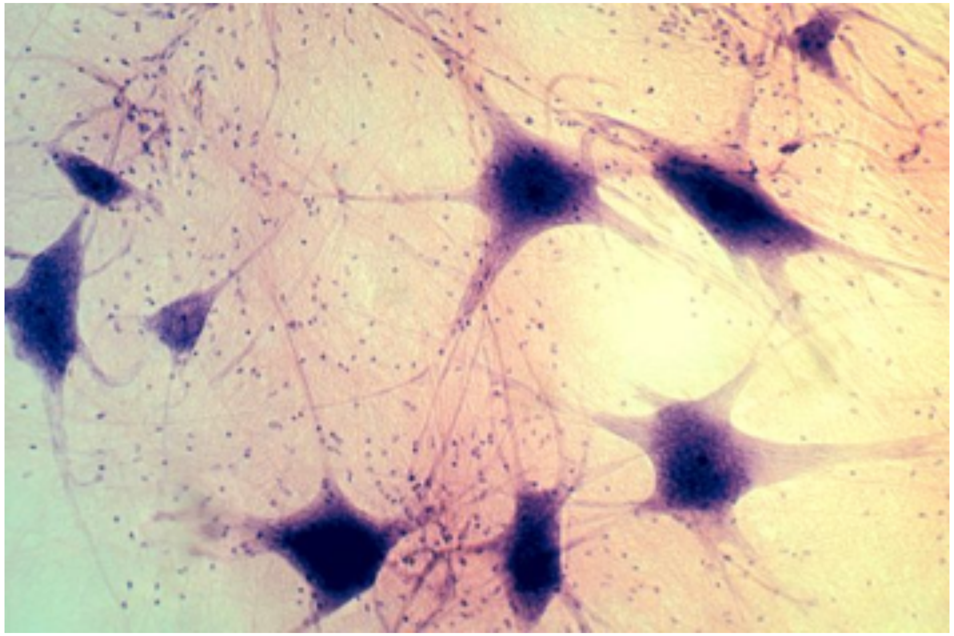
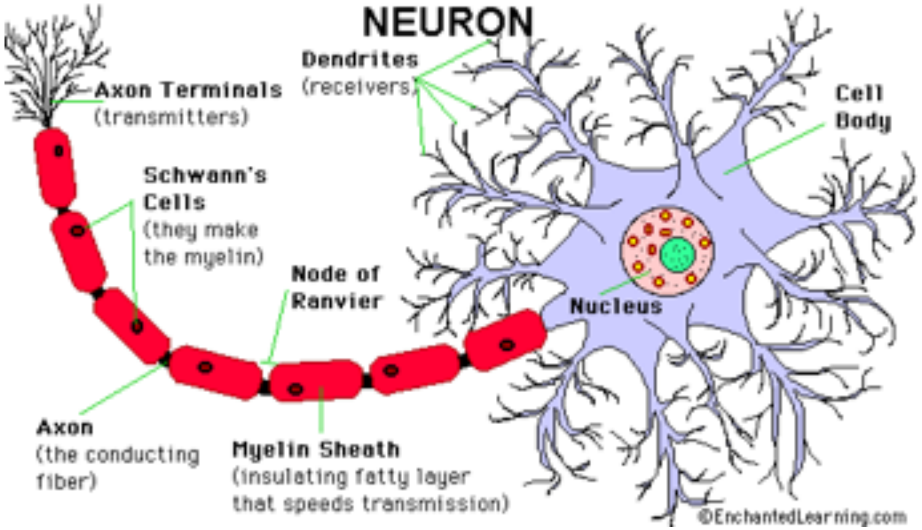
Lecture 9: Machine Learning VII

Neural Networks and Nearest Neighbors

http://cs.nju.edu.cn/yuy/course_dm13ms.ashx

Neural networks





Neuron / perceptron

output a function of sum of input

linear function:

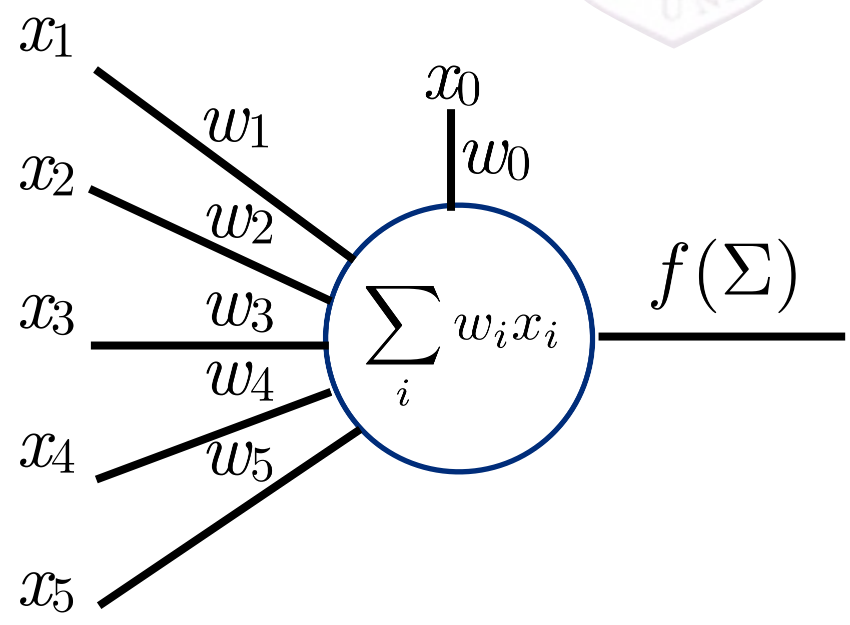
$$f(\sum_i w_i x_i) = \sum_i w_i x_i$$

threshold function:

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

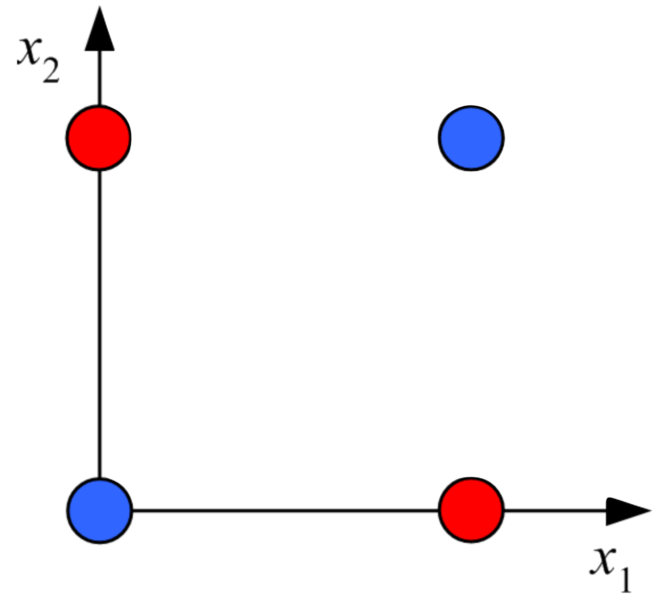
sigmoid function:

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



Limitation of single neuron

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



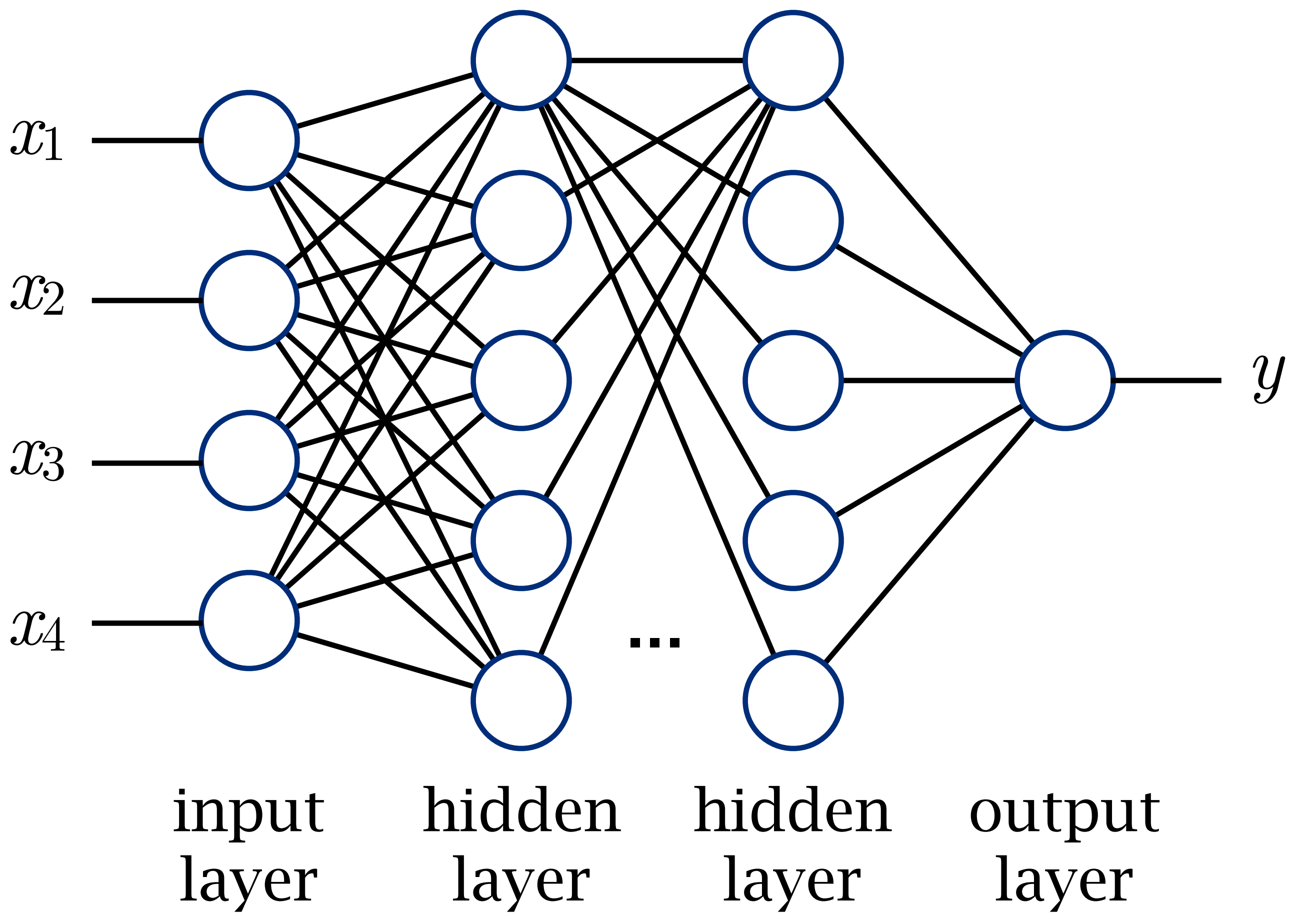
[Minsky and Papert, *Perceptrons*, 1969]

Marvin Minsky
Turing Award 1969

AI Winter

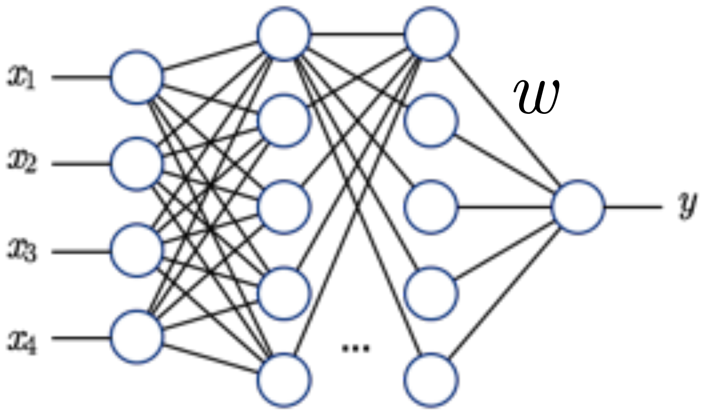
Multi-layer perceptrons

feed-forward network



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

Back-propagation algorithm



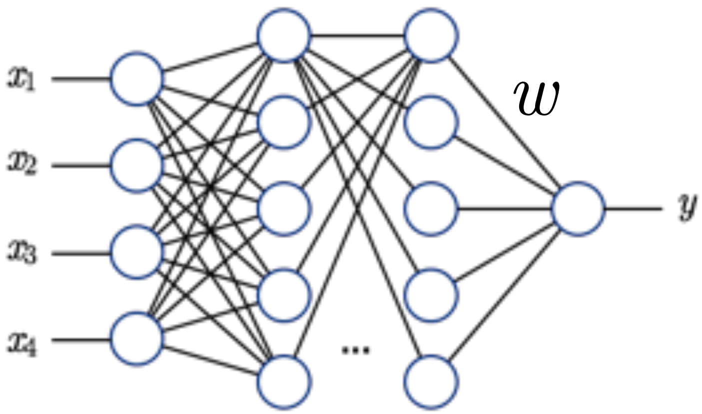
$$\hat{y} = F(\boldsymbol{x}) \quad f(\sum_i w_i x_i) = \frac{1}{1+e^{-\Sigma}}$$

gradient descent

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

[Rumelhart, Hinton, Williams, Nature 1986]

