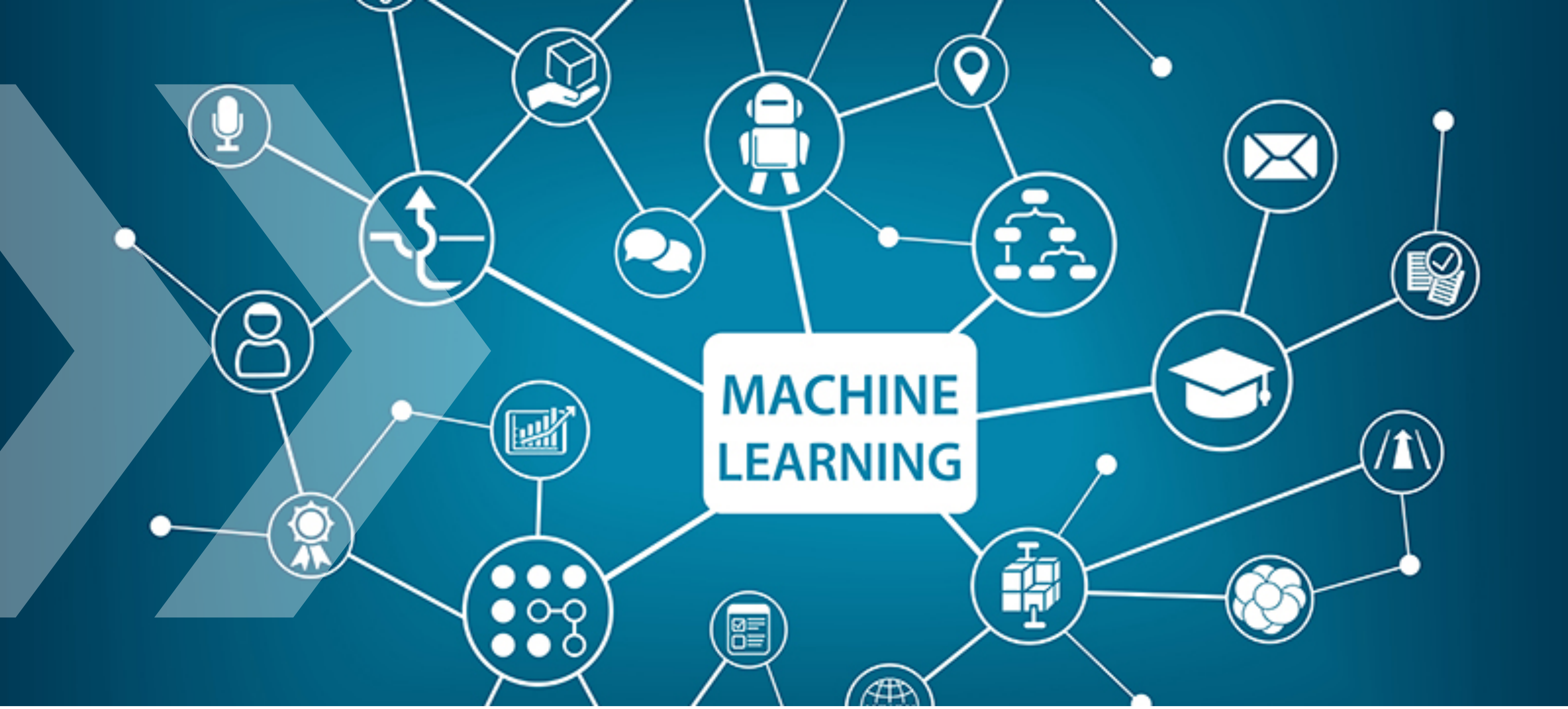


CS489/698: Intro to ML

Lecture 05: *K*-nearest neighbors

UNIVERSITY OF WATERLOO

Outline

- Announcements
- Algorithm
- Theory
- Application

Announcements

- Assignment 1 extension to Thursday?

$$\frac{1}{2}\|\underbrace{X_{:j}}_{\vec{u}} z + \underbrace{\sum_{k\neq j} X_{:k} w_k - y}_{\vec{v}}\|_2^2 + \lambda |z|^2$$

$$\frac{1}{2}\|\vec{u}\|^2 z^2 + \vec{u}^T\vec{v} \cdot z + \frac{1}{2}\|\vec{v}\|_2^2 + \lambda z^2$$

$$\frac{1}{2} z^2 + \frac{\vec{u}^T\vec{v}}{\|\vec{u}\|_2^2} z + \frac{1}{2\|\vec{u}\|_2^2}\|\vec{v}\|_2^2 + \frac{\lambda}{\|\vec{u}\|_2^2} z^2$$

