数字图像处理

第A章

机器学习/模式识别基础

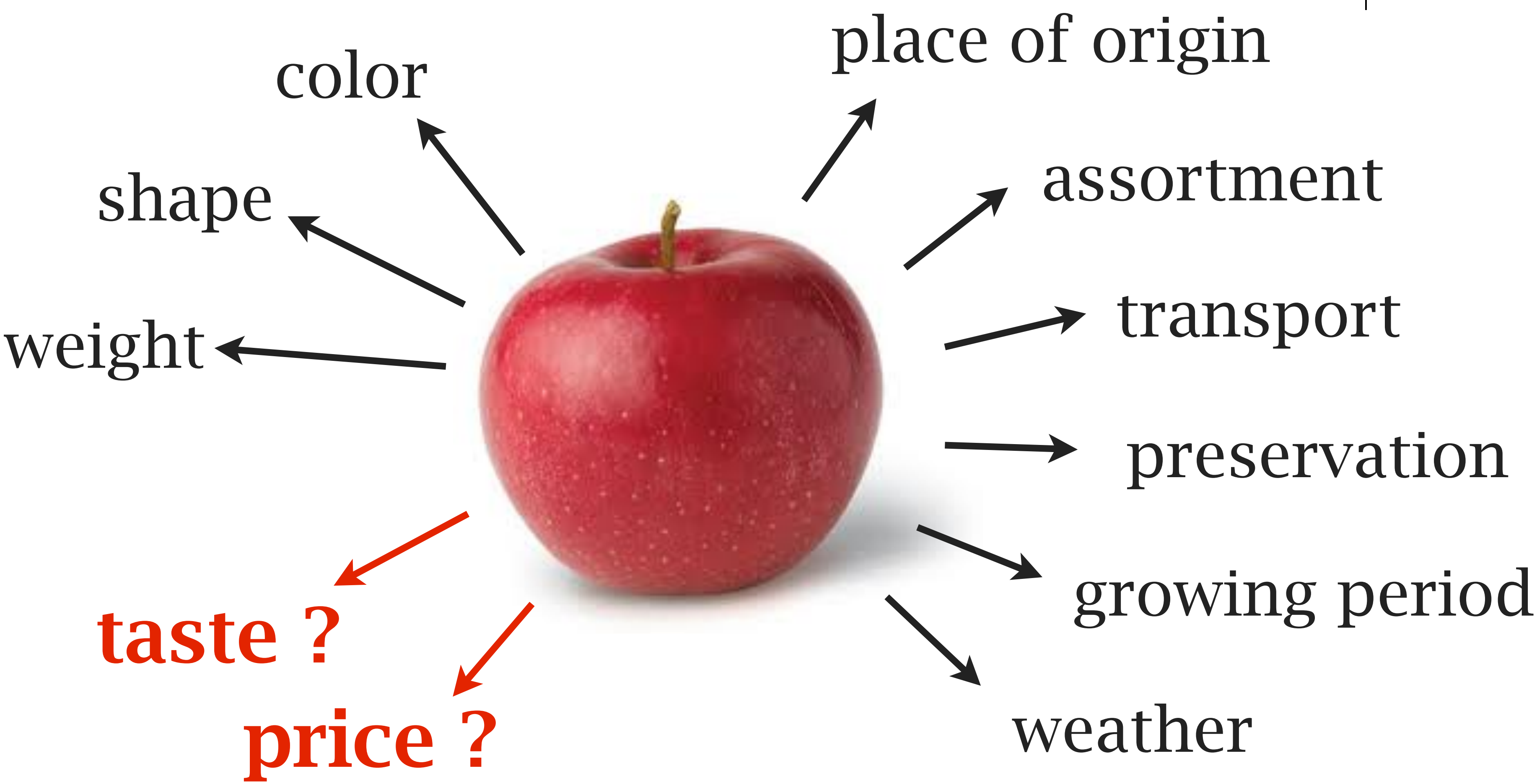
机器学习/模式识别基础

- 预测与识别
- 预测算法
- 特征提取

1 预测与识别

- 预测：根据当前的观测，预测未观测事件
- 识别：根据当前的观测，判断是否是预定模式
- 如何定义“事件”或“模式”?
- 机器学习方法：基于数据的定义

1 预测与识别


color
shape
weight
taste ?
price ?
place of origin
assortment
transport
preservation
growing period
weather

1 预测与识别

color={0,1,2,3} weight={0,1,2,3,4}

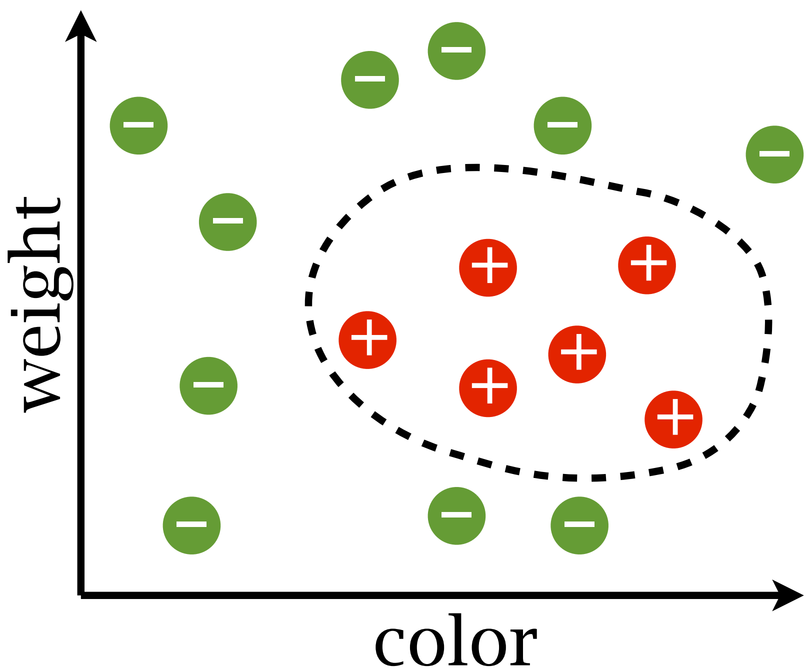
color	weight	sweet?
3	4	yes
2	3	yes
0	3	no
3	2	no
1	4	no

机器学习/模式识别基础

- 预测与识别
- 预测算法
- 特征提取

Classification

Features: color, weight
Label: taste is sweet (positive/+) or not (negative/-)



(color, weight) $\rightarrow$ sweet ?

$$\mathcal{X} \rightarrow \{-1, +1\}$$

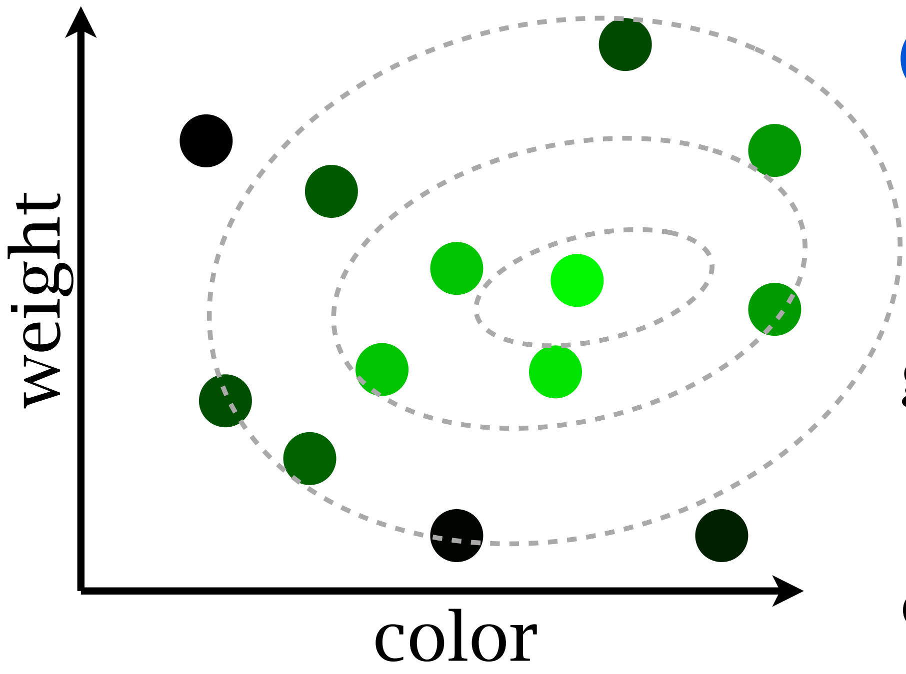
ground-truth function f

examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$$y_i = f(\boldsymbol{x}_i)$$

Regression

Features: color, weight
Label: sweetness [0,1]



(color, weight) → sweetness

$\mathcal{X} \rightarrow [-1, +1]$

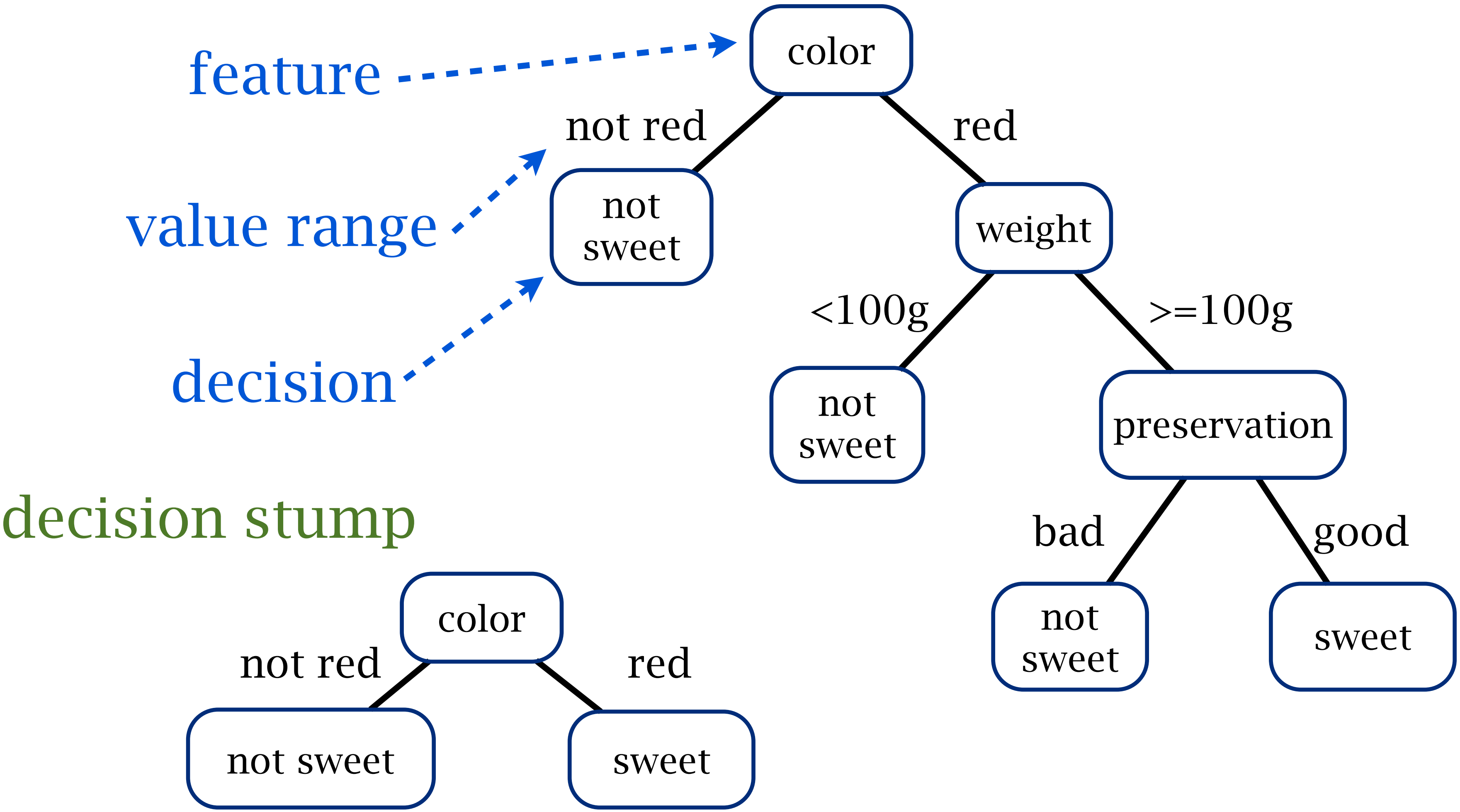
ground-truth function f

examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

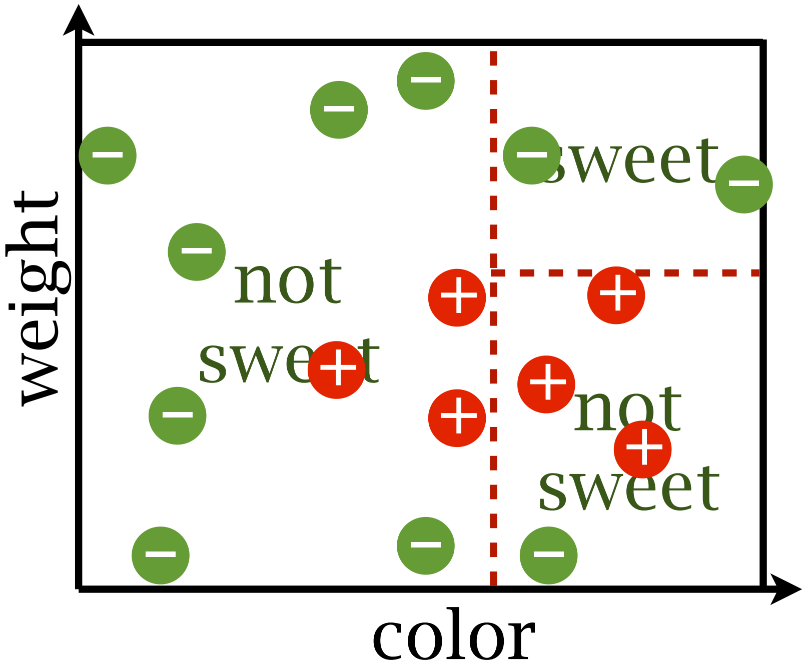
$y_i = f(\boldsymbol{x}_i)$

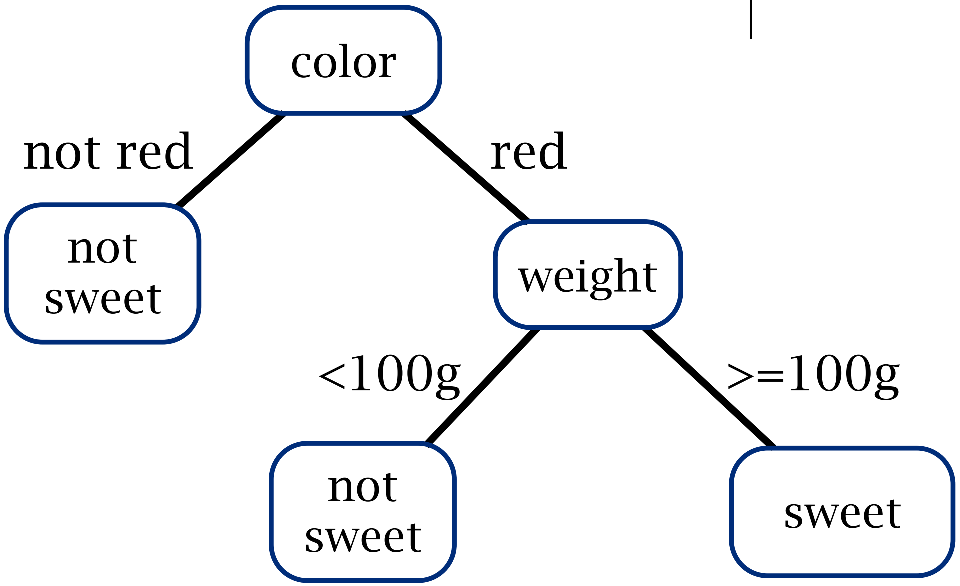
Decision tree model

decision process
with a tree structure



Decision tree model

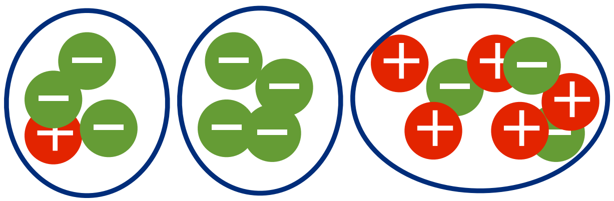


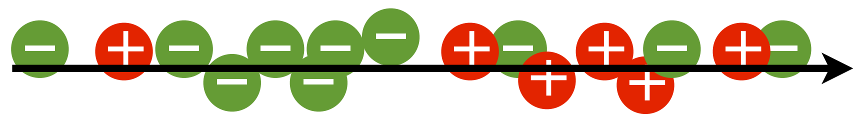


find a decision tree that matches the data?

