Supervised Learning

吉建民

USTC
jianmin@ustc.edu.cn

2024 年 5 月 7 日

Used Materials

Disclaimer: 本课件采用了 S. Russell and P. Norvig's Artificial Intelligence –A modern approach slides, 徐林莉老师课件和其他网络课程课件，也采用了 GitHub 中开源代码，以及部分网络博客内容

Table of Contents

Supervised Learning

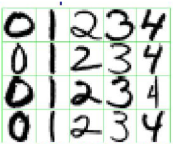
- Learning Decision Trees
- K Nearest Neighbor Classifier
- Linear Predictions
- Logistic Regression
- Support Vector Machines

Supervised learning

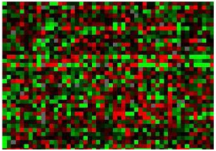
- Input data space $\mathcal{X}$
- Output (label, target) space $\mathcal{Y}$
- Unknown function $f\colon \mathcal{X} \to \mathcal{Y}$
- we are given a set examples $(x_i, y_i), i = 1 ... N$ with $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Finite $\mathcal{Y} \Rightarrow$ classification
- Continuous $\mathcal{Y} \Rightarrow$ regression

Classification (分类)

- We are given a set of N observations $\{(\mathbf{x}_i, y_i)\}_{i=1..N}$
- Need to map $x \in \mathcal{X}$ to a label $y \in \mathcal{Y}$
- Example:



digits recognition;
$\mathcal{Y} = \{0, \dots, 9\}$



prediction from microarray data;
$\mathcal{Y} = \{\text{desease present/absent}\}$

Table of Contents

Supervised Learning

- Learning Decision Trees
- K Nearest Neighbor Classifier
- Linear Predictions
- Logistic Regression
- Support Vector Machines

Learning decision trees

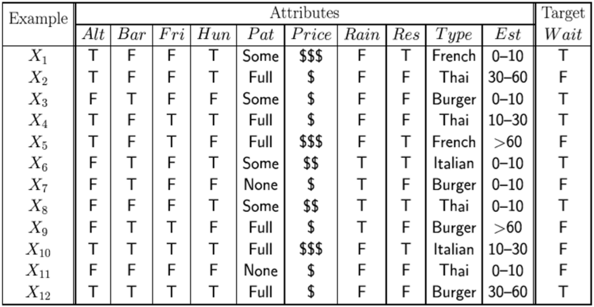
Problem: decide whether to wait for a table at a restaurant, based on the following attributes (属性):

1. Alternate (别的选择): is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons (顾客): number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation (预约): have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

Examples described by attribute values（属性）(Boolean, discrete, continuous)

E.g., situations where I will/won't wait for a table

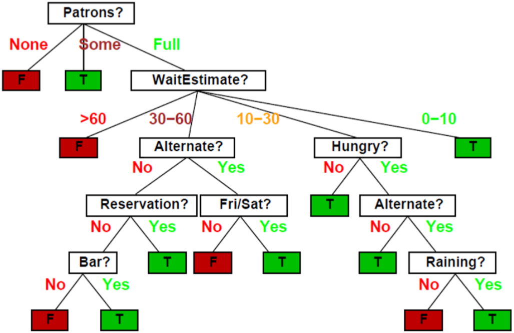


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

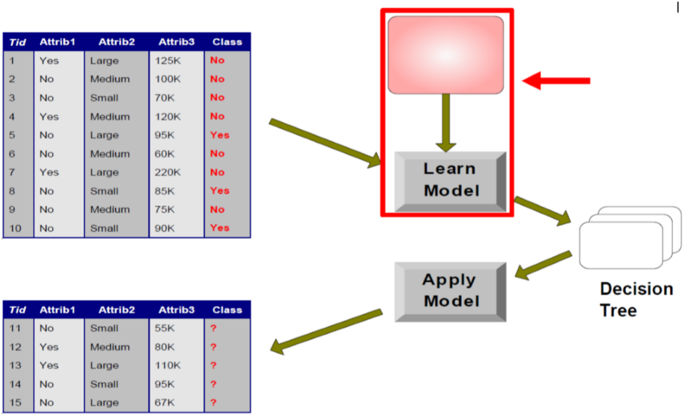
Classification（分类）of examples is positive (T) or negative (F)

Decision trees

One possible representation for hypotheses
i.e. here is the “true” tree for deciding whether to wait:



Decision Tree Learning

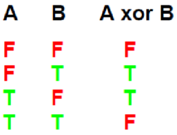


Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

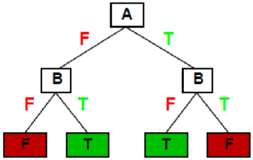
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Expressiveness (表达能力)

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf (函数真值表的每行对应于树中的一条路径):



A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples
- Prefer to find more compact decision trees

Hypothesis spaces (假设空间)

How many distinct decision trees with n Boolean attributes?

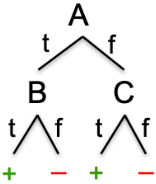
$=$ number of Boolean functions

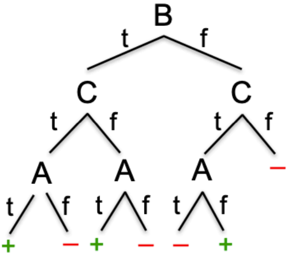
$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are
18,446,744,073,709,551,616 trees

Are all decision trees equal?

- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., ((A and B) or (not A and C))





Which tree do we prefer?

Learning *simplest* decision tree is NP-hard

- Formal justification – statistical learning theory
- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest' 76]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurs

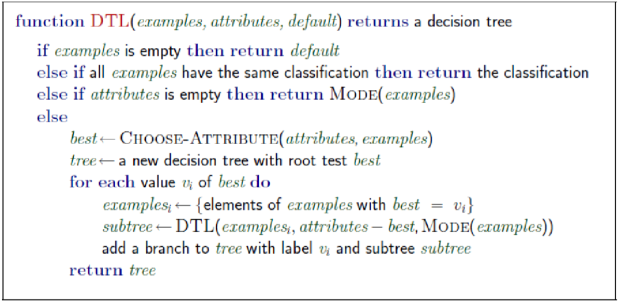
Learning Algorithm for Decision Trees

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}, \quad \mathbf{x} = (x_1, \ldots, x_d),\ x_j, y \in \{0, 1\}$$

```
GrowTree($S$)
**if** ($y = 0$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new *leaf*($0$)
**else if** ($y = 1$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new *leaf*($1$)
**else**
        choose best attribute $x_j$ //DT algorithms differ on this choice
        $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
        $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
        **return** new *node*($x_j$, GrowTree($S_0$), GrowTree($S_1$))
```

Decision Tree Learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree

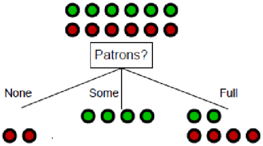


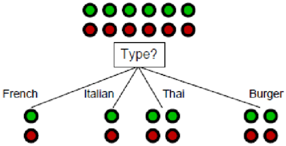
```
function DTL($examples, attributes, default$) returns a decision tree
    if $examples$ is empty then return $default$
    else if all $examples$ have the same classification then return the classification
    else if $attributes$ is empty then return MODE($examples$)
    else
        $best \leftarrow$ CHOOSE-ATTRIBUTE($attributes, examples$)
        $tree \leftarrow$ a new decision tree with root test $best$
        for each value $v_i$ of $best$ do
            $examples_i \leftarrow$ {elements of $examples$ with $best = v_i$}
            $subtree \leftarrow$ DTL($examples_i, attributes - best$, MODE($examples$))
            add a branch to $tree$ with label $v_i$ and subtree $subtree$
        return $tree$
```

MODE(examples): selects the most common output value among a set of examples

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





Patrons? is a better choice

Using information theory (信息论)

- Idea: To implement `Choose-Attribute` in the DTL algorithm

- Information Content 信息量 (Entropy 熵):

$$
\begin{aligned}
H(V) &= E_V[I(V = v_i)] \\
&= I\left(P\left(v_1\right), \ldots, P\left(v_n\right)\right) = \sum_{i=1}^{n} -P\left(v_i\right) \log_2 P\left(v_i\right)
\end{aligned}
$$

- For a training set containing p positive examples and n negative examples:

$$
I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}
$$

Entropy

Entropy $H(Y)$ of a random variable Y

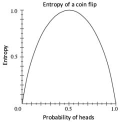
$$H(Y) = E_Y[I(Y = y_i)] = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

Conditional Entropy $H(Y \mid X)$ of a random variable Y conditioned on a random variable X

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

More uncertainty, more entropy.