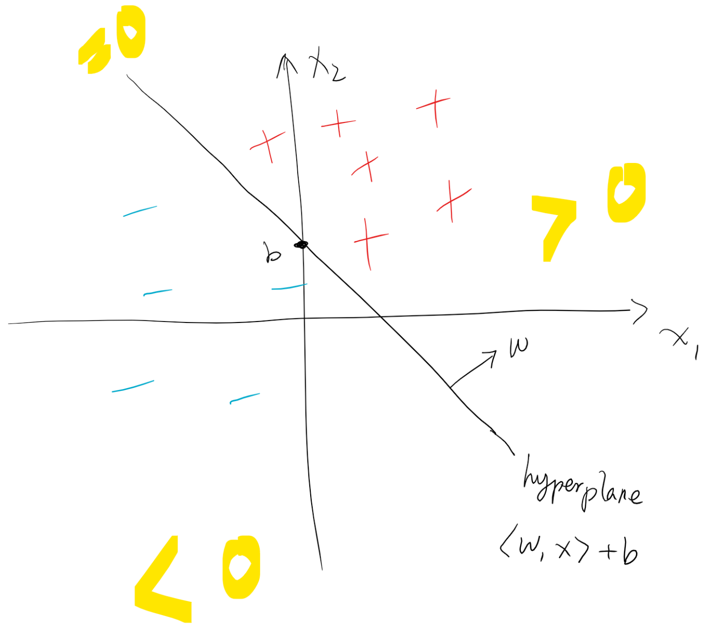
$$\underline{\frac{1}{2}\left(z + \frac{\vec{u}^T\vec{v}}{\|\vec{u}\|_2^2}\right)^2} + \frac{\lambda}{\|\vec{u}\|_2^2} z^2 + \frac{\|\vec{v}\|_2^2}{2\|\vec{u}\|_2^2} - \frac{(u^T v)^2}{2\|u\|_2^2}$$

Outline

- Announcements
- 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 **w**
- Non-parametric: no specific form (or inf-dim **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 x X → 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)

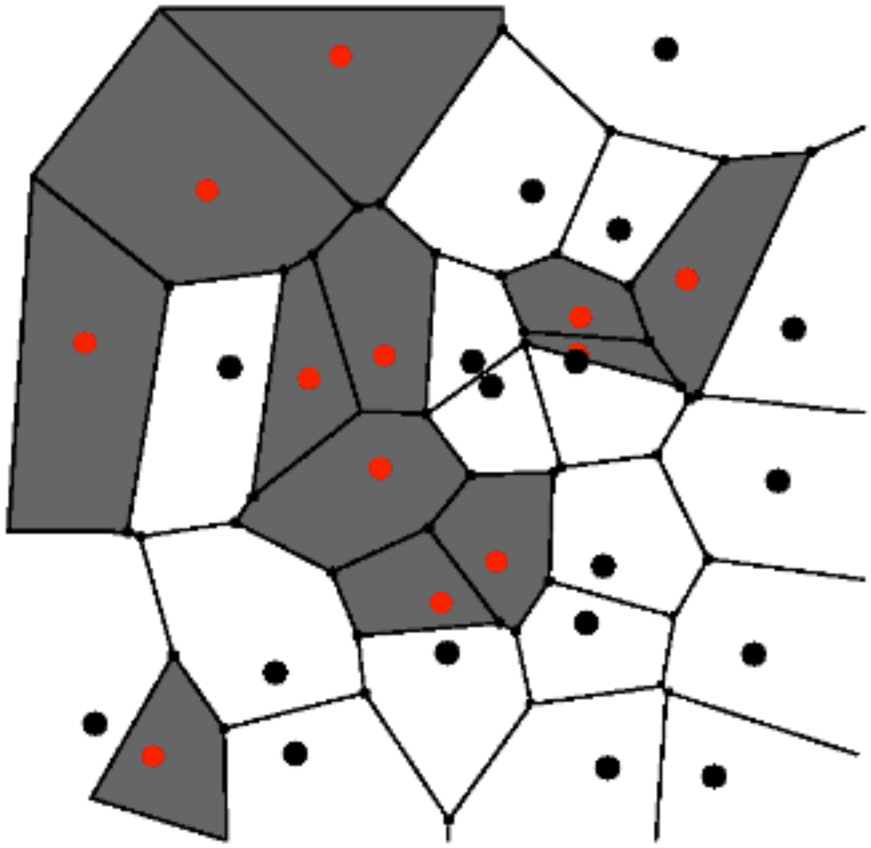
L_p distance: $d_p(x, x') = || x - x' ||_p$
- p=2: Euclidean distance
- p=1: Manhattan distance
- p=inf: Chebyshev distance

