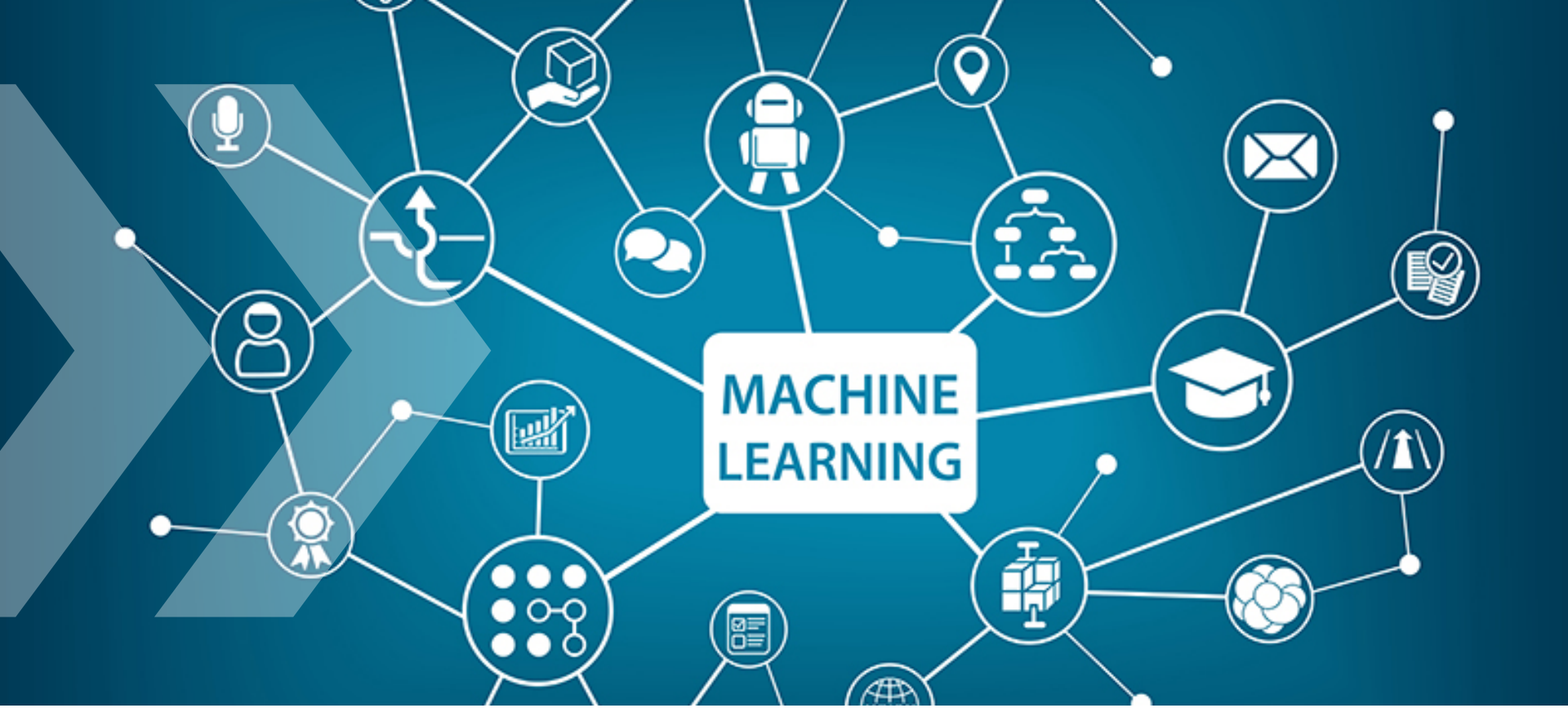


CS480/680: Intro to ML

Lecture 05: *K*-nearest neighbors

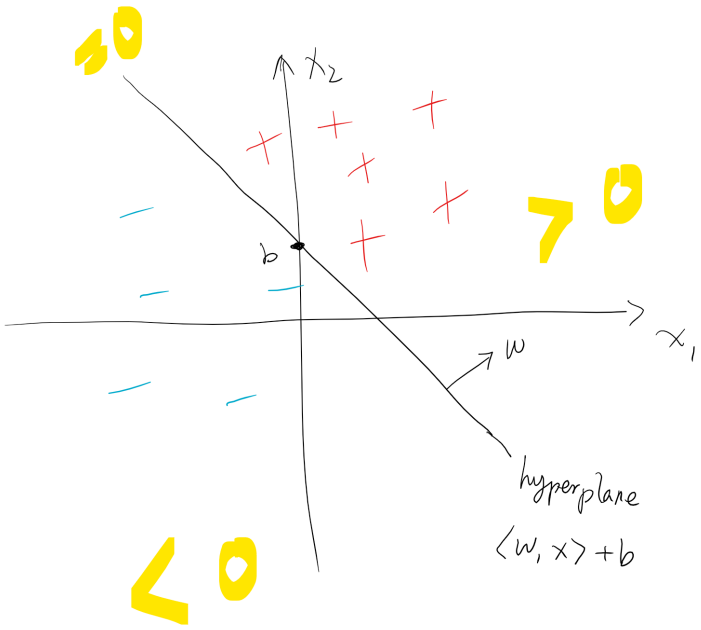
UNIVERSITY OF WATERLOO

Outline

- Algorithm
- Theory
- Application

Classification revisited

- $\hat{y} = \text{sign}(\mathbf{x}^T\mathbf{w} + b)$
- Decision boundary: $\mathbf{x}^T\mathbf{w} + b = 0$
- Parametric: finite-dim $\mathbf{w}$
- Non-parametric: no specific form (or inf-dim $\mathbf{w}$)



1-Nearest Neighbour

- Store training set (X, **y**)

- For query (test point) **x**'
  - find nearest point **x** in X
