Supervised learning

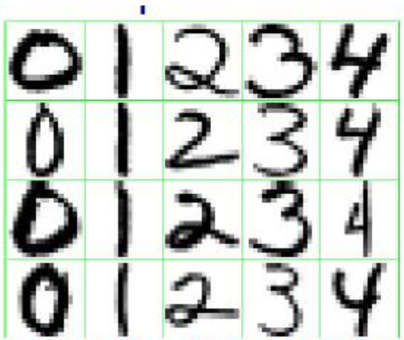
Supervised learning

Formal setup

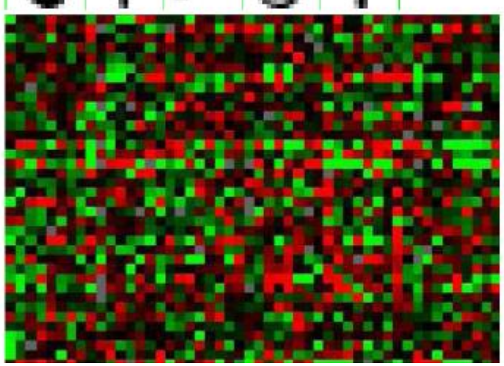
- Input data space $\mathcal{X}$
- Output (label, target) space $\mathcal{Y}$
- Unknown function $f : \mathcal{X} \rightarrow \mathcal{Y}$
- We are given a set of labeled examples $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, with $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.
- Finite $\mathcal{Y} \Rightarrow$ 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 $y \in \mathcal{Y}$
- Examples:



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



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

Decision Trees
决策树

Section 18.3

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:

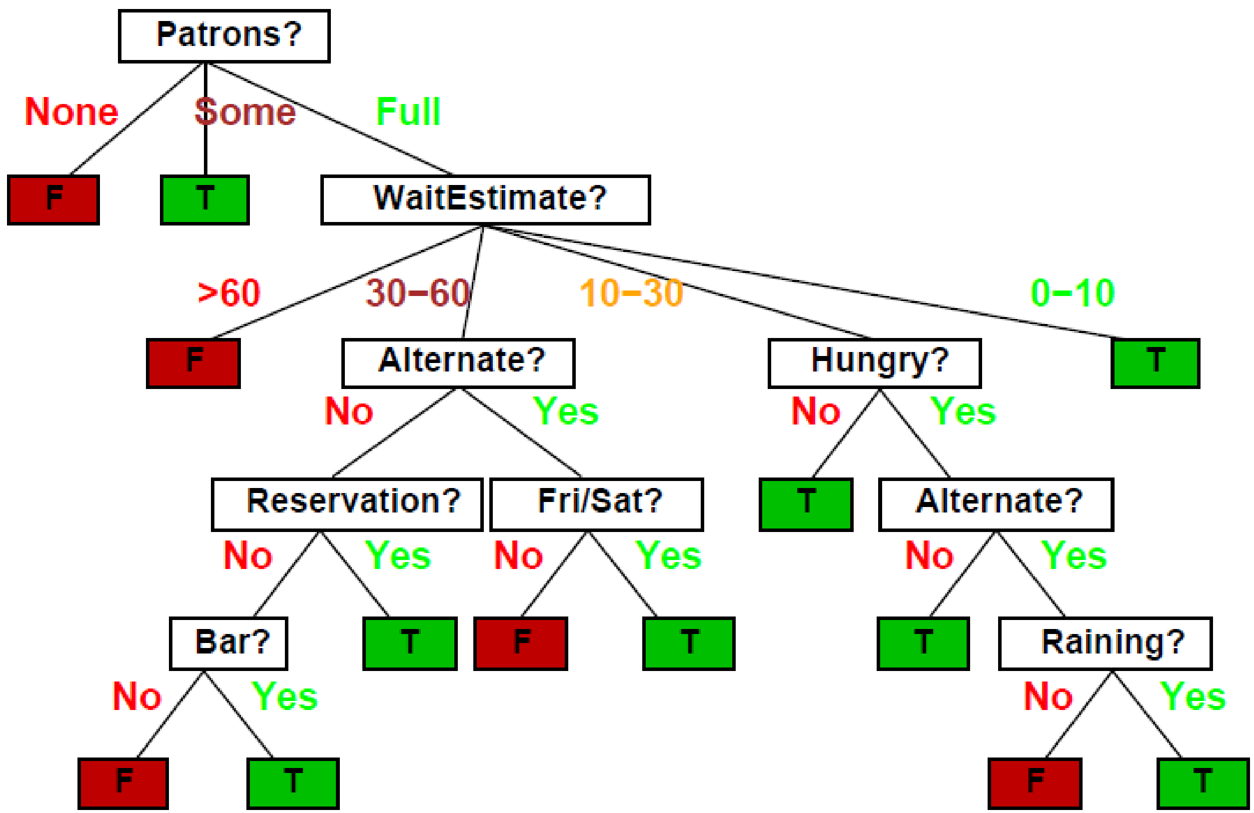
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	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

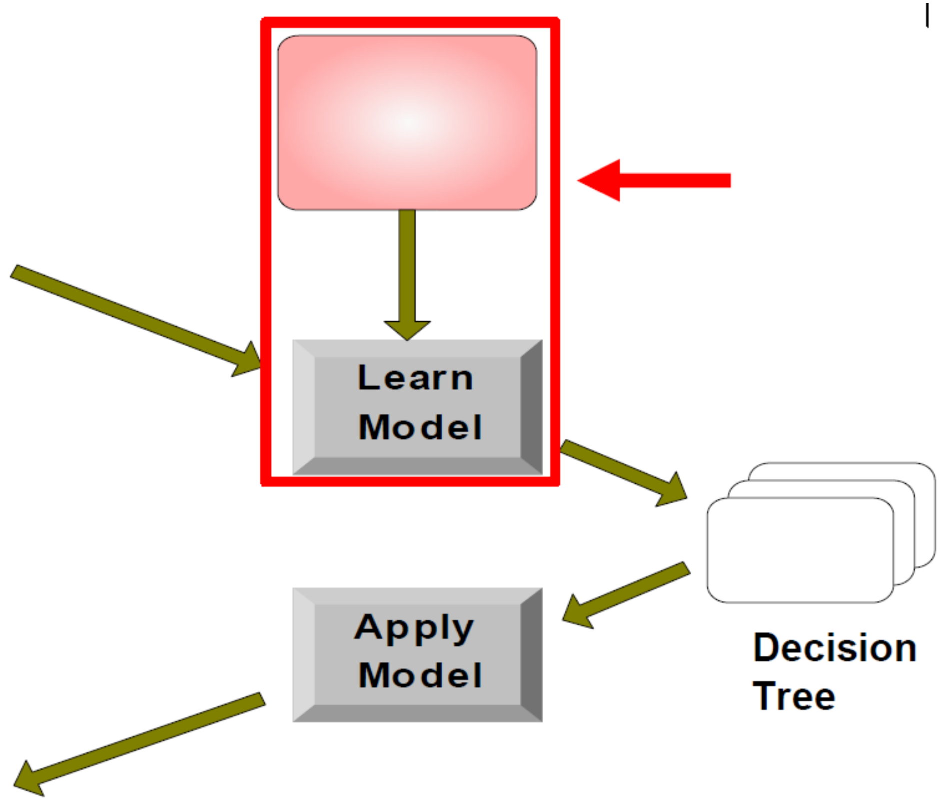
One possible representation for hypotheses