Complexity of 1-NN

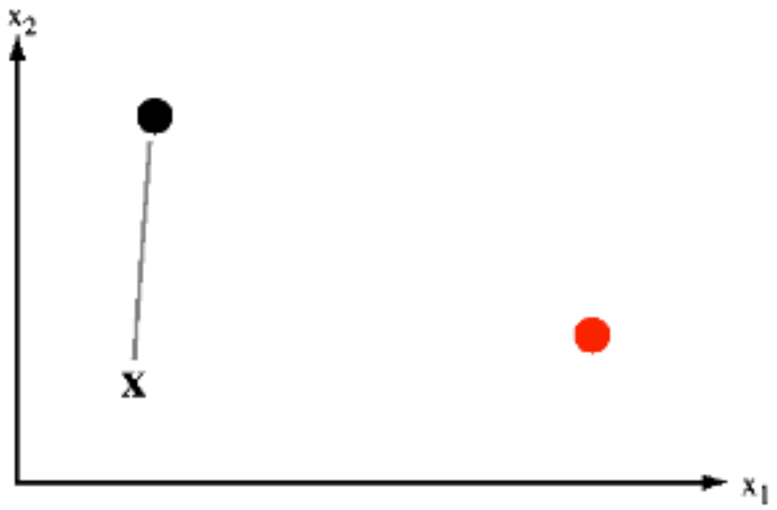
- Training: 0… but O(nd) space
- Testing: O(nd) for each query point
  - n: # of training samples
  - d: # of features
- Can we do better?

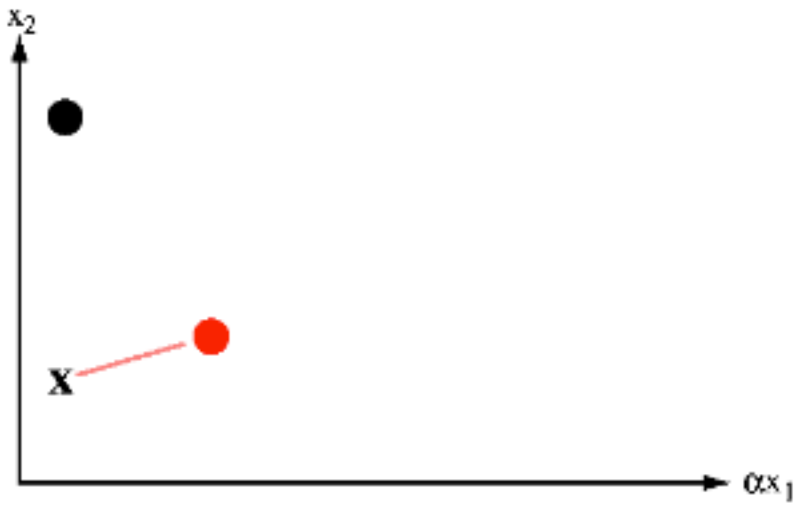
Voronoi diagram

- 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



Normalization





- Usually, for each feature, subtract the mean and divide by standard deviation

- Or, equivalently, use a different distance metric

Learning the metric

- Mahalanobias distance

$$d_M(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^\top M (\mathbf{x} - \mathbf{x}')} \quad M \in \mathbb{S}_+^d$$

- Or equivalently let $M = LL^\top \qquad L \in \mathbb{R}^{d \times h}$
  - First perform linear transformation $\mathbf{x} \mapsto L^\top \mathbf{x}$
  - Then use the usual Euclidean distance