  - predict y' = y(**x**)

What do you mean "nearest"

- Need to measure distance or similarity

- d: $X \times X \rightarrow R_+$ such that
  - symmetry: d(x, x') = d(x, x')
  - definite: d(x, x') = 0 iff x = x'
  - triangle inequality: d(a, b) <= d(a,c) + d(c,b)

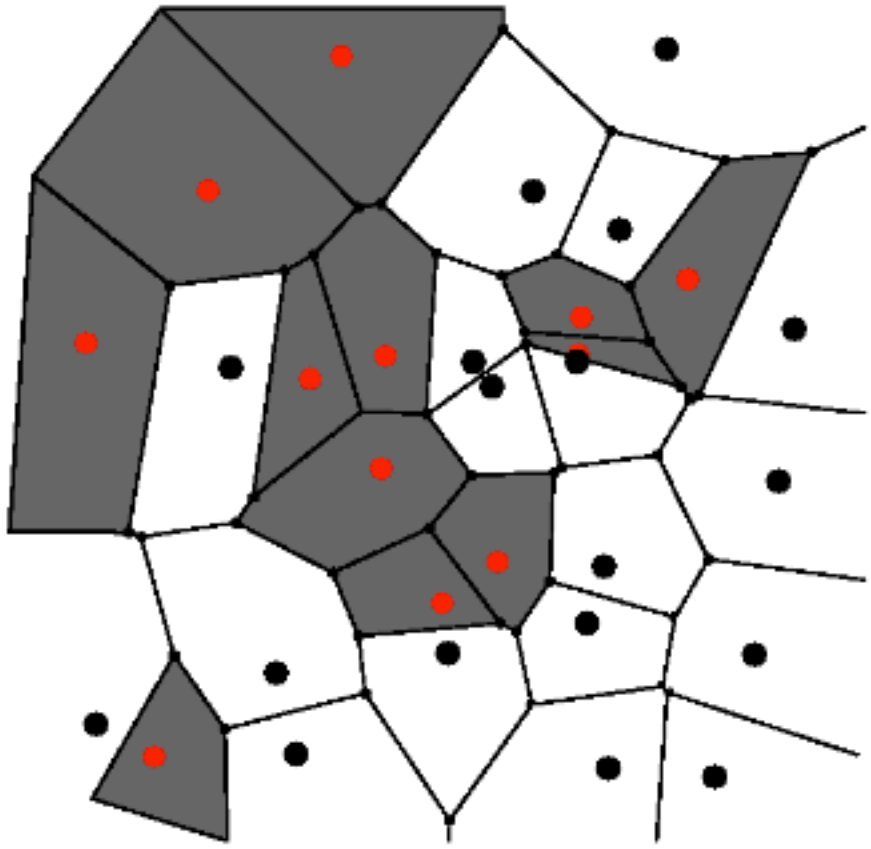
L_p distance: $d_p(x, x') = \| x - x' \|_p$
- p=2: Euclidean distance
- p=1: Manhattan distance
- p=inf: Chebyshev distance

Complexity of 1-NN

- Training: 0… but O(nd) space
- Testing: O(nd) for each query point
  - n: # of training samples
  - d: # of features
- Can we do better?