E.g., 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



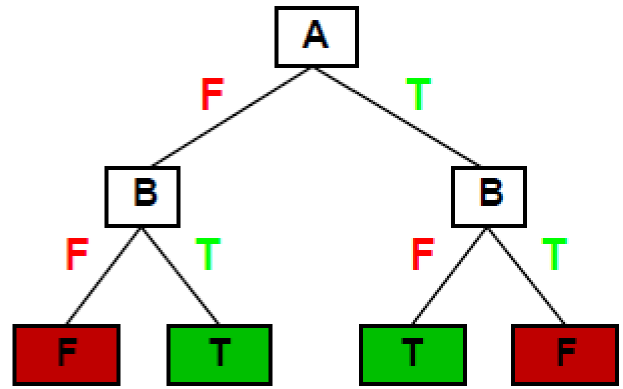
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

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

Decision tree learning

Aim: find a small tree consistent with the training examples

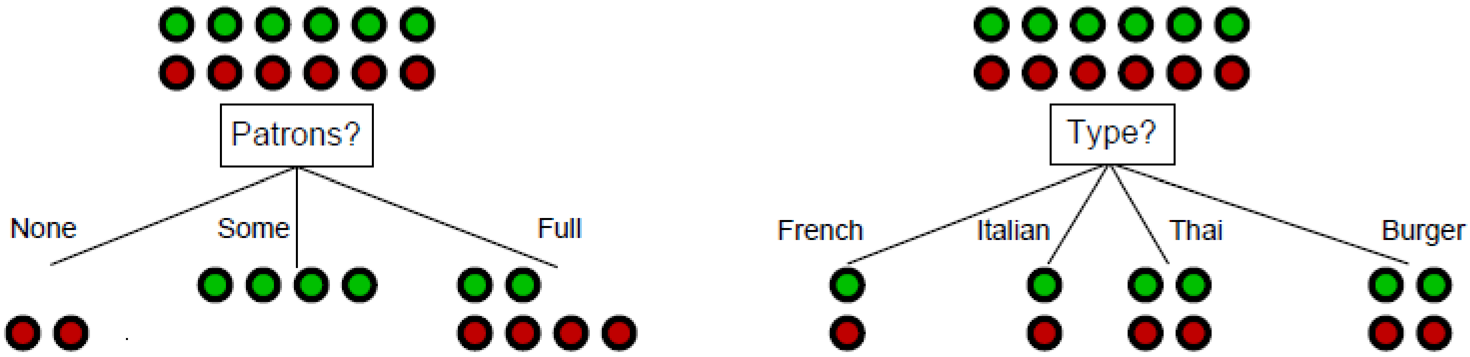
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

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 ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        tree ← a new decision tree with root test *best*
        for each value v_i of *best* **do**
            $examples_i$ ← {elements of *examples* with *best* $= v_i$}
            subtree ← DTL($examples_i$, *attributes* − *best*, MODE(*examples*))
            add a branch to *tree* with label v_i and subtree *subtree*
        return *tree*

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）

To implement `Choose-Attribute` in the DTL algorithm

Information Content 信息量(Entropy熵):

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$

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

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

Information gain（信息增益）

A chosen attribute *A* divides the training set *E* into subsets E_1, … , 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

Information gain

For the training set, $p = n = 6$, *I(6/12, 6/12) = 1* bit

Consider the attributes *Patrons* and *Type* (and others too):

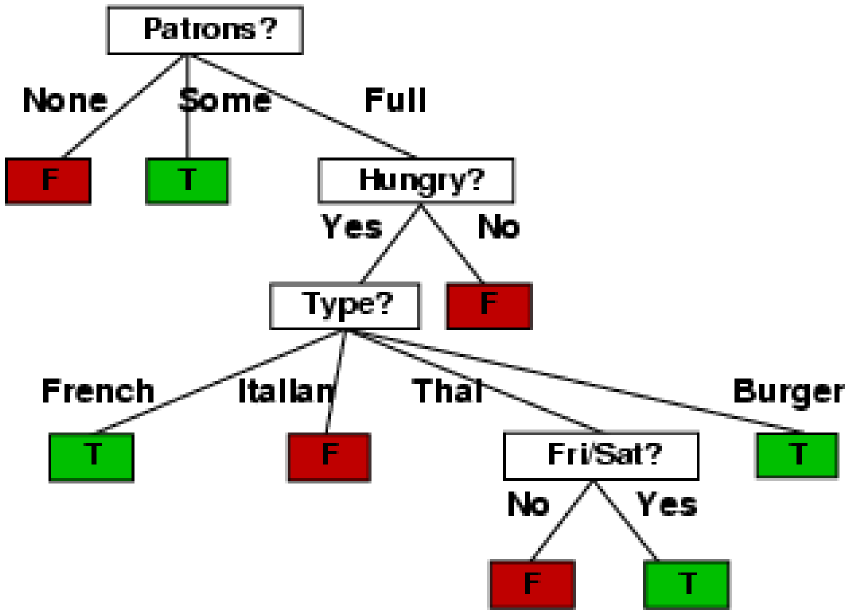
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

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 set size

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

Example: C4.5

- Simple depth-first construction.
- Uses Information Gain

19

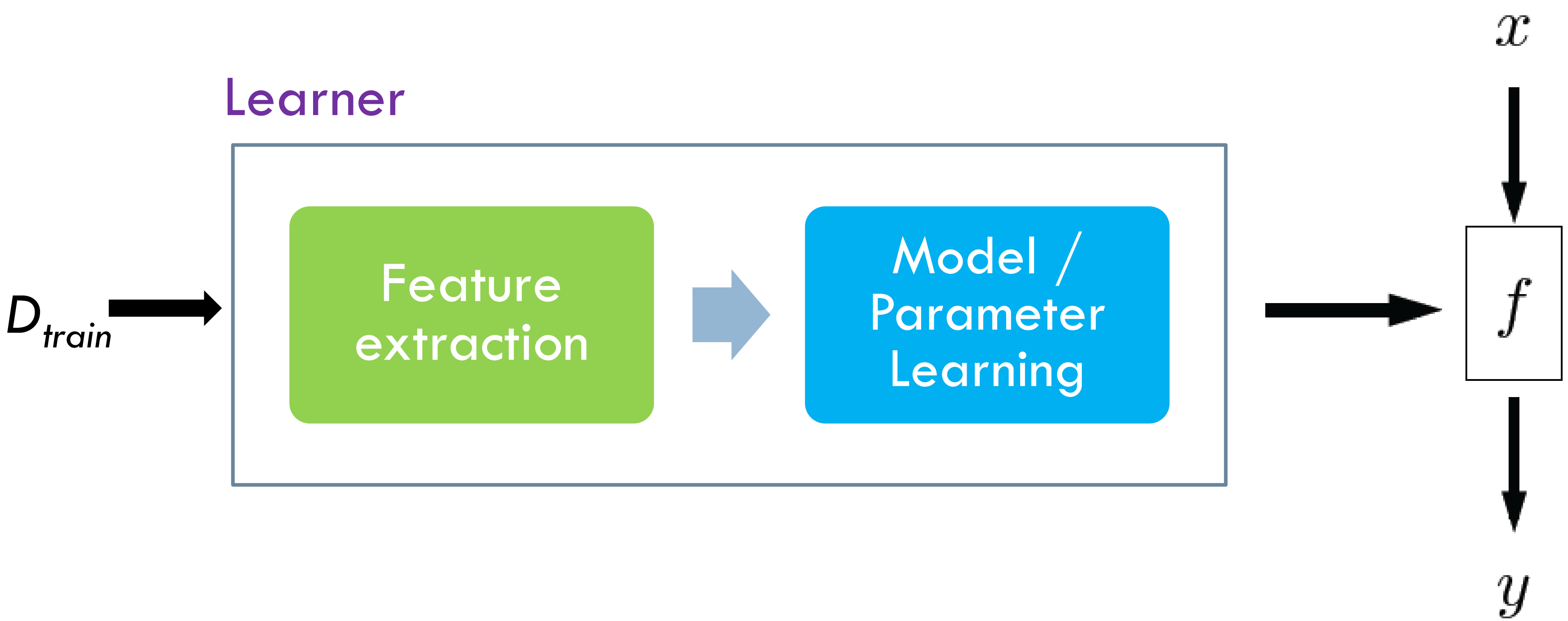
K nearest neighbor classifier
最近邻模型

Section 20.4

20

Linear predictions
线性预测

Learning Framework



Focus of this part

- Binary classification (e.g., predicting spam or not spam):