$\min_{M \in \mathbb{S}_+^d} \; f(M)$ such that $d_M(\mathbf{x}, \mathbf{x}')$ is small iff $y = y'$

k-NN

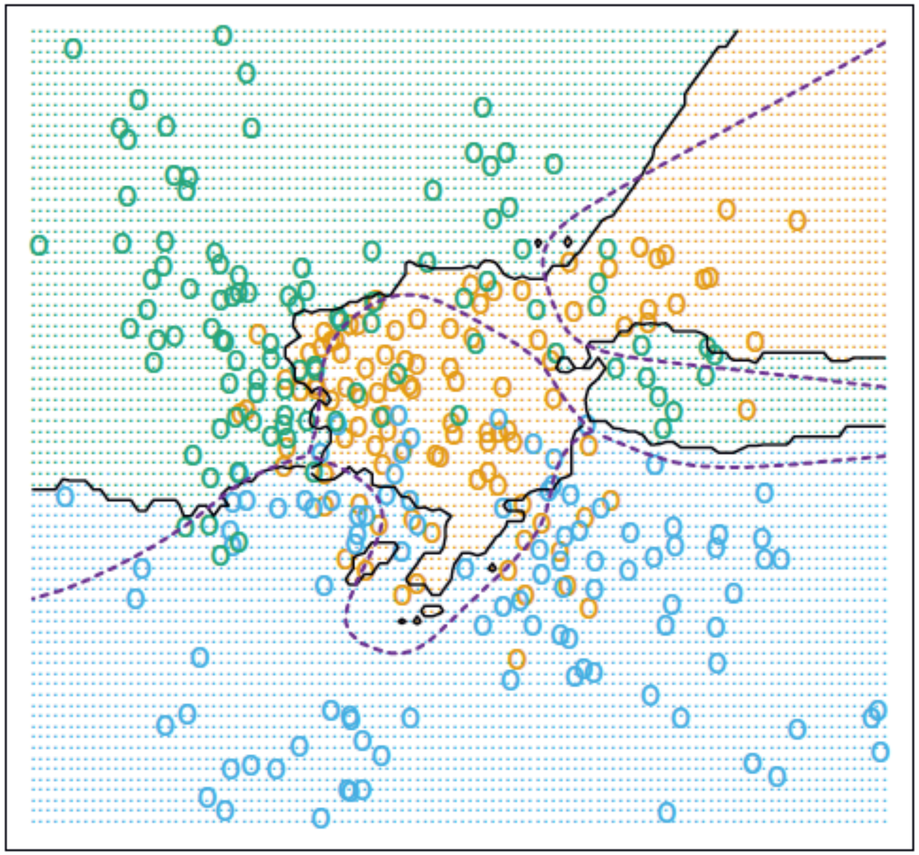
- Store training set (X, **y**)

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

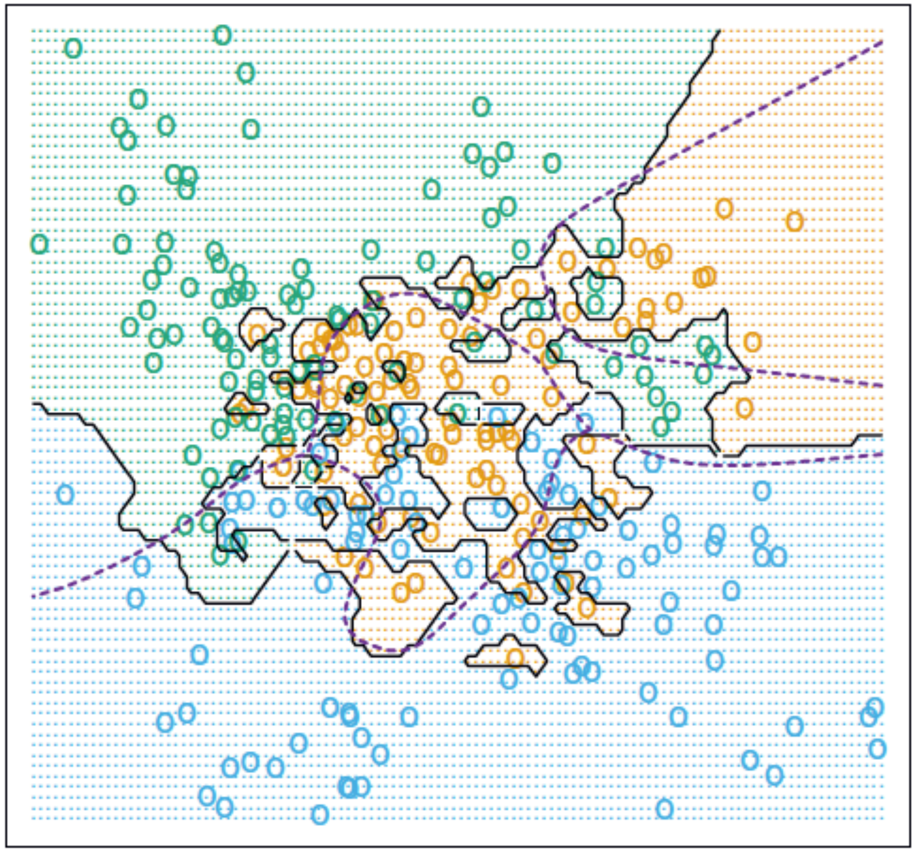
- Test complexity: O(nd)