Voronoi diagram

- In 2D, can construct in O(n logn) time and O(n) space (Fortune, 1989)
- In 2D, query costs O(log n)
- Large d? $n^{O(d)}$ space with O(d log n) query time ☹
  - approximate NN



k-NN

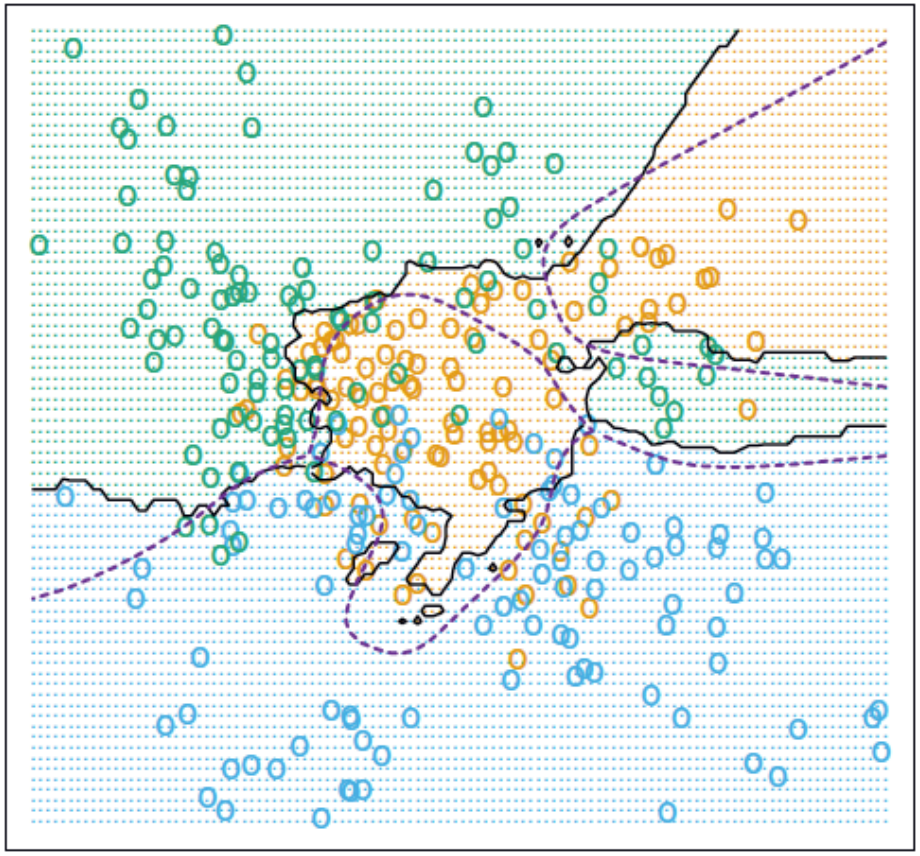
- Store training set (X, **y**)

- For query (test point) **x**'
  - find k nearest points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ in X
  - predict y' = y($\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$)
    - usually a majority vote among the labels of $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$
    - say y_1=1, y_2=1, y_3=-1, y_4=1, y5=-1 → y'=

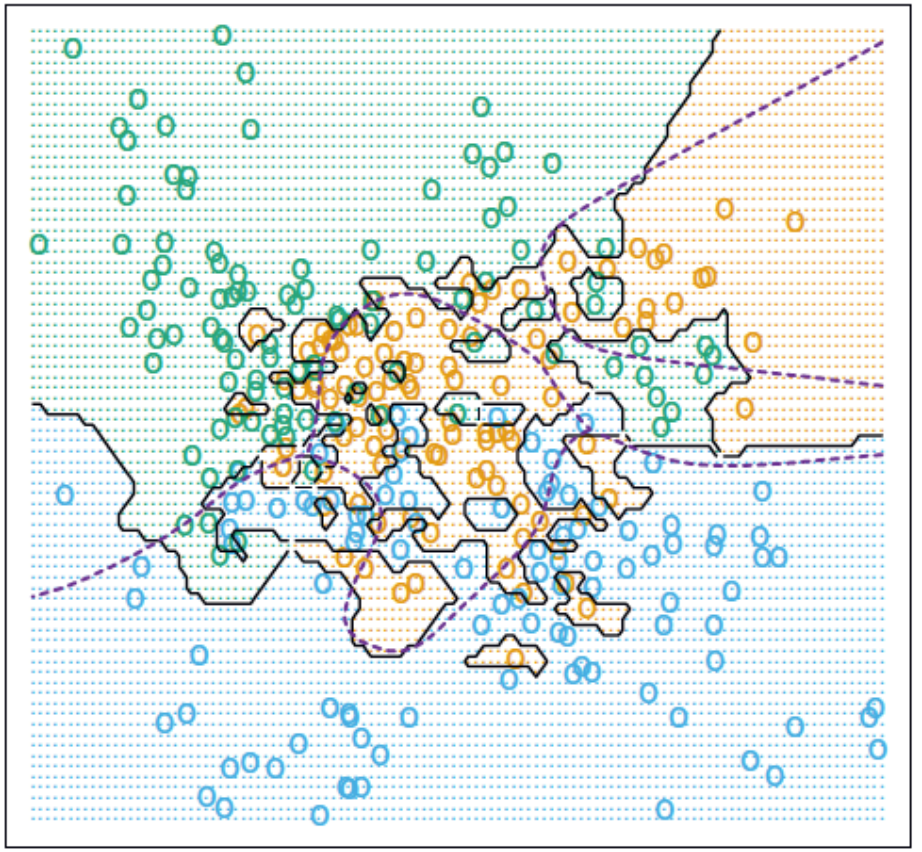
- Test complexity: O(nd)