Back-propagation algorithm



$$\hat{y} = F(\boldsymbol{x}) \quad f(\sum_i w_i x_i) = \frac{1}{1+e^{-\Sigma}}$$

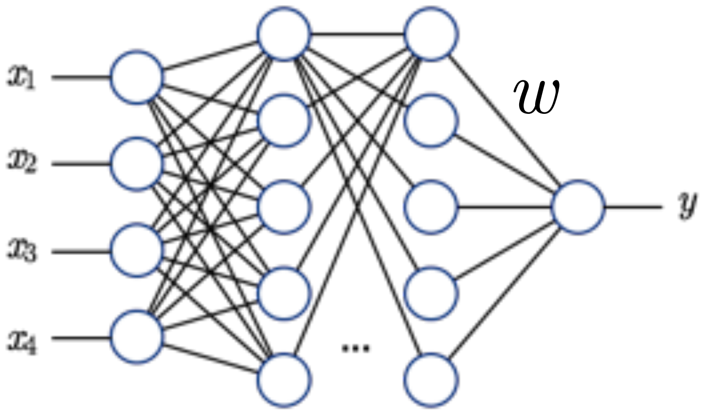
gradient descent

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

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

[Rumelhart, Hinton, Williams, Nature 1986]

Back-propagation algorithm



$\hat{y} = F(\boldsymbol{x})$ $\quad f(\sum_i w_i x_i) = \frac{1}{1+e^{-\Sigma}}$

gradient descent

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

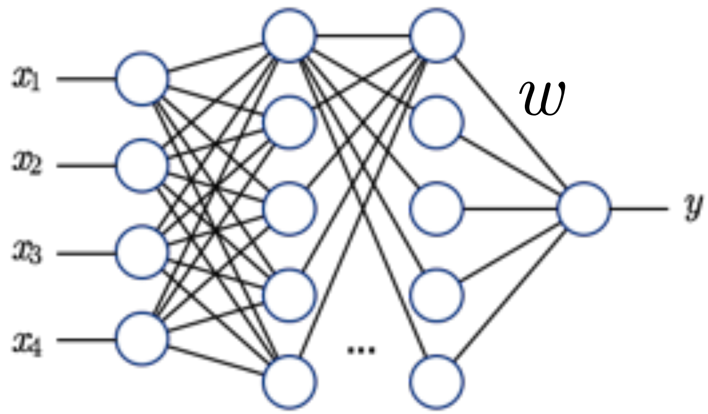
update one weight: $\Delta w_{i,j} = -\eta \frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}}$

weight of the laster layer

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

[Rumelhart, Hinton, Williams, Nature 1986]

Back-propagation algorithm



$$\hat{y} = F(\boldsymbol{x}) \quad f(\sum_i w_i x_i) = \frac{1}{1 + e^{-\Sigma}}$$

gradient descent

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

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

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

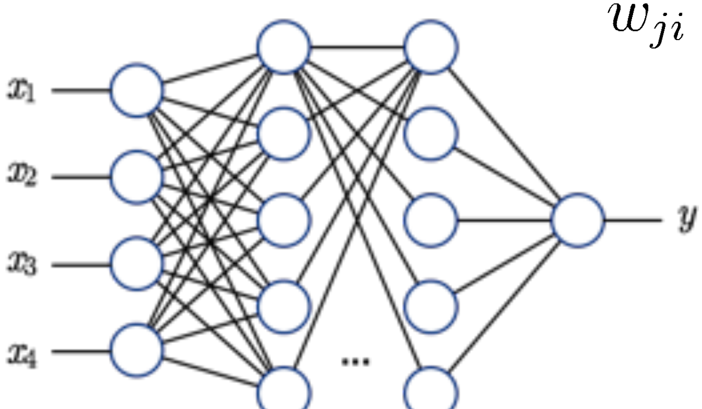
$$\delta_k \leftarrow o_k(1 - o_k)(y_k - o_k)$$

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

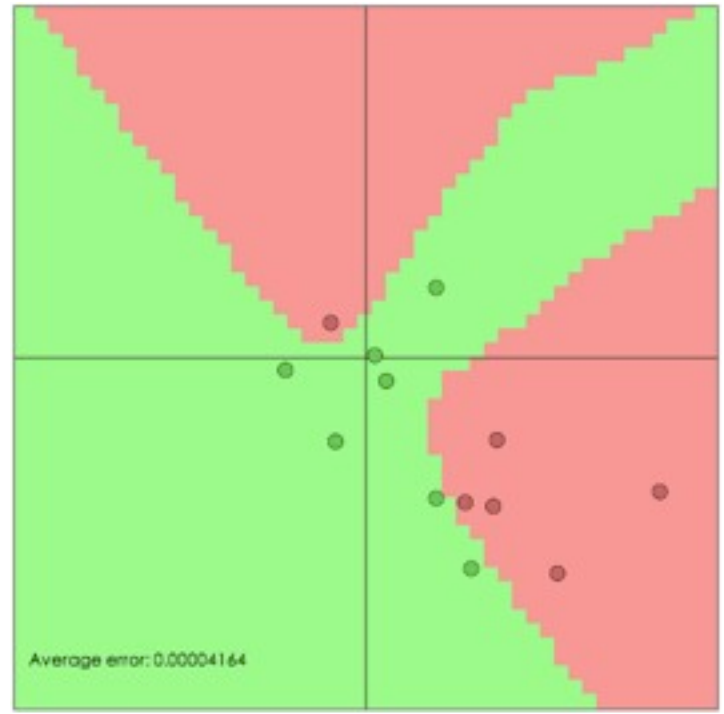
$$w_{ji} \leftarrow w_{ji} + \eta \delta_j x_{ji}$$



