Course ID 80250993 Dates: 9/16-12/23/2021 @ I-205
Tencent/VooV Meeting 521 4678 6257 (passwd 1205)

Chapter 4
Multi-category Classification and Nonlinear Classification

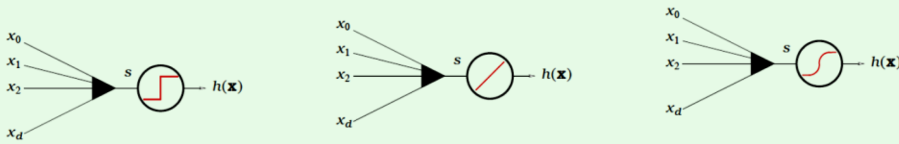
Xuegong Zhang
September 30, 2021

4.1
Multi-category Linear Classifiers

Recall: Three major types of linear machines

$$s = \sum_{i=0}^{d} w_i x_i$$

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$

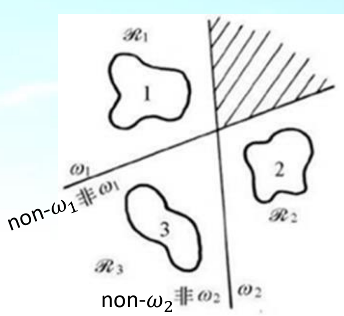


Abu-Mostafa, Magdon-Ismail, Lin, *Learning from Data, Lecture 9*

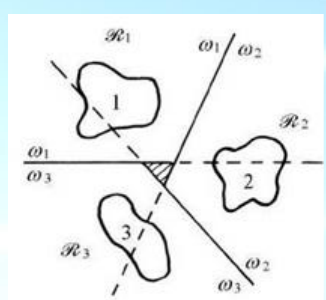
→ Binary Classification

Multi-category classification

- Strategies for multi-class classification using binary methods
  - One-vs-rest, aka one-vs-all, one-over-all (OVA)
  - Pairwise classification



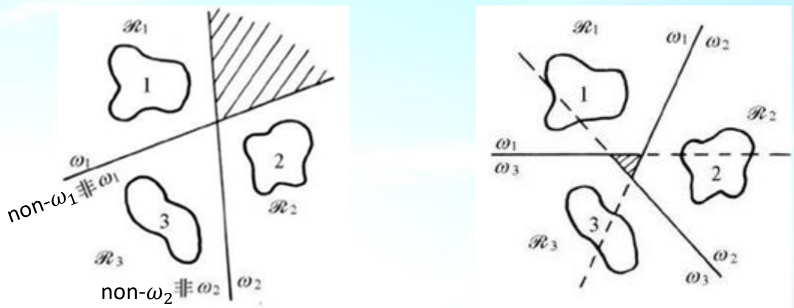
ω_i vs. non-ω_i



ω_i vs. ω_j

Challenges

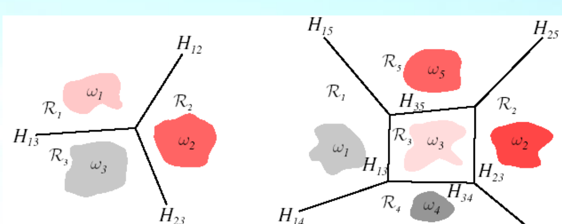
- Need a “code-book” to index the binary classifiers
- Need comprehensive voting schemes to make decisions based on multiple binary classifiers
- There could be “undefined” voting results



ω_i vs. non-ω_i ω_i vs. ω_j

Multi-category Linear Discriminants

- Given C categories, define C discriminant functions $g_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{y}$
- Classify $\boldsymbol{x}$ as a member of c_i if $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x})$ for all $j \neq i$



from Duda *et al.*

Algorithm:

If $y^k \in \omega_i$ but $a_i(k)^T y^k \leq a_j(k)^T y^k, j \neq i$

then $\begin{cases} a_i(k+1) = a_i(k) + \rho_k y^k \\ a_j(k+1) = a_j(k) - \rho_k y^k \\ a_l(k+1) = a_l(k), \; l \neq i, j \end{cases}$

——“Multi-class linear machine”

- Converge when data are linearly separable

Multiclass Fisher's Linear Discriminant

R

- a.k.a. Multi-class LDA
- Within-class scatter matrix $\mathrm{S_w} = \sum_k \sum_{x\in\omega_i} (\boldsymbol{x}-\boldsymbol{m}_k)(\boldsymbol{x}-\boldsymbol{m}_k)^{\mathrm{T}}$
- Between-class scatter matrix $\mathrm{S_b} = \sum_k n_k\, (\boldsymbol{m}_k-\boldsymbol{m})(\boldsymbol{m}_k-\boldsymbol{m})^{\mathrm{T}}$

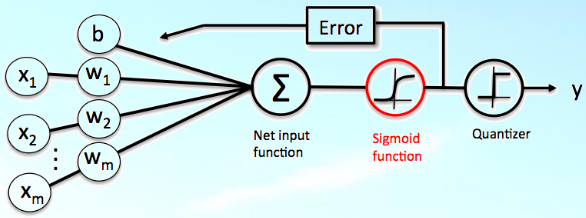
$$W\!:\ \max\quad J(W) = \frac{|W^{\mathrm{T}}\mathrm{S_b}\mathrm{W}|}{|W^{\mathrm{T}}\mathrm{S_w}\mathrm{W}|}\quad \text{or}\quad \hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^{\mathrm{T}}\mathrm{S_b}\mathbf{w}}{\boldsymbol{w}^{\mathrm{T}}\mathrm{S_w}\mathbf{w}}$$

- Solution: the $\le c-1$ eigenvector solutions of $\mathrm{S_b}\boldsymbol{w} = \lambda S_{\mathrm{w}}\boldsymbol{w}$