Effect of k

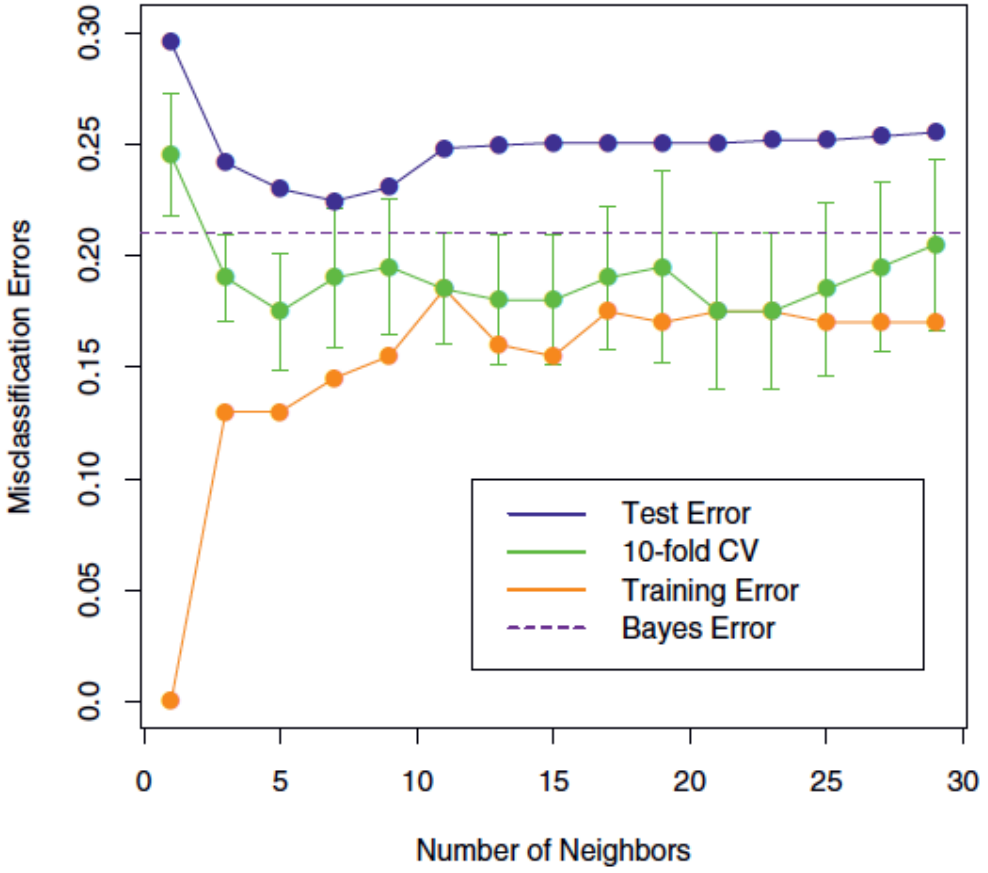
15-Nearest Neighbors

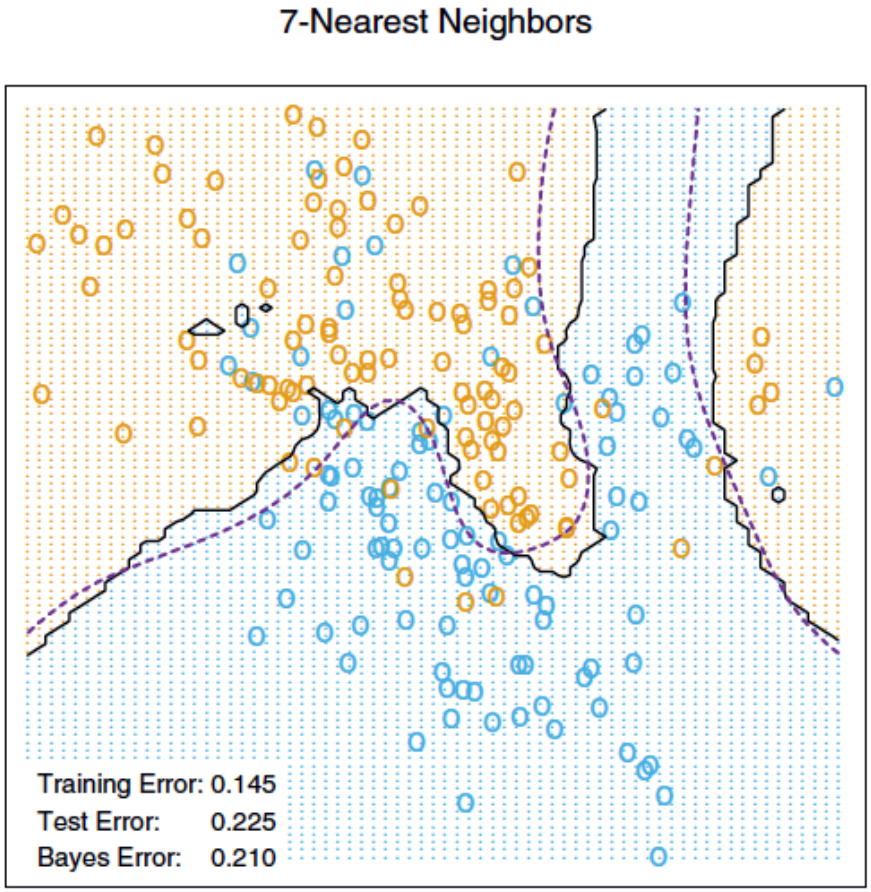


1-Nearest Neighbor



How to select k?





UNIVERSITY OF WATERLOO

Does k-NN work?

MNIST: 60k train, 10k test

linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions]	none	0.35	Ciresan et al. Neural Computation 10, 2010 and arXiv 1003.0358, 2010
SVM, Gaussian Kernel	none	1.4	
Virtual SVM, deg-9 poly, 2-pixel jittered	deskewing	0.56	DeCoste and Scholkopf, MLJ 2002
Convolutional net LeNet-4	none	1.1	LeCun et al. 1998
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23	Ciresan et al. CVPR 2012
K-nearest-neighbors, Euclidean (L2)	none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	none	2.83	Kenneth Wilder, U. Chicago
K-NN with non-linear deformation (P2DHMDM)	shiftable edges	0.52	Keysers et al. IEEE PAMI 2007

Outline

- Algorithm
- Theory
- Application

Bayes rule

- Bayes error $P^* = \min_{f:\mathcal{X}\to\{\pm 1\}} \mathbf{P}(f(X) \neq Y)$
- Bayes rule

$$\eta(X) = \mathbf{P}(Y = 1|X)$$

$f^*(X) = 1$ iff $\eta(X) \geq ½$

Multi-class

$$f^*(X) = \operatorname*{argmax}_{m=1,\ldots,c} \mathrm{P}(Y = m|X)$$

$$P^* = \mathbf{E}[1 - \max_{m=1,\ldots,c} \mathrm{P}(Y = m|X)]$$

- This is the best we can do even when we know the distribution of (X,Y)
- How big can P* be?

At most twice worse (Cover & Hart'67)

$$\lim_{n\to\infty} P(Y_1^{(n)} \neq Y) \leq 2P^* - \frac{c}{c-1}(P^*)^2$$

Asymptotically!