Effect of k

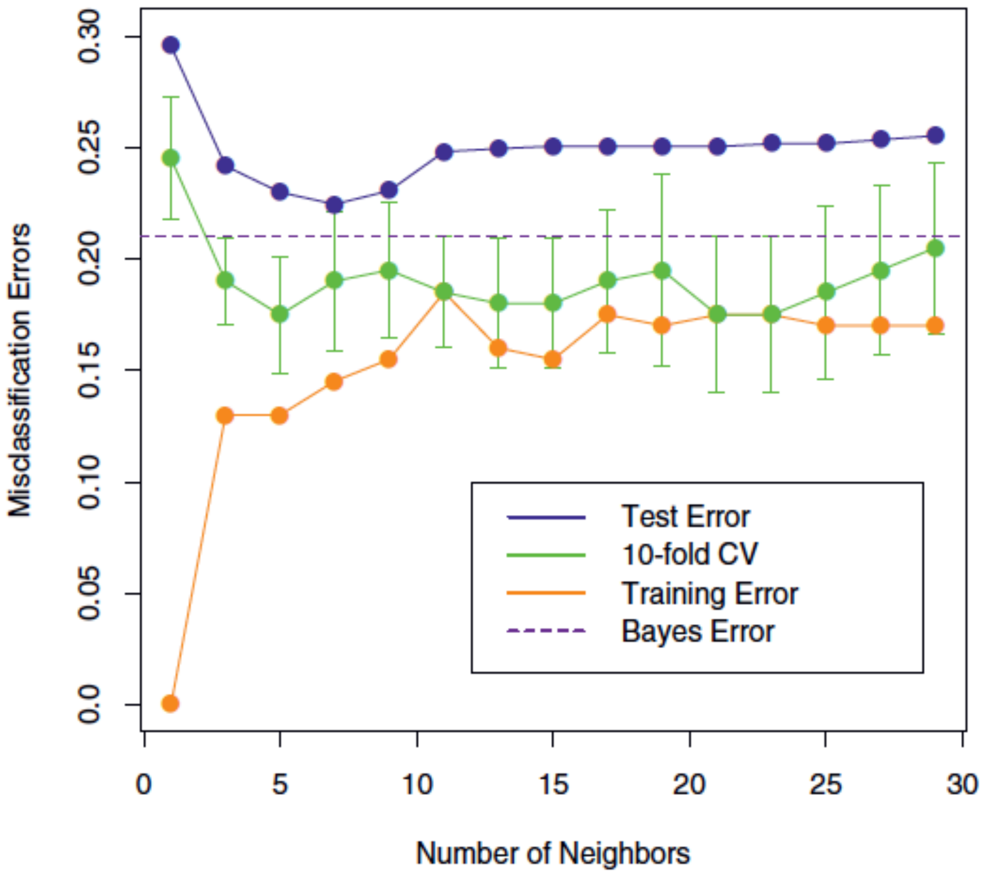
15-Nearest Neighbors

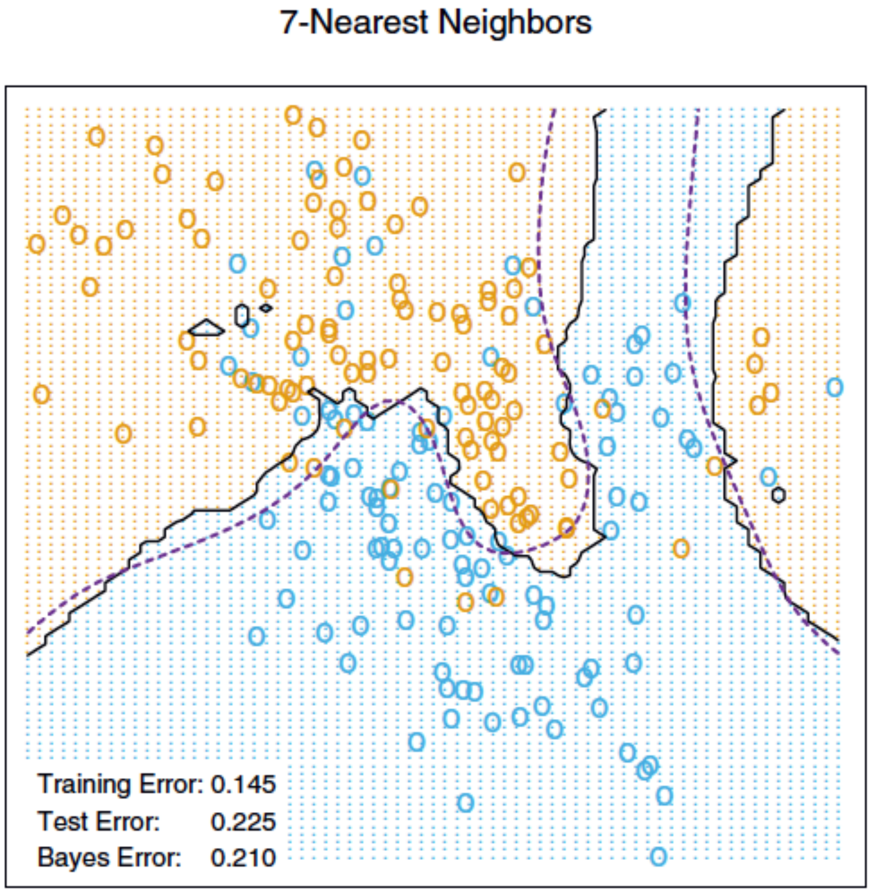