Information theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



Information gain (信息增益)

- Decrease in entropy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y \mid X)$$

- A chosen attribute A divides the training set E into subsets $E_1, \ldots, E_v$ according to their values for A, where A has v distinct values.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Information Gain (IG) or reduction in entropy from the attribute test:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- Choose the attribute with the largest IG

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

Information gain (信息增益)

For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit
Consider the attributes Patrons and Type (and others too):

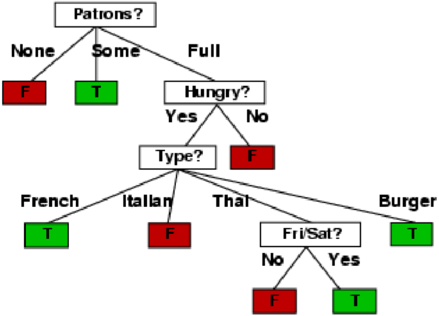
$$IG(Patrons) = 1 - \left[\frac{2}{12}I(0,1) + \frac{4}{12}I(1,0) + \frac{6}{12}I(\frac{2}{6},\frac{4}{6})\right] = 0.541 \text{ bits}$$

$$\begin{aligned} IG(Type) &= 1 - \left[\frac{2}{12}I(\frac{1}{2},\frac{1}{2}) + \frac{2}{12}I(\frac{1}{2},\frac{1}{2}) + \frac{4}{12}I(\frac{2}{4},\frac{2}{4}) + \frac{4}{12}I(\frac{2}{4},\frac{2}{4})\right] \\ &= 0 \text{ bits} \end{aligned}$$

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Example contd.

Decision tree learned from the 12 examples:



Substantially simpler than “true” tree – a more complex hypothesis isn’t justified by small amount of data

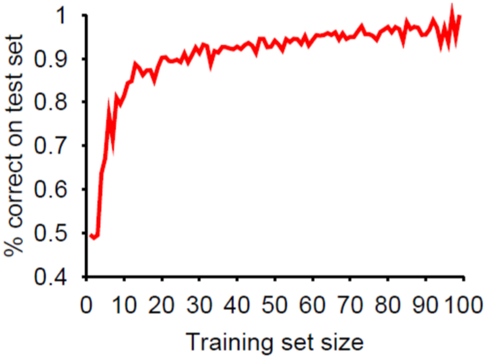
Performance measurement

How do we know that h $\approx$ f ?

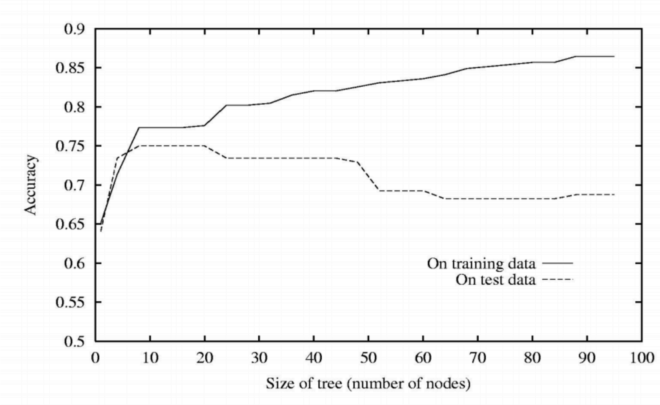
1. Use theorems of computational/statistical learning theory
2. Try *h* on a new test set (测试集) of examples

(use same distribution over example space as training set)

Learning curve (学习曲线) = % correct on test set as a function of training



Decision trees will overfit



Decision trees will overfit

- Standard decision trees have no learning bias
  - Training set error is always zero!
    If there is no label noise
  - Lots of variance
  - Must introduce some bias towards simpler trees
- Many strategies for picking simpler trees
  - Fixed depth
  - Minimum number of samples per leaf
- Random forests: 随机且有放回地从原始训练集中生成 N 个样本集，训练 N 个决策树，根据 N 棵树的 N 个分类结果投票，得到最终结果

Avoiding overfitting in decision trees

How can we avoid overfitting?

- Stop growing when data split is not statistically significant
- Acquire more training data
- Remove irrelevant attributes
- **Grow full tree, then post-prune**

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

Reduced-Error Pruning

Split training data further into *training* and *validation* sets

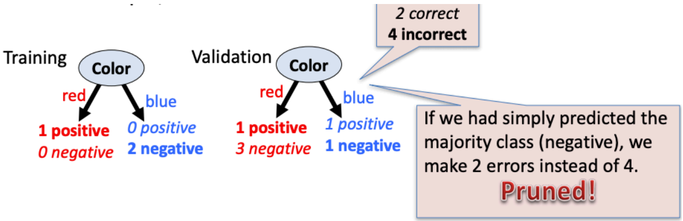
Grow tree based on *training* set

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the node that most improves *validation set* accuracy

Pruning Decision Trees

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf
- For example,

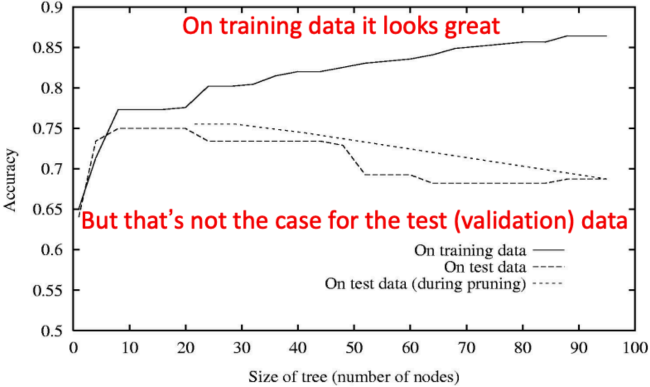


2 correct
4 incorrect

If we had simply predicted the majority class (negative), we make 2 errors instead of 4.

Pruned!

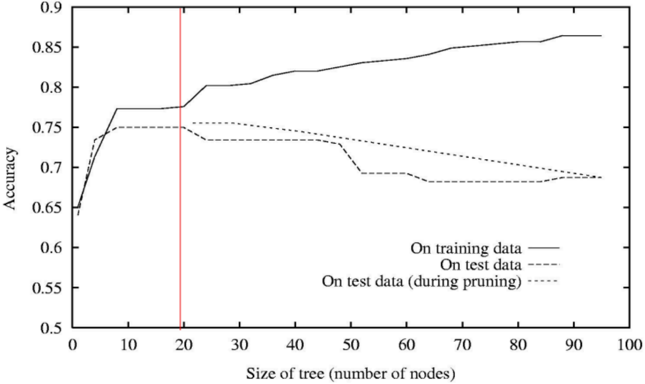
Effect of Reduced-Error Pruning



On training data it looks great

But that’s not the case for the test (validation) data

Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data

Comments on decision tree based classification

Advantages:

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

C4.5 algorithm

Example: C4.5 algorithm

- Simple depth-first construction.
- Uses information gain ratio (normalized information gain):

$$IG(Y, X) = H(Y) - H(Y \mid X),$$

$$H_X(Y) = -\sum_{j=1}^{v} \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j),$$

$$IG_{ratio}(Y, X) = \frac{IG(Y, X)}{H_X(Y)}.$$

  - ID3 algorithm uses information gain
- You can download the software from:
- http://www.cse.unsw.edu.au/∼quinlan/c4.5r8.tar.gz

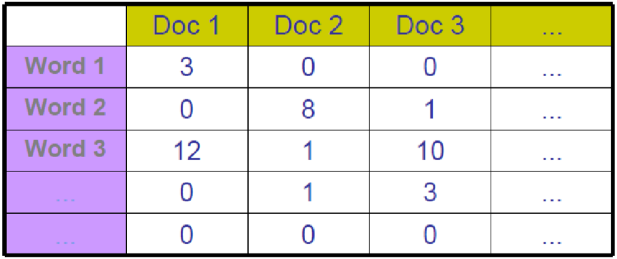
Table of Contents

Supervised Learning

- Learning Decision Trees
- K Nearest Neighbor Classifier
- Linear Predictions
- Logistic Regression
- Support Vector Machines

Vector Space (向量空间) Representation

Web page classification
Each document is a vector, one component for each term (=word).

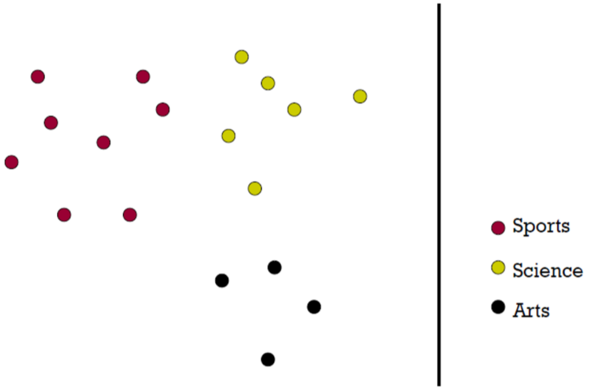


	Doc 1	Doc 2	Doc 3	...
Word 1	3	0	0	...
Word 2	0	8	1	...
Word 3	12	1	10	...
...	0	1	3	...
...	0	0	0	...

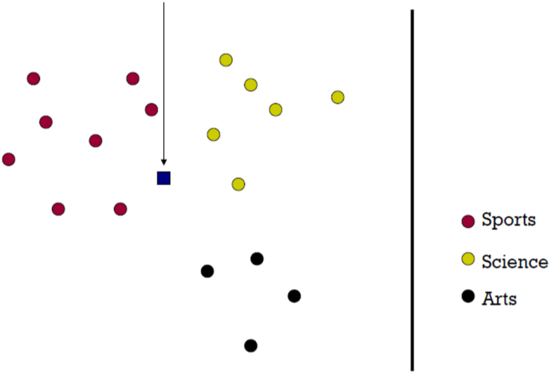
High-dimensional vector space:

- Terms are axes, 10,000+ dimensions, or even 100,000+
- Docs are vectors in this space

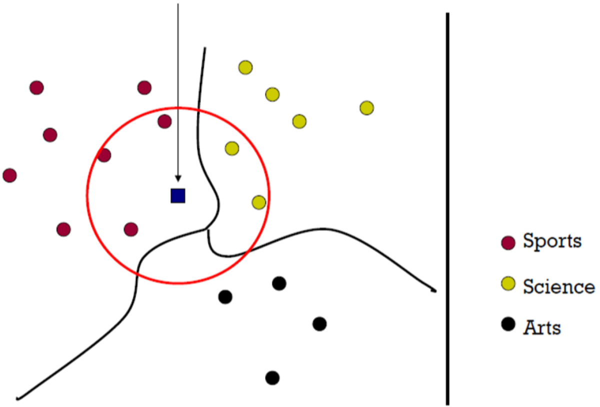
Classes in a Vector Space



Classes in a Vector Space



Classes in a Vector Space



Key ingredients of kNN

- A distance metric (距离度量)
- How many nearby neighbors to look at?
- A weighting function (optional)
- How to relate to the local points?

Euclidean Distance Metric

- Euclidean Distance (欧式距离):

$$D(\mathbf{x},\mathbf{x}') = \sqrt{\sum_i (x_i - x_i')^2} = \sqrt{(\mathbf{x}-\mathbf{x}')^\top(\mathbf{x}-\mathbf{x}')}$$

- Standardized Euclidean Distance (标准化欧氏距离): 任意维度上标度变化将影响整个距离，一个简单方法，将量纲正则化，将变量 x_i 变成 $\dfrac{x_i - \mu_i}{\sigma_i}$ （使各个维度的数据分别满足标准正态分布 $\mathcal{N}(0,1)$)

$$D(\mathbf{x},\mathbf{x}') = \sqrt{\sum_i \frac{(x_i - x_i')^2}{\sigma_i^2}} = \sqrt{(\mathbf{x}-\mathbf{x}')^\top\Sigma^{-1}(\mathbf{x}-\mathbf{x}')}$$

其中 Σ 为对角阵，第 i,i 项为 σ_i，其他项为 0。

Other Metric

- p-norm (范数, Minkowski distance):

$$||\mathbf{x}-\mathbf{x}'||_p = \left(\sum_i |x_i - x_i'|^p\right)^{1/p}$$

  - L_1 norm (Manhattan distance): $\sum_i |x_i - x_i'|$
  - L_2 norm (欧式距离): $\sqrt{\sum_i (x_i - x_i')^2}$
  - L_∞ norm: $\max_i |x_i - x_i'|$
- Hamming distance: 对于布尔属性值，两个点中具有不同值的属性数目
- Mahalanobis distance (马氏距离):

$$\sqrt{(\mathbf{x}-\mathbf{x}')^\top \Sigma^{-1} (\mathbf{x}-\mathbf{x}')}$$

  - Σ 为 $\mathbf{x}$ 与 $\mathbf{x}'$ 的协方差矩阵。
  - 若 Σ^{-1} 为单位矩阵，则马氏距离简化为欧式距离；
  - 若 Σ^{-1} 为对角阵，则简化为标准化欧式距离。
- . . .

方差、协方差、协方差矩阵

- 方差 (Variance)：方差是标准差 (Standard Deviation) 的平方，而标准差的意义是数据集中各个点到均值点距离的平均值。反应数据的离散程度。

  $$\sigma^2 = var(X) = E((X-\mu)^2)$$

  $$\sigma = \sqrt{E((X-\mu)^2)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}, \quad \text{where } \mu = \frac{1}{N}\sum_{i=1}^{N}x_i$$

- 协方差 (Covariance)：标准差与方差描述一维数据。协方差衡量两个随机变量的联合变化程度。两个变量倾向于表现出相似的行为，协方差为正；两变量倾向于表现出相反的行为，协方差为负。协方差的数值大小因取决于变量的大小，所以不容易解释。不过，正态形式的协方差大小可以显示两变量线性关系的强弱。

  $$cov(X,Y) = E((X-\mu)(Y-\nu)) = E(X\cdot Y) - \mu\nu$$

  如果 X 与 Y 是统计独立的，则 $con(X,Y)=0$，因为 $E(X\cdot Y)=\mu\nu$。

方差、协方差、协方差矩阵 (con’t)

- 协方差 (Covariance)：

$$\eta = \frac{cov(X,Y)}{\sqrt{var(X)\cdot var(Y)}}$$

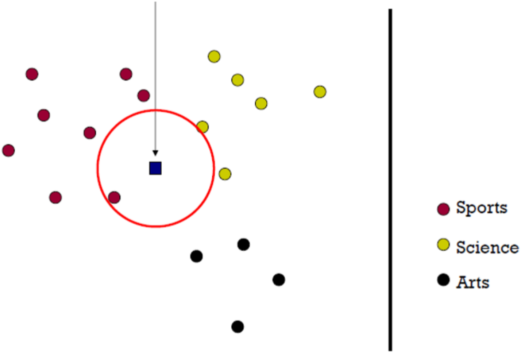
  η 表示线性相关性，$\eta \in [-1,1]$。当 $\eta = 1$，完全线性相关；$\eta = -1$，完全线性负相关。