$$x \longrightarrow \boxed{f} \longrightarrow y \in \{-1, +1\}$$

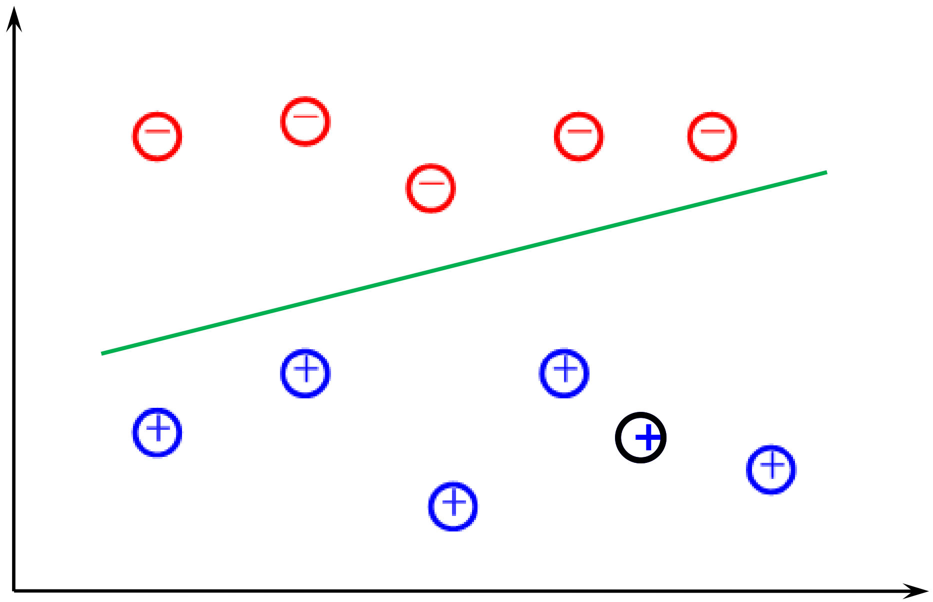
- Regression (e.g., predicting housing price):

$$x \longrightarrow \boxed{f} \longrightarrow y \in \mathbb{R}$$

Classification

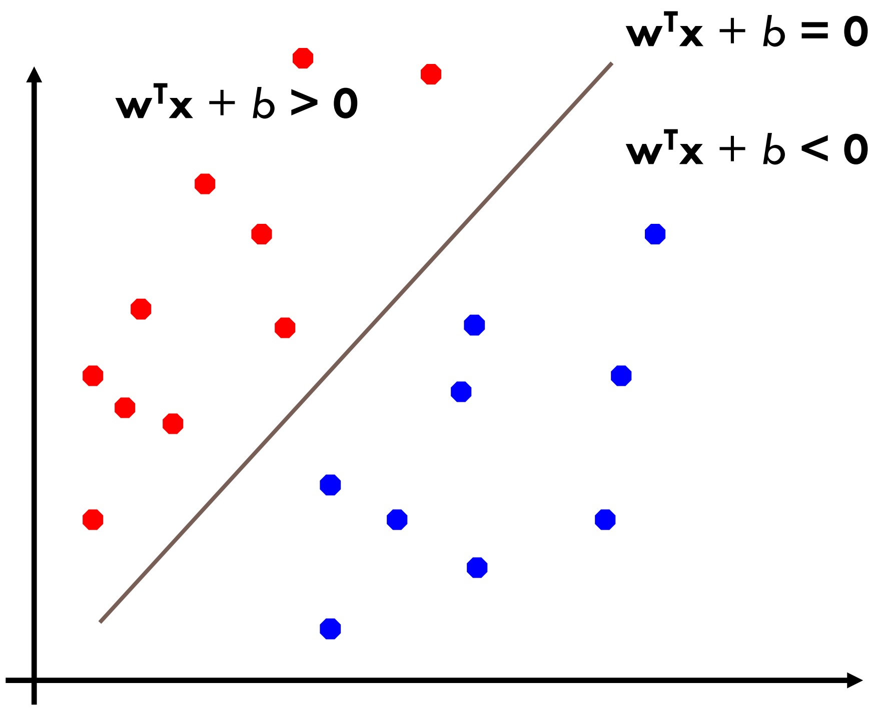
Classification

= learning from data with finite discrete labels. 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 $\mathbf{w}^T\mathbf{x} + b > 0$,
otherwise $\hat{y} = -1$

$$\hat{y} = h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Roadmap

Linear Prediction

Loss Minimization

Linear Classifiers

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

- 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: $\mathcal{X} \rightarrow \mathcal{Y}$, which is

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

Gold standard (ideal case)

Linear Classifiers → Loss Minimization

Ideally we want to find a classifier

$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$ to minimize the 0/1 loss

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

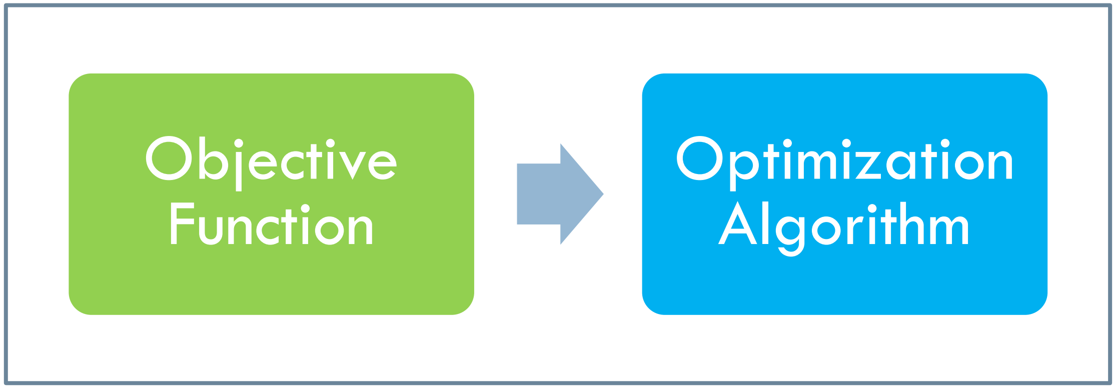
Unfortunately, this is a hard problem..