—— reduction to $\le c-1$ dimensions, but not solving the classifiers

4.2
Multi-category Logistic Regression and Softmax

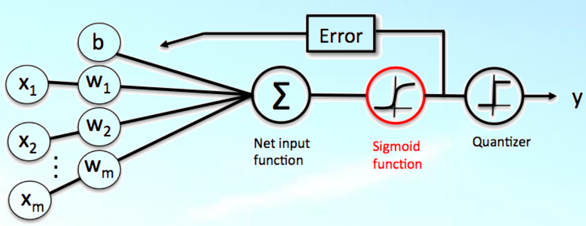
Binary logistic regression classifier



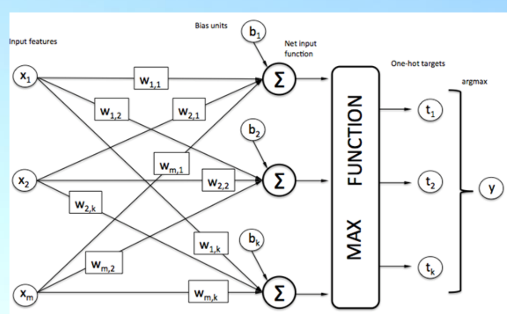
Logistic regression classifier

- Binary classification:
  - Weighted sum → Logistic function → Compare with threshold→ Classification

Multiclass classification with logistic regression

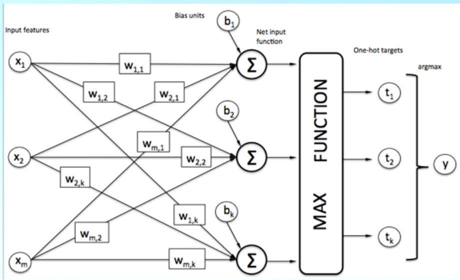


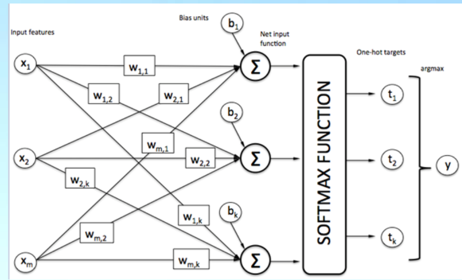
Logistic regression classifier



- Binary classification:
  - Weighted sum → Logistic function → Compare with threshold→ Classification
- Multiclass classification:
  - Weighted sum → Compare among peers (Max function) → Classification

Multiclass logistic regression and SoftMax

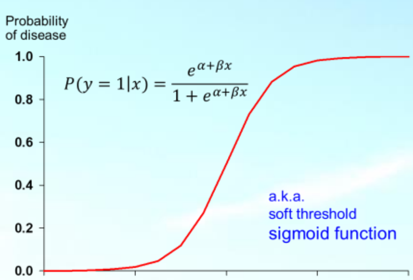


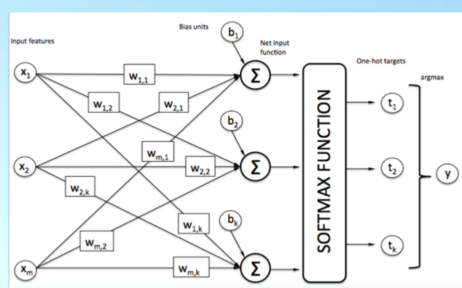


- Problem: no consideration on competition among class probabilities
- Solution: Max functions → SoftMax function

Xuegong Zhang Modified from https://rasbt.github.io/mlxtend/user_guide/classifier/SoftmaxRegression/ 11

SoftMax (Normalized exponential function)





$$P(y=1|\boldsymbol{x})=\frac{e^{\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}{1+e^{\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}$$

$$\boxed{P(y=j|\boldsymbol{x})=\frac{e^{\boldsymbol{w}_j\cdot\boldsymbol{x}}}{\sum_{k=1}^{K} e^{\boldsymbol{w}_k\cdot\boldsymbol{x}}},\quad j=1,\cdots,K}$$

--- SoftMax function

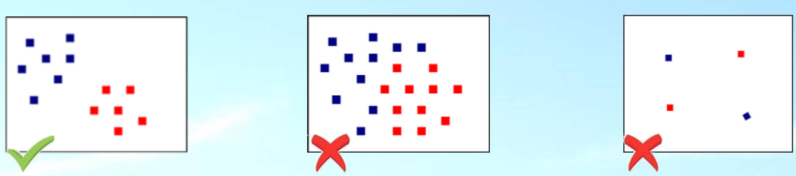
Warning: many random names in ML ☺

Xuegong Zhang 12

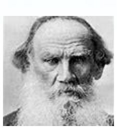
4.3
Quadratic Discriminant

The need for nonlinear classifiers

- Linearly non-separable cases → The need for nonlinear classifiers



- Linearly non-separable ≠ nonlinearly separable
- “nonlinearities” are not the same



Leo Tolstoy:
“All happy families resemble one another, but each un-happy family is unhappy in its own way.”

Quadratic discriminant analysis (QDA)

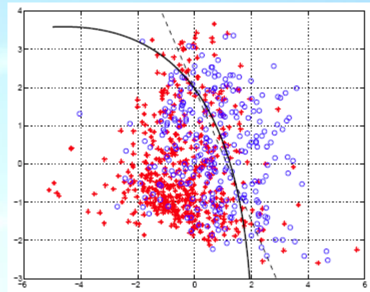
- Quadratic functions: might be the simplest nonlinear functions
- Quadratic discriminant:

$$g_i(\boldsymbol{x}) = k_i^2 - (\boldsymbol{x} - \widehat{\boldsymbol{m}}_i)^T \widehat{\Sigma}_i^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{m}}_i),\ i = 1, \cdots, C$$

mean $\widehat{\boldsymbol{m}}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{x}_j$

covariance matrix $\widehat{\Sigma}_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\boldsymbol{x}_j - \boldsymbol{m}_i)(\boldsymbol{x}_j - \boldsymbol{m}_i)^T$