- 协方差矩阵 (Covariance Matrix)：高维随机变量中随机变量之间的相关性。
  - 给定 n 维度随机变量 $\mathbf{X} = (X_1, X_2, \ldots, X_n)^\top$，其中 μ_i 是 X_i 的期望值，即 $\mu_i = E(X_i)$。
  - 协方差矩阵的第 i,j 项是随机变量 X_i 与 X_j 的协方差

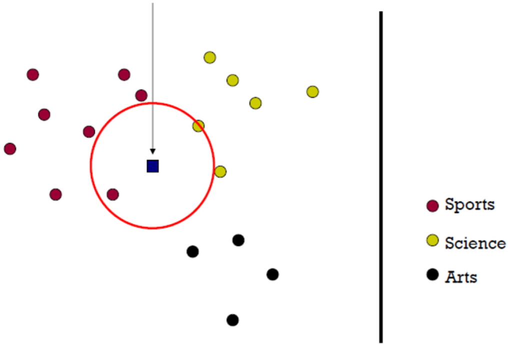
$$\sigma_{i,j} = cov(X_i, X_j) = E((X_i - \mu_i)(X_j - \mu_j))$$

  - $con(X_i, X_j) = con(X_j, X_i)$，协方差矩阵是对称的。

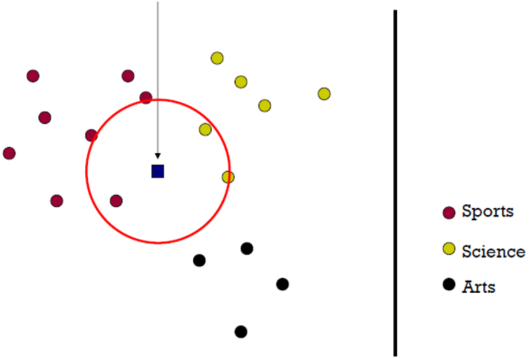
1-Nearest Neighbor (kNN) classifier



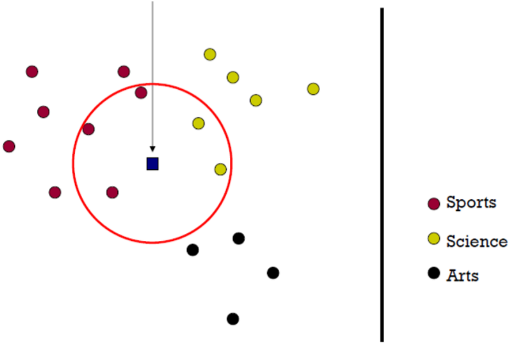
2-Nearest Neighbor (kNN) classifier



3-Nearest Neighbor (kNN) classifier



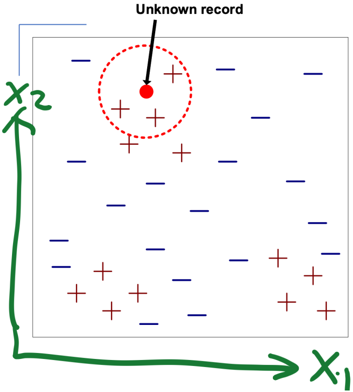
5-Nearest Neighbor (kNN) classifier



Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in D.
- Testing instance x:
    - Compute similarity between x and all examples in D.
    - Assign x the category of the majority of the k most similar examples in D.
- Also called
    - Case-based learning (基于实例的学习)
    - Memory-based learning
    - Lazy learning

Nearest neighbor classifiers

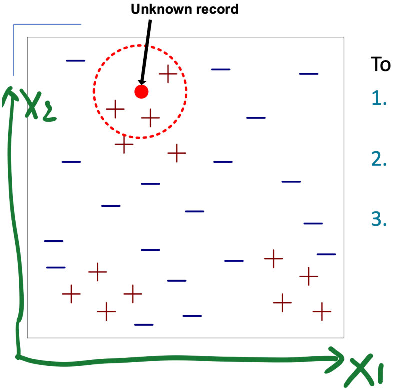


Requires three inputs:

1. The set of stored training samples
2. Distance metric to compute distance between samples
3. The value of *k*, i.e., the number of nearest neighbors to retrieve

$y \in \{+,-\}$

Nearest neighbor classifiers



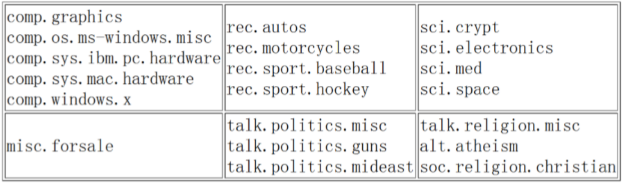
To classify unknown sample:

1. Compute distance to training records
2. Identify *k* nearest neighbors
3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Case Study: kNN for Web Classification

Dataset

- 20 News Groups (20 classes)
- Download : (http://people.csail.mit.edu/jrennie/20Newsgroups/)
- 61,118 words, 18,774 documents
- Class labels descriptions



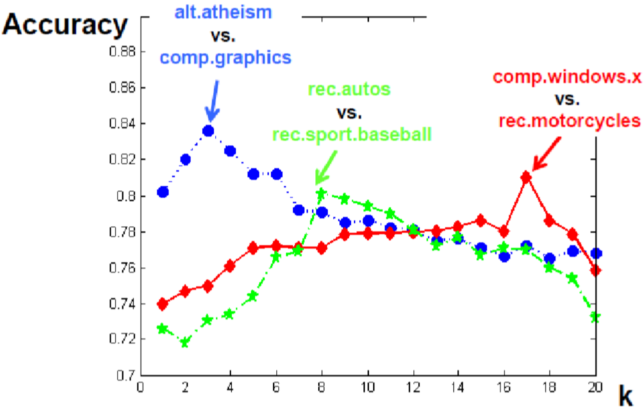
comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Experimental Setup

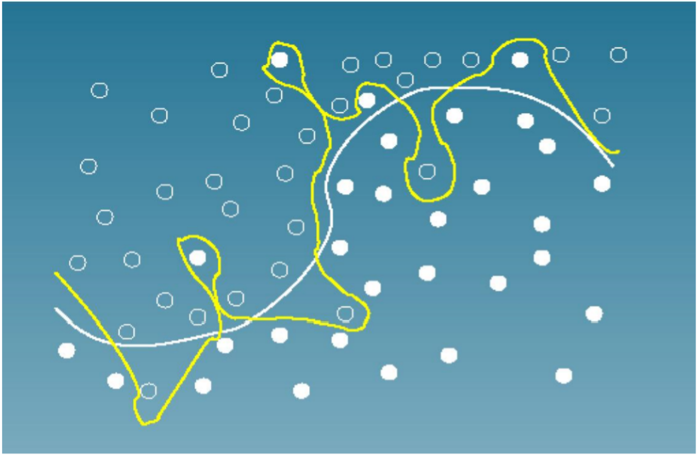
- Training/Test Sets:
  - 50%-50% randomly split
  - 10 runs
  - report average results
- Evaluation Criteria:

$$Accuracy = \frac{\sum_{i \in \text{testset}} \mathbf{1}\left(predict_i = truelabel_i\right)}{\#\text{ of test samples}}$$

Results: Binary Classes



Is kNN ideal?