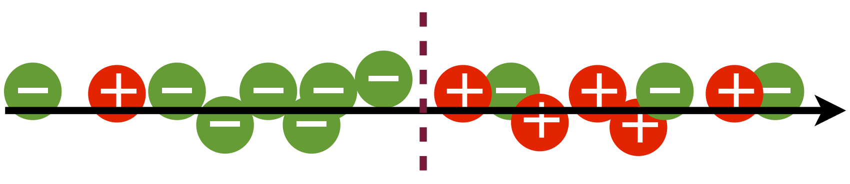
Split-criterion: classification

for every possible split of every feature:





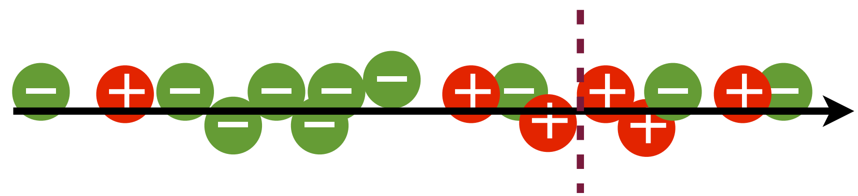
Training error:



prediction: –
error: 1

prediction: +
error: 3

total error: 4

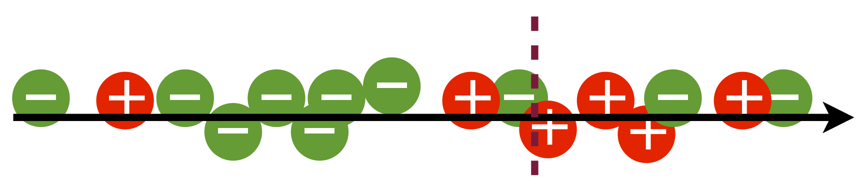


prediction: –
error: 3

prediction: +
error: 2

total error: 5

total error: 4



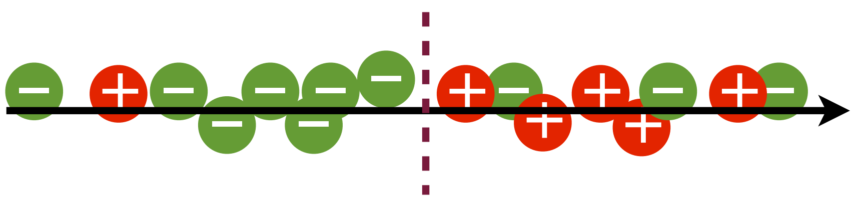
Split-criterion: classification

Information gain (ID3):

Entropy: $H(X) = -\sum_i p_i \ln(p_i)$

Entropy after split: $I(X;\text{split}) = \frac{\#\text{left}}{\#\text{all}} H(\text{left}) + \frac{\#\text{right}}{\#\text{all}} H(\text{right})$

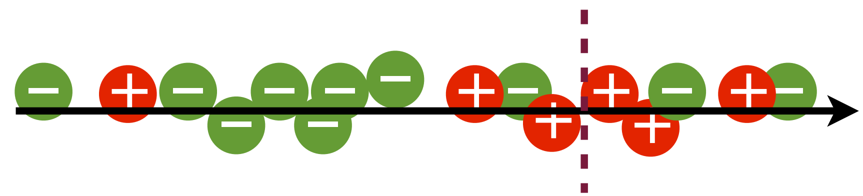
Information gain: $H(X)$-$I(X;\text{split})$



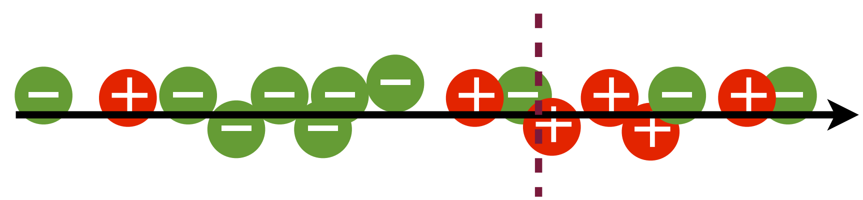
$$H(\text{left}) = -\frac{1}{8}\ln\frac{1}{8} - \frac{7}{8}\ln\frac{7}{8} = 0.3768$$

$$H(\text{right}) = -\frac{5}{8}\ln\frac{5}{8} - \frac{3}{8}\ln\frac{3}{8} = 0.6616$$

$$\text{IG} = H(X) - (0.5 \times 0.3768 + 0.5 \times 0.6616)$$
$$= H(X) - 0.5192$$



$$\text{IG} = H(X) - 0.6132$$



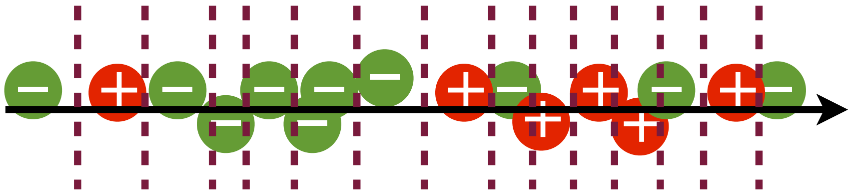
$$\text{IG} = H(X) - 0.5514$$

Split-criterion: classification

Gain ratio (C4.5):

$$\text{Gain ratio}(X) = \frac{H(X) - I(X;\text{split})}{IV(\text{split})}$$

$$IV(\text{split}) = H(\text{split})$$



e.g. student ID

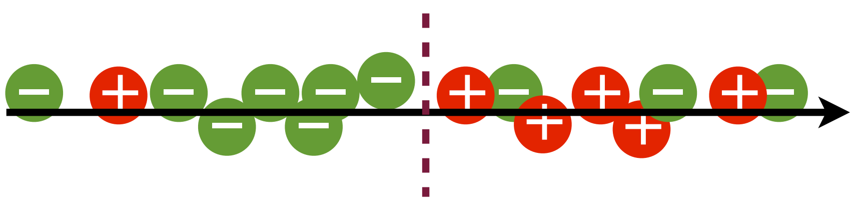
$$\text{IG} = H(X) - 0$$

Split-criterion: classification

Gini index (CART):

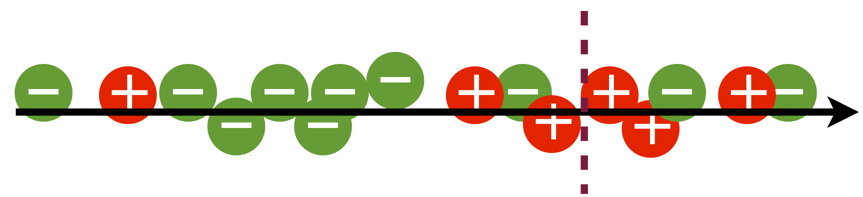
Gini: $Gini(X) = 1 - \sum_i p_i^2$

Gini after split: $\frac{\#\text{left}}{\#\text{all}} Gini(\text{left}) + \frac{\#\text{right}}{\#\text{all}} Gini(\text{right})$



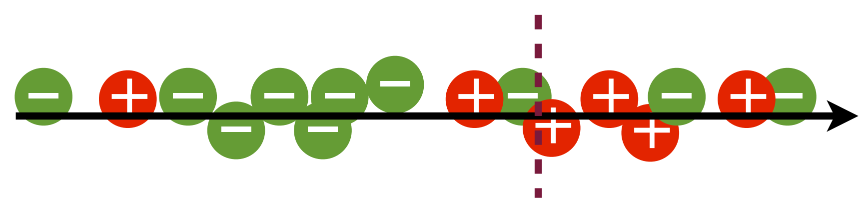
$\text{IG} = H(X) - 0.5192$

$Gini = 0.3438$



$\text{IG} = H(X) - 0.6132$

$Gini = 0.4427$

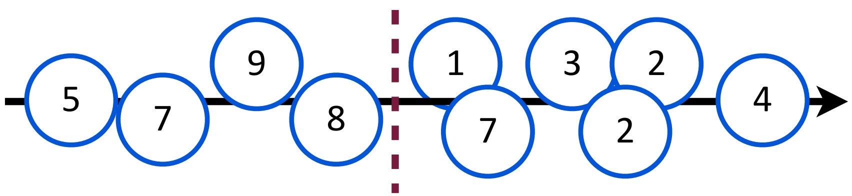


$\text{IG} = H(X) - 0.5514$

$Gini = 0.3667$

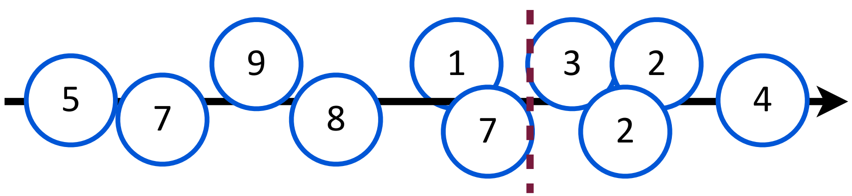
Split-criterion: regression

Training error:



mean: 7.25 mean: 3.1667

MSE: 8.75+22.83=31.583



mean: 6.1667 mean: 2.75

MSE: 43.5833

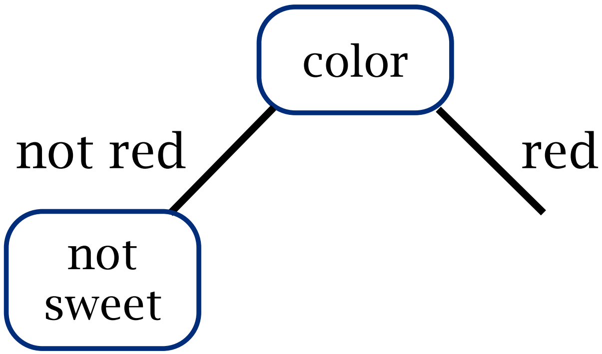
Split-criterion: stop

Stop criterion:
no feature to use

Classification: examples are pure of class

Regression: variance small enough

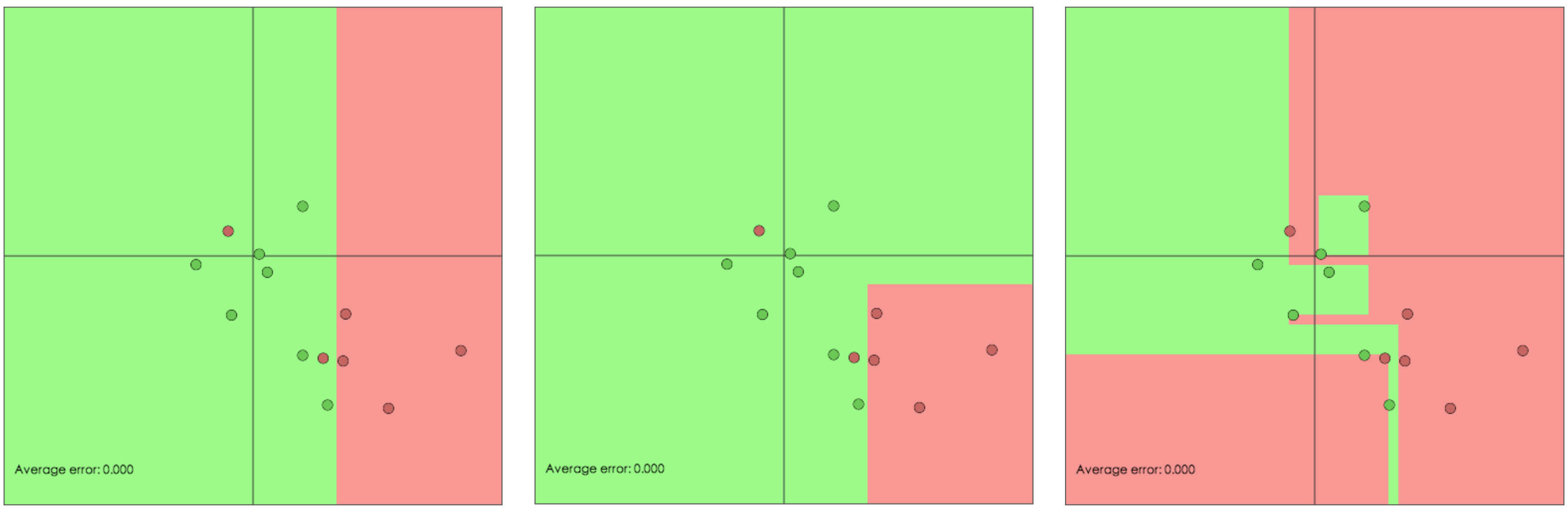
Make-leaf



Classification: major class

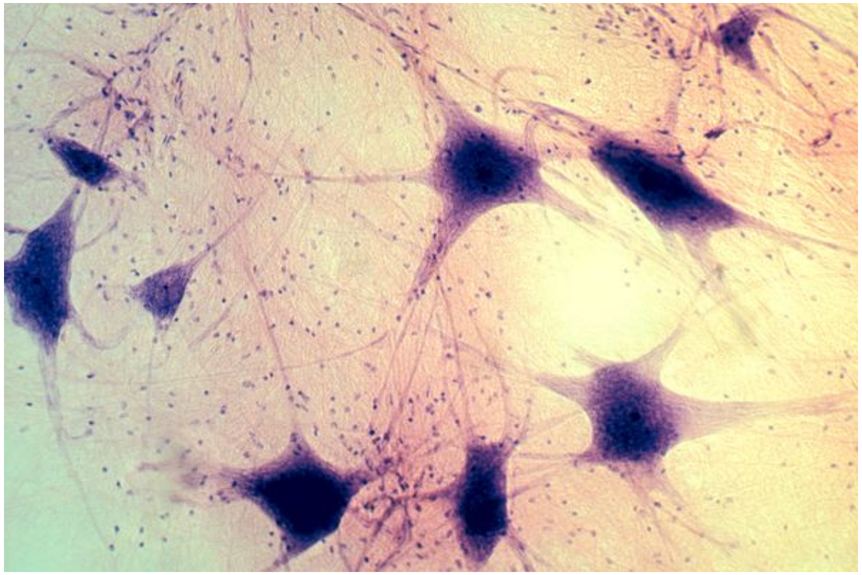
Regression: mean value

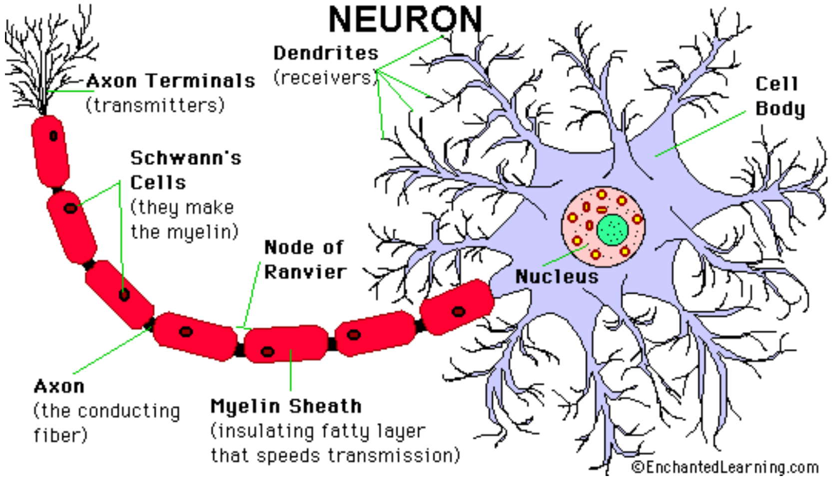
DT boundary visualization



decision stump max depth=2 max depth=12

Neural networks





Neuron / perceptron

output a function of sum of input

linear function:

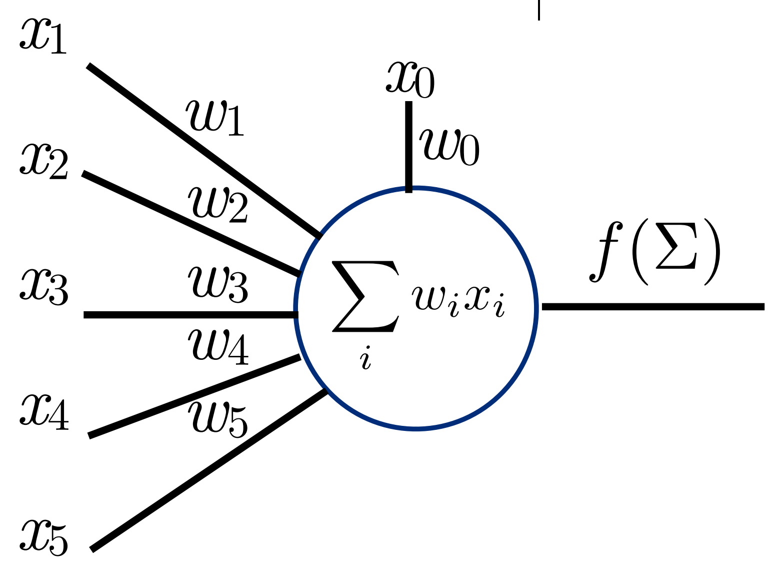
$$f(\sum_i w_i x_i) = \sum_i w_i x_i$$

threshold function:

$$f(\sum_i w_i x_i) = I(\sum_i w_i x_i > 0)$$

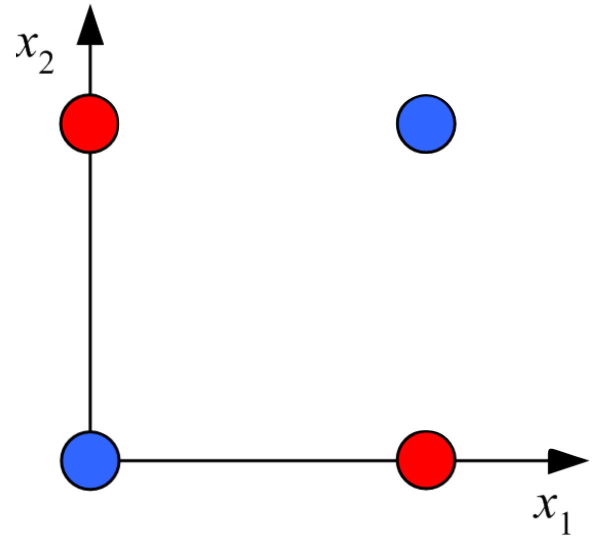
sigmoid function:

$$f(\sum_i w_i x_i) = \frac{1}{1 + e^{-\Sigma}}$$



Limitation of single neuron

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0



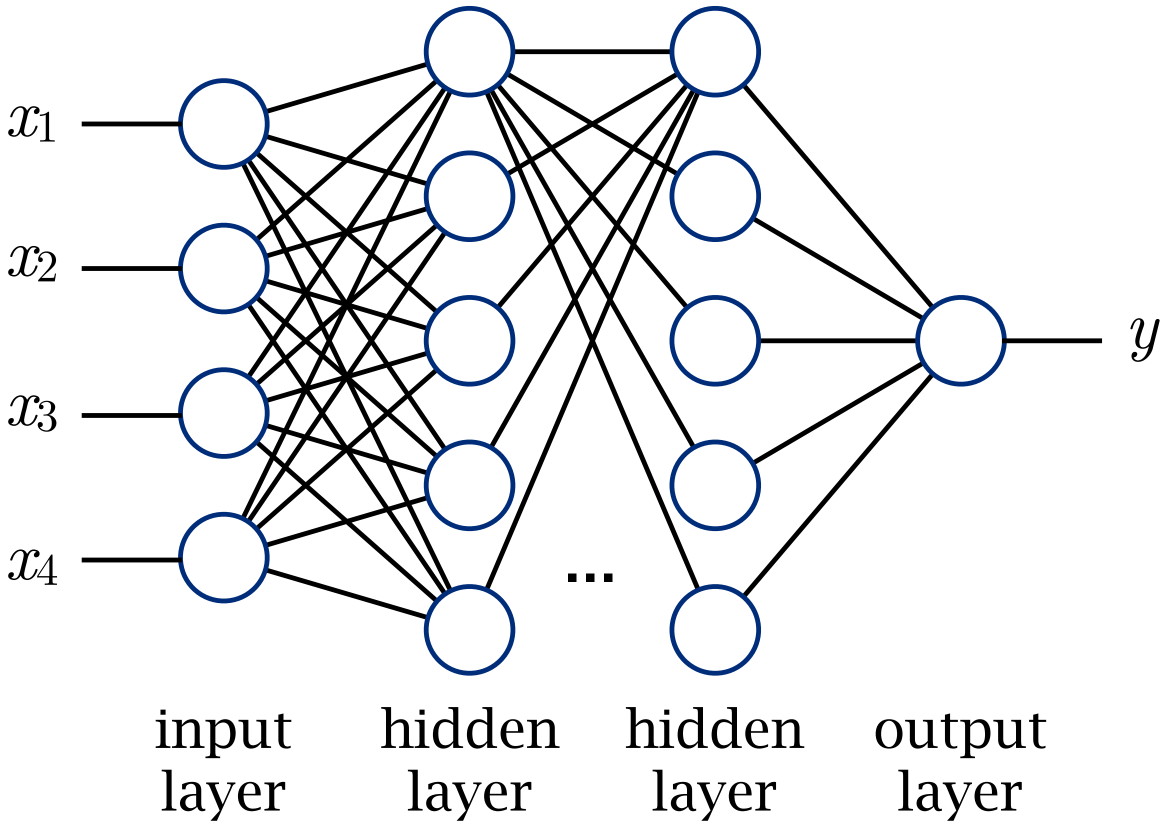
[Minsky and Papert, *Perceptrons,* 1969]

Marvin Minsky
Turing Award 1969

AI Winter

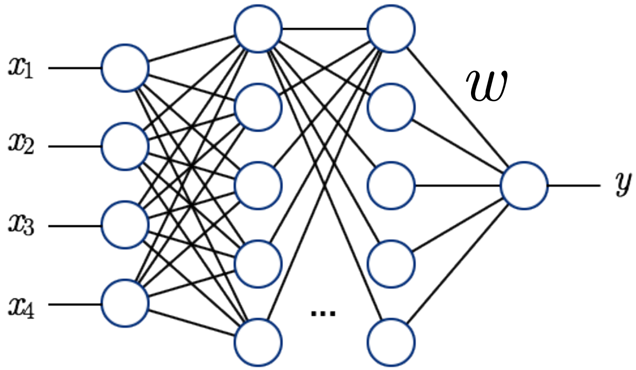
Multi-layer perceptrons

feed-forward network



sigmoid network with one hidden layer can approximate arbitrary function [Cybenko 1989]

Back-propagation algorithm



$$\hat{y} = F(\boldsymbol{x})$$

gradient descent

error: $E(\boldsymbol{w}) = (F(\boldsymbol{x}) - y)^2$

update one weight: $\Delta w_{i,j} = -\eta \frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}}$

weight of the laster layer

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}} = \frac{\partial E(\boldsymbol{w})}{\partial F(\boldsymbol{x})} \frac{\partial F(\boldsymbol{x})}{\partial w_{i,j}}$$

weight of the first layer

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}} = \frac{\partial E(\boldsymbol{w})}{\partial F(\boldsymbol{x})} \frac{\partial F(\boldsymbol{x})}{\partial \mathrm{HL2}} \frac{\partial \mathrm{HL2}}{\partial \mathrm{HL1}} \frac{\partial \mathrm{HL1}}{\partial w_{i,j}}$$

[Rumelhart, Hinton, Williams, Nature 1986]

Back-propagation algorithm

For each given training example (**x**, **y**), **do**

1. Input the instance **x** to the NN and compute the output value o_u of every output unit u of the network

2. **For each** network output unit k, calculate its error term δ_k

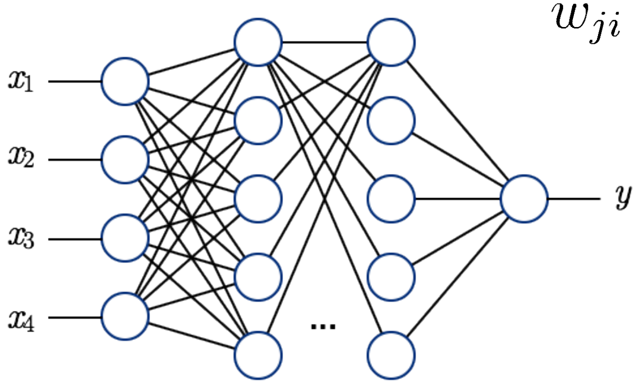
$$\delta_k \leftarrow o_k(1 - o_k)(y_k - o_k)$$

3. **For each** hidden unit k, calculate its error term δ_h

$$\delta_h \leftarrow o_k(1 - o_k) \sum_{k \in outputs} w_{kh} \delta_k$$

4. Update each network weight w_{ji} which is the weight associated with the i-th input value to the unit j

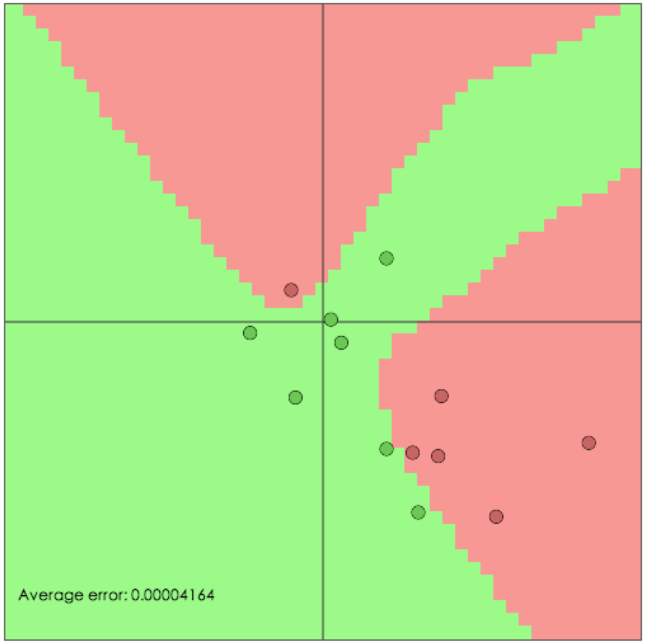
$$w_{ji} \leftarrow w_{ji} + \eta \delta_j x_{ji}$$



[Rumelhart, Hinton, Williams, Nature 1986]

Advantage and disadvantages

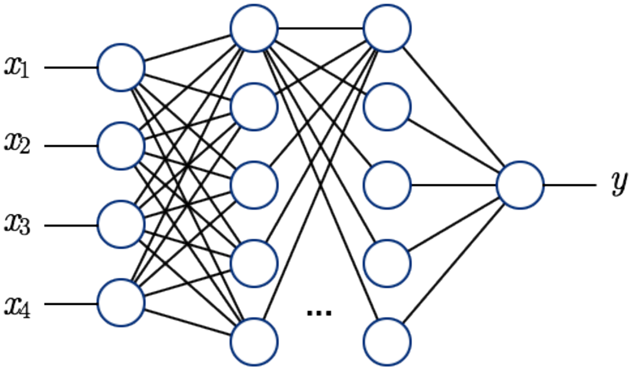
Smooth and nonlinear decision boundary



Slow convergence

Many local optima

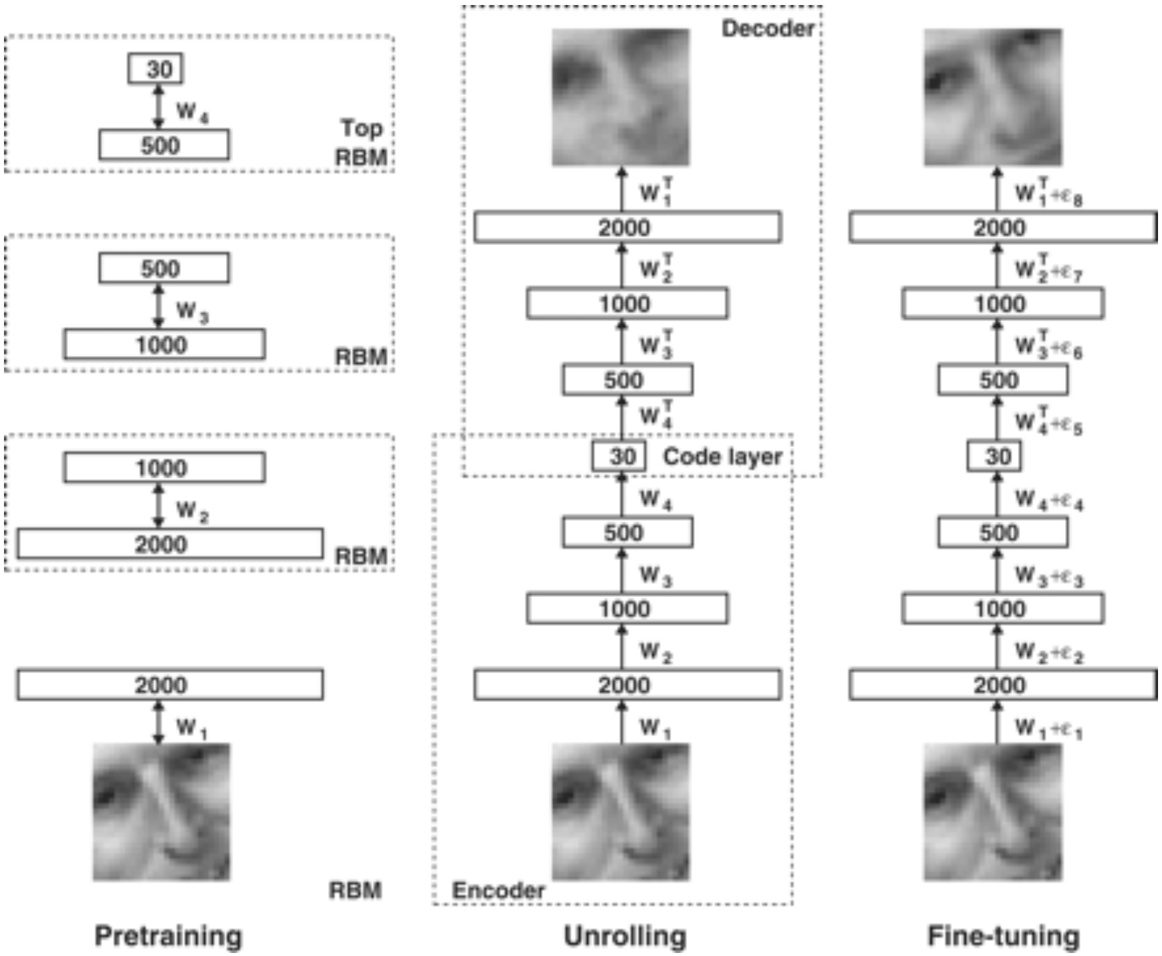
Best network structure unknown



Hard to handle nominal features

Deep network

autoencoder:



[Hinton and Salakhutdinov, Science 2006]

Bayes rule

classification using posterior probability

for binary classification

$$f(x) = \begin{cases} +1, & P(y=+1 \mid \boldsymbol{x}) > P(y=-1 \mid \boldsymbol{x}) \\ -1, & P(y=+1 \mid \boldsymbol{x}) < P(y=-1 \mid \boldsymbol{x}) \\ \text{random}, & otherwise \end{cases}$$

in general

$$\begin{aligned} f(x) &= \arg\max_y P(y \mid \boldsymbol{x}) \\ &= \arg\max_y P(\boldsymbol{x} \mid y) P(y) / P(\boldsymbol{x}) \\ &= \arg\max_y P(\boldsymbol{x} \mid y) P(y) \end{aligned}$$

how the probabilities be estimated

Naive Bayes

$$f(x) = \arg\max_y P(\boldsymbol{x} \mid y) P(y)$$

estimation the a priori by frequency:

$$P(y) \leftarrow \tilde{P}(y) = \frac{1}{m} \sum_i I(y_i = y)$$

assume features are conditional independence given the class (naive assumption):

$$\begin{aligned} P(\boldsymbol{x} \mid y) &= P(x_1, x_2, \ldots, x_n \mid y) \\ &= P(x_1 \mid y) \cdot P(x_2 \mid y) \cdot \ldots P(x_n \mid y) \end{aligned}$$

decision function:

$$f(x) = \arg\max_y \tilde{P}(y) \prod_i \tilde{P}(x_i \mid y)$$

Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

color	weight	sweet?
3	4	yes
2	3	yes
0	3	no
3	2	no
1	4	no

$$P(y = yes) = 2/5$$

$$P(y = no) = 3/5$$

$$P(color = 3 \mid y = yes) = 1/2$$

…

$$f(y \mid color = 3, weight = 3) \rightarrow$$

$$P(color = 3 \mid y = yes) P(weight = 3 \mid y = yes) P(y = yes) = 0.5 \times 0.5 \times 0.4 = 0.1$$

$$P(color = 3 \mid y = no) P(weight = 3 \mid y = no) P(y = no) = 0.33 \times 0.33 \times 0.6 = 0.06$$

$$f(y \mid color = 0, weight = 1) \rightarrow$$

$$P(color = 0 \mid y = yes) P(weight = 1 \mid y = yes) P(y = yes) = 0$$

$$P(color = 0 \mid y = no) P(weight = 1 \mid y = no) P(y = no) = 0$$

Naive Bayes

color={0,1,2,3} weight={0,1,2,3,4}

color	weight	sweet?
3	4	yes
2	3	yes
0	3	no
3	2	no
1	4	no

+

color	sweet?
0	yes
1	yes
2	yes
3	yes

smoothed (Laplacian correction) probabilities:

$$P(color = 0 \mid y = yes) = (0+1)/(2+4)$$

$$P(y = yes) = (2+1)/(5+2)$$

for counting frequency, assume every event has happened once.

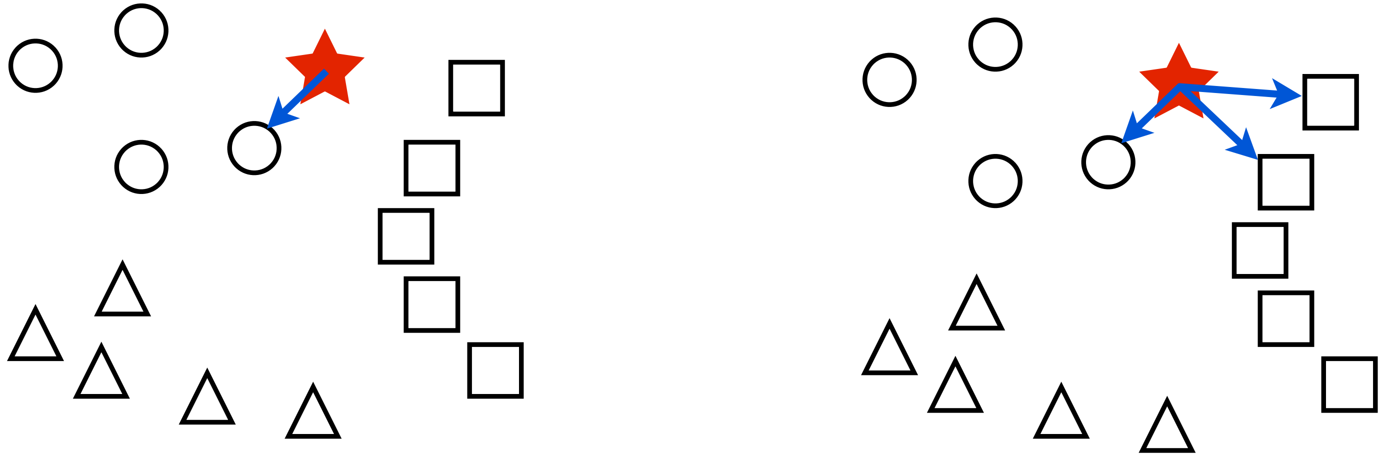
$$f(y \mid color = 0, weight = 1) \rightarrow$$

$$P(color = 0 \mid y = yes)P(weight = 1 \mid y = yes)P(y = yes) = \frac{1}{6} \times \frac{1}{7} \times \frac{3}{7} = 0.01$$

$$P(color = 0 \mid y = no)P(weight = 1 \mid y = no)P(y = no) = \frac{2}{7} \times \frac{1}{8} \times \frac{4}{7} = 0.02$$

Nearest neighbor classifier

1-nearest neighbor: *k*-nearest neighbor:

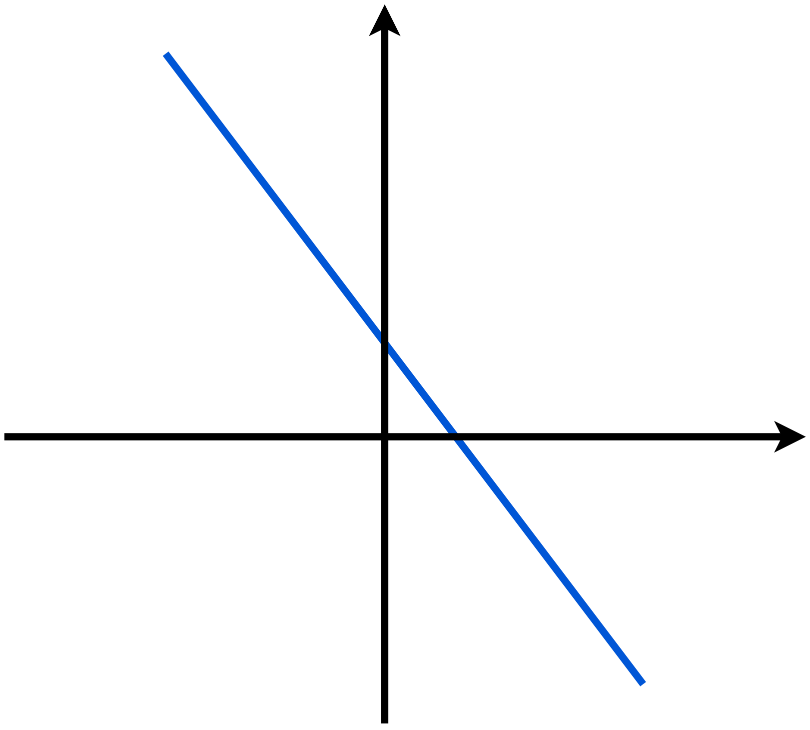


- asymptotically less than 2 times of the optimal Bayes error
- naturally handle multi-class
- no training time
- nonlinear decision boundary
- slow testing speed for a large training data set
- have to store the training data
- sensitive to similarity function

Linear model

model space: $\mathbb{R}^{n+1}$

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$



we sometimes omit the bias

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$$

1. w with a constant element
2. practically as good as with bias (centered data)

Least square regression

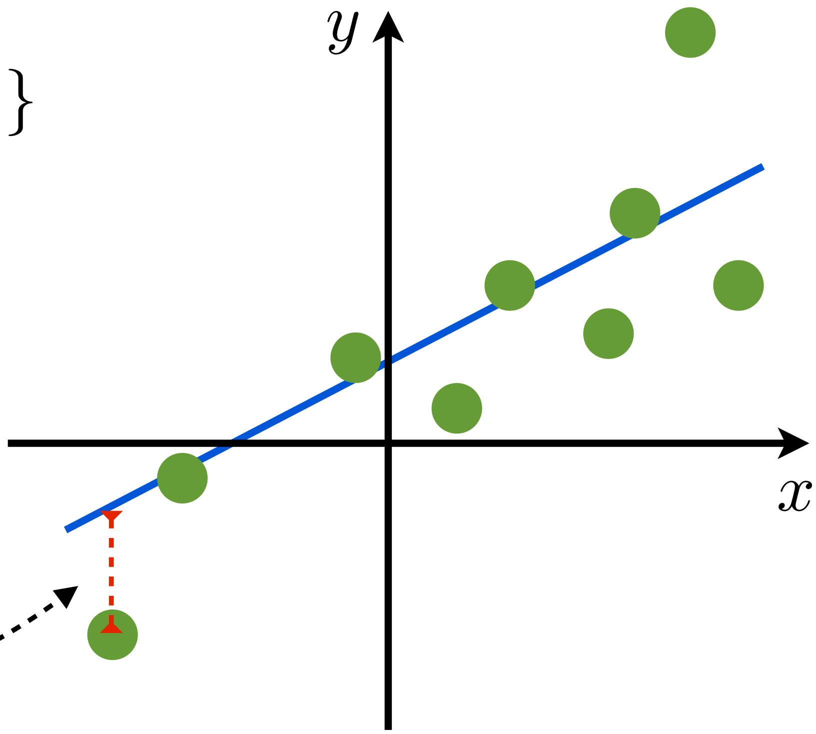
Regression: $y \in \mathbb{R}$

Training data:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_m, y_m)\}$$

Least square loss:

$$\frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i)^2$$



Least square regression

$$L(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i)^2$$

$$\frac{\partial L(\boldsymbol{w}, b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} 2(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i) = 0$$

$$\frac{\partial L(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} = \frac{1}{m}\sum_{i=1}^{m} 2(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i)\boldsymbol{x}_i = 0$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - \boldsymbol{w}^\top \boldsymbol{x}_i) = \bar{y} - \boldsymbol{w}^\top \bar{\boldsymbol{x}}$$

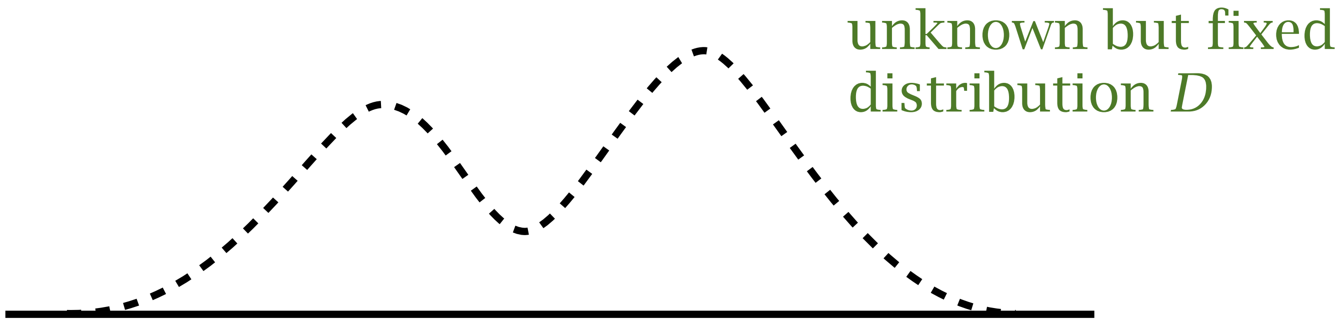
$$\boldsymbol{w} = \Big(\frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i \boldsymbol{x}_i^\top - \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\top\Big)^{-1}\Big(\frac{1}{m}\sum_{i=1}^{m}(y_i \boldsymbol{x}_i) - \bar{y}\bar{\boldsymbol{x}}\Big)$$

$$= var(\boldsymbol{x})^{-1} cov(\boldsymbol{x}, y) = (X^\top X)^{-1} X^\top Y$$

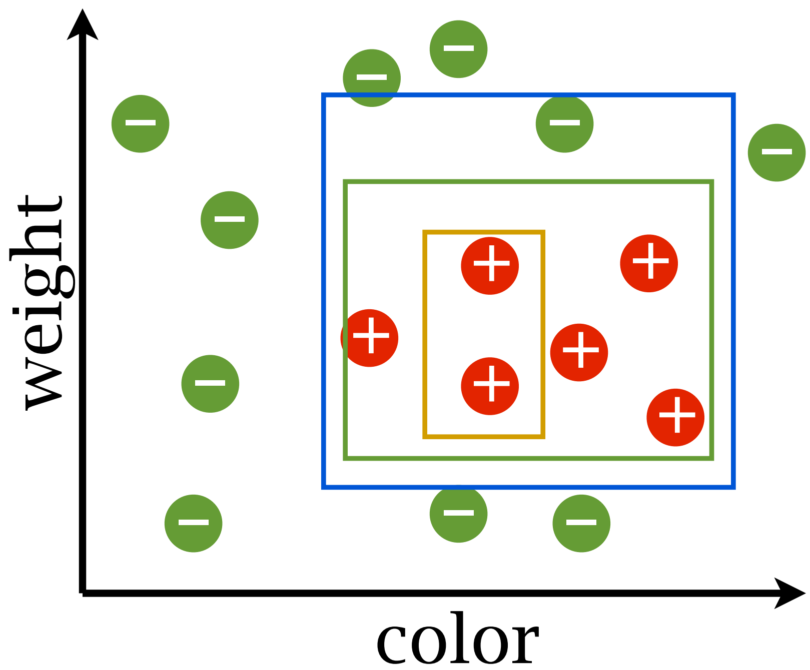
closed form solution

I.I.D. assumption

all training examples and future (test) examples are drawn *independently* from an *identical distribution*