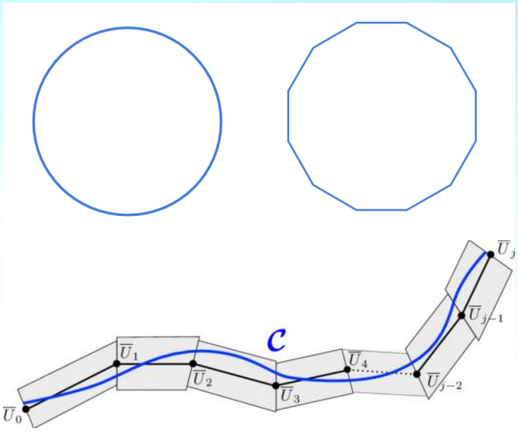
- Decision: $\text{class}(\boldsymbol{x}) = \underset{i}{\operatorname{argmax}}\, g_i(\boldsymbol{x})$
- When to choose QDA (instead of LDA)?
  - Large training data set, roughly normal distribution
  - Covariance matrixes different between classes

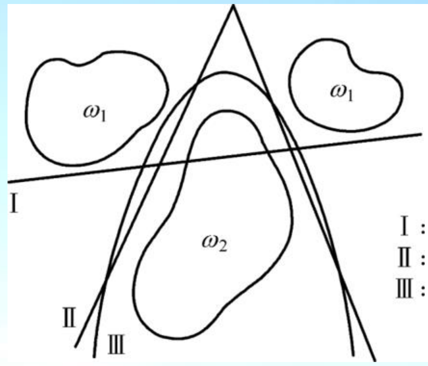


https://online.stat.psu.edu/stat508/lesson/9/9.2/9.2.8

4.4 Piecewise Linear Classifiers

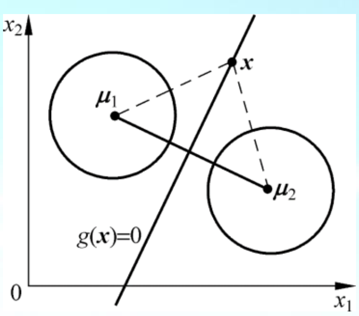
Piecewise linearity to approximate any nonlinearity

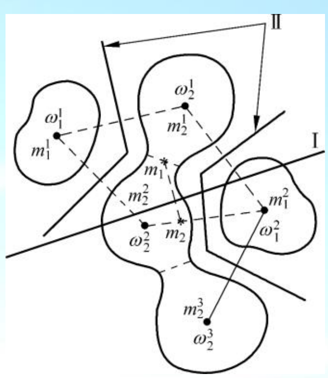




Ⅰ：Linear classifier
Ⅱ：Piecewise linear classifier
Ⅲ：Quadratic classifier

Minimal distance classifier





- With given sub-class regions

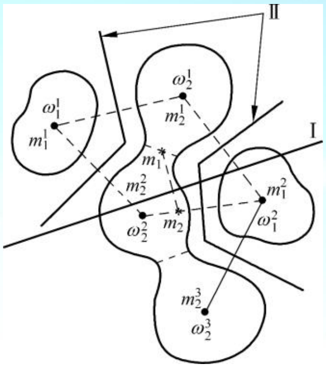
  $R_i^l, l = 1,2,\cdots,l,\ i = 1,2,\cdots,c$

  $$g_i(\boldsymbol{x}) = \min_{l=1,\cdots,l_i} \left\|\boldsymbol{x} - \boldsymbol{m}_i^l\right\|$$

- Decision:

  $$\text{class}(\boldsymbol{x}) = \underset{i}{\text{argmin}}\, g_i(\boldsymbol{x})$$

Piecewise linear discriminant



- With given sub-classes

$$\omega_i = \left\{\omega_i^1, \omega_i^2, \cdots, \omega_i^{l_i}\right\}, i = 1,2,\cdots,c$$

- Linear discriminants

$$g_i^l(\boldsymbol{x}) = \boldsymbol{w}_i^l \cdot \boldsymbol{x} + \omega_{i0}^l, l = 1,\cdots,l_i, i = 1,\cdots,c$$

$$g_i(\boldsymbol{x}) = \max_{l=1,\cdots,l_i} g_i^l(\boldsymbol{x}), i = 1,\cdots,c$$

- Decision

$$\text{class}(\boldsymbol{x}) = \operatorname*{argmax}_{i=1,\cdots,c} g_i(\boldsymbol{x})$$

An algorithm for piecewise linear classification

① Initialization:
- Set: l_i subclasses in class ω_i, $i = 1,\cdots,c$
- Weights of linear discriminant of subclass l_i at iteration t: $\alpha_i^l(t)$
- Initialize $\boldsymbol{\alpha}_i^l(0)$, $i = 1,\cdots,c$, $l = 1,\cdots,l_i$

② For sample $\boldsymbol{y}_k \in \omega_j$, find $\boldsymbol{\alpha}_j^m(t)^T\boldsymbol{y}_k = \max_{l=1,\cdots,l_j}\left\{\boldsymbol{\alpha}_j^l(t)^T\boldsymbol{y}_k\right\}$, check
- If $\boldsymbol{\alpha}_j^m(t)^T\boldsymbol{y}_k > \boldsymbol{\alpha}_i^l(t)^T\boldsymbol{y}_k, \forall i = 1,\cdots,c$, $i \neq j$, $l = 1,\cdots,l_i$, continue;
- Else if $\exists i \neq j$, $\boldsymbol{\alpha}_j^m(t)^T\boldsymbol{y}_k \leq \boldsymbol{\alpha}_i^n(t)^T\boldsymbol{y}_k$ in subclass n, find the subclass n with largest discriminant, do correction:

③ $\boldsymbol{\alpha}_j^m(t+1) = \boldsymbol{\alpha}_j^m(t) + \rho_k\boldsymbol{y}_k$

④ $\boldsymbol{\alpha}_i^n(t+1) = \boldsymbol{\alpha}_i^n(t) - \rho_k\boldsymbol{y}_k$

③ Repeat ② with the next sample, until convergence.