Comments on kNN

- Instance-based learning: kNN – a Nonparametric (无参数的) classifier
  - A nonparametric method does not rely on any assumption concerning the structure of the underlying density function.
  - Very little “learning” is involved in these methods
- Sample size
  - The more the better
  - Need efficient search algorithm for NN
- Good news:
  - Simple and powerful methods
  - Flexible and easy to apply to many problems.
- Bad news:
  - High memory requirements
  - Very dependent on the scale factor for a specific problem

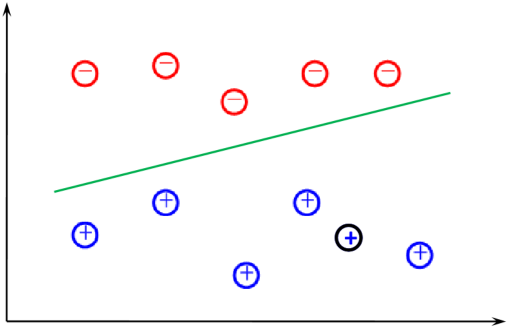
Table of Contents

Supervised Learning

- Learning Decision Trees
- K Nearest Neighbor Classifier
- Linear Predictions
- Logistic Regression
- Support Vector Machines

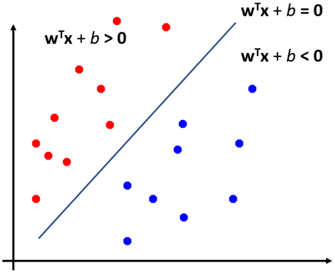
Classification

Classification
= learning from labeled data. Dominant problem in Machine Learning



Linear Classifiers

Binary classification can be viewed as the task of separating classes in feature space (特征空间):



Decide $\hat{y} = 1$ if $\mathbf{w}^\top \mathbf{x} + b > \mathbf{0}$
otherwise $\hat{y} = -1$
$\hat{y} = h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right)$

Linear Classifiers

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

- Need to find $\mathbf{w}$ (direction) and b (location) of the boundary
- Want to minimize the expected zero/one loss (损失) for classifier $h : \mathcal{X} \to \mathcal{Y}$, which is

$$L(h(\mathrm{x}), y) = \begin{cases} 0 & \text{if } h(\mathrm{x}) = y \\ 1 & \text{if } h(\mathrm{x}) \neq y \end{cases}$$

Linear Classifiers

Ideally we want to find a classifier
$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right)$ to minimize the $0/1$ loss

$$\min_{\mathbf{w},b} \sum_i L_{0/1}\left(h\left(\mathbf{x}_i\right), y_i\right)$$

Unfortunately, this is a hard problem.
Alternate loss functions:

$$\begin{aligned}
L_2(h(\mathbf{x}), y) &= (y - \mathbf{w}^\top \mathbf{x} - b)^2 = (1 - y(\mathbf{w}^\top \mathbf{x} + b))^2 \\
L_1(h(\mathbf{x}), y) &= |y - \mathbf{w}^\top \mathbf{x} - b| = |1 - y(\mathbf{w}^\top \mathbf{x} + b)| \\
L_{hinge}(h(\mathbf{x}), y) &= (1 - y(\mathbf{w}^\top \mathbf{x} + b))_+ = \max(0, 1 - y(\mathbf{w}^\top \mathbf{x} + b))
\end{aligned}$$

Linear Classifiers

Least squares loss function:

$$L_2(h(\mathbf{x}), y) = \left(y - \mathbf{w}^\top \mathbf{x} - b\right)^2$$

The goal:
to learn a classifier $h(x) = sign(w^\top x + b)$ to minimize the least squares loss:

$$\begin{aligned} \text{Loss} &= \min_{\mathbf{w},b} \sum_i L_2\left(h\left(\mathbf{x}_i\right), y_i\right) \\ &= \min_{\mathbf{w},b} \sum_i \left(y_i - \mathbf{w}^\top \mathbf{x}_i - b\right)^2 \end{aligned}$$

Solving Least Squares Classification

Let

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots & \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} b \\ \vdots \\ w_d \end{bmatrix}$$

$$\begin{aligned} \text{Loss} = \min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^2 &= \min_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^2 \\ &= \min_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \end{aligned}$$

Solving for *w*

$$\begin{aligned} \frac{\partial \text{Loss}}{\partial \mathbf{w}} = 2(\mathbf{X}\mathbf{w}-\mathbf{y})^\top \mathbf{X} &= 0 \\ \mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{X}^\top \mathbf{y} &= 0 \\ \mathbf{w}^* &= \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \end{aligned}$$

Note:

$$d(\mathbf{A}\mathbf{x}+\mathbf{b})^\top \mathbf{C}(\mathbf{D}\mathbf{x}+\mathbf{e}) = \left((\mathbf{A}\mathbf{x}+\mathbf{b})^\top \mathbf{C}\mathbf{D} + (\mathbf{D}\mathbf{x}+\mathbf{e})^\top \mathbf{C}^\top \mathbf{A}\right) d\mathbf{x}$$

$$d(\mathbf{A}\mathbf{x}+\mathbf{b})^\top (\mathbf{A}\mathbf{x}+\mathbf{b}) = \left(2(\mathbf{A}\mathbf{x}+\mathbf{b})^\top \mathbf{A}\right) d\mathbf{x}$$

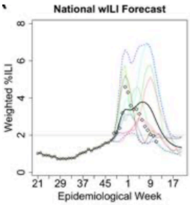
要求 $\mathbf{X}^\top\mathbf{X}$ 可逆 i.e., **满秩；若不满秩，则可解出多个 $\mathbf{w}^*$**

- $\mathbf{X}^+ = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top$ is called the Moore-Penrose pseudoinverse (伪逆) of X
- Prediction for x_0:
  $$\hat{y} = \text{sign}\left(\mathbf{w}^{*\top} \begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix}\right) = \text{sign}\left(\mathbf{y}^\top \mathbf{X}^{+\top} \begin{bmatrix} 1 \\ \mathbf{x}_0 \end{bmatrix}\right)$$

Regression

Goal:

- Given a training dataset of pairs (**x**,y) where
  - **x** is a vector
  - y is a scalar
- Learn a function (aka. curve or line) y' = h(x) that best fits the training data



Example Applications:

- Stock price prediction
- Forecasting epidemics
- Speech synthesis
- Generation of images (e.g. *Deep Dream*)
- Predicting the number of tourists on Machu Picchu on a given day

Regression

Example Application:
Forecasting Epidemics

- Input features, **x**: attributes of the epidemic
- Output, y: Weighted %ILI, prevalence of the disease
- Setting: observe past prevalence to predict future prevalence

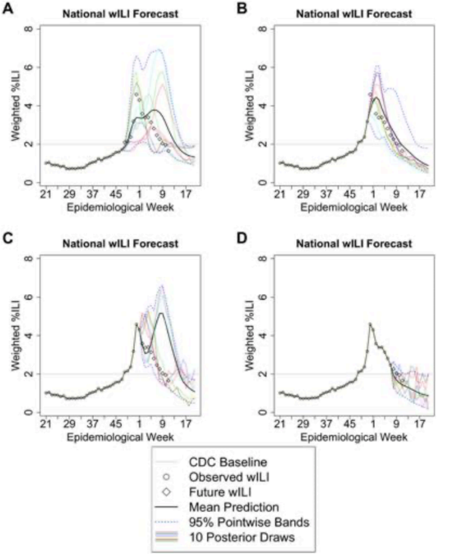
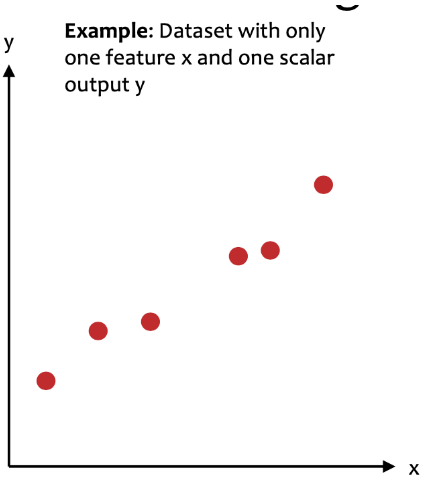


Fig 2. 2013–2014 national forecast, retrospectively, using the final revisions of wILI values, using revised wILI data through epidemiological weeks (A) 47, (B) 51, (C) 1, and (D) 7.