Hypothesis class

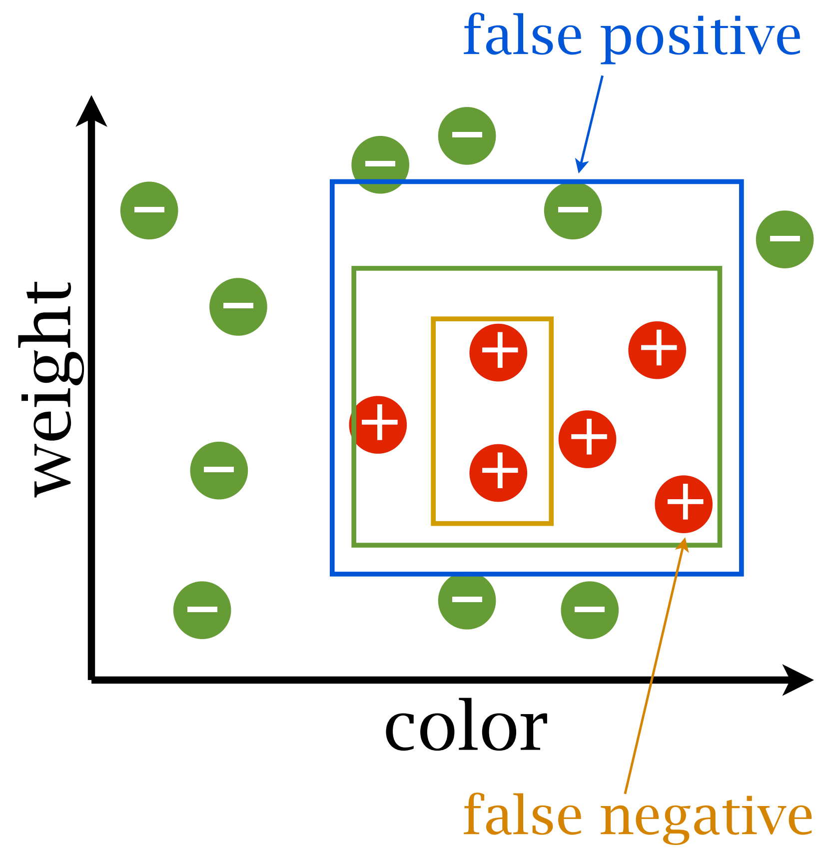


box hypothesis class $\mathcal{H}$ contains all boxes

$h \in \mathcal{H}$ is a hypothesis

$$h(\boldsymbol{x}) = \begin{cases} +1, \text{if } x \text{ is inside the box} \\ -1, \text{if } x \text{ is outside the box} \end{cases}$$

Training and generalization errors



training error

$$\epsilon_t = \frac{1}{m} \sum_{i=1}^{m} I(h(\boldsymbol{x}_i) \neq y_i)$$

generalization error

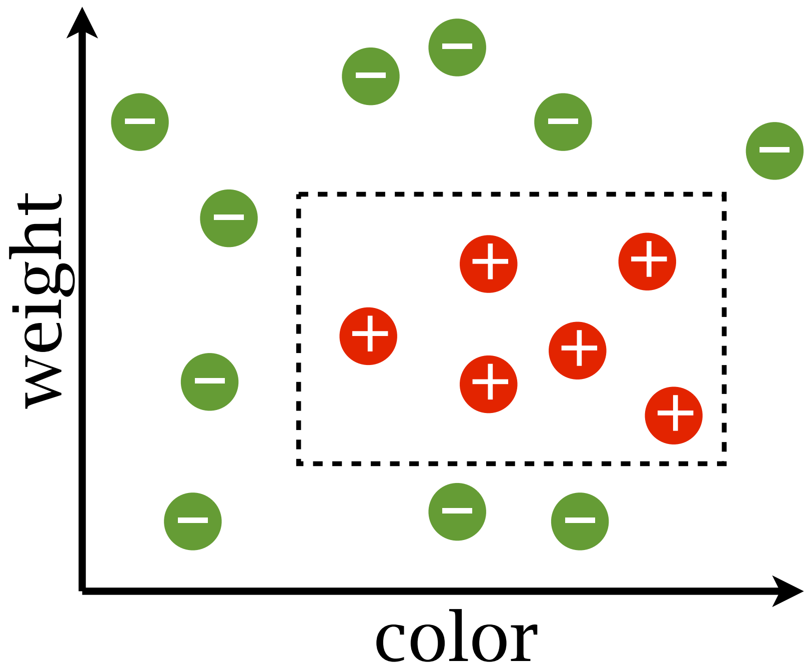
$$\epsilon_g = \mathbb{E}_x[I(h(\boldsymbol{x}) \neq f(\boldsymbol{x}))]$$

$$= \int_{\mathcal{X}} p(x) I(h(\boldsymbol{x}) \neq f(\boldsymbol{x}))]\mathrm{d}x$$

find a hypothesis minimizes the generalization error

Generalization error

assume i.i.d. examples, and the ground-truth hypothesis is a box



the error of picking a consistent hypothesis:

with probability at least $1-\delta$

$$\epsilon_g < \frac{1}{m} \cdot (\ln|\mathcal{H}| + \ln\frac{1}{\delta})$$

smaller generalization error:
- more examples
- smaller hypothesis space

Generalization error

for one *h*

What is the probability of

h is consistent

$\epsilon_g(h) \geq \epsilon$

assume *h* is ***bad***: $\epsilon_g(h) \geq \epsilon$

h is consistent with 1 example:

$$P \leq 1 - \epsilon$$

h is consistent with ***m*** example:

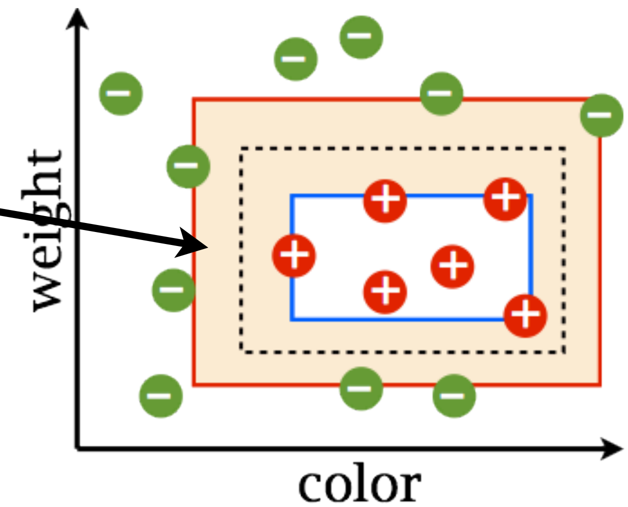
$$P \leq (1 - \epsilon)^m$$

Generalization error

h is consistent with $\boldsymbol{m}$ example:

$$P \leq (1-\epsilon)^m$$

There are $\boldsymbol{k}$ consistent hypotheses



Probability of choosing a bad one:

h_1 is chosen and h_1 is bad $P \leq (1-\epsilon)^m$

h_2 is chosen and h_2 is bad $P \leq (1-\epsilon)^m$

...

h_k is chosen and h_k is bad $P \leq (1-\epsilon)^m$

overall:

$\exists h$: h can be chosen (consistent) but is bad

Generalization error

h_1 is chosen and h_1 is bad $P \leq (1-\epsilon)^m$
h_2 is chosen and h_2 is bad $P \leq (1-\epsilon)^m$
…
h_k is chosen and h_k is bad $P \leq (1-\epsilon)^m$

overall:

$\exists h$: h can be chosen (consistent) but is bad

Union bound: $P(A \cup B) \leq P(A) + P(B)$

$$P(\exists h \text{ is consistent but bad}) \leq k \cdot (1-\epsilon)^m \leq |\mathcal{H}| \cdot (1-\epsilon)^m$$

Generalization error

$$P(\exists h \text{ is consistent but bad}) \leq k \cdot (1-\epsilon)^m \leq |\mathcal{H}| \cdot (1-\epsilon)^m$$

$$\Downarrow$$

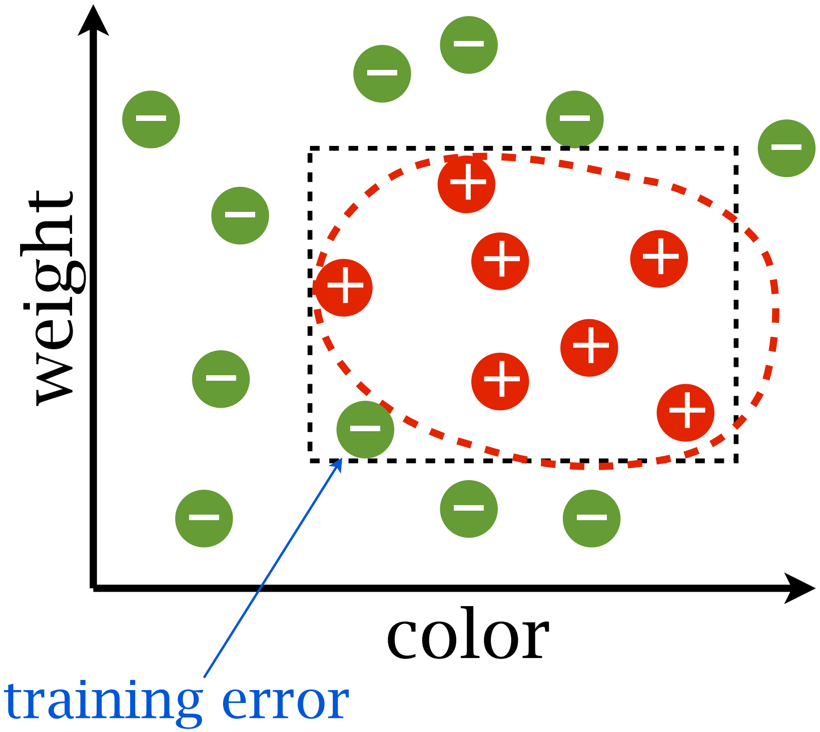
$$P(\epsilon_g \geq \epsilon) \leq \underbrace{|\mathcal{H}| \cdot (1-\epsilon)^m}_{\delta}$$

with probability at least $1-\delta$

$$\epsilon_g < \frac{1}{m} \cdot (\ln|\mathcal{H}| + \ln\frac{1}{\delta})$$

Inconsistent hypothesis

What if the ground-truth hypothesis is NOT a box: non-zero training error



with probability at least $1 - \delta$

$$\epsilon_g < \epsilon_t + \sqrt{\frac{1}{m}(\ln|\mathcal{H}| + \ln\frac{1}{\delta})}$$

smaller generalization error:

- more examples
- smaller hypothesis space
- **smaller training error**

机器学习/模式识别基础

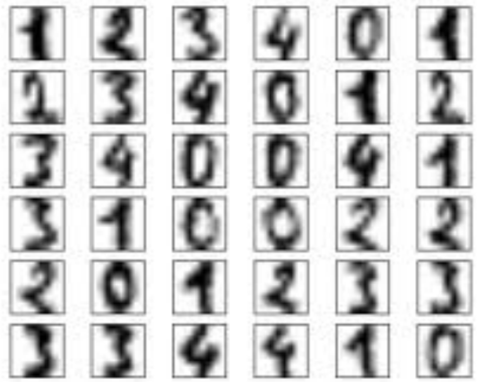
- 预测与识别
- 预测算法
- 特征提取

Feature extraction

disclosure the inner structure of the data
to support a better mining performance

- feature extraction construct new features
- commonly followed by a feature selection
- usually used for low-level features

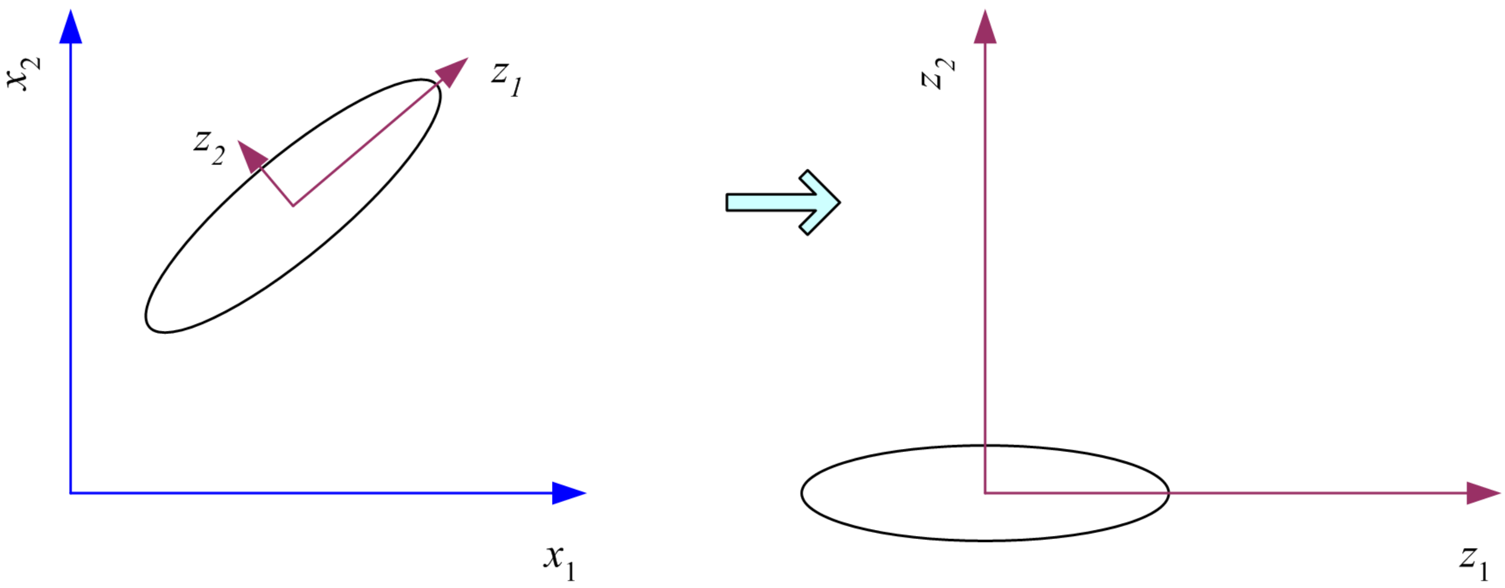
digits bitmap:



Linear methods

Principal components analysis (PCA)

rotate the data to align the directions of the variance



Linear methods

Principal components analysis (PCA)

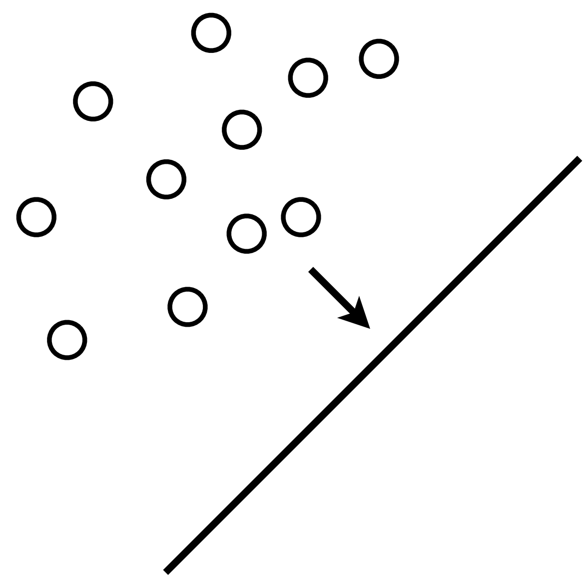
the first dimension = the largest variance direction

$$z = \boldsymbol{w}^T \boldsymbol{x}$$

$$\mathrm{Var}(z_1) = \boldsymbol{w}_1^T \Sigma \boldsymbol{w}_1$$

find a unit $\boldsymbol{w}$ to maximize the variance

$$\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^T \Sigma \boldsymbol{w}_1 - \alpha(\boldsymbol{w}_1^T \boldsymbol{w}_1 - 1)$$



$2\Sigma \boldsymbol{w}_1 - 2\alpha \boldsymbol{w}_1 = 0,$ and therefore $\Sigma \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$

$$\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1^T \boldsymbol{w}_1 = \alpha$$

$\boldsymbol{w}$ is the eigenvector with the largest eigenvalue

Linear methods

Principal components analysis (PCA)

the second dimension = the largest variance direction orthogonal to the first dimension