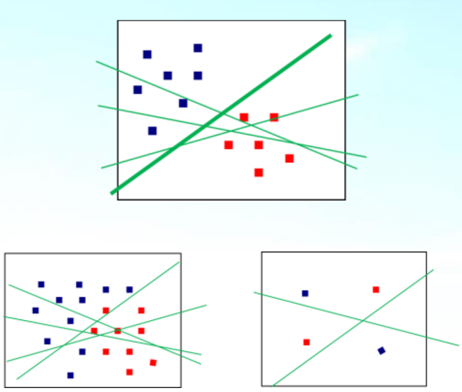
1-minute break

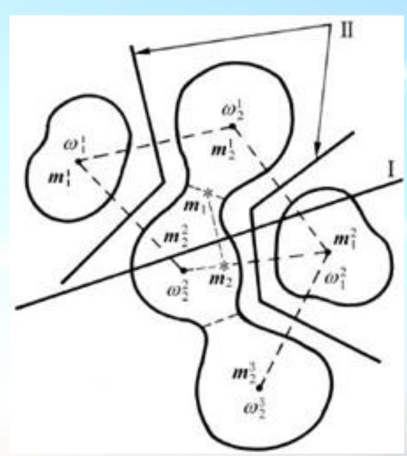
4.5
Nearest Neighbor Classifiers

4.5.1
Nearest Neighbor Methods

Piece-wise linear classifier

- One way for nonlinearity: Piece-wise linear classifier
  → Classification of multiple sub-classes





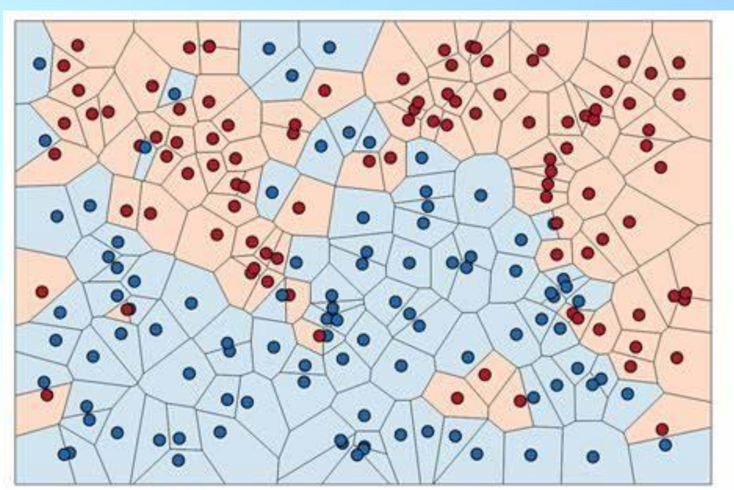
Extreme case: ➔ Nearest-Neighbor (NN) method

- Sample set: $S_N = \{(x_1, \theta_1), (x_2, \theta_2), \cdots (x_N, \theta_N)\}$
  - x_i: samples, labels: $\theta_i = \{1, 2, \cdots, c\}$
- Distance measure $\delta(x_i, x_j)$, e.g. $\delta(x_i, x_j) = \|x_i - x_j\|$
- Nearest Neighbor Decision (1-NN):

  If $\delta(x, x') = \min\limits_{j=1,\cdots N} \delta(x, x_j)$ and $\omega(x') = \theta'$, then $\widehat{\omega}_1(x) = \theta'$
- Or:

  Discriminant function of ω_i: $g_i(x) = \min\limits_{k=1,\cdots N_i} \|x - x_i^k\|$

  Decision: If $g_j(x) = \min\limits_{i=1,\cdots c} g_i(x)$, then $x \in \omega_j$

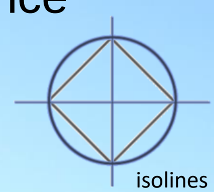


Some popular distance measurements

- Minkovski Metric (of order s): $\delta(\boldsymbol{x}_k, \boldsymbol{x}_l) = \left[\sum_{i=1}^{d} |x_{ki} - x_{li}|^s\right]^{\frac{1}{s}}$
- Euclidean Distance: $\delta_E(\boldsymbol{x}_k, \boldsymbol{x}_l) = \left[(\boldsymbol{x}_k - \boldsymbol{x}_l)^{\mathrm{T}}(\boldsymbol{x}_k - \boldsymbol{x}_l)\right]^{\frac{1}{2}}$
- City-Block Distance: $\delta(\boldsymbol{x}_k, \boldsymbol{x}_l) = \sum_{i=1}^{d} |x_{ki} - x_{li}|$
- Chobychev Distance: $\delta(\boldsymbol{x}_k, \boldsymbol{x}_l) = \max_i |x_{ki} - x_{li}|$
- Squared Distance: $\delta(\boldsymbol{x}_k, \boldsymbol{x}_l) = (\boldsymbol{x}_k - \boldsymbol{x}_l)^{\mathrm{T}} Q (\boldsymbol{x}_k - \boldsymbol{x}_l)$
- Nonlinear distances, e.g., $\delta(\boldsymbol{x}_k, \boldsymbol{x}_l) = \begin{cases} H & \text{if } \delta_E(\boldsymbol{x}_k, \boldsymbol{x}_l) \geq T \\ 0 & \text{if } \delta_E(\boldsymbol{x}_k, \boldsymbol{x}_l) < T \end{cases}$

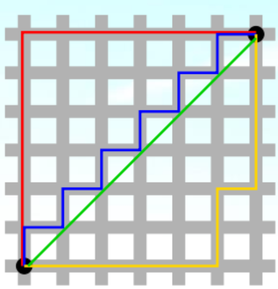
Manhattan Distance vs. Euclidean Distance

- Euclidean (L2) Distance: $\delta_E(\boldsymbol{x}_k, \boldsymbol{x}_l) = \left[(\boldsymbol{x}_k - \boldsymbol{x}_l)^{\mathrm{T}}(\boldsymbol{x}_k - \boldsymbol{x}_l)\right]^{\frac{1}{2}}$
- City-Block (L1) Distance: $\delta(\boldsymbol{x}_k, \boldsymbol{x}_l) = \sum_{i=1}^{d} |x_{ki} - x_{li}|$