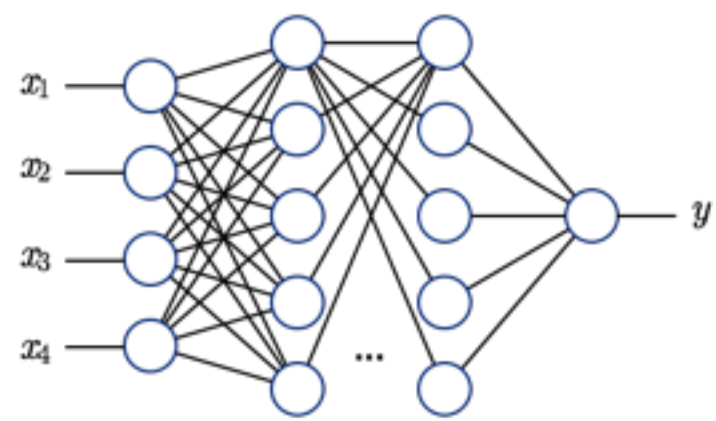
[Rumelhart, Hinton, Williams, Nature 1986]

Advantage and disadvantages

Smooth and nonlinear decision boundary



Slow convergence



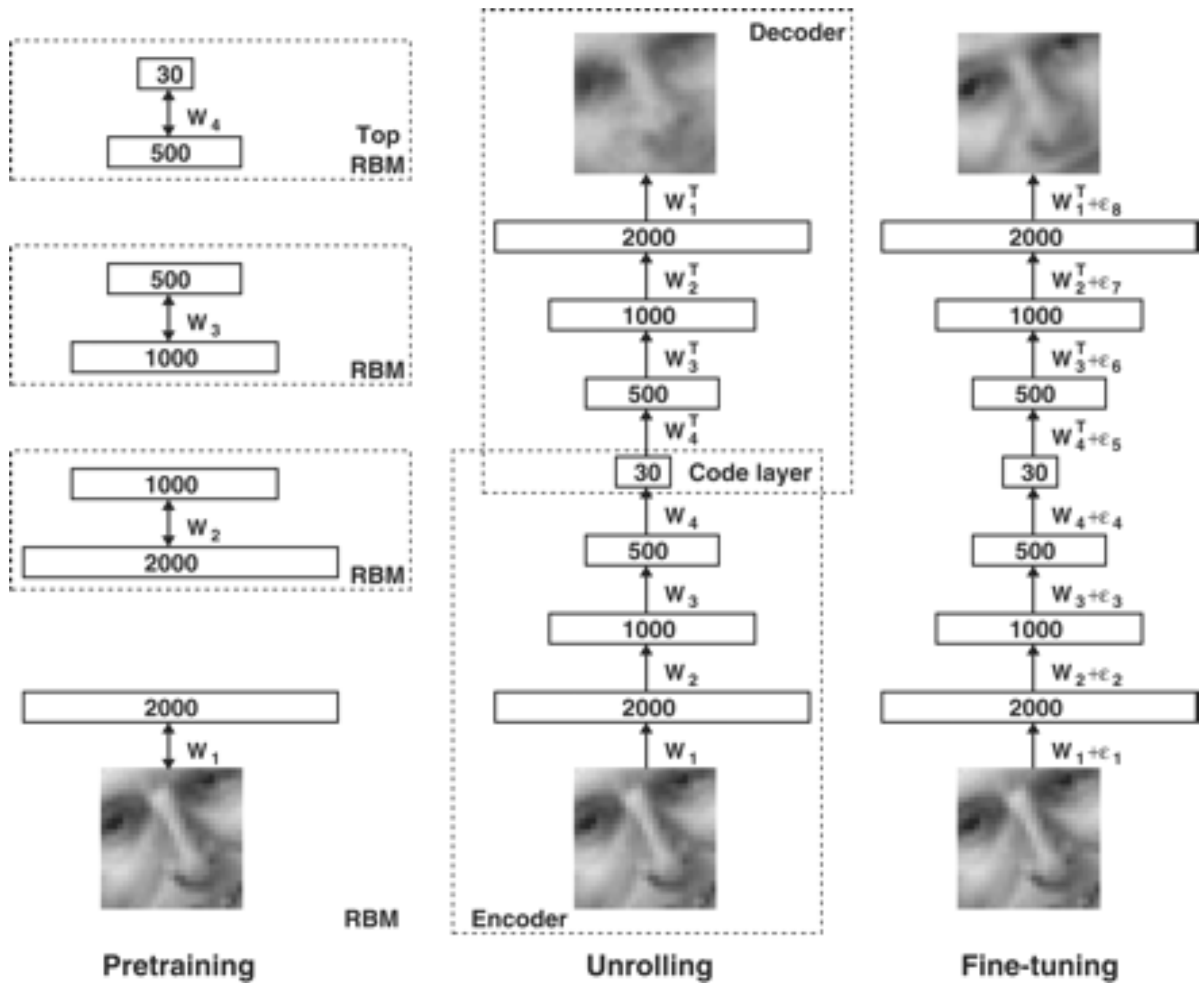
Many local optima

Best network structure unknown

Hard to handle nominal features

Deep network

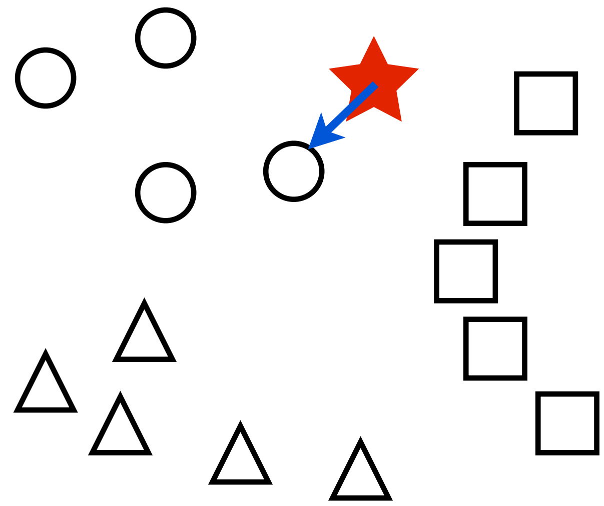
autoencoder:



[Hinton and Salakhutdinov, Science 2006]