Alternate loss functions:

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

Learning as Optimization

Parameter Learning



Least Squares Classification

Least squares loss function:

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

The goal:

to learn a classifier $h(\mathbf{x}) = \text{sign}(\mathbf{w}^\mathsf{T}\mathbf{x} + b)$ to minimize the least squares loss

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

Solving Least Squares Classification

Let

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

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

Solving for w

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

Note: $d(\mathbf{Ax}+\mathbf{b})^T\mathbf{C}(\mathbf{Dx}+\mathbf{e}) = ((\mathbf{Ax}+\mathbf{b})^T\mathbf{CD} + (\mathbf{Dx}+\mathbf{e})^T\mathbf{C}^T\mathbf{A})\, d\mathbf{x}$

$d(\mathbf{Ax}+\mathbf{b})^T(\mathbf{Ax}+\mathbf{b}) = (2(\mathbf{Ax}+\mathbf{b})^T\mathbf{A})\, d\mathbf{x}$

- $X^+ = (X^\top X)^{-1}X^\top$ is called the *Moore-Penrose* pseudoinverse（伪逆）of X
- Least squares classification in Matlab

```
% X(i: ,) is the i-th example, y(i) is the i-th label
wLSQ = pinv([ones(size(X, 1), 1) X])*y;
```

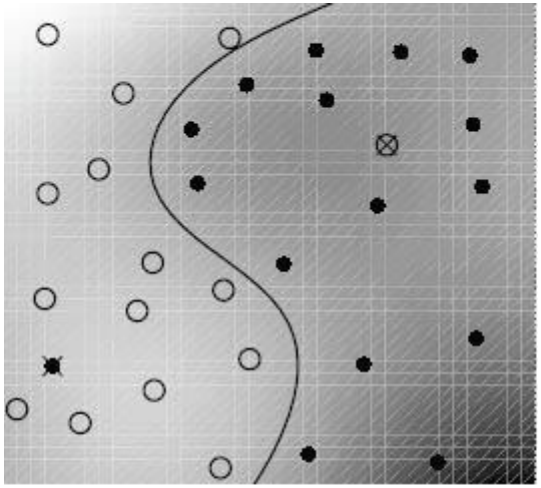
- Prediction for $\mathbf{x}_0$

$$\hat{y} = \operatorname{sign}\left(\mathbf{w}^{*\top}\begin{bmatrix}1\\ \mathbf{x}_0\end{bmatrix}\right) = \operatorname{sign}\left(\mathbf{y}^\top X^{+\top}\begin{bmatrix}1\\ \mathbf{x}_0\end{bmatrix}\right)$$

General linear classification

Basis (nonlinear) functions （基函数）

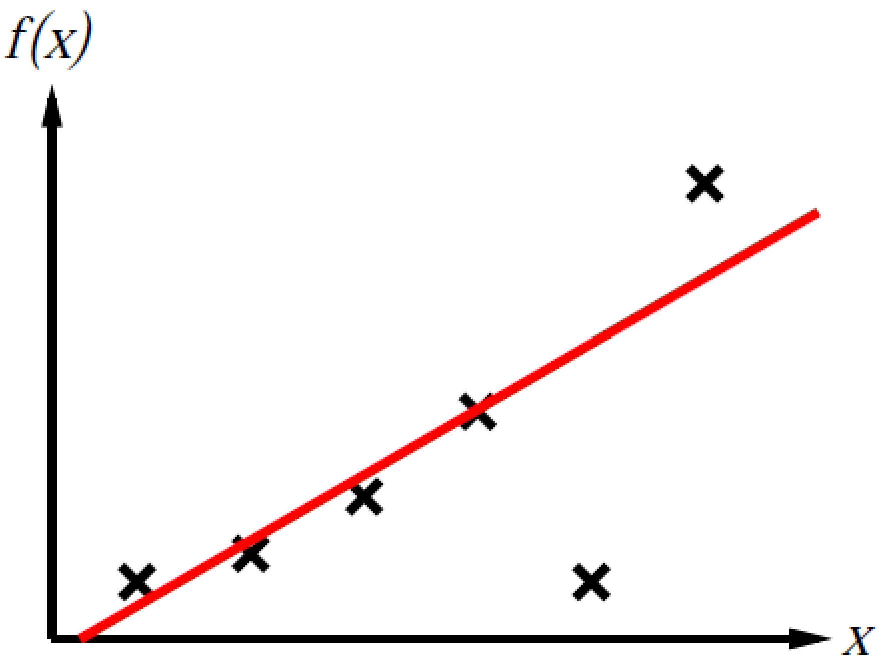
$$f(\mathbf{x}, \mathbf{w}) = b + w_1\phi_1(\mathbf{x}) + w_2\phi_2(\mathbf{x}) + \cdots + w_m\phi_m(\mathbf{x})$$



Regression（回归）

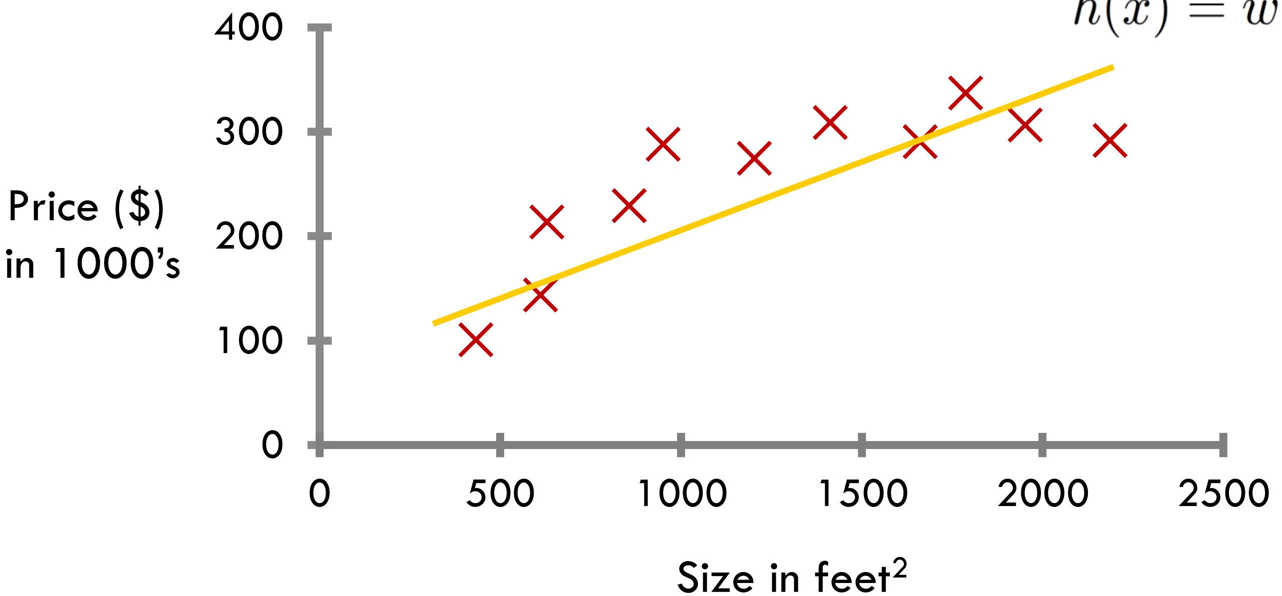
Regression

= learning from continuously labeled data.