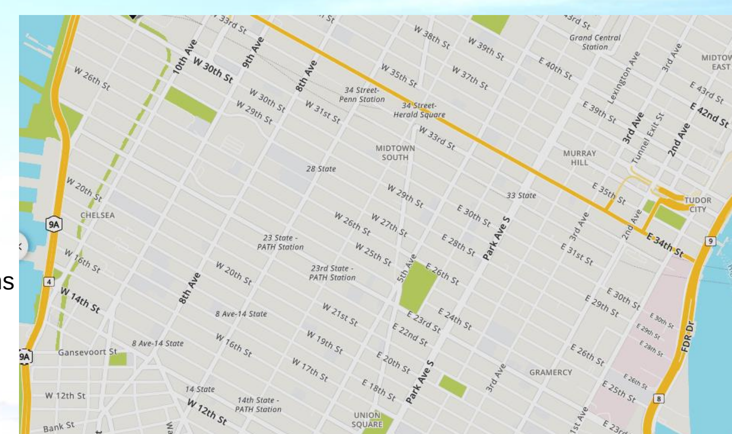


isolines

a.k.a. Manhattan distance, taxicab distance



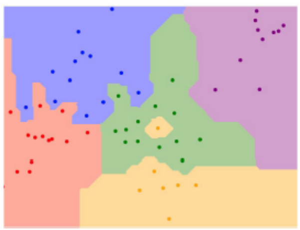
Taxicab geometry versus Euclidea distance: In taxicab geometry, the red, yellow, and blue paths all have the same shortest path length of 12. In Euclidean geometry, the green line has length $6\sqrt{2} \approx 8.49$, and is the unique shortest path.



K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

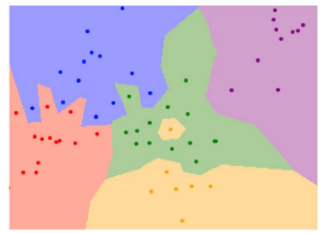
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

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



K = 1

Fei-Fei Li & Justin Johnson & Serena Yeung   Lecture 2 - 33   April 5, 2018

k-Nearest Neighbors (k-NN) method

- A generalization of 1-NN

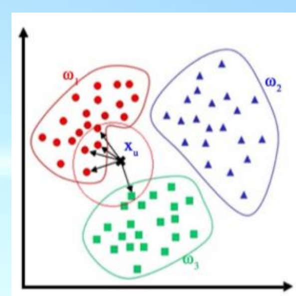
$$S_N = \{(x_1, \theta_1), (x_2, \theta_2), \cdots (x_N, \theta_N)\}, \quad \theta_i = \{1,2,\cdots,c\}$$

- Discriminant for ω_i:

$$g_i(x) = k_i$$

where $k_i, i = 1, \cdots, c$ is the number of samples belonging to ω_i among the k nearest neighbors of x

- Decision: If $g_j(x) = \max\limits_{i=1,\cdots c} g_i(x)$, then $x \in \omega_j$



4.5.2 Asymptotic Errors of Nearest Neighbor Methods

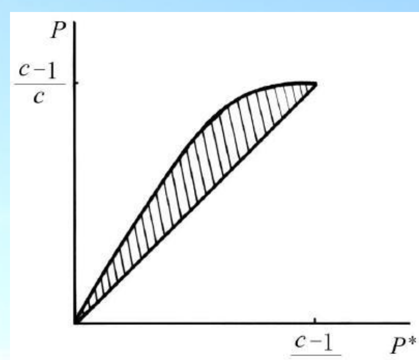
Conclusion:

- Error rate P_1 of NN method for i.i.d. samples, comparing with the best possible error rate

$$P^* \leq P_1 \leq P^* \left(2 - \frac{c}{c-1} P^*\right)$$

where P^* is the Bayesian error (smallest error given the distributions),

and P_1 is the **asymptotic** error of NN



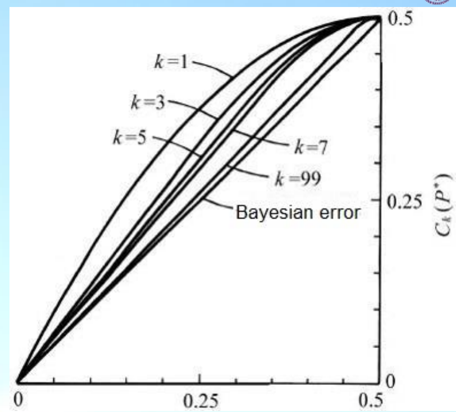
- Asymptotic error of k-NN

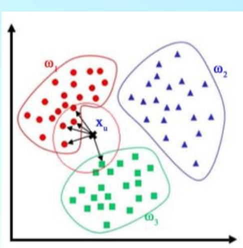
$$P^* \le P_k \le P^*\left(2 - \frac{c}{c-1}P^*\right)$$

Or:

$$P^* \le P_k \le 2P^*$$

- When $N \to \infty$, $k \uparrow$, $P_k \downarrow$
  given that $k \ll N$

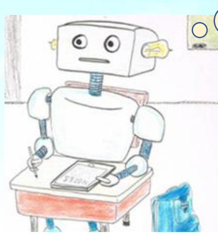




So simple, so easy, and so good!

Who told me to learn? No need! I just need to memorize training samples.

D



单选题 1分

设置

4-min break

A Yes, I’m back to my computer.

B Sorry, not yet.

提交

清華大學

4.5.3
Improved Nearest Neighbor Methods

- So simple, so easy, and so good!

—— Wait! No free lunch!