Nearest neighbor

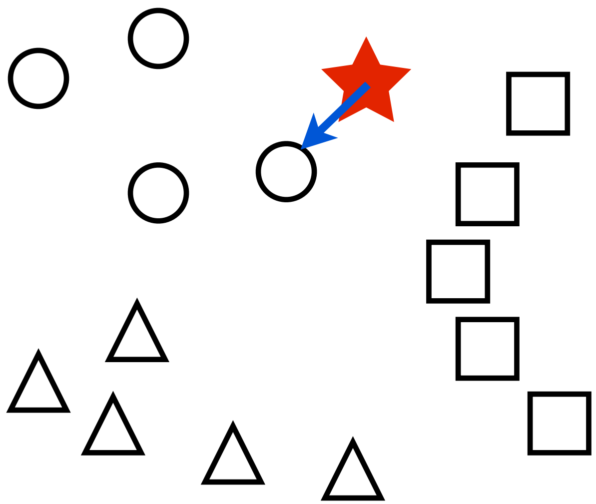
what looks similar are similar



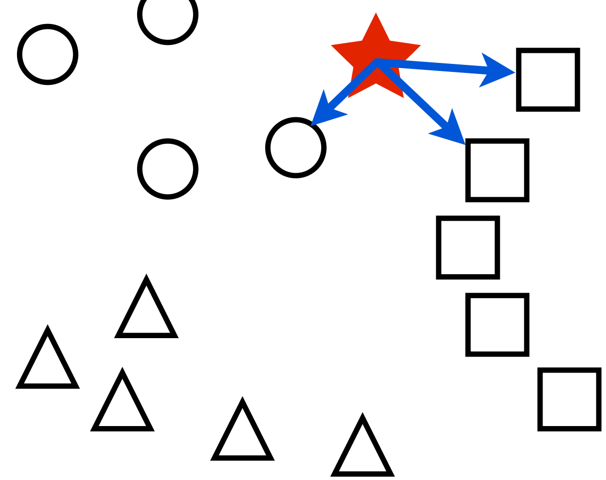
Nearest neighbor

for classification:

1-nearest neighbor:



k-nearest neighbor:

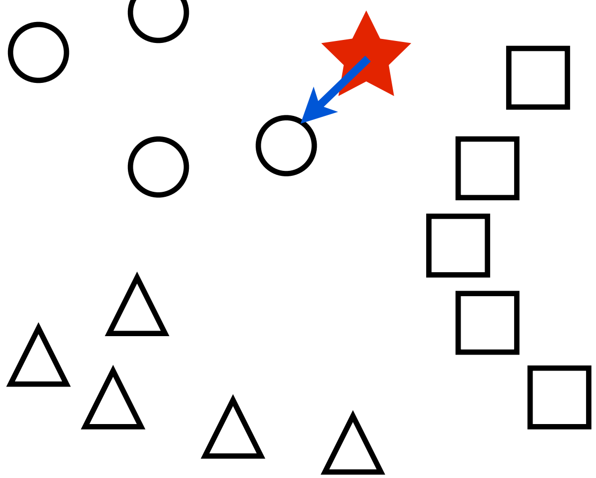


Predict the label as that of the NN
or the (weighted) majority of the k-NN

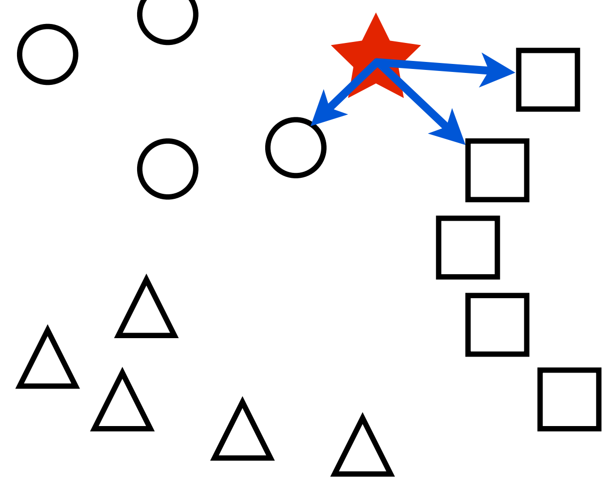
Nearest neighbor

for regression:

1-nearest neighbor:



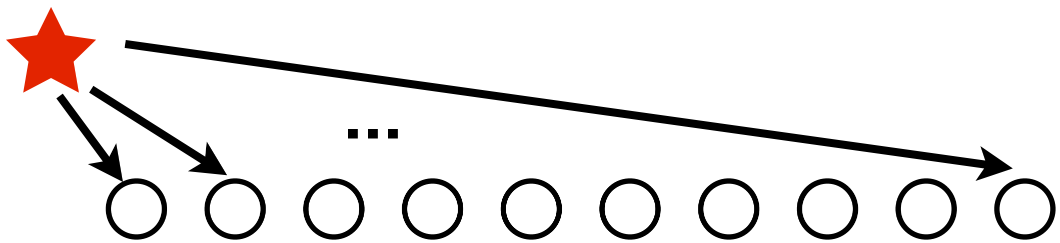
k-nearest neighbor:



Predict the label as that of the NN
or the (weighted) average of the k-NN

Search for the nearest neighbor

Linear search

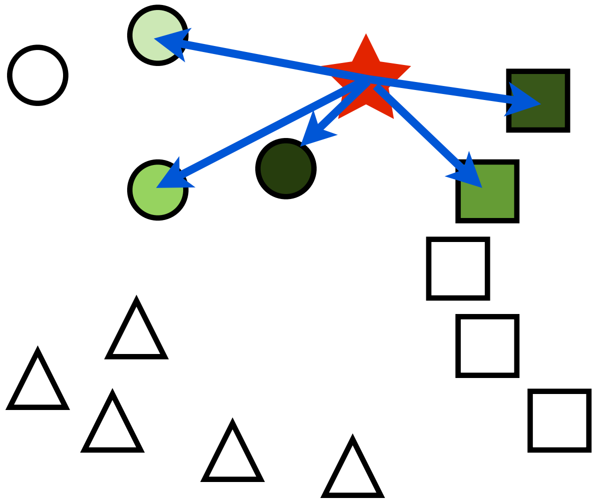


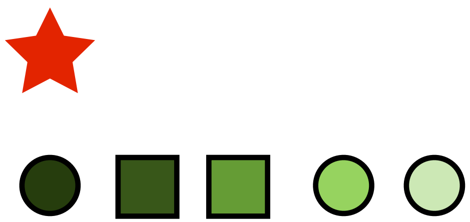
n times of distance calculations

$O(nk)$

Nearest neighbor

for retrieval:





Nearest neighbor classifier

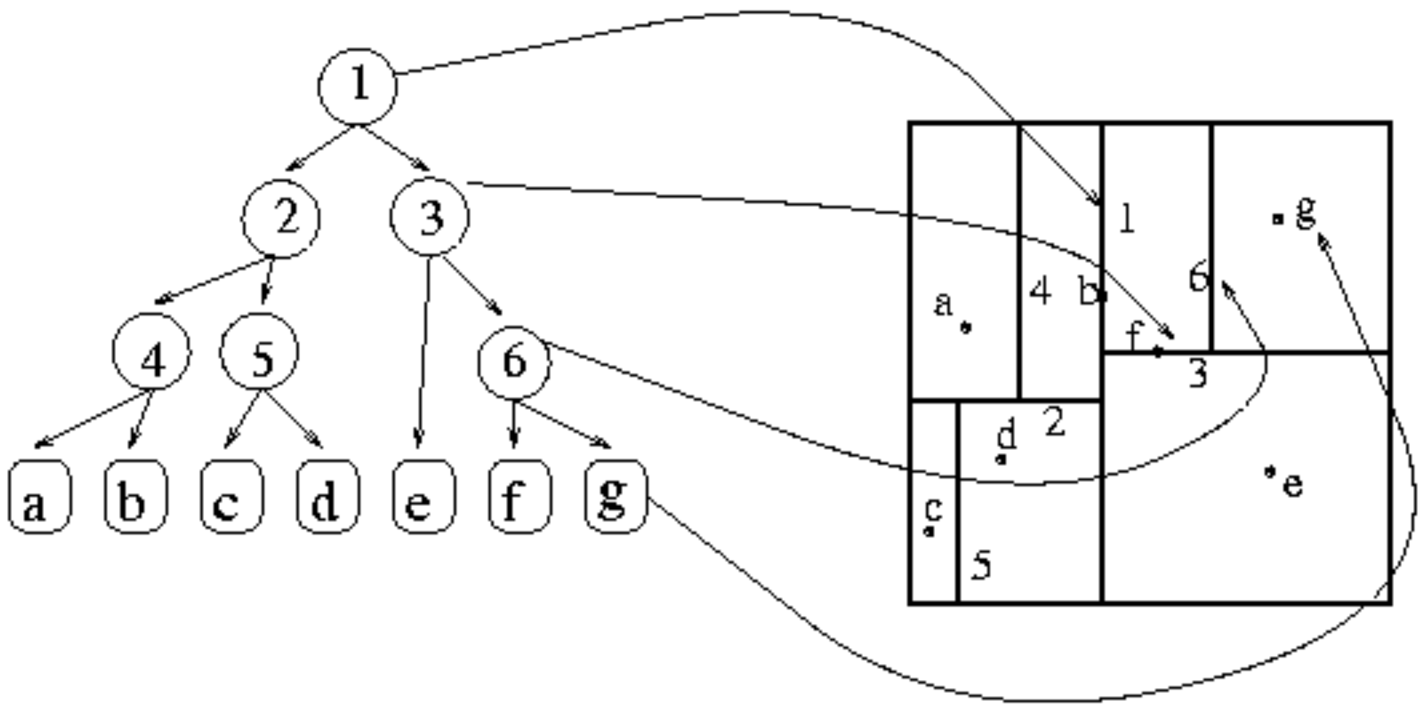
Nearest neighbor classifier

- as classifier, 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

Accelerate NN search: branch-and-bound

k-d tree:

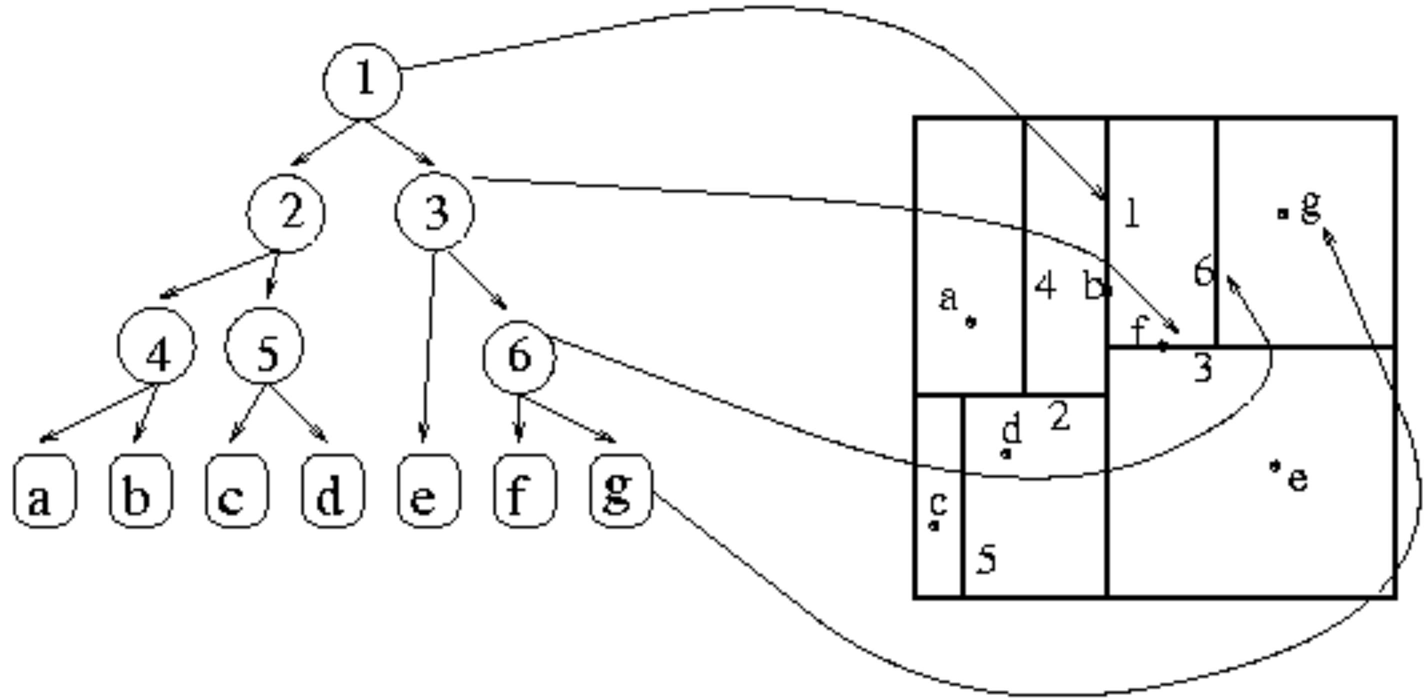


construction:
alternatively choose one dimension,
make a split by the median value.