Linear Regression

$$h(x) = w_1 \cdot x + b$$

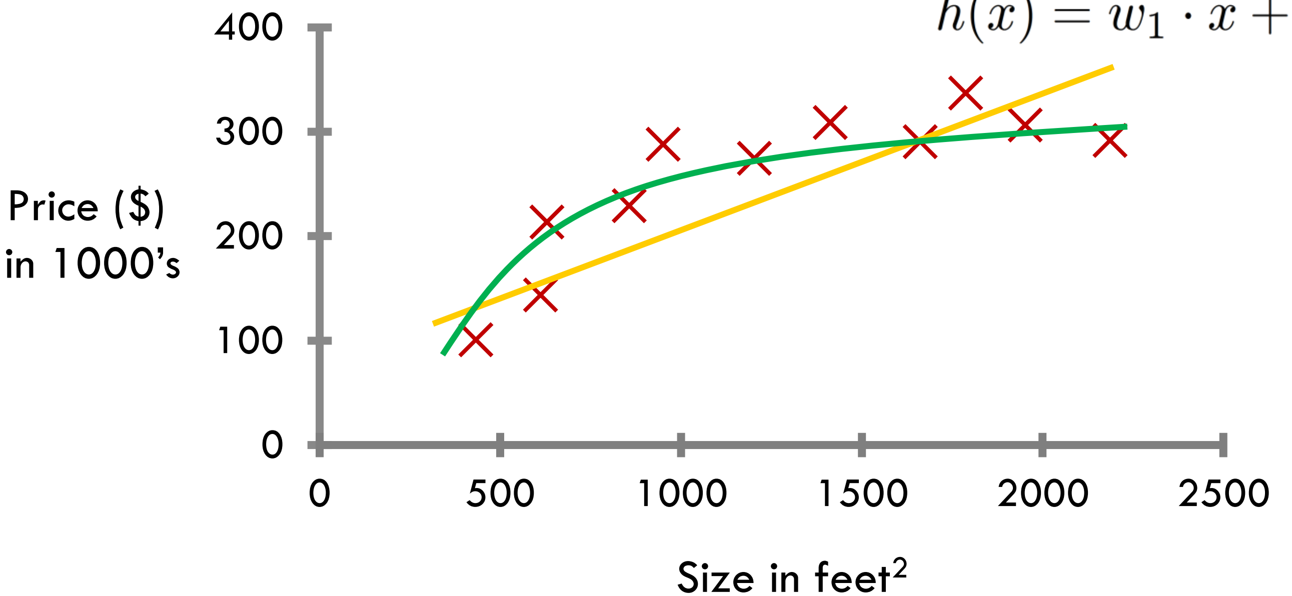


$$\begin{aligned} Loss &= \min_{\mathbf{w},b} \sum_i L_2(h(x_i), y_i) \\ &= \min_{\mathbf{w},b} \sum_i (y_i - w_1 \cdot x_i - b)^2 \end{aligned}$$

Least Squares

General Linear/Polynomial Regression

$$h(x) = w_1 \cdot x + w_2 \cdot x^2 + b$$

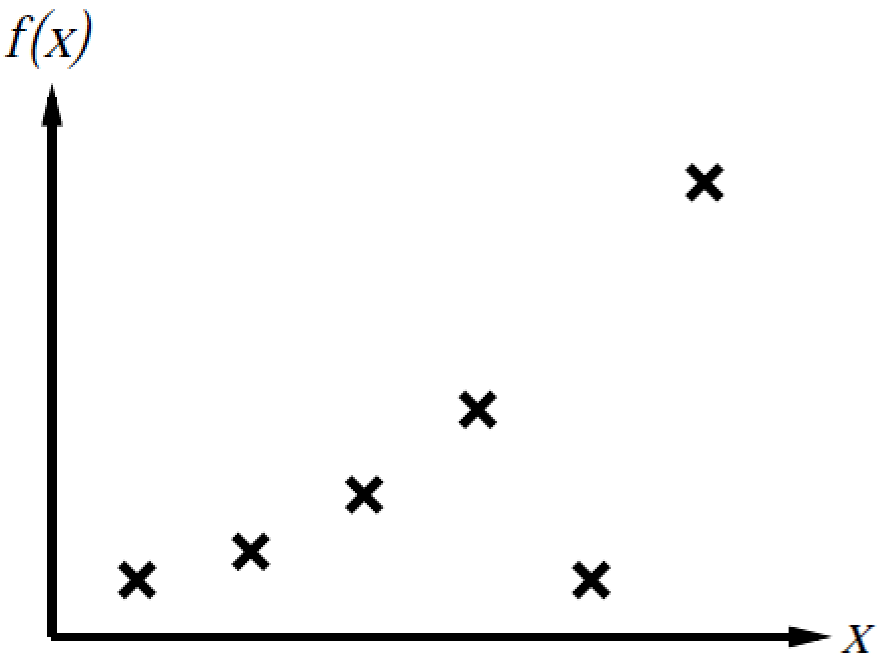


$$\begin{aligned} Loss &= \min_{\mathbf{w},b} \sum_i L_2(h(x_i), y_i) \\ &= \min_{\mathbf{w},b} \sum_i (y_i - w_1 \cdot x_i - w_2 \cdot x_i^2 - b)^2 \end{aligned}$$