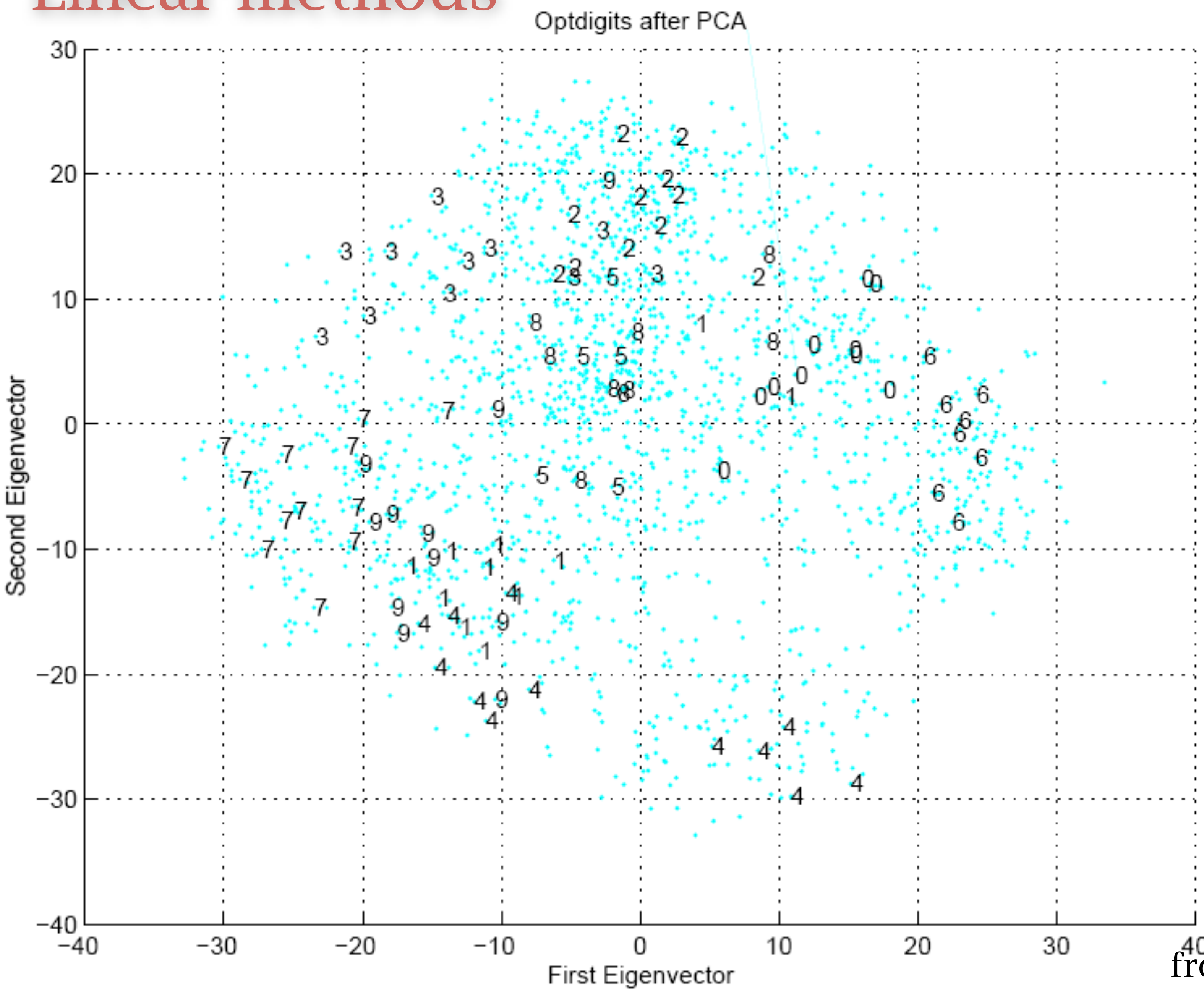
$$\max_{\boldsymbol{w}_2} \boldsymbol{w}_2^T \boldsymbol{\Sigma} \boldsymbol{w}_2 - \alpha(\boldsymbol{w}_2^T \boldsymbol{w}_2 - 1) - \beta(\boldsymbol{w}_2^T \boldsymbol{w}_1 - 0)$$

$$2\boldsymbol{\Sigma}\boldsymbol{w}_2 - 2\alpha\boldsymbol{w}_2 - \beta\boldsymbol{w}_1 = 0$$

$$\beta = 0 \qquad \boldsymbol{\Sigma}\boldsymbol{w}_2 = \alpha\boldsymbol{w}_2$$

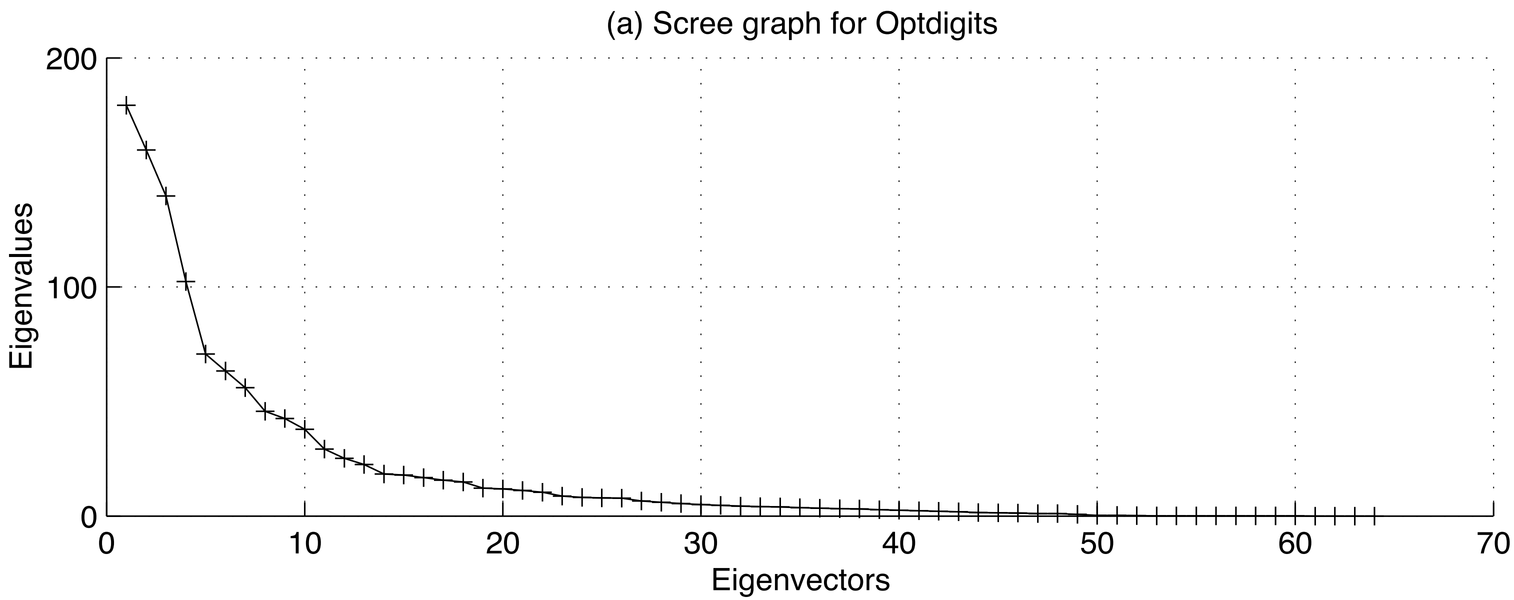
$\boldsymbol{w}$'s are the eigenvectors sorted by the eigenvalues

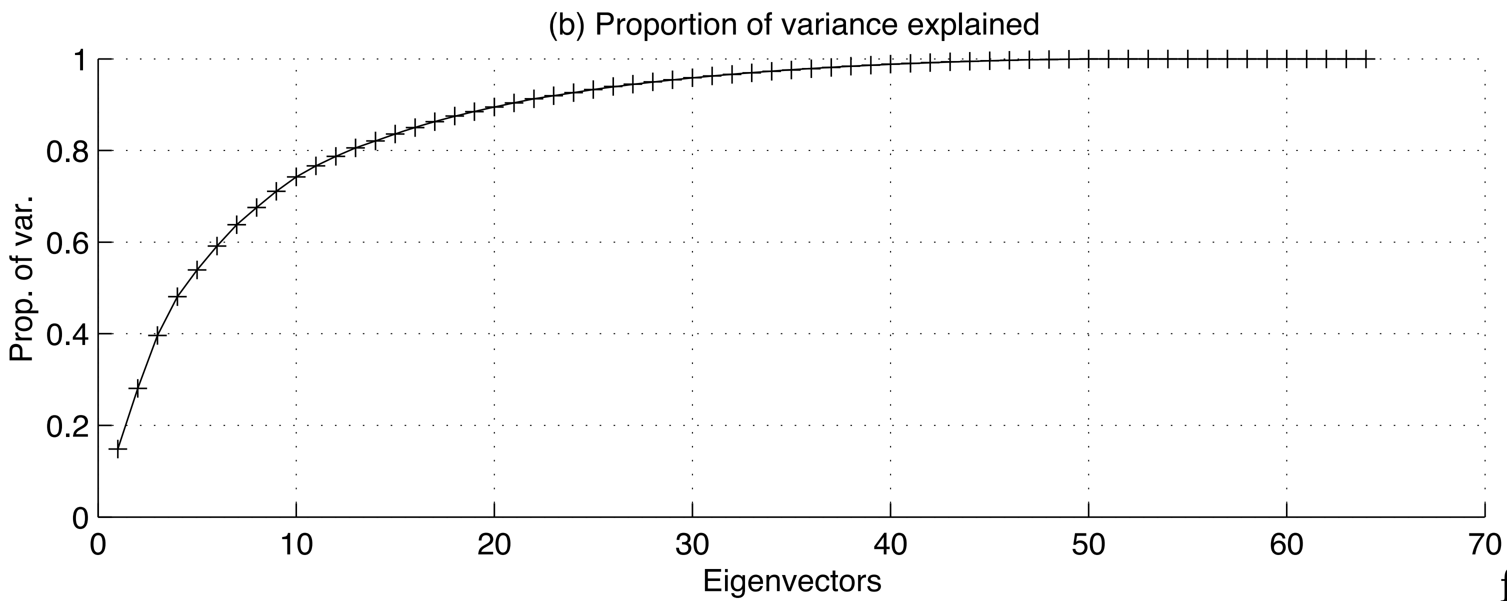
Linear methods



from [Intro. ML]

Linear methods





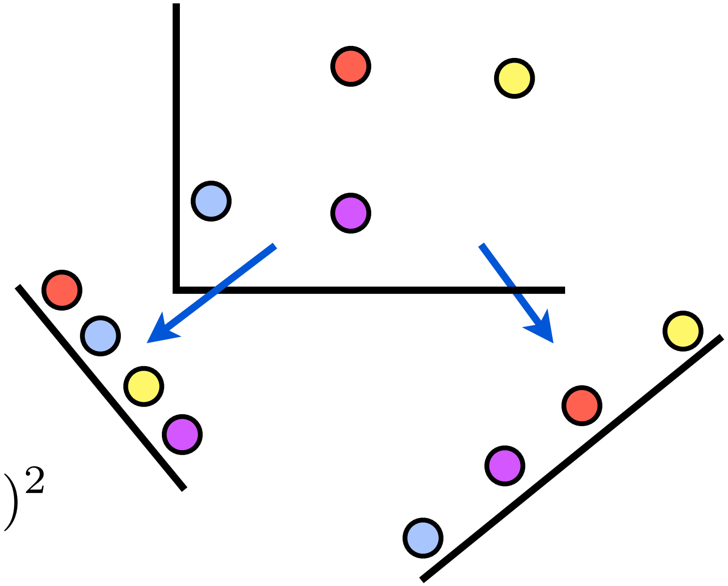
from [Intro. ML]

Linear methods

Multidimensional Scaling (MDS)

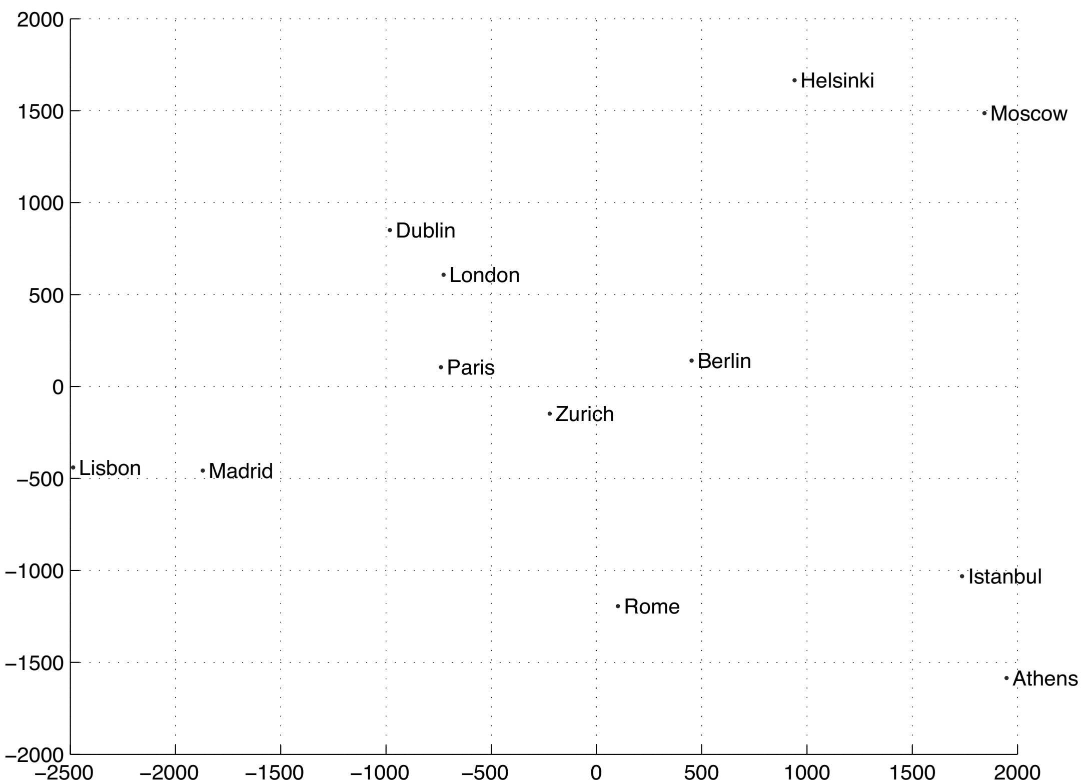
keep the distance into a lower dimensional space

for linear transformation,
W is an n*k matrix



$$\arg\min_{W} \sum_{i,j} (\|\boldsymbol{x}_i^\top W - \boldsymbol{x}_j^\top W\| - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|)^2$$

Linear methods



from [Intro. ML]