- Potential challenges:
  - Huge memory and computation
  - “Voting democracy” can be risky, esp. with noisy samples
    - Rejection when voting is too close
    - Weight voting, …
  - May not work well when N is too small

- Computational Issues
  - Need to store all training samples → Memory Cost
  - Need to compare with all samples → Computation Cost

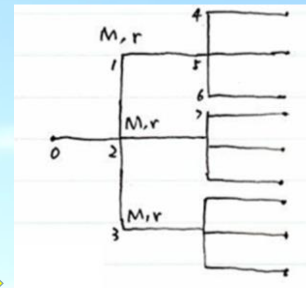
  Q: What is the time complexity of finding the nearest neighbor in N training samples?
- Fast k-NN algorithms

  →Basic idea: Reduce the online time by offline preparation beforehand

- Basic idea of fast k-NN algorithms

  →Reduce the online time by offline preparation beforehand

- The Branch-Bound (BB) Algorithm
  - Organize samples as subsets in a tree structure
  - Use few parameters to represent a subset (node)
  - Compare with nodes instead of all samples
  - Only compare with individual samples at the end nodes



Branch-Bound (BB) Algorithm (for 1-NN)

- Symbols:

  $\mathcal{X}_p$: Sample subset of node p

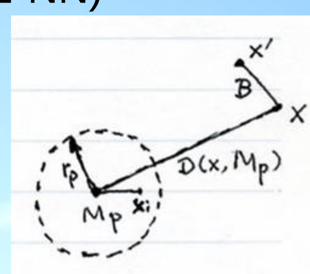
  N_p: Number of samples in $\mathcal{X}_p$

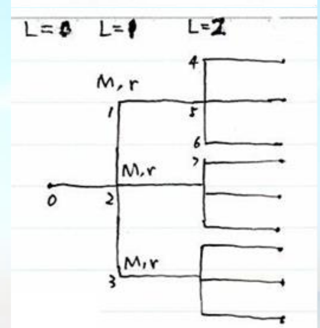
  M_p: Sample mean of $\mathcal{X}_p$

  $r_p = \max_{x_i \in \mathcal{X}_p} D(x_i, M_p)$: the farthest distance in $\mathcal{X}_p$ from its center

  B: Record on the current nearest distance

- Two procedures:
  1. Divide samples into subsets (e.g., using clustering algorithms), compute and store the M_p, r_p and all $D(x_i, M_p)$ of $\mathcal{X}_p$
  2. Using the BB algorithm to search through the tree





- Rule 1:

  For new sample x and node $\mathcal{X}_p$:

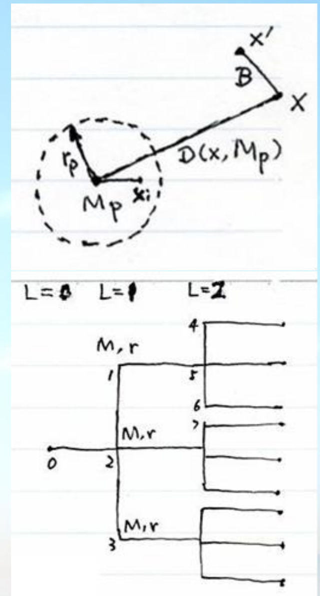
  If $D(x, M_p) > B + r_p$, the nearest neighbor of x cannot be in $\mathcal{X}_p$.

- Rule 2:

  For new sample x and training sample $x_i \in \mathcal{X}_p$:

  If $D(x, M_p) > B + D(x_i, M_p)$,

  x_i is NOT the nearest neighbor of x .



- BB Algorithm for the nearest neighbor:

1° (Initialize)

Set $B = \infty, L = 1, p = 0$.

2° (Expand)

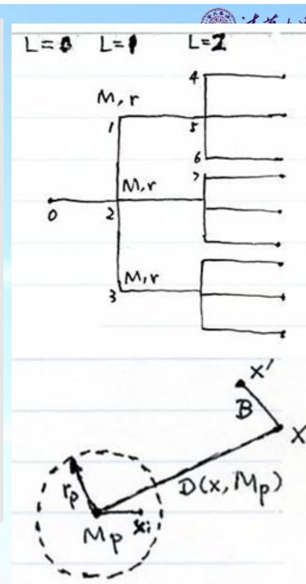
Put children of current node in the Active Table of the current level, and compute and store their $D(x, M_p)$.

3° (Check)

For each node in the Active Table, if $D(x, M_p) > B + r_p$, remove it from the Table (according to Rule 1).

4° (Trace back)

If no node in the Active Table, go back to the upper level and set $L = L - 1$.
If $L == 0$, stop; otherwise go to step 3°.
If there are nodes in the Active Table, continue to 5°.



5° （Choose nearest node）

Choose the nearest node with smallest $D(x, M_p)$ in the Active Table, say, p', set it as the current node.

If current level L is the end level, go to 6° ; otherwise, set $L = L + 1$, go to 2°**.**

6° （Check）

For every x_i in current node p',

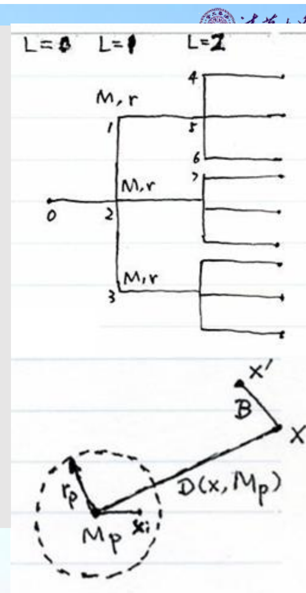
If $D\left(x, M_p\right) > B + D(x_i, M_p)$ then not the nearest (Rule 2)

Otherwise, compute$D(x, x_i)$. If $D(x, x_i) < B$, set $NN = i, B = D(x, x_i)$

After all x_i in p' being checked, go to 3°**.**

- When algorithm ends, output the x_{NN} and $D(x, x_{NN}) = B$ as the nearest neighbor and the nearest distance.