Least Squares

Model complexity and overfitting

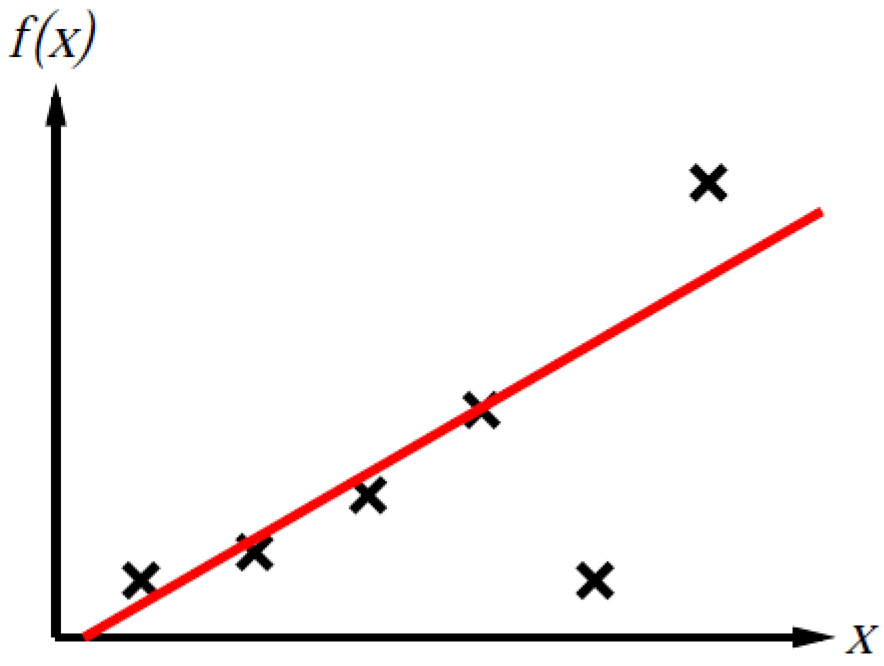
E.g., curve fitting（曲线拟合）：



Model complexity and overfitting

E.g., curve fitting（曲线拟合）：

$$f(x) = w_1 \cdot x + b$$



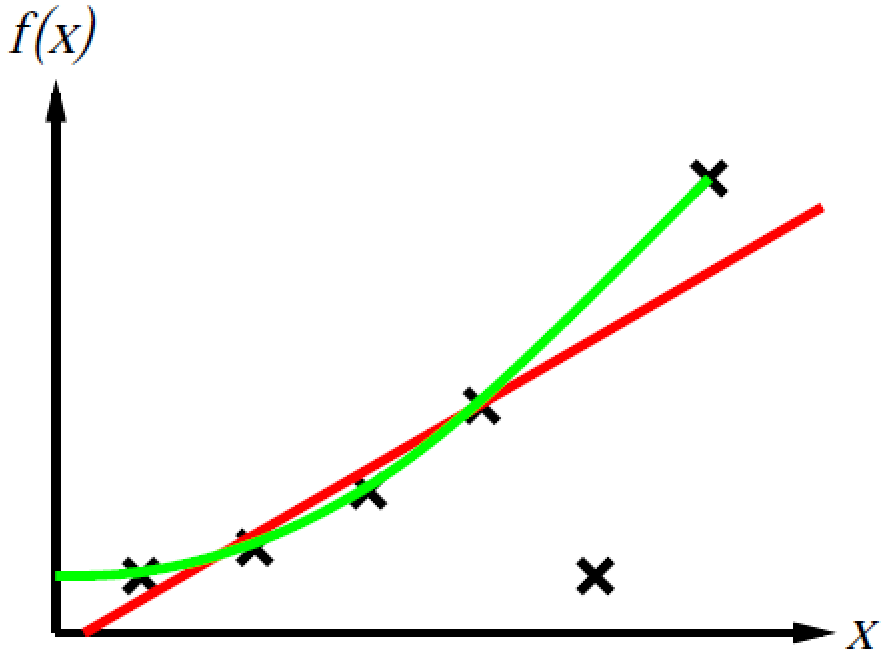
Underfitting

High **Bias**

Model complexity and overfitting

E.g., curve fitting（曲线拟合）：

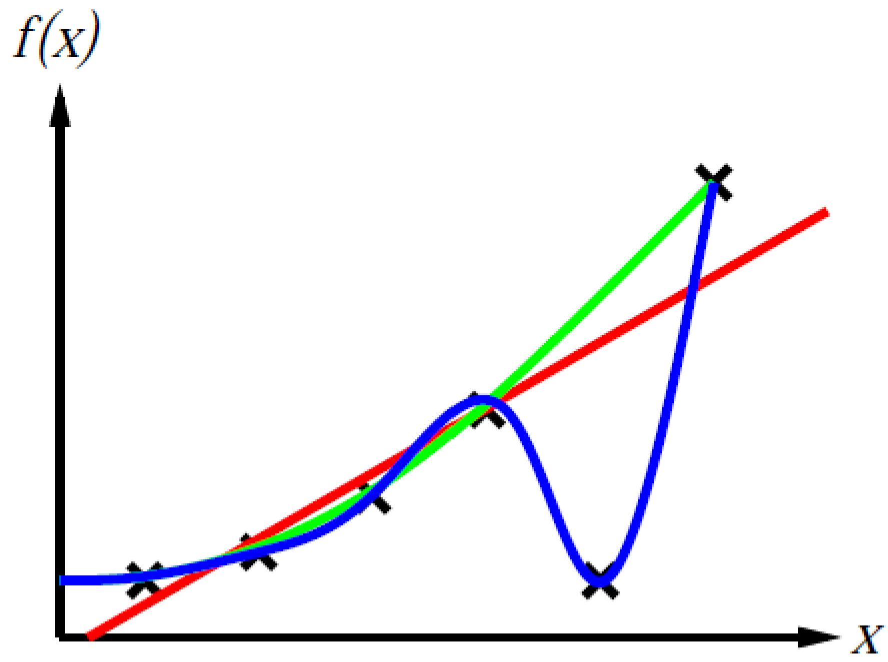
$$f(x) = w_1 \cdot x + w_2 \cdot x^2 + b$$



Model complexity and overfitting

E.g., curve fitting（曲线拟合）：

$$f(x) = w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5 + b$$



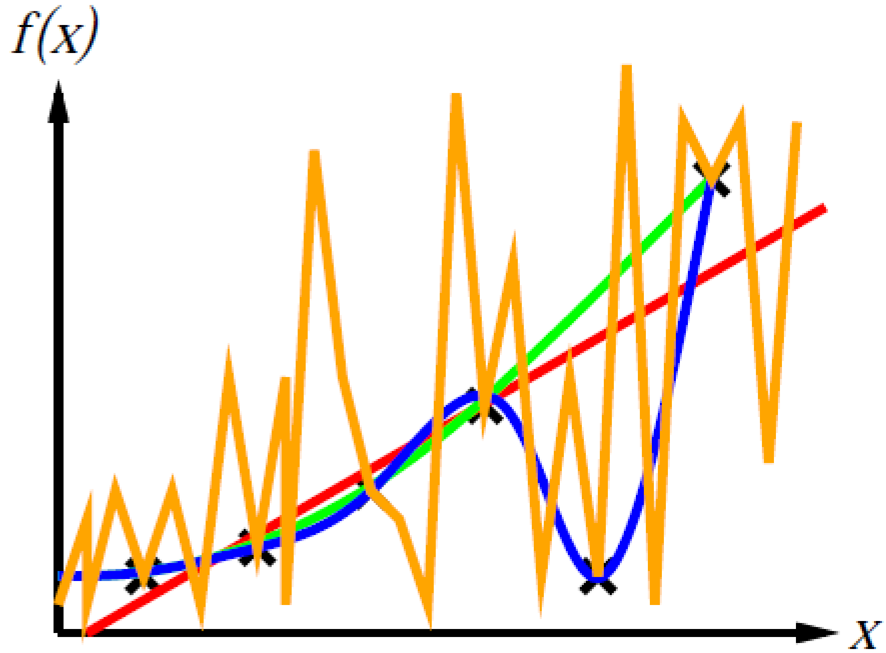
Overfitting

High **Variance**

Model complexity and overfitting

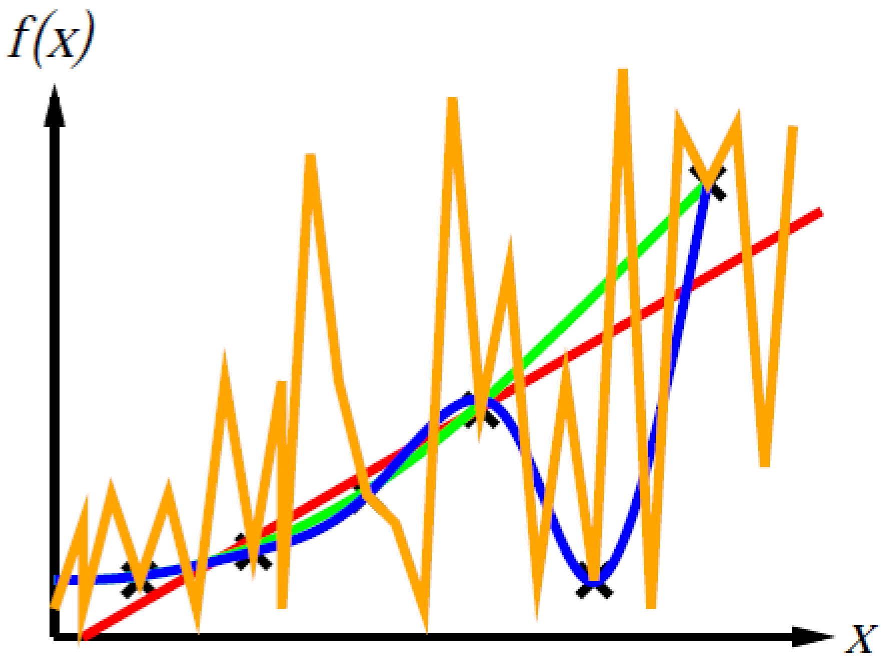
E.g., curve fitting（曲线拟合）：

$$f(x) = w_1 \cdot x + w_2 \cdot x^2 + \ldots + w_n \cdot x^n + b$$



Model complexity and overfitting

E.g., curve fitting（曲线拟合）:



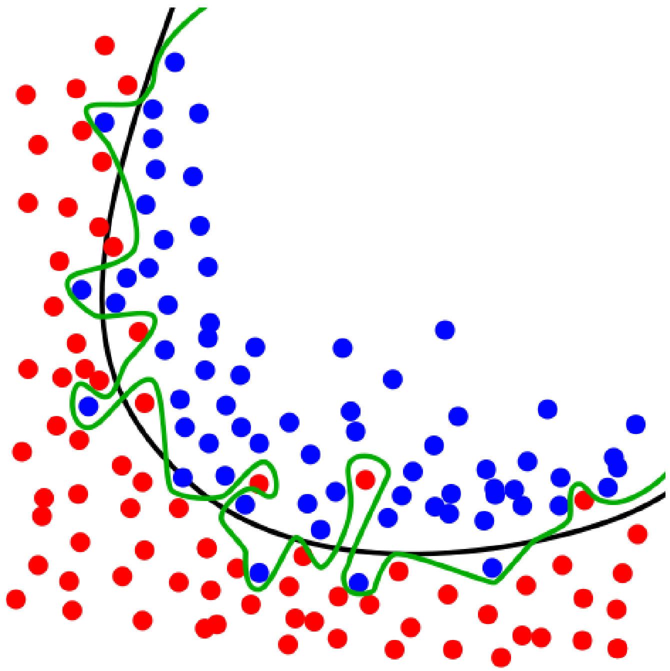
Ockham's razor（奥卡姆剃刀原则）: maximize a combination of consistency and simplicity

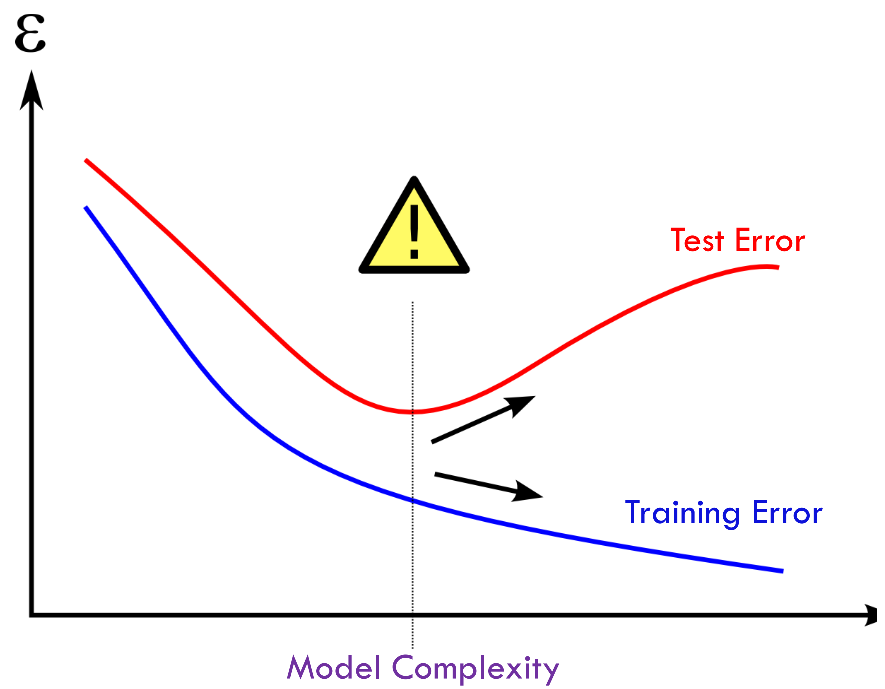
优先选择与数据一致的最简单的假设

Prediction Errors

- Training errors (apparent errors) — 训练误差
  - Errors committed on the training set
- Test errors — 测试误差
  - Errors committed on the test set
- Generalization errors — 泛化误差
  - Expected error of a model over random selection of records from same distribution（未知记录上的期望误差）

Model complexity and overfitting





Underfitting: when model is too simple, both training and test errors are large

Overfitting: when model is too complex, training error is small but test error is large

Incorporating Model Complexity

- Rationale: Ockham's Razor
  - Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
  - A complex model has a greater chance of being fitted accidentally by errors in data
  - Therefore, one should include model complexity when evaluating a model

Regularization（规范化）

Intuition: small values for parameters

- "Simpler" hypothesis
- Less prone to overfitting

$$L_p - norm: \|\boldsymbol{v}\|_p = \left(\sum_i |v_i|^p\right)^{1/p}$$

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

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

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

Regularization parameter

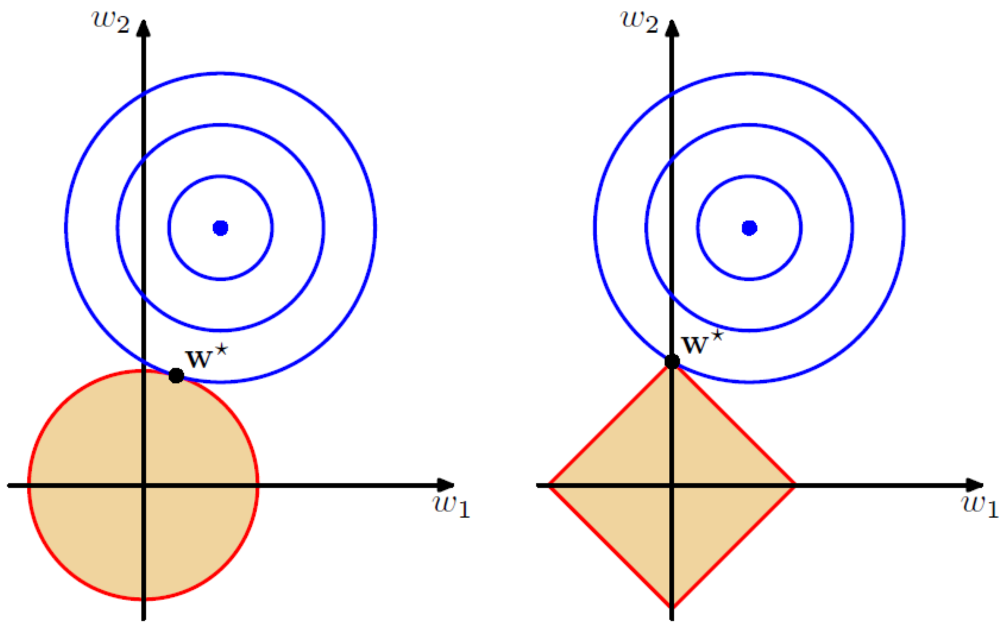
- Solving L2-regularized LS

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

Solution?

L-2 and L-1 regularization

- L-2: easy to optimize, closed form solution
- L-1: sparsity



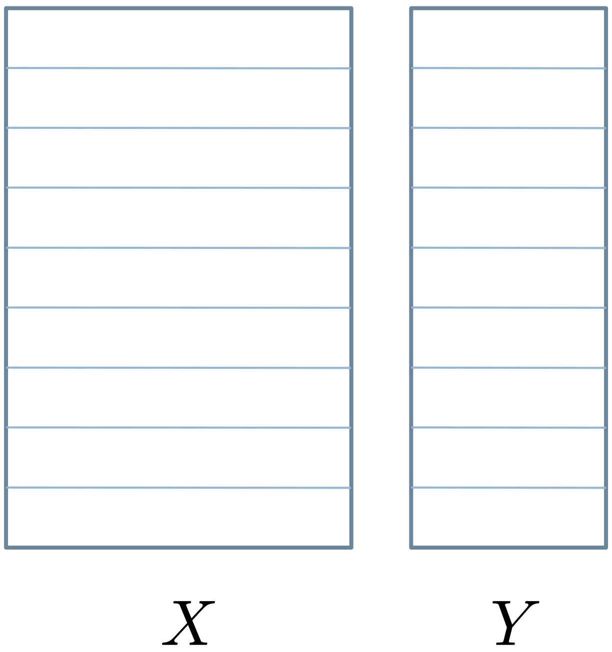
More than two classes?