[image from http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html]

Accelerate NN search: branch-and-bound

k-d tree:



linear search on *k*-d tree:

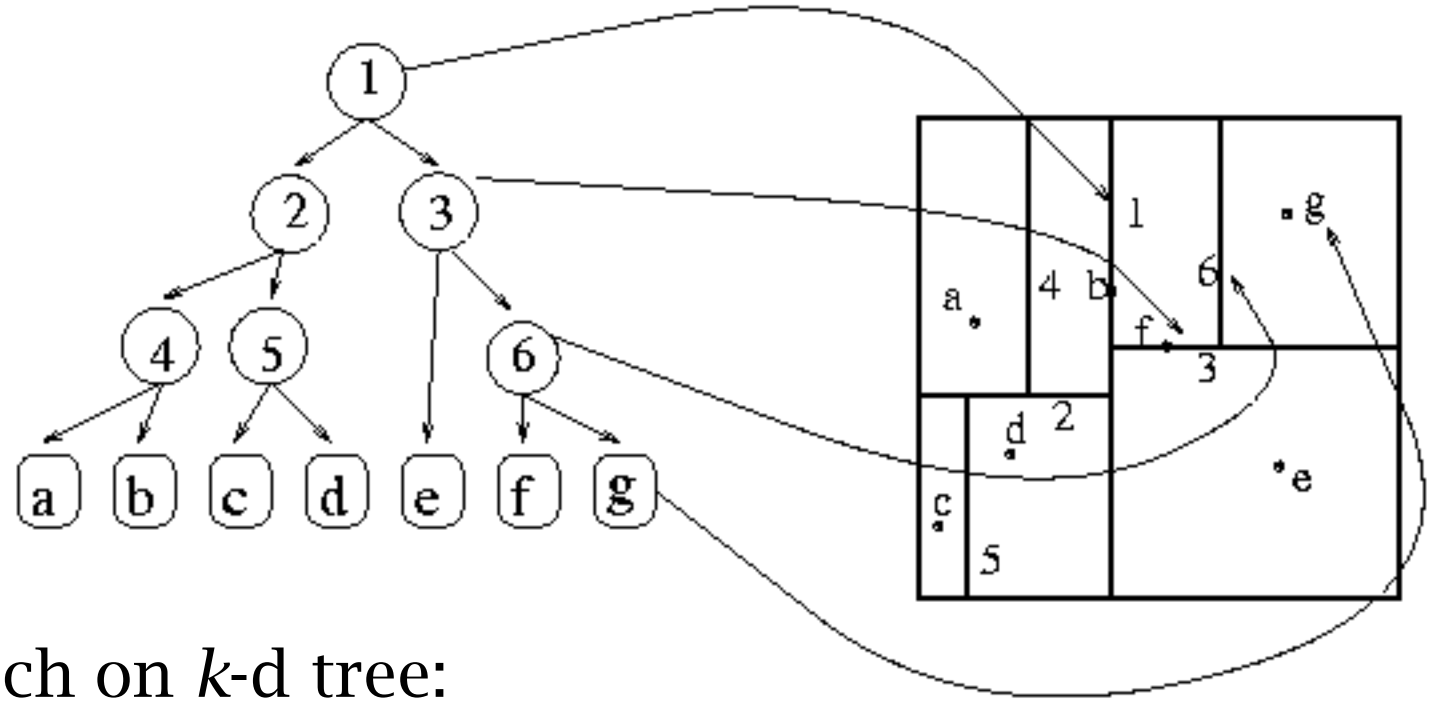
search(node,x):

1. if node is a leave, return the distance and the instance
2. compare *search*(left branch,x) and *search*(right branch,x)
3. return the instance with smaller distance

[image from http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html]

Accelerate NN search: branch-and-bound

k-d tree:



a smarter search on *k*-d tree:

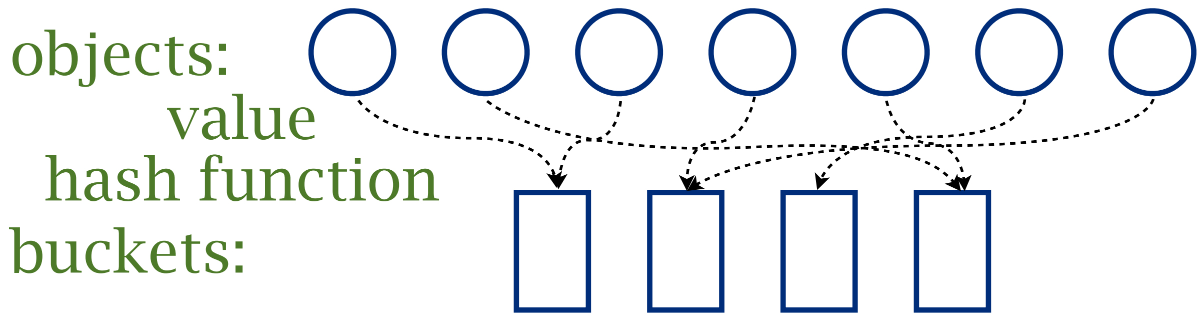
search(node,x):

1. if node is a leave, return the distance and the instance
2. if ***out-of-best-range***, return infinity distance
2. compare *search*(left branch,x) and *search*(right branch,x)
3. return the instance with smaller distance

[image from http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html]