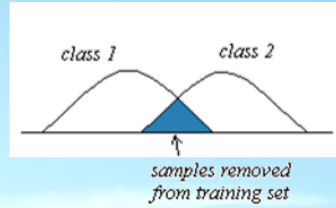
(for k-NN, we only need to modify step 6°)



Examples of Improved Nearest Neighbor Methods

Editing Nearest Neighbor Method

- Consideration
  - Samples in the overlapping region of two classes can confuse the algorithm
  - Removing them should help to improve



Editing Nearest Neighbor Method

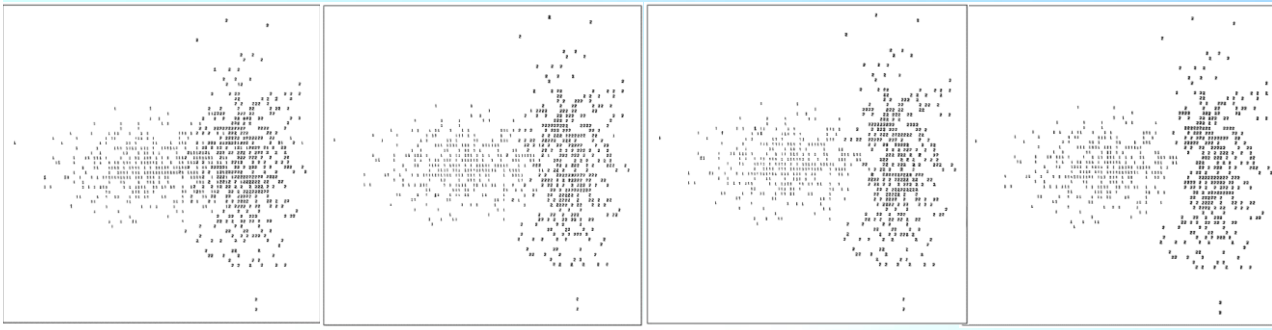
- How to know samples in the confusing zone?
- Idea:
  – Pre-classification to detect samples in confusing zone
  – Edit: remove samples in confusing zone
  – Can be done for multiple rounds if we have sufficient samples

- Asymptotic error of 1-NN using 1-NN pre-classification editing

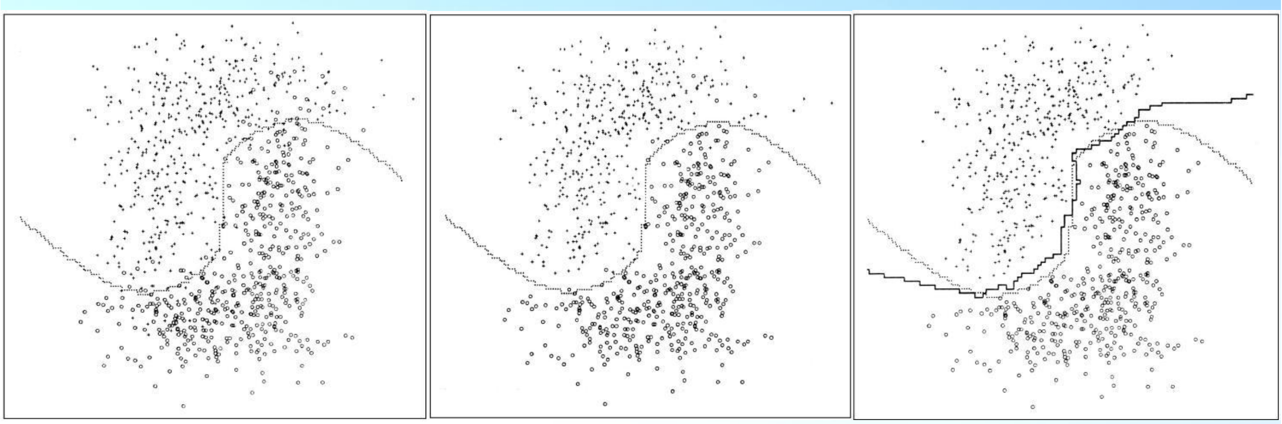
$$P_1^E(e) = \frac{P_1(e)}{2[1 - P_1(e)]}$$

ref. $P^* \leq P_1 \leq 2P^*$

- Examples (multi-edit)



- Examples (multi-edit)

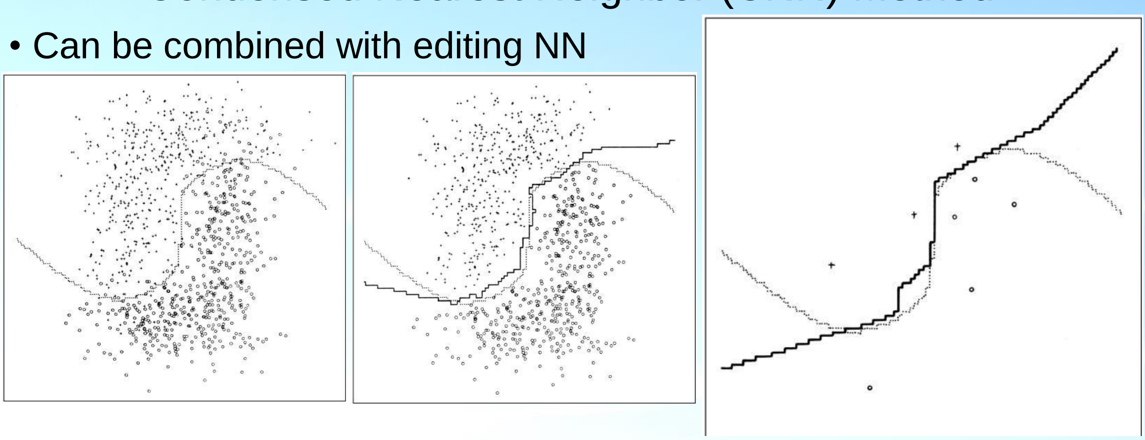


Condensed Nearest Neighbor (CNN) Method

- Find a few representative samples to significantly reduce storage
  - Separate $\mathcal{X}^N$ as $\mathcal{X}_S$ and $\mathcal{X}_G$
  - Start with only 1 sample in $\mathcal{X}_S$, all others in $\mathcal{X}_G$
  - Consider each sample in $\mathcal{X}_G$, if correctly classified with $\mathcal{X}_S$ then stay, otherwise move to $\mathcal{X}_S$
  - …
  - Use only samples in $\mathcal{X}_S$ as the final set