1-Nearest Neighbor



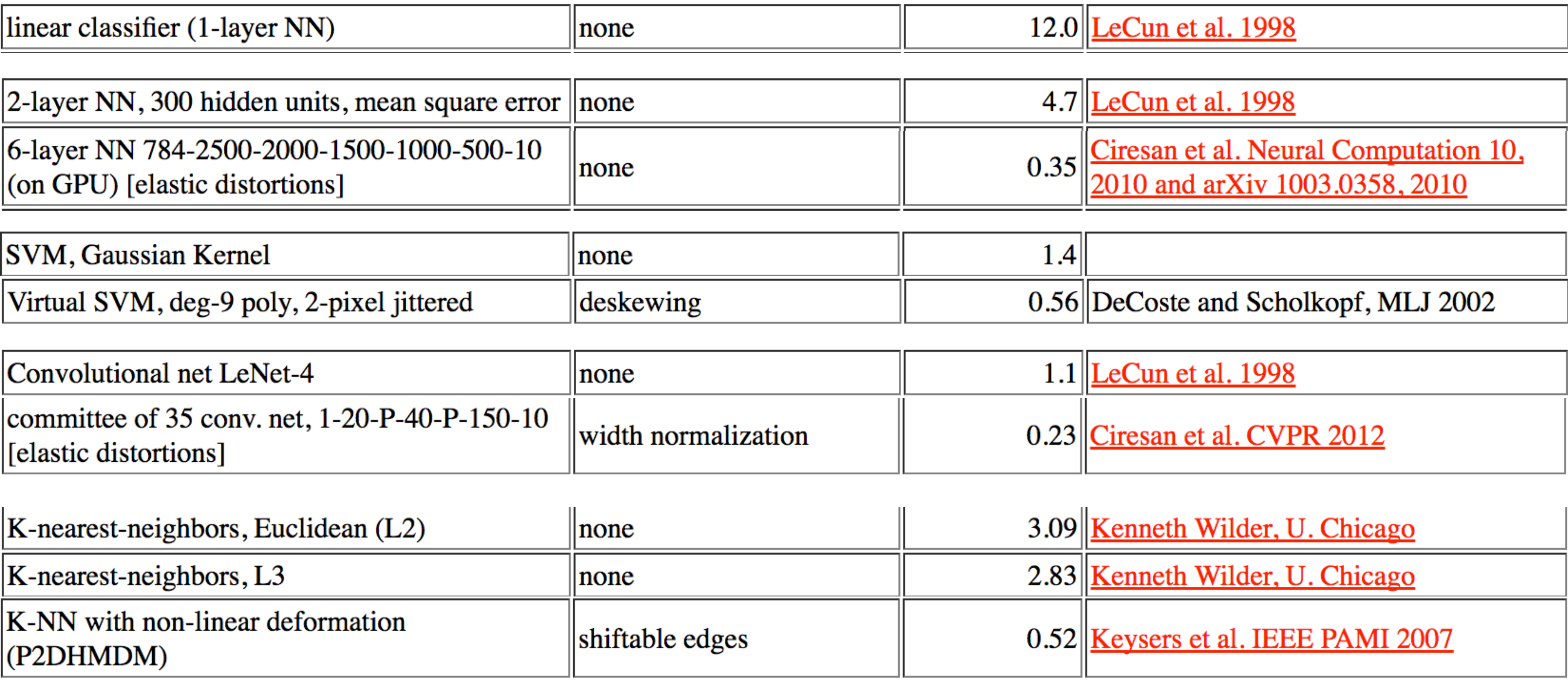
How to select k?