Figure from Brooks et al. (2015)

Regression

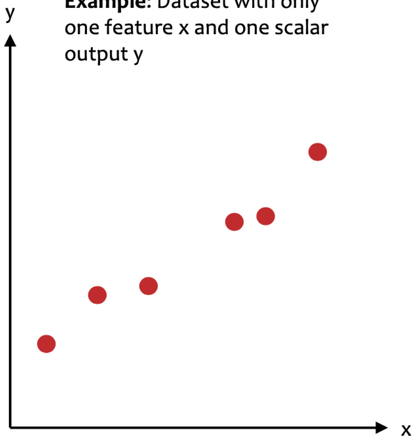


Example: Dataset with only one feature x and one scalar output y

Q: What is the function that best fits these points?

k-NN Regression

Example: Dataset with only one feature x and one scalar output y



k=1 Nearest Neighbor Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest x in training data and return its y

k=2 Nearest Neighbor Distance Weighted Regression

- *Train*: store all (x, y) pairs
- *Predict*: pick the nearest two instances $x^{(n1)}$ and $x^{(n2)}$ in training data and return the weighted average of their y values

Linear Regression（线性回归）

$$f(x) = wx + b \quad \text{s.t. } f(x_i) \approx y_i$$

- 令均方误差最小化，得到：

$$\begin{aligned}(w^*, b^*) &= \arg\min_{(w,b)} \sum_{i=1}^{m} (f(x_i) - y_i)^2 \\ &= \arg\min_{(w,b)} \sum_{i=1}^{m} (y_i - wx_i - b)^2\end{aligned}$$

- 也就是对 $E_{(w,b)} = \sum_{i=1}^{m} (y_i - wx_i - b)^2$ 进行最小二乘参数估计

线性回归

- 分别对 w 和 b 求导：

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)x_i\right)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m}(y_i - wx_i)\right)$$

- 令导数为 0，得到闭式（closed-form）解：

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

多元（multi-variate）线性回归

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \quad \text{s.t. } f(\mathbf{x}_i) \approx \mathbf{y}_i$$

Let

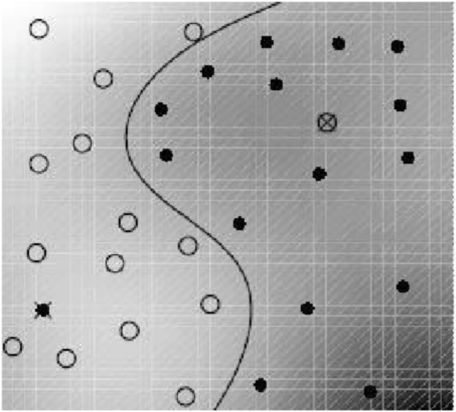
$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots & \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} b \\ \vdots \\ w_d \end{bmatrix}$$

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ \frac{\partial E_{\mathbf{w}}}{\partial \mathbf{w}} &= 2(\mathbf{X}\mathbf{w} - \mathbf{y})^\top \mathbf{X} \\ \mathbf{w}^* &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}, \qquad \mathbf{X}^\top \mathbf{X} \text{ 满秩} \end{aligned}$$

General linear classification

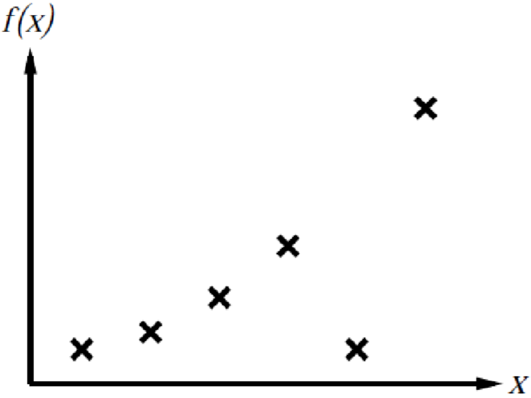
Basis (nonlinear) functions (基函数)

$f(\mathbf{x}, \mathbf{w}) = b + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \cdots + w_m\phi_m(\mathbf{x})$



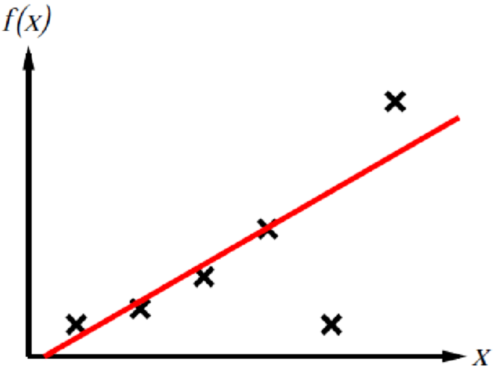
Model complexity and overfitting

E.g., curve fitting (曲线拟合):



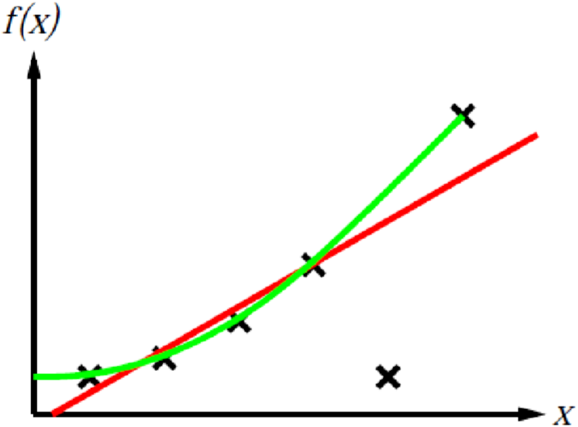
Model complexity and overfitting

E.g., curve fitting (曲线拟合):



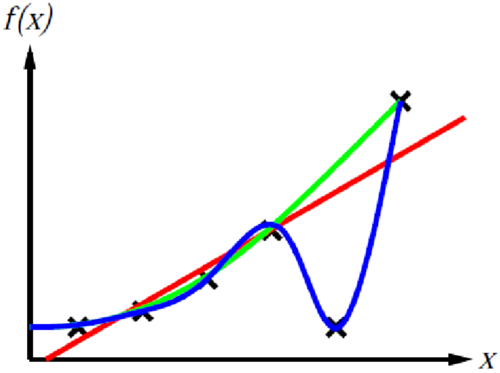
Model complexity and overfitting

E.g., curve fitting (曲线拟合):



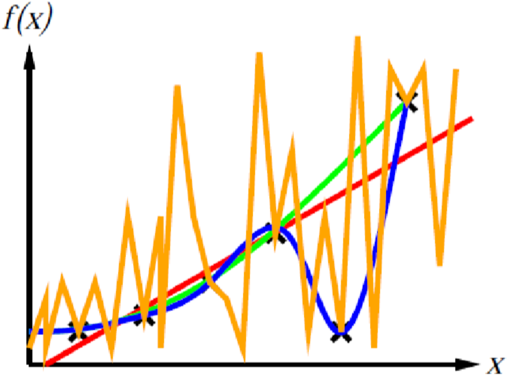
Model complexity and overfitting

E.g., curve fitting (曲线拟合):



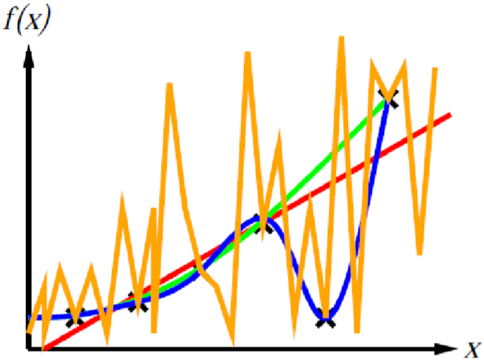
Model complexity and overfitting

E.g., curve fitting (曲线拟合):