1-NN error

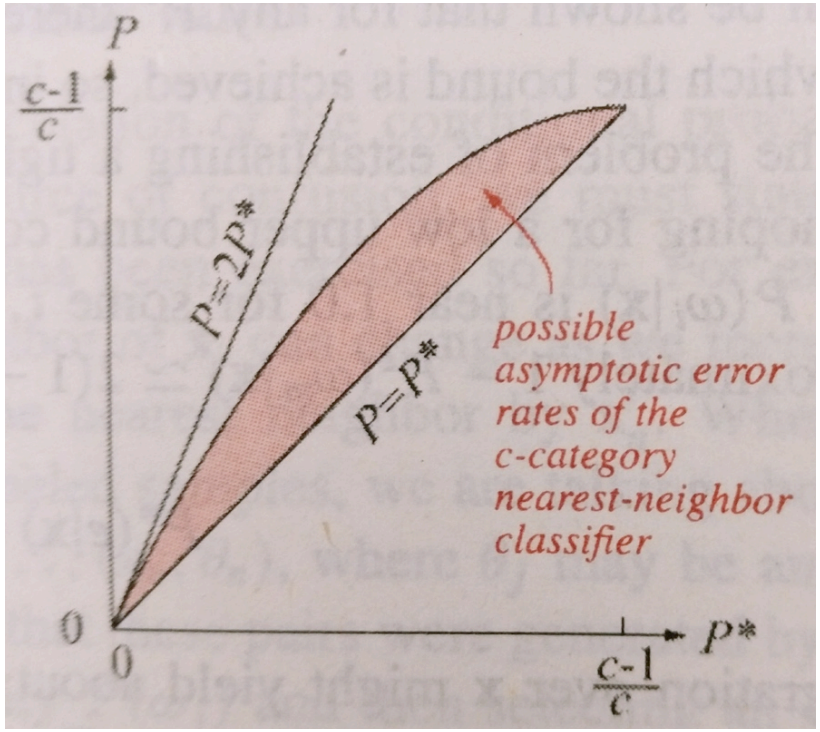
Bayes error

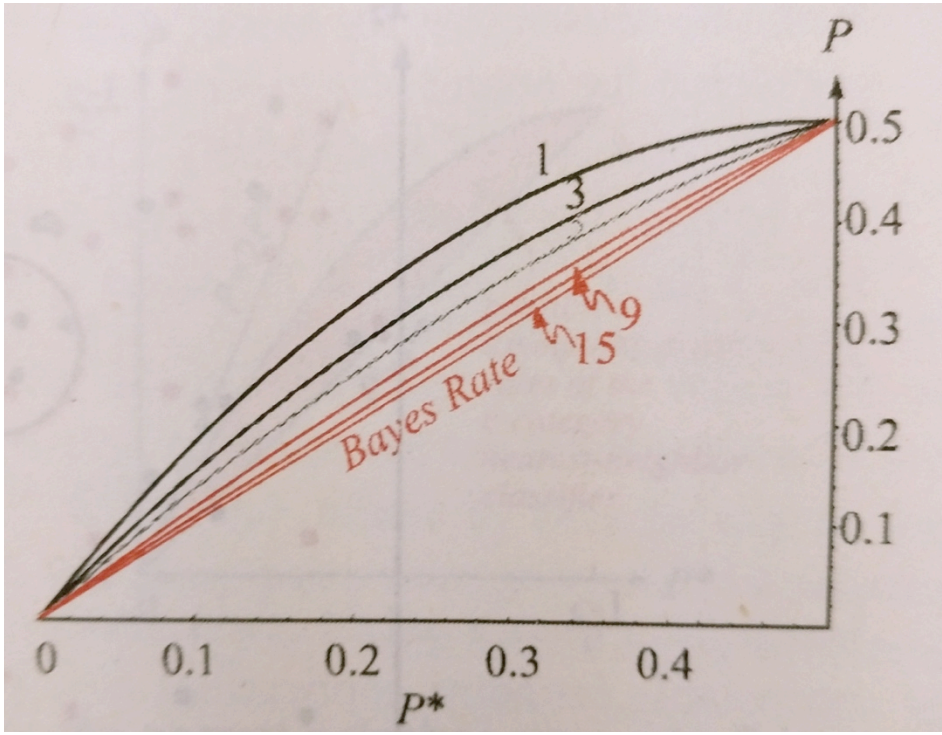
# of classes

$1 / (\max P^*)$

- If P^* close to 0, then 1-NN error close to $2P^*$
- How big does n have to be?

The picture





Both assume we have infinite amount of training data!