Given

- $N \times d$ data matrix X
- $N\times\ k$ label matrix Y
- $N = \#$ training instances
- $d = \#$ features
- $k = \#$ targets

Assume

- $k < d$



More than two classes

- Learn:
  - parameters W ($d \times k$) for a model $f_W : X \to Y$
- Objective $\min_W \, tr\left((XW - Y)(XW - Y)^\top\right)$
  - A convex quadratic, so just solve for a critical point:

$$\frac{d}{dW} = 2X^\top(XW - Y) = 0$$

  - Thus

$$\begin{aligned} X^\top X W &= X^\top Y \\ W &= (X^\top X)^{-1} X^\top Y = X^\dagger Y \end{aligned}$$

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

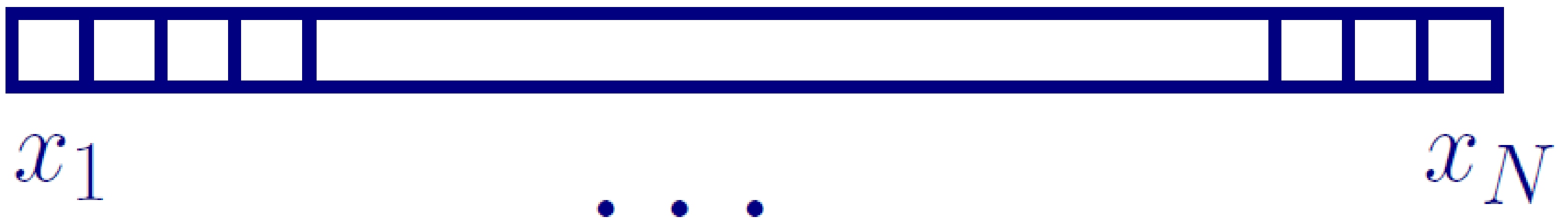
Cross-validation（交叉验证）

- 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 heldout set.

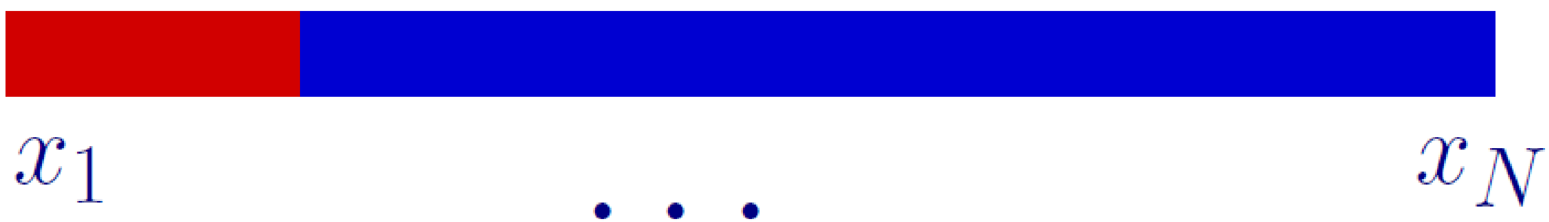
Cross-validation

- 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

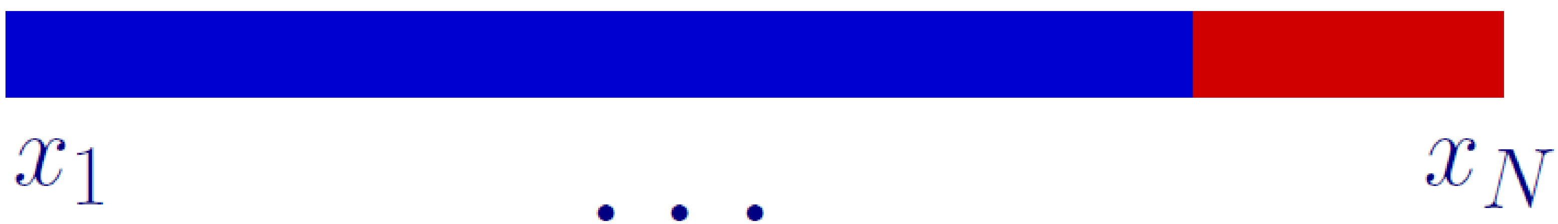


Cross-validation

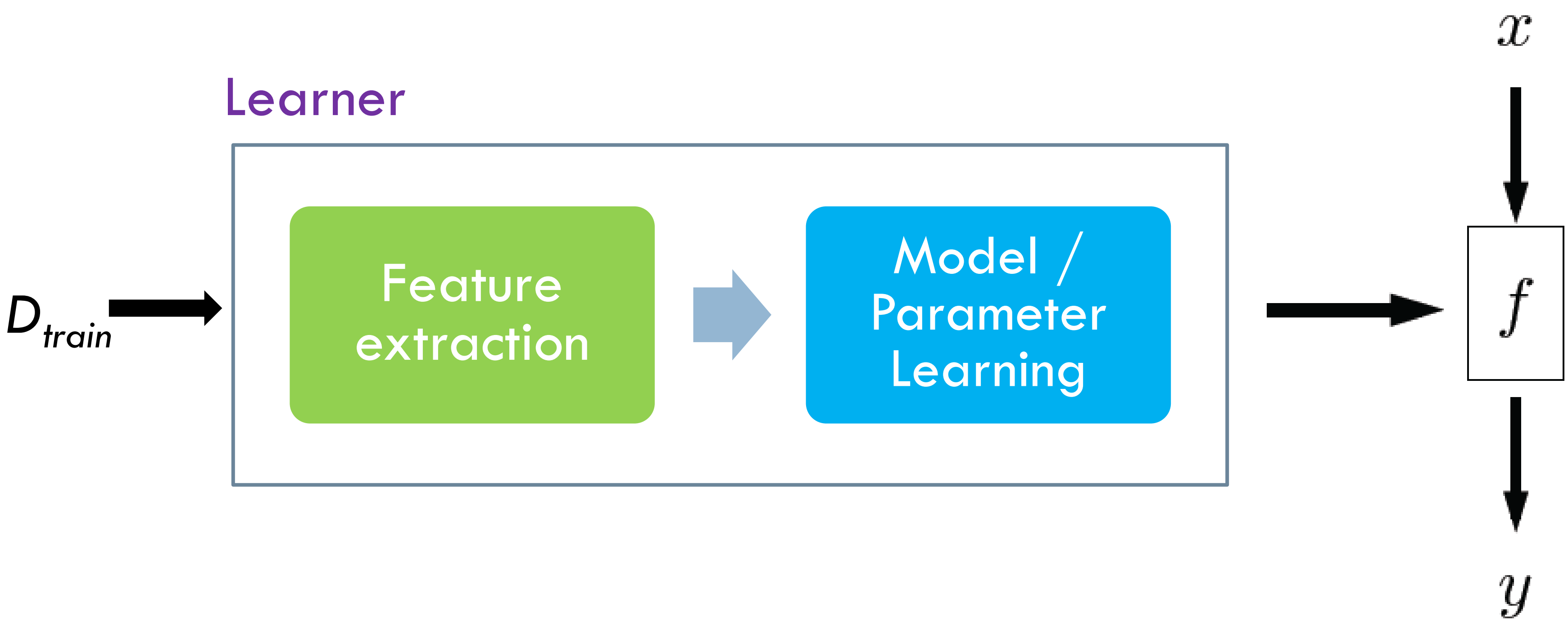
- The improved holdout method: k-fold *cross-validation*
  - Partition data into k roughly equal parts;
  - Train on all but j-th part, test on j-th part

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



Learning Framework



Model/parameter learning paradigm

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

Summary

Supervised learning

- Classification
  - Naïve Bayes model
  - Decision tree
  - Least squares classification
- Regression
  - Least squares regression

作业

- 试证明对于不含冲突数据（即特征向量完全相同但标记不同）的训练集，必存在与训练集一致（即训练误差为0）的决策树。