Model complexity and overfitting

E.g., curve fitting (曲线拟合):



Occam's razor (奥卡姆剃刀原则): maximize a combination of consistency and simplicity 优先选择与数据一致的最简单的假设

Regularization (规范化)

Intuition: should penalize not the parameters, but the number of bits required to encode the parameters

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \ \text{Loss} \ + \lambda \cdot \text{penalty}(\mathbf{w})$$

$$\text{L2 regularization } \mathbf{w}^* = \arg\min_{\mathbf{w}} \ \text{Loss} \ + \lambda \|\mathbf{w}\|^2$$

$$\text{L1 regularization } \mathbf{w}^* = \arg\min_{\mathbf{w}} \ \text{Loss} \ + \lambda |\mathbf{w}|$$

Solving L2-regularized LS

$$\text{min}_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^2 + \lambda \|\mathbf{w}\|^2$$

Comments on least squares classification

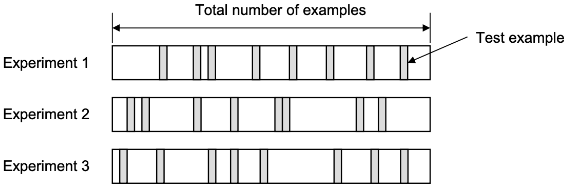
- Not the best thing to do for classification
- But
  - Easy to train, closed form solution (闭式解)
  - Ready to connect with many classical learning principles

Cross-validation (交叉验证)

- The basic idea: if a model overfits (is too sensitive to data) it will be unstable, i.e., removal part of the data will change the fit significantly
- We can hold out (取出) part of the data, fit the model to the rest, and then test on the holdout set.

Random Subsampling

- **Random Subsampling performs K data splits of the entire dataset**
  - Each data split randomly selects a (fixed) number of examples without replacement
  - For each data split we retrain the classifier from scratch with the training examples and then estimate E_i with the test examples

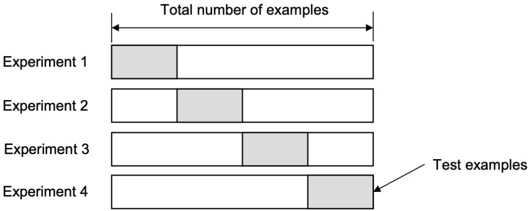


- **The true error estimate is obtained as the average of the separate estimates E_i**
  - This estimate is significantly better than the holdout estimate

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

K-Fold Cross-validation

- **Create a K-fold partition of the the dataset**
  - For each of K experiments, use K-1 folds for training and a different fold for testing
    - This procedure is illustrated in the following figure for K=4

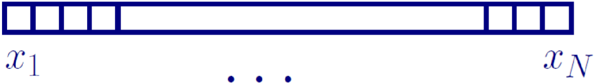


- **K-Fold Cross validation is similar to Random Subsampling**
  - The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing
- **As before, the true error is estimated as the average error rate on test examples**

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

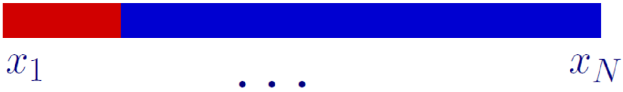
Cross-validation

- The improved holdout method: *k*-fold cross-validation
  - Partition data into *k* roughly equal parts;
  - Train on all but *j*-th part, test on *j*-th part



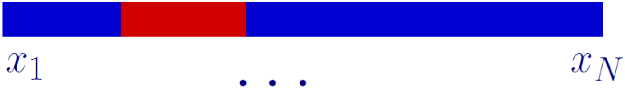
Cross-validation

- The improved holdout method: *k*-fold cross-validation
  - Partition data into *k* roughly equal parts;
  - Train on all but *j*-th part, test on *j*-th part



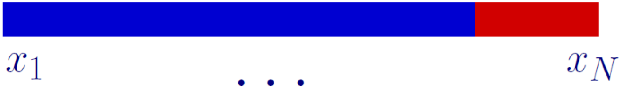
Cross-validation

- The improved holdout method: *k*-fold cross-validation
  - Partition data into *k* roughly equal parts;
  - Train on all but *j*-th part, test on *j*-th part



Cross-validation

- The improved holdout method: *k*-fold cross-validation
    - Partition data into *k* roughly equal parts;
    - Train on all but *j*-th part, test on *j*-th part



Machine learning paradigm

- Choose a model class
  - NB, kNN, loss/regularization combination
- Model selection
  - Cross validation
- Training
- Testing

Table of Contents

Supervised Learning

- Learning Decision Trees
- K Nearest Neighbor Classifier
- Linear Predictions
- Logistic Regression
- Support Vector Machines

Logistic regression

- 中文通常称为“逻辑回归”，西瓜书称为“对数几率回归”
- Name is somewhat misleading. Really a technique for classification, not regression.
  - “Regression” comes from fact that we fit a linear model to the feature space
- Involves a more probabilistic view of classification

Different ways of expressing probability

- Consider a two-outcome probability space, where:
  - $p(O_1) = p$
  - $p(O_2) = 1 - p = q$
- Can express probability of O_1 as:

	notation	range equivalents		
standard probability	p	0	0.5	1
odds	p / q	0	1	+ ∞
log odds (logit)	log(p / q)	- ∞	0	+ ∞

Log odds (几率)

- Numeric treatment of outcomes O_1 and O_2 is equivalent
  - If neither outcome is favored over the other, then log odds $= 0$
  - If one outcome is favored with log odds $= x$, then other outcome is disfavored with log odds $= -x$
- Especially useful in domains where relative probabilities can be miniscule (微小)
  - Example: multiple sequence alignment in computational biology

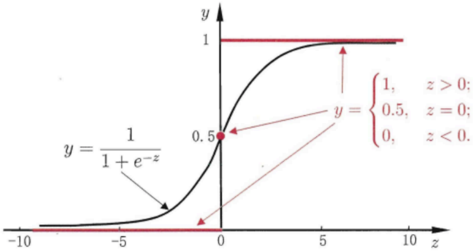
From probability to log odds (and back again)

$$z = \log\left(\frac{p}{1-p}\right) \qquad \text{logit function}$$

$$e^{z} = \frac{p}{1-p}$$

$$p = \frac{e^{z}}{1+e^{z}} = \frac{1}{1+e^{-z}} \qquad \text{logistic function}$$

Standard logistic function



$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$, Logistic function, $\sigma(x) = \frac{1}{1 + e^{-x}}$

- x_0, the x value of the sigmoid's midpoint;
- L, the curve's maximum value;
- k, the logistic growth rate or steepness of the curve.