1-NN vs k-NN

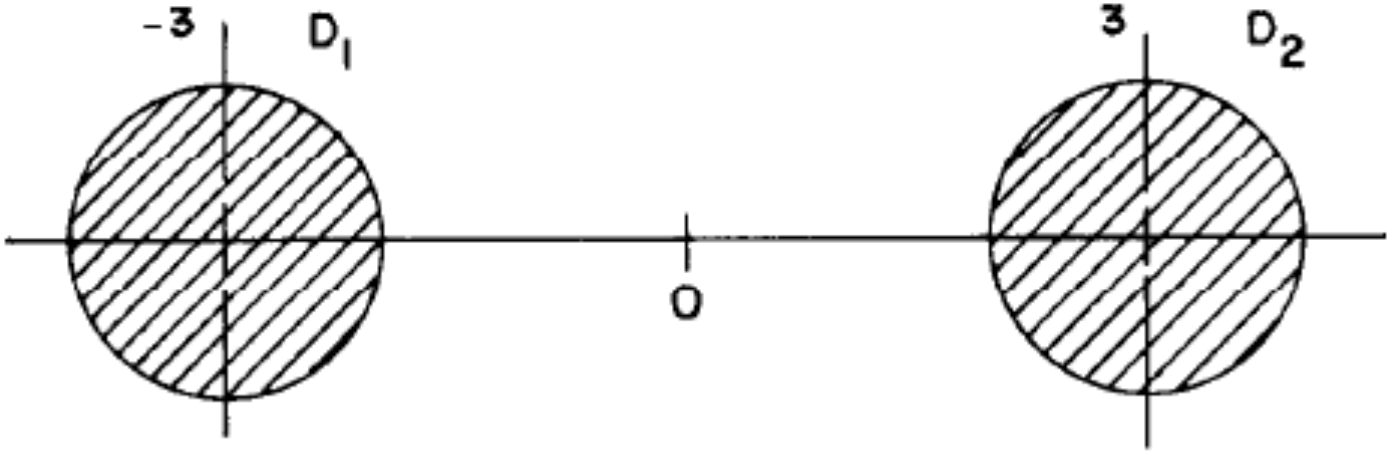


Fig. 1. Admissibility of nearest neighbor rule.

- error(1NN) = $1/2^n$
- error(kNN) for k = 2t+1: $\frac{1}{2^n}\sum_{i=0}^{t}\binom{n}{i}$

Curse of dimensionality

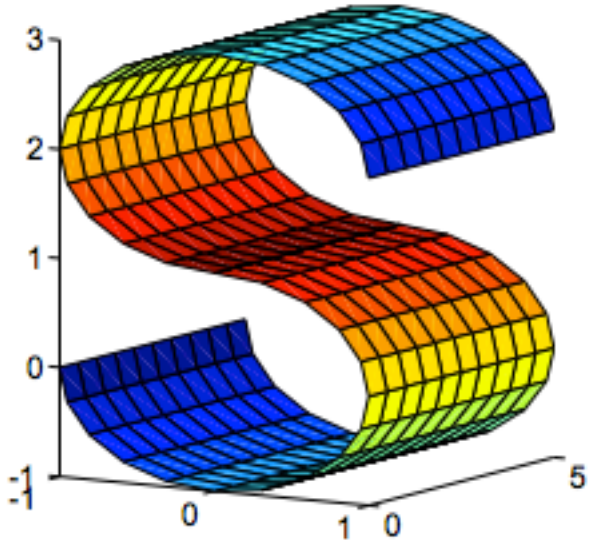
Theorem (SSBD, p224). For any $c > 1$ and **any** learning algorithm L, there exists a distribution over $[0,1]^d \times \{0,1\}$ such that the Bayes error is 0 but for sample size $n \leq (c+1)^d/2$, the error of the rule L is greater than ¼.

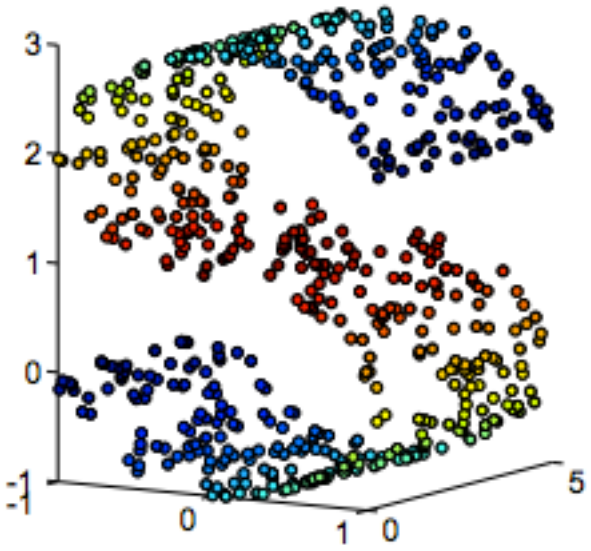
- k-NN is effective when have **many** training samples
- Dimensionality reduction may be helpful