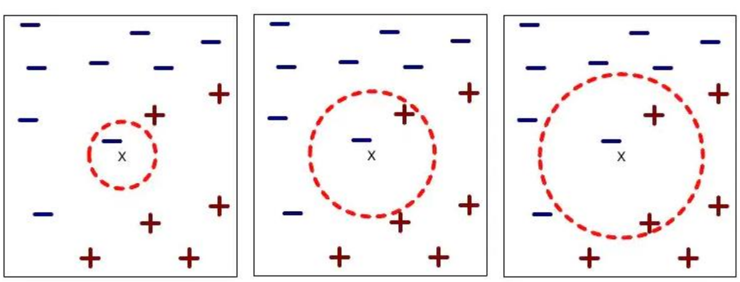
Condensed Nearest Neighbor (CNN) Method

- Can be combined with editing NN

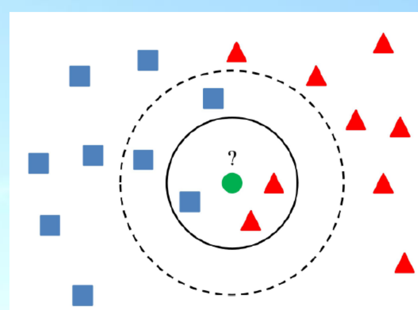


4.5.4
Discussions on Nearest Neighbor Methods

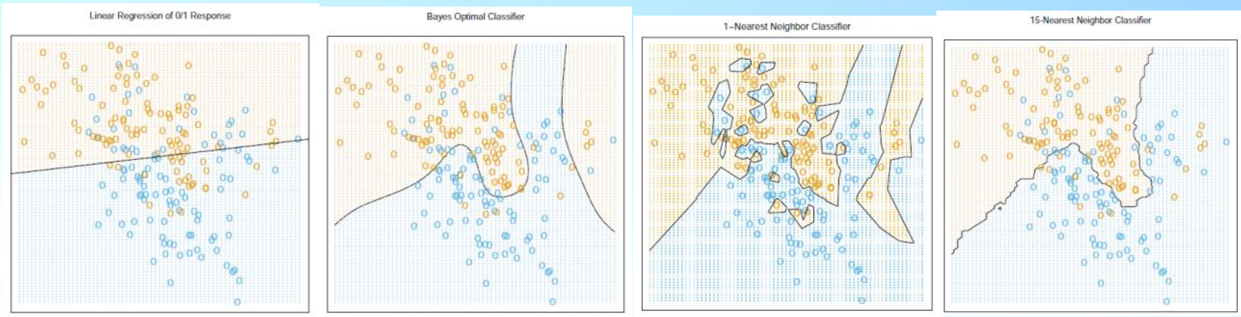
k-Nearest Neighbors (k-NN)



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

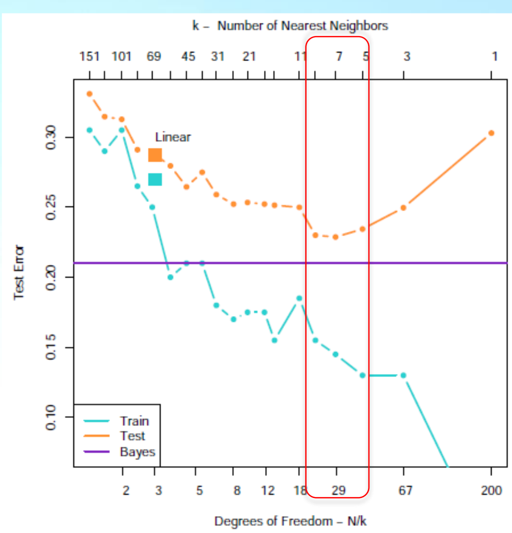


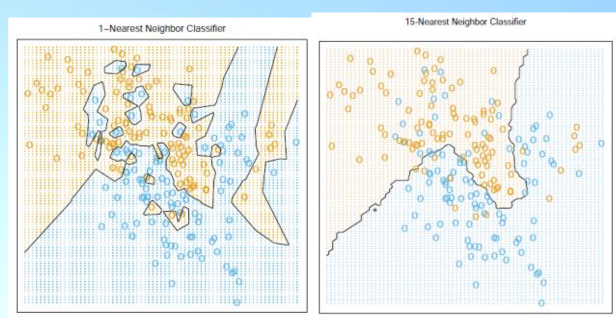
About the choice of k in k-NN



Ref. Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning*, Springer

About the choice of k in k-NN





Ref. Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning*, Springer

- The curse of dimensionality
  - In high dimension space, your nearest neighbor can be far away
  - You may need tremendous amount of samples to make NN work

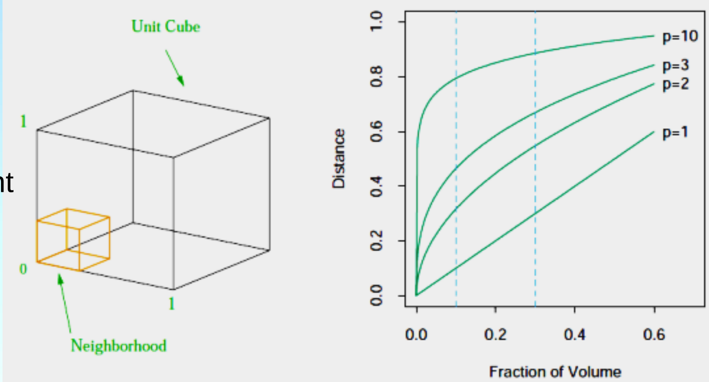


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.*

Ref. Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning*, Springer

Discussion

- In what scenario would you prefer using k-NN?
  - Dimensionality, Sample size
  - Expectation on the complexity of classification
  - Interpretability
- How do you choose *k* in a practical application?
  - Rule of thumb
  - Exploratory experiments
- What distance/preprocessing would you use in a practical application?
  - Characteristics of features
  - Relative importance of features

D

H

Homework

- Computer exercises (Ex2)
  - Find a package of KNN
  - Describe its algorithm
  - Write your own code of MLP
  - Experiment on the medical dataset
- D Deadline:
  - Oct. 13 (Sunday), 23:00

Xuegong Zhang 57

单选题 1分 设置

5-min break

A Yes, I’m back to my computer.

B Sorry, not yet.

提交

Xuegong Zhang 58