偏置^2+方差+噪声
  - 估计误差时，如没有测试集，需多次平均（减小方差）

对分解的解读

- 偏置与数据无关，是由模型（的复杂度）决定的
  - 例如，线性分类器（1阶多项式）的偏置大
  - 但是，7阶多项式的复杂度高，偏置小
- 但是，方差$\mathrm{Var}_D(f(\boldsymbol{x};D))$和抽样得到的训练集以及模型两者都有关系
  - 例如，高阶多项式的方差大
- 怎么减少误差？
  - 对于样本标签噪声，机器学习没有办法—高质量的数据获取！
  - 减少偏置和方差
    - 如集成方法(ensemble methods)

统计测试

- 可以决定我们对于评估的结果是否有信心
  - 自学吴建鑫《模式识别》第4.5节
  - 目标是理解统计假设检验的基本思想
  - 知识要点
    - 怎么理解置信度（吴建鑫《模式识别》第4.5.2节最后部分）
    - T检验的实际操作流程
    - 因假设“概率与数理统计”是前置课程，吴建鑫《模式识别》第4.5节只是一个简要回顾

进一步的阅读

- FLANN：https://github.com/flann-lib/flann
  - 相关软件、文档
- Hash： http://en.wikipedia.org/wiki/Locality-sensitive_hashing 及其页面中的资源
- 其他见各页面的进一步阅读资源