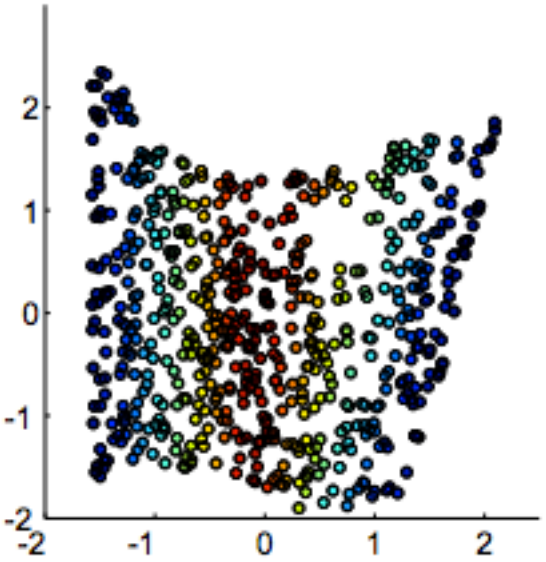
Outline

- Algorithm
- Theory
- Application

Locally linear embedding (Saul & Roweis'00)





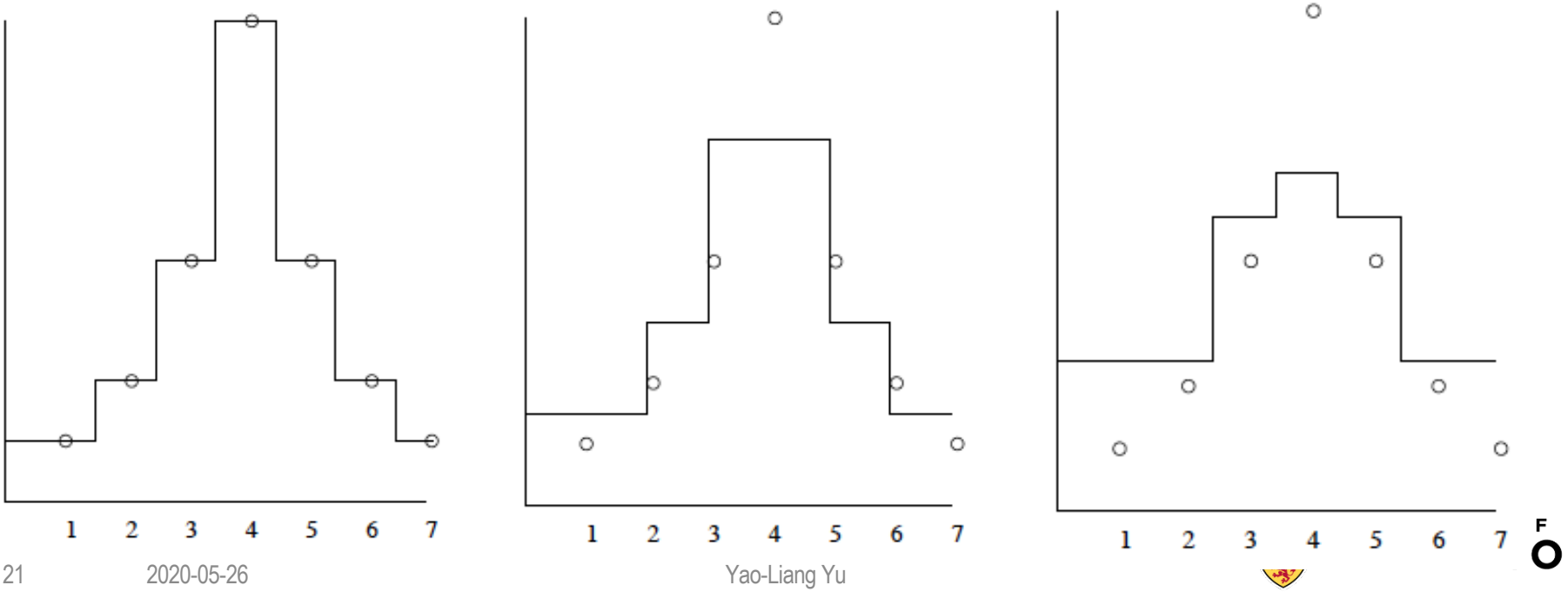


$$\min_{W1=1} \|\mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j\|_2^2$$

$$\min_{Z} \sum_i \|\mathbf{z}_i - \sum_j W_{ij}\mathbf{z}_j\|_2^2$$

k-NN for regression

- Training: store training set (X, y)
- Query x'
  - Find k-nns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ in X
  - output $y' = y(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k) = (y_1+y_2+\ldots+y_k)/k$



Questions?