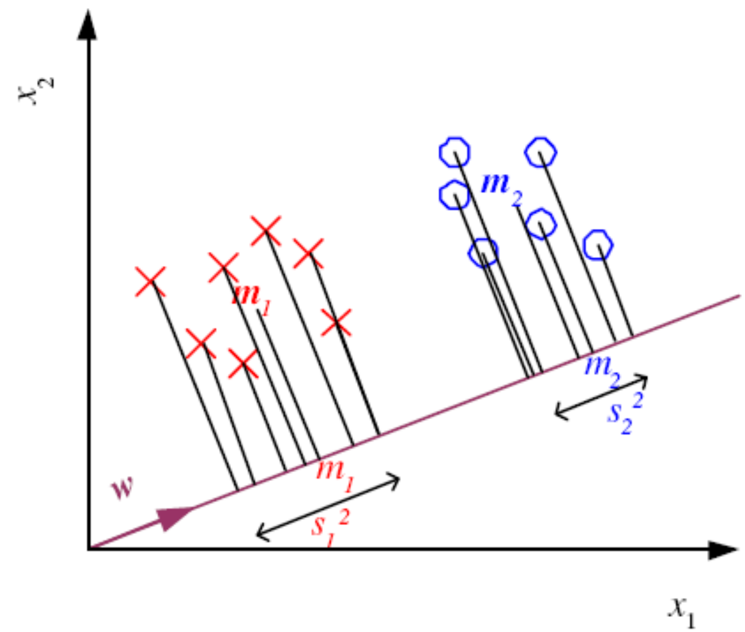
Linear methods

Linear Discriminant Analysis (LDA)

find a direction such that the two classes are well separated

$$z = \boldsymbol{w}^T \boldsymbol{x}$$

m be the mean of a class
s^2 be the variance of a class



maximize the criterion

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

Linear methods

Linear Discriminant Analysis (LDA)

$$
\begin{aligned}
(m_1 - m_2)^2 &= (\boldsymbol{w}^T \boldsymbol{m}_1 - \boldsymbol{w}^T \boldsymbol{m}_2)^2 \\
&= \boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} \\
&= \boldsymbol{w}^T \mathbf{S}_B \boldsymbol{w}
\end{aligned}
$$

$$
\begin{aligned}
s_1^2 &= \sum_t (\boldsymbol{w}^T \boldsymbol{x}^t - m_1)^2 r^t \\
&= \sum_t \boldsymbol{w}^T (\boldsymbol{x}^t - \boldsymbol{m}_1)(\boldsymbol{x}^t - \boldsymbol{m}_1)^T \boldsymbol{w} r^t \\
&= \boldsymbol{w}^T \mathbf{S}_1 \boldsymbol{w}
\end{aligned}
$$

$$s_1^2 + s_2^2 = \boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w} \qquad \mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

The objective becomes:

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^T \mathbf{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}} = \frac{|\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)|^2}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}}$$

Linear methods

Linear Discriminant Analysis (LDA)

The objective becomes:

$$J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^T \mathbf{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}} = \frac{|\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)|^2}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}}$$

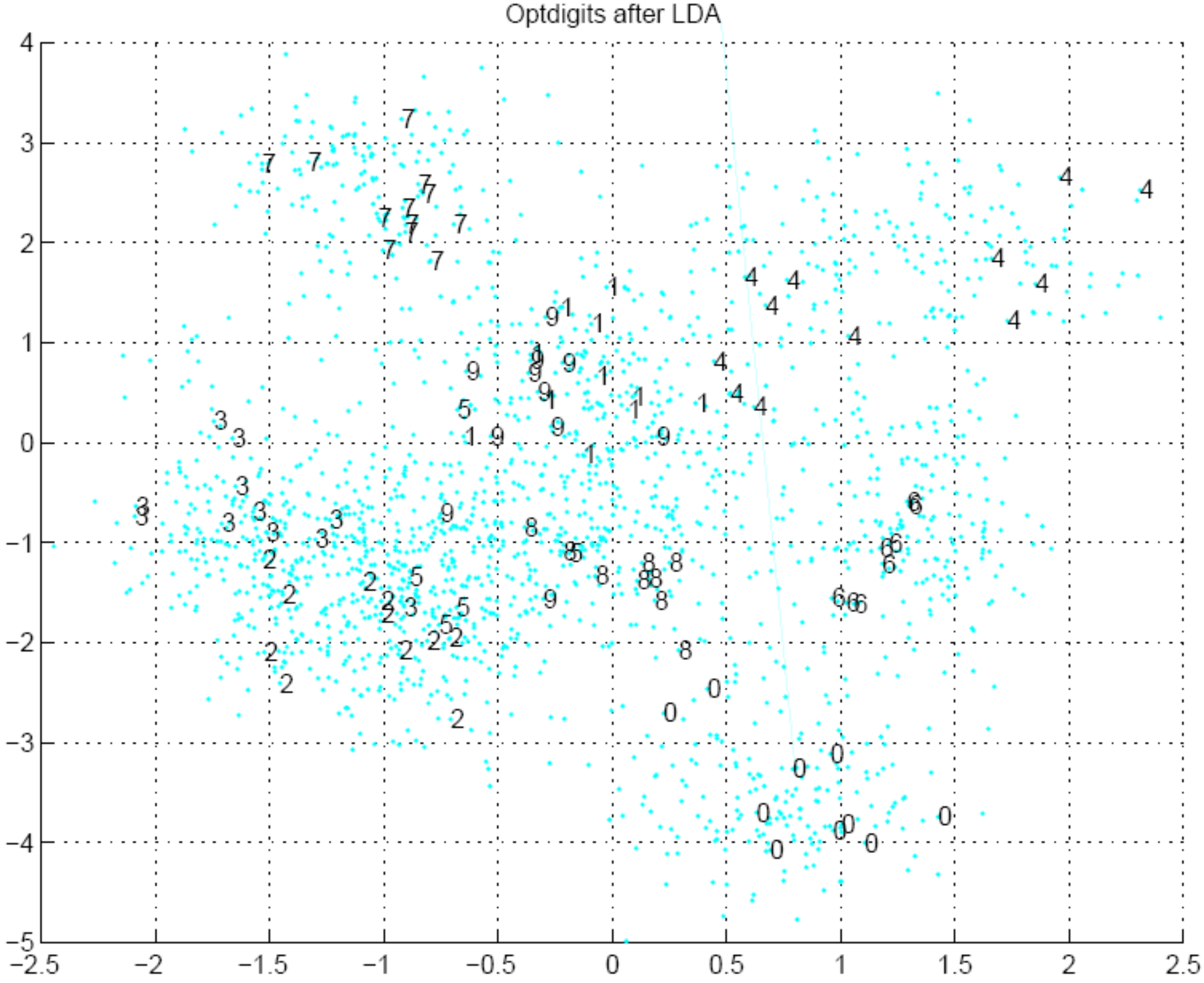
$$\frac{\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}} \left(2(\boldsymbol{m}_1 - \boldsymbol{m}_2) - \frac{\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)}{\boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}} \mathbf{S}_W \boldsymbol{w} \right) = 0$$

Given that $\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2) / \boldsymbol{w}^T \mathbf{S}_W \boldsymbol{w}$ is a constant, we have

$$\boldsymbol{w} = c \mathbf{S}_W^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

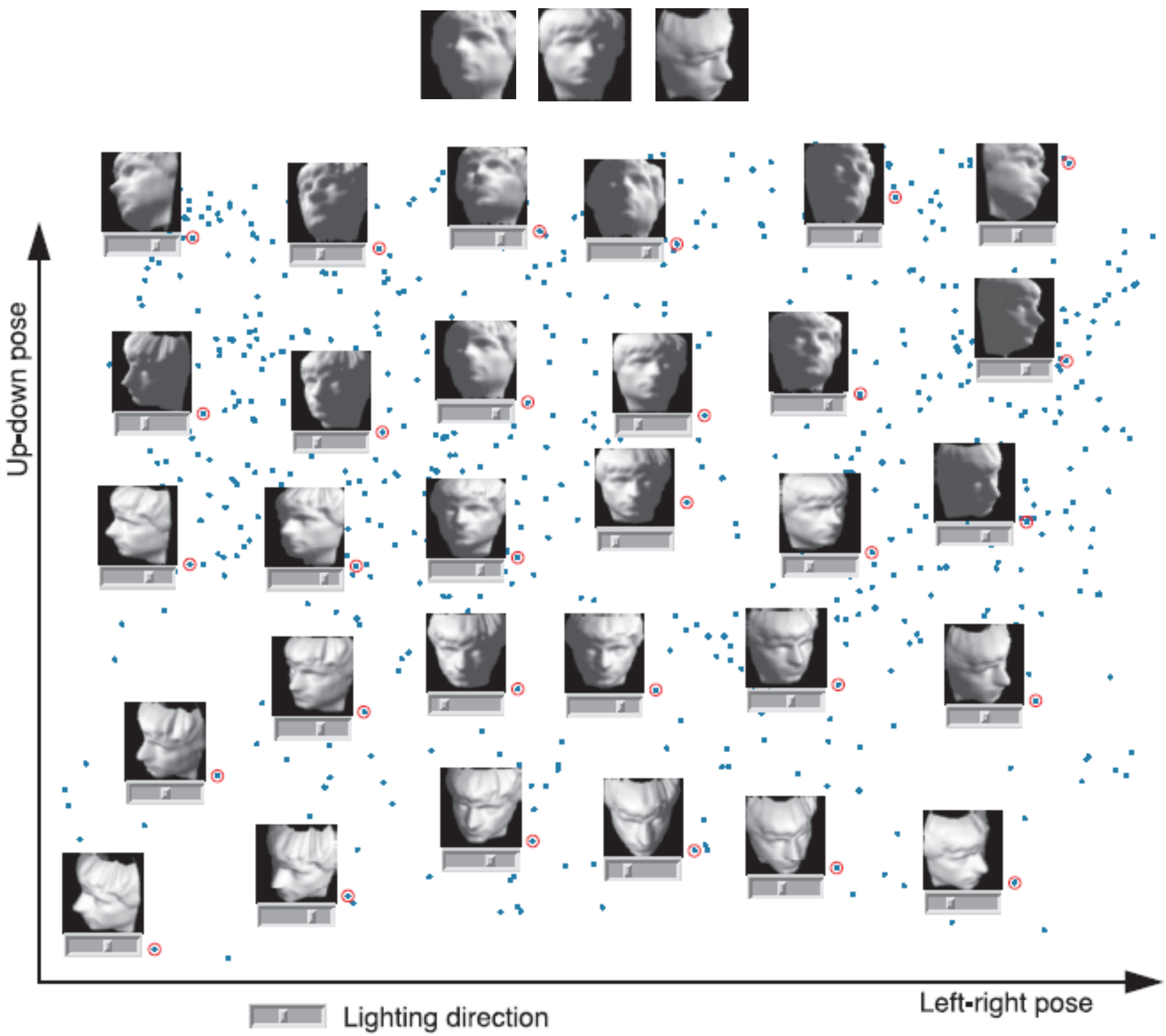
just take $c = 1$ and find $\boldsymbol{w}$

Linear methods



from [Intro. ML]

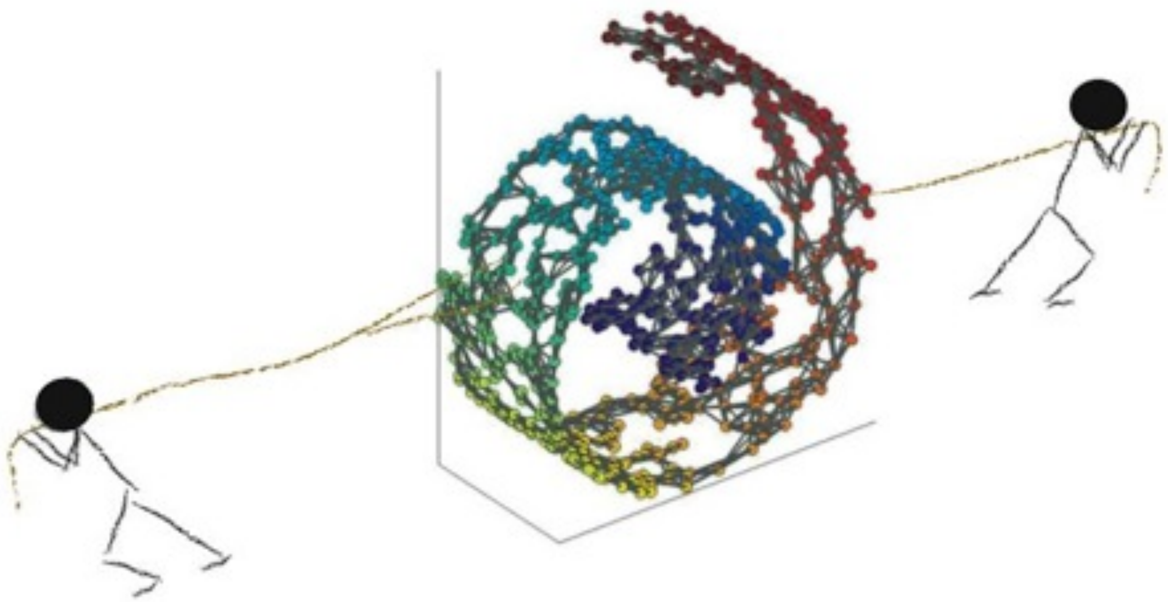
Manifold learning



Manifold learning

A low intrinsic dimensional data embedded in a high dimensional space

cause a bad distance measure



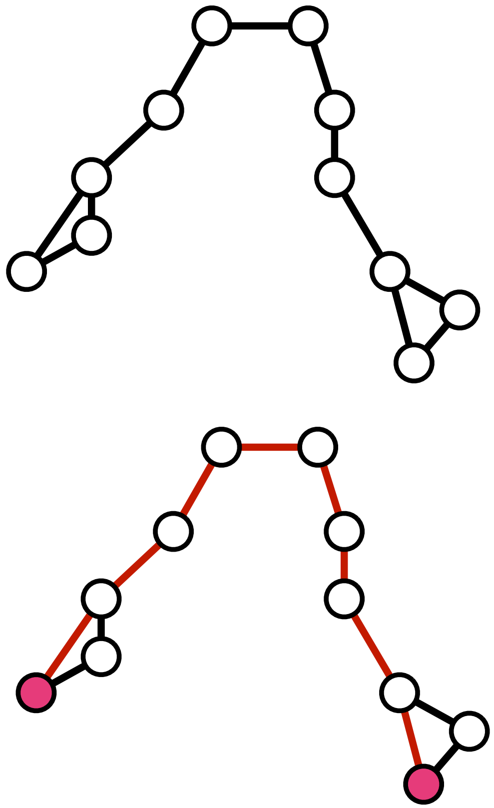
Manifold learning

ISOMAP

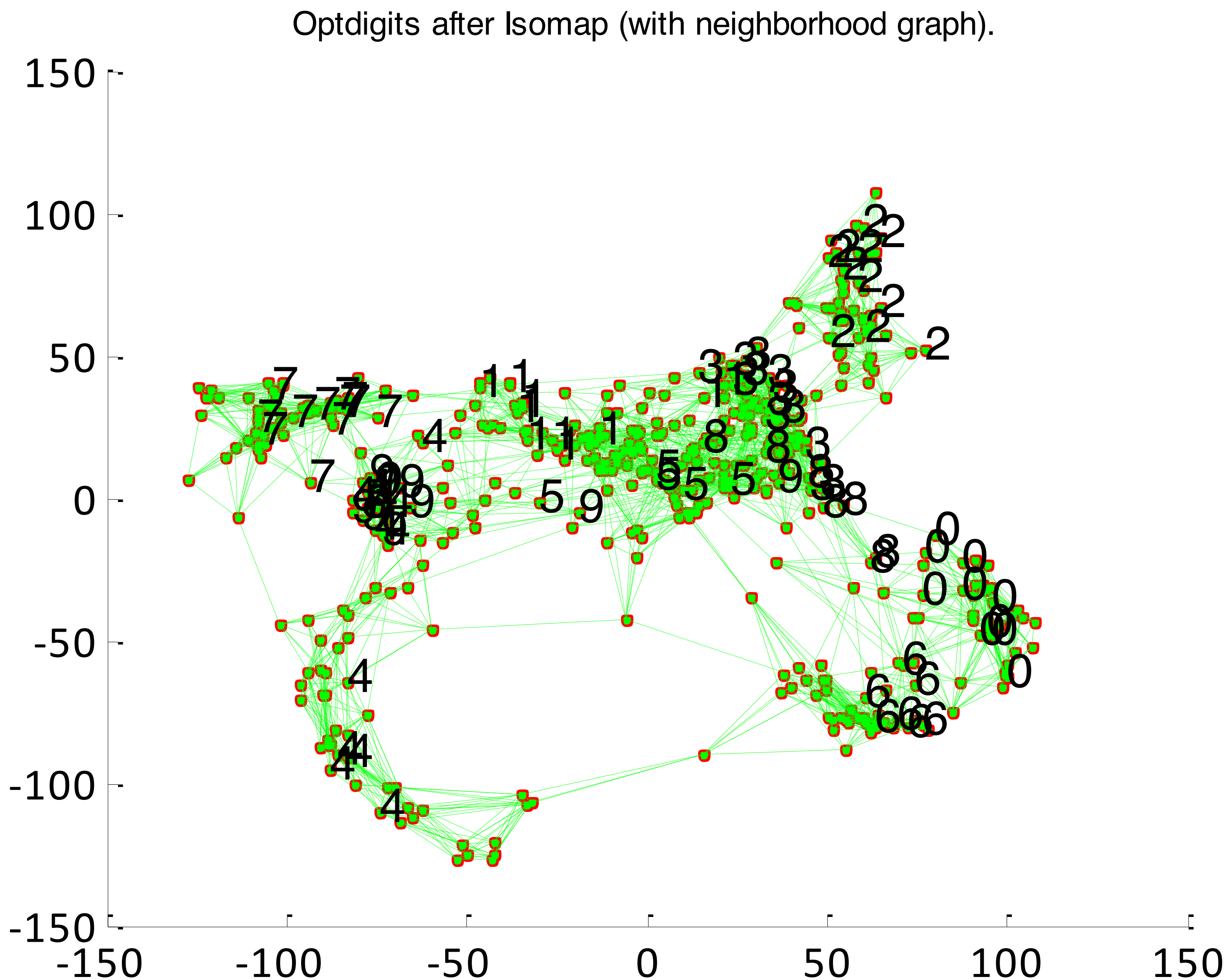
1. construct a neighborhood graph (kNN and ε-NN)