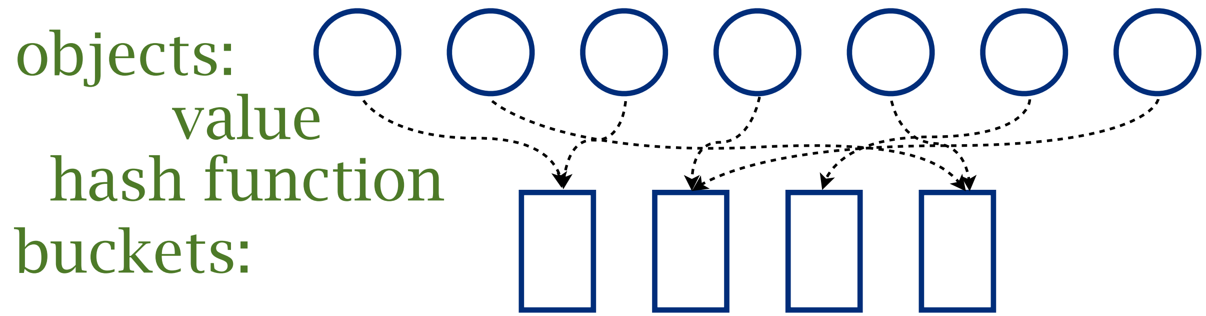
Accelerate NN search: hashing

hashing



Accelerate NN search: hashing

hashing



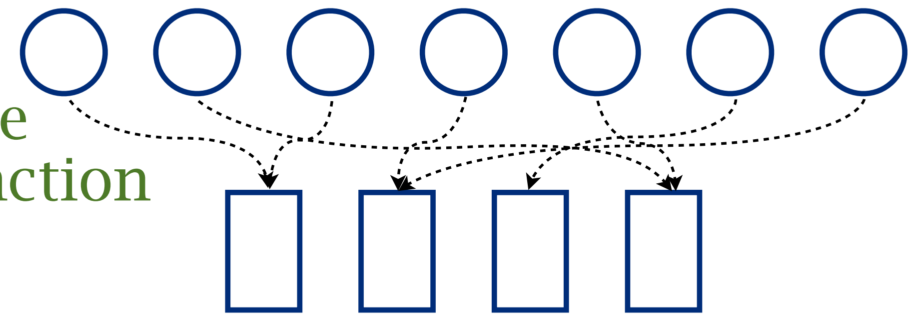
locality sensitive hashing:

similar objects in the same bucket

Accelerate NN search: hashing

hashing

objects:
value
hash function
buckets:



locality sensitive hashing:

similar objects in the same bucket

A LSH function family $\mathcal{H}(c, r, P_1, P_2)$ has the following properties for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$

$$\text{if } \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \leq r, \text{ then } P_{h \in \mathcal{H}}(h(\boldsymbol{x}_1) = h(\boldsymbol{x}_2)) \geq P_1$$

similar objects should be hashed in the same bucket with high probability

$$\text{if } \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| \geq cr, \text{ then } P_{h \in \mathcal{H}}(h(\boldsymbol{x}_1) = h(\boldsymbol{x}_2)) \leq P_2$$

dissimilar objects should be hashed in the same bucket with low probability

Accelerate NN search: hashing

Binary vectors in Hamming space

objects: (1100101101)

Hamming distance: count the number of positions with different elements

$$\|110101001, 110001100\|_H = 3$$

Accelerate NN search: hashing

Binary vectors in Hamming space

objects: (1100101101)

Hamming distance: count the number of positions with different elements

$$\|110101001, 110001100\|_H = 3$$

LSH functions: $\mathcal{H} = \{h_1, \ldots, h_n\}$ where $h_i(\boldsymbol{x}) = x_i$

	h_2	h_5	h_9
110101001	1	0	1
110010100	1	1	0
000110110	0	1	0
111001001	1	0	1
000011101	0	1	1

Accelerate NN search: hashing

Binary vectors in Hamming space

objects: (1100101101)

Hamming distance: count the number of positions with different elements

$$\|110101001, 110001100\|_H = 3$$

LSH functions: $\mathcal{H} = \{h_1, \ldots, h_n\}$ where $h_i(\boldsymbol{x}) = x_i$

	h_2	h_5	h_9
110101001	1	0	1
110010100	1	1	0
000110110	0	1	0
111001001	1	0	1
000011101	0	1	1

$$P(h_i(\boldsymbol{x}_1) = h_i(\boldsymbol{x}_2)) = 1 - \frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|}{d}$$

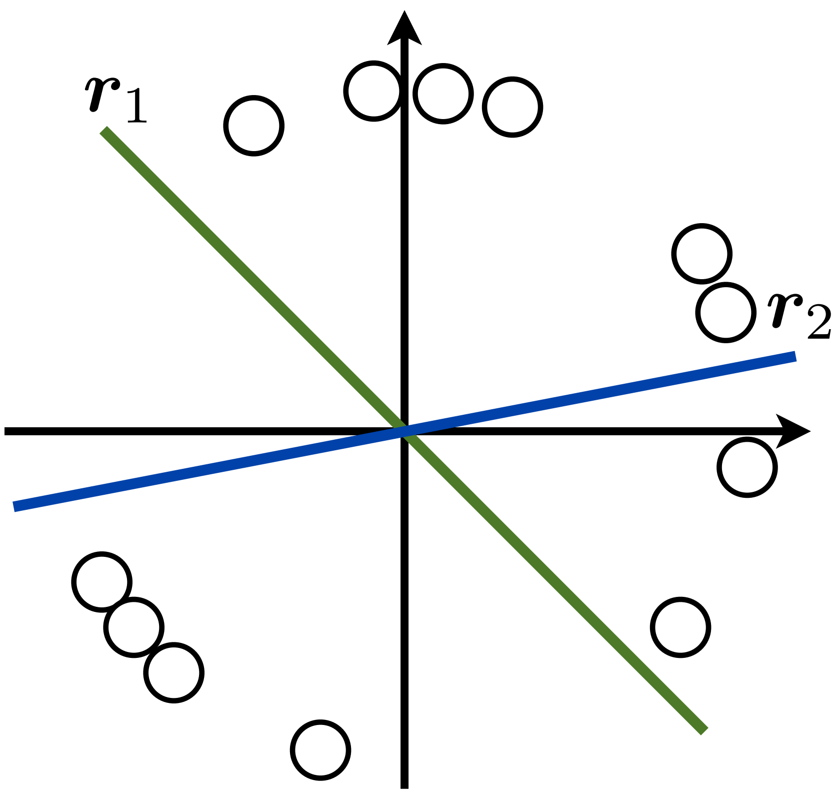
frequency in the same bucket for a sample of hashing functions

Accelerate NN search: hashing

Real vectors with angle similarity

$$\theta(\boldsymbol{x}_1, \boldsymbol{x}_2) = \arccos \frac{\boldsymbol{x}_1^\top \boldsymbol{x}_2}{\|\boldsymbol{x}_1\| \|\boldsymbol{x}_2\|}$$

LSH functions: $\mathcal{H} = \{h_{\boldsymbol{r}}\}(\boldsymbol{r} \in \mathbb{B}^n)$ where $h_r(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{r}^\top \boldsymbol{x})$



$$P(h_{\boldsymbol{r}}(\boldsymbol{x}_1) = h_{\boldsymbol{r}}(\boldsymbol{x}_2)) = 1 - \frac{\theta(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\pi}$$

frequency in the same bucket for a sample of hashing functions

习题

多层神经网络为何能实现非线性分类?

BP算法能否收敛到全局最优解?

k近邻分类算法是否需要训练预测模型?