Standard logistic function where $L = 1,\ k = 1,\ x_0 = 0$

Logistic regression

- Model consists of a vector $\theta = (\mathbf{w}, b)$ in $(d+1)$-dimensional feature space
- For a point $\mathbf{x}$ in feature space, project it onto θ to convert it into a real number z in the range in the range $-\infty$ to $+\infty$

$$z = \mathbf{w}^\top \mathbf{x} + b = w_1 x_1 + \cdots + w_d x_d + b$$

- Map z to the range 0 to 1 using the logistic function

$$p = \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Overall, logistic regression maps a point $\mathbf{x}$ in $(d+1)$-dimensional feature space to a value in the range 0 to 1

Logistic regression

$$y = \sigma(\mathbf{w}^\top \cdot \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \cdot \mathbf{x} + b)}}$$

等价变换为:

$$\ln \frac{y}{1-y} = \mathbf{w}^\top \cdot \mathbf{x} + b$$

若视 y 视为样本 $\mathbf{x}$ 作为正例的可能性，则 $1-y$ 是其反例可能性，两者的比值 $\frac{y}{1-y}$ 称为 “几率” (odds)，反映了 $\mathbf{x}$ 作为正例的相对可能性，对几率取对数则得到 “对数几率” (log odds，亦称 logit)，$\ln \frac{y}{1-y}$

Logistic regression 实际上是在用线性回归模型的预测结果去逼近真实标记的对数几率。

Learning in logistic regression

- How are parameters of the model ($\mathbf{w}$ and b) learned?
- This is an instance of supervised learning
  - We have labeled training examples
- We want model parameters such that
  - For training examples $\mathbf{x}$, the prediction of the model $\hat{y}$ is as close as possible to the true y
  - Or equivalently so that the distance between $\hat{y}$ and y is small

Learning in logistic regression

将 y 看作后验概率 $P(y=1 \mid \mathbf{x})$，则

$$\ln \frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} = \mathbf{w}^\top \mathbf{x} + b$$

$$P(y=1 \mid \mathbf{x}) = \frac{e^{\mathbf{w}^\top \mathbf{x}+b}}{1+e^{\mathbf{w}^\top \mathbf{x}+b}}$$

$$P(y=0 \mid \mathbf{x}) = \frac{1}{1+e^{\mathbf{w}^\top \mathbf{x}+b}}$$

通过“极大似然法”（maximum likelihood method）来估计 $\mathbf{w}$ 和 b

给定数据集 $\{(x_i, y_i)\}_{i=1}^m$，最大化“对数似然”（log-likelihood）

$$L(\mathbf{w}, b) = \sum_{i=1}^{m} \ln P(y_i \mid \mathbf{x}_i;\ \mathbf{w}, b)$$

令每个样本属于其真实标记的概率越大越好

Cross-entropy loss

令 $y \in \{0, 1\}$，则

$$\ln P(y \mid \mathbf{x};\, \mathbf{w}, b) = y \ln \sigma(\mathbf{w}^\top \mathbf{x} + b) + (1 - y)\ \ln(1 - \sigma(\mathbf{w}^\top \mathbf{x} + b))$$

则最大化 $L(\mathbf{w}, b)$，等价于最小化 $E(\mathbf{w}, b)$

$$E(\mathbf{w}, b) = -\sum_{i=1}^{m} [y_i \ln \sigma(\mathbf{w}^\top \mathbf{x}_i + b) + (1 - y_i)\ \ln(1 - \sigma(\mathbf{w}^\top \mathbf{x}_i + b))]$$

- The **cross-entropy** of the distribution q relative to a distribution p over a given set is defined as:

$$H(p, q) = -\sum_{x \in X} p(x) \ln q(x)$$

- Let $\hat{y} = \sigma(\mathbf{w}^\top \mathbf{x} + b)$, then

$$\begin{aligned} H(y, \hat{y}) &= -\sum_{i=1}^{m} [y \ln \hat{y} + (1 - y) \ln(1 - \hat{y})] \\ &= -\sum_{i=1}^{m} [y \ln \sigma(\mathbf{w}^\top \mathbf{x} + b) + (1 - y)\ \ln(1 - \sigma(\mathbf{w}^\top \mathbf{x} + b))] \end{aligned}$$

Gradient descent

- Goal:
  - find parameters $\theta = (\mathbf{w}, b)$
  - such that
$$\hat{\theta} = \arg\min_{\theta} \frac{1}{m} L_{CE}(y_i, \mathbf{x}_i;\ \theta)$$

  where

  $$L_{CE}(y_i, \mathbf{x}_i;\ \theta) = -[y_i \ln \sigma(\mathbf{w}^{\top}\mathbf{x}_i + b) + (1 - y_i) \ln(1 - \sigma(\mathbf{w}^{\top}\mathbf{x}_i + b))]$$

- For logistic regression, the loss is **convex**

Illustrating gradient descent

- The **gradient** indicates the direction of greatest increase of the cost/loss function
- **Gradient descent** finds parameters $(\mathbf{w}, b)$ that decrease the loss by taking a step in the opposite direction of the gradient

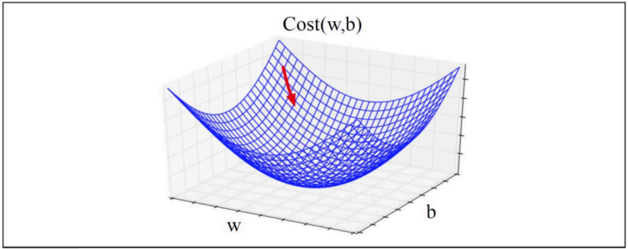
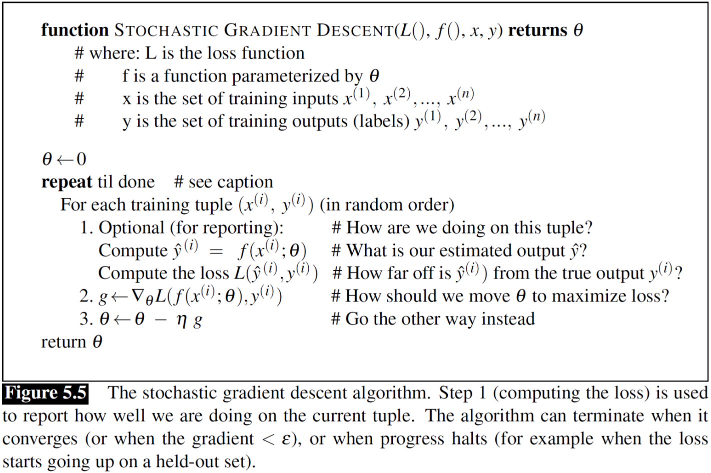


Figure 5.4 Visualization of the gradient vector in two dimensions *w* and *b*.

Stochastic gradient descent



```
function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, $x$, $y$) returns $\theta$
    # where: L is the loss function
    #       f is a function parameterized by $\theta$
    #       x is the set of training inputs $x^{(1)}, x^{(2)}, ..., x^{(n)}$
    #       y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, ..., y^{(n)}$

$\theta \leftarrow 0$
repeat til done   # see caption
   For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
      1. Optional (for reporting):          # How are we doing on this tuple?
         Compute $\hat{y}^{(i)} = f(x^{(i)};\theta)$     # What is our estimated output $\hat{y}$?
         Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$  # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?
      2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)};\theta), y^{(i)})$       # How should we move $\theta$ to maximize loss?
      3. $\theta \leftarrow \theta - \eta g$                     # Go the other way instead
return $\theta$
```

Figure 5.5 The stochastic gradient descent algorithm. Step 1 (computing the loss) is used to report how well we are doing on the current tuple. The algorithm can terminate when it converges (or when the gradient $< \varepsilon$), or when progress halts (for example when the loss starts going up on a held-out set).

The gradient for logistic regression

$$L_{CE}(w,b) = -[y \log \sigma(w \cdot x + b) + (1-y) \log (1 - \sigma(w \cdot x + b))]$$

$$\frac{\partial L_{CE}(w,b)}{\partial w_j} = [\sigma(w \cdot x + b) - y] x_j$$

Difference between the model prediction and the correct answer y

Feature value for dimension j

Logistic regression

- Advantages:
  - Makes no assumptions about distributions of classes in feature space
  - Easily extended to multiple classes (multinomial regression)
  - Natural probabilistic view of class predictions
  - Quick to train
  - Very fast at classifying unknown records
  - Good accuracy for many simple data sets
  - Resistant to overfitting
  - Can interpret model coefficients as indicators of feature importance
- Disadvantages:
  - Linear decision boundary

Table of Contents

Supervised Learning

- Learning Decision Trees
- K Nearest Neighbor Classifier
- Linear Predictions
- Logistic Regression
- Support Vector Machines