2. calculate distance matrix as the shortest path on the graph

3. apply MDS on the distance matrix



Manifold learning



from [Intro. ML]

Manifold learning

Local Linear Embedding (LLE):

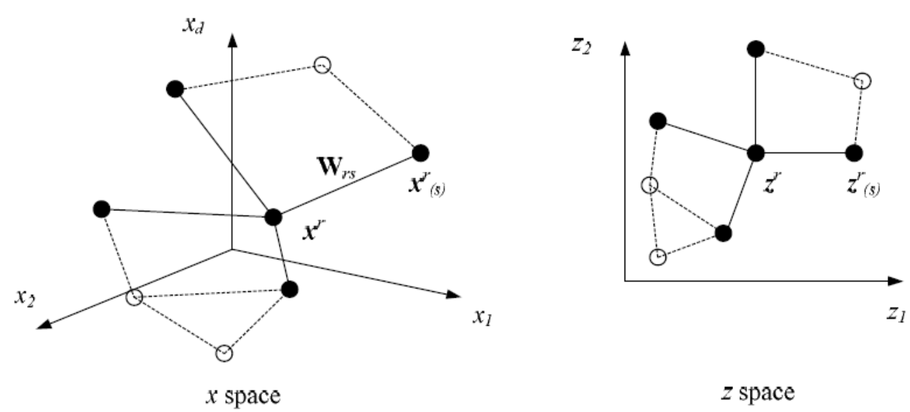
1. find neighbors for each instance

2. calculate a linear reconstruction for an instance

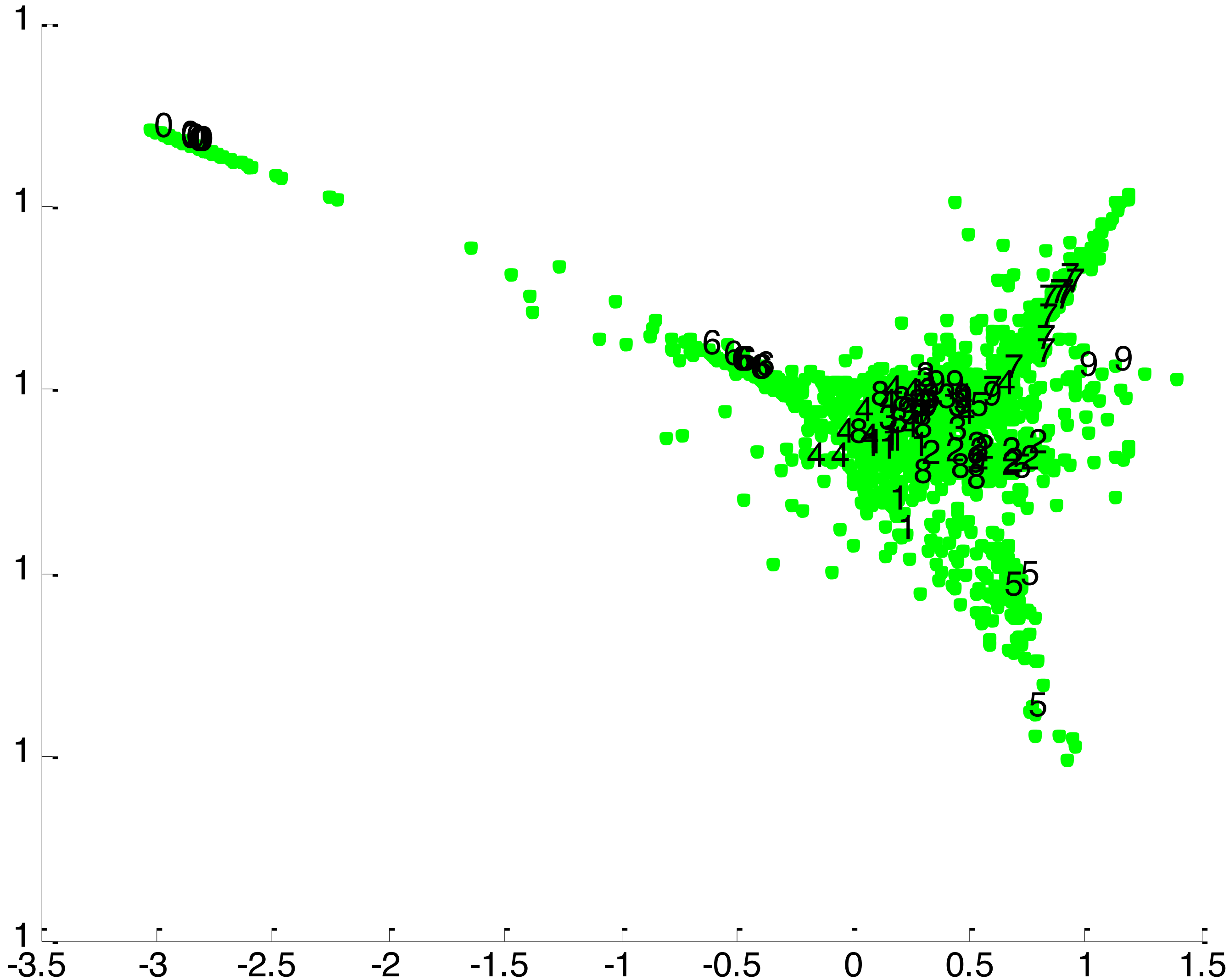
$$\sum_r \|\boldsymbol{x}^r - \sum_s \mathbf{W}_{rs}\boldsymbol{x}^s_{(r)}\|^2$$

3. find low dimensional instances preserving the reconstruction

$$\sum_r \|\boldsymbol{z}^r - \sum_s \mathbf{W}_{rs}\boldsymbol{z}^s\|^2$$



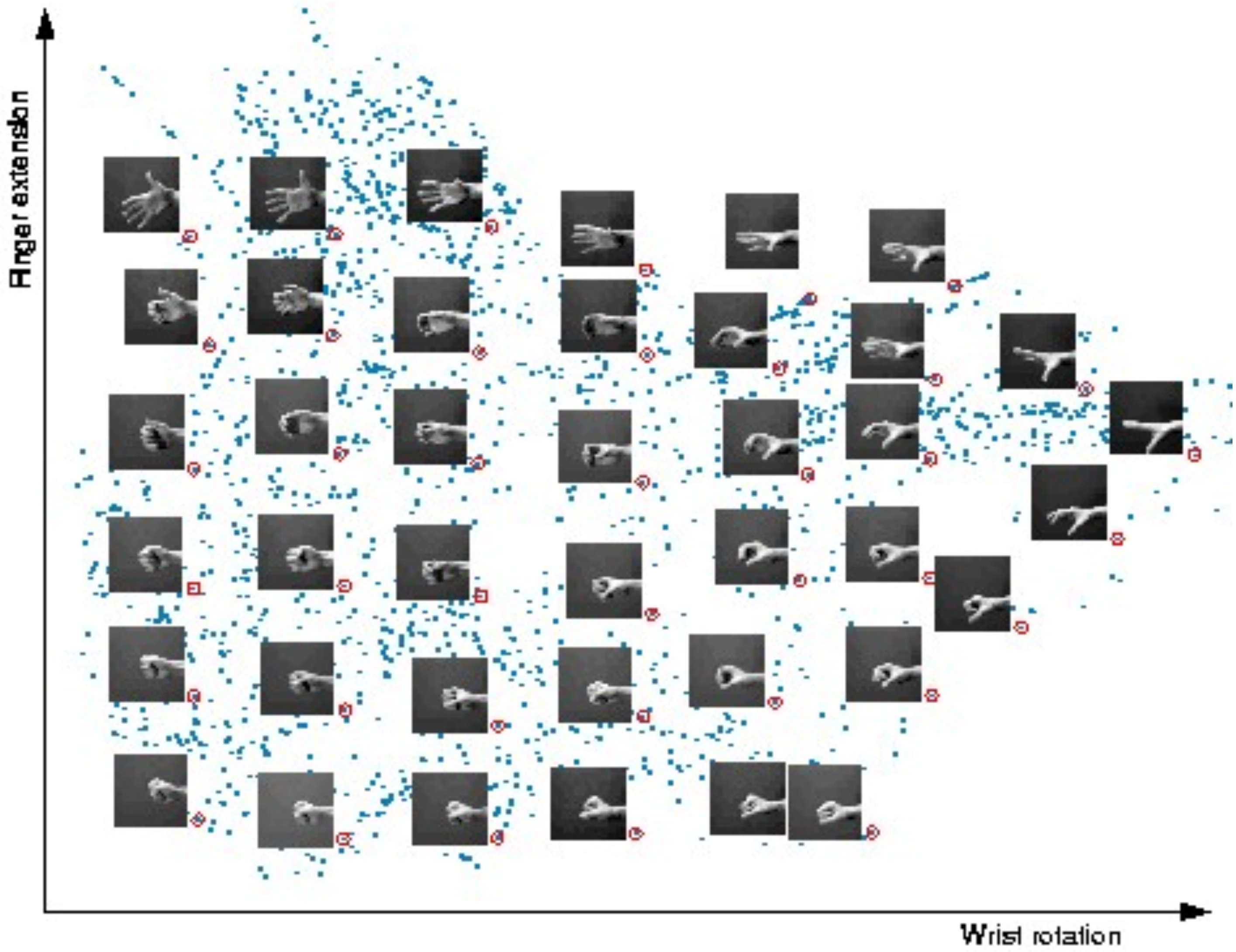
Manifold learning



from [Intro. ML]

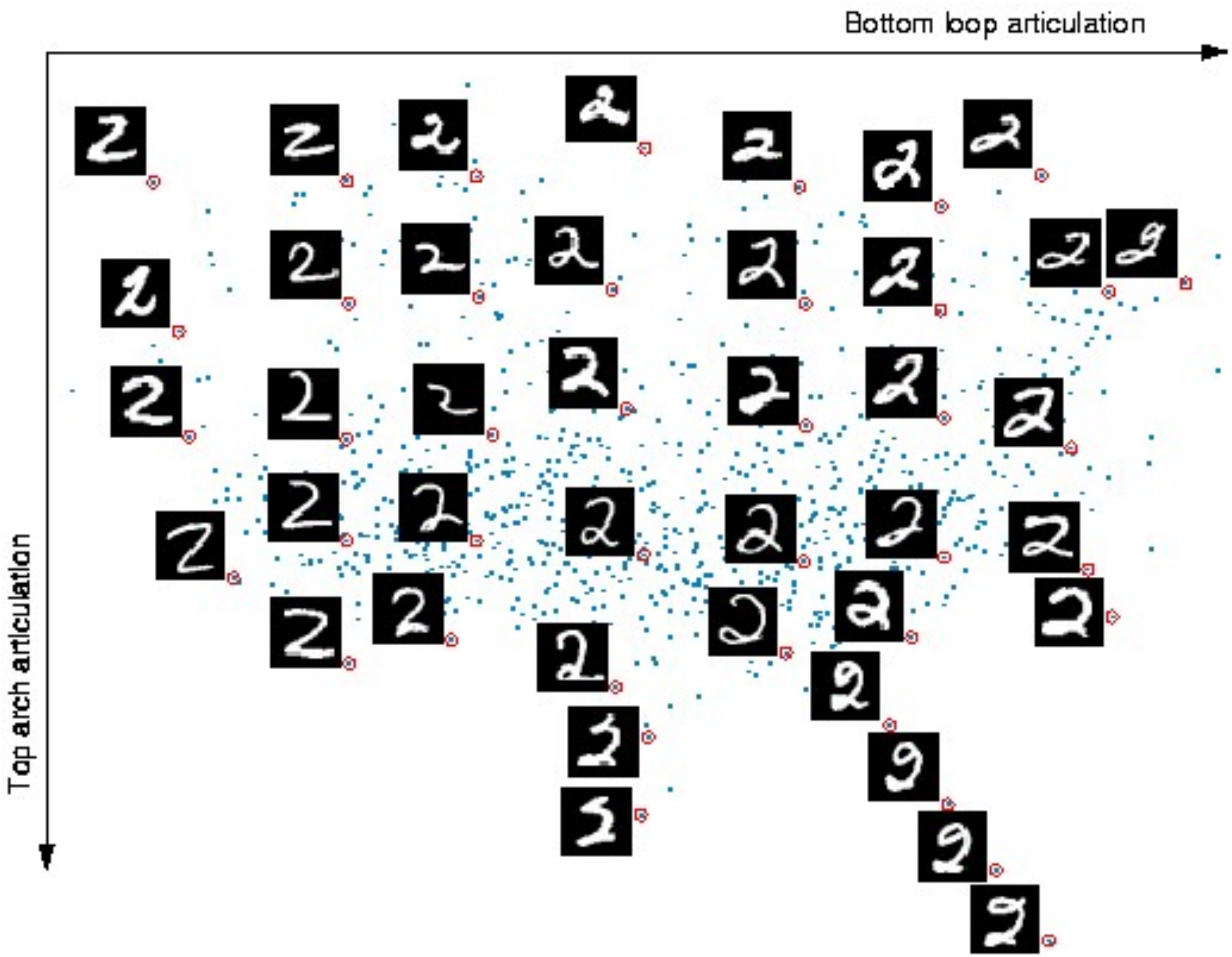
Manifold learning

more manifold learning examples



Manifold learning

more manifold learning examples



机器学习/模式识别基础

小结

- 预测与识别
  - 用数据定义“模式”
- 预测算法

- 特征提取