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

UNIVERSITY OF WATERLOO

Outline

- Announcements
- Algorithm
- Theory
- Application

Bayes rule

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

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

$$\begin{aligned}
\mathbf{P}(f(X) \neq Y) &= 1 - \mathbf{P}(f(X) = 1, Y = 1) - \mathbf{P}(f(X) = -1, Y = -1) \\
&= 1 - \mathbf{E}[\mathbf{P}(f(X) = 1, Y = 1|X)] - \mathbf{E}[\mathbf{P}(f(X) = -1, Y = -1|X)] \\
&= 1 - \mathbf{E}[1_{f(X)=1}\mathbf{P}(Y = 1|X)] - \mathbf{E}[1_{f(X)=-1}\mathbf{P}(Y = -1|X)] \\
&= 1 - \mathbf{E}[1_{f(X)=1}\eta(X) + 1_{f(X)=-1}(1 - \eta(X))] \\
&= \mathbf{E}[\eta(X) + 1_{f(X)=1}(1 - 2\eta(X))]
\end{aligned}$$

$$\boxed{f^*(X) = 1 \text{ iff } \eta(X) \geq ½}$$

$$1_{f(X)=1} + 1_{f(X)=-1} = 1$$

Derivation

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)

$$\boxed{\lim_{n\to\inf} P(Y_1^{(n)} \neq Y) \;\leq\; 2P^* - c/(c-1)\,(P^*)^2}$$

Asymptotically!

1-NN error

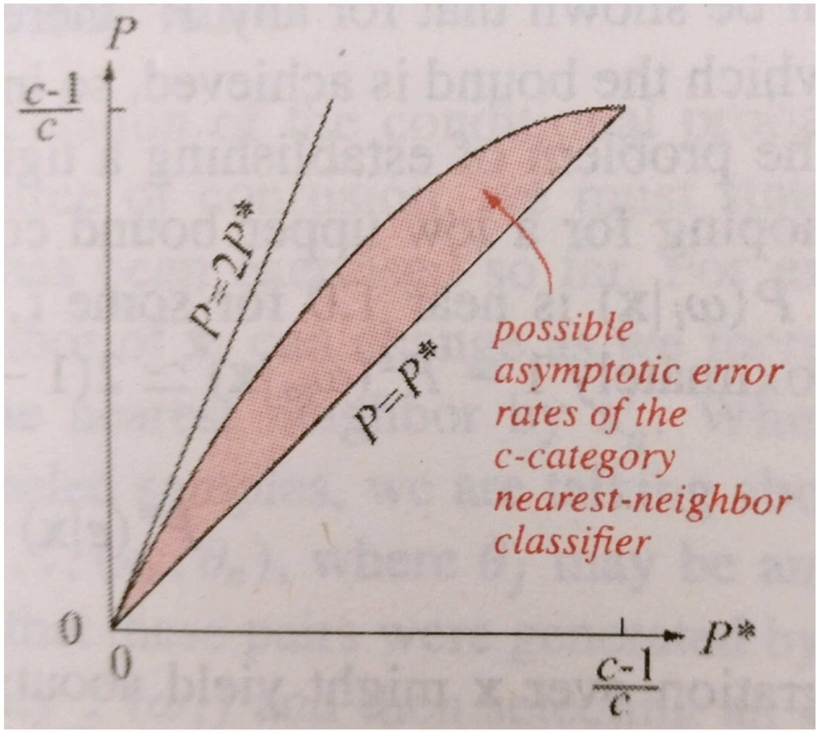
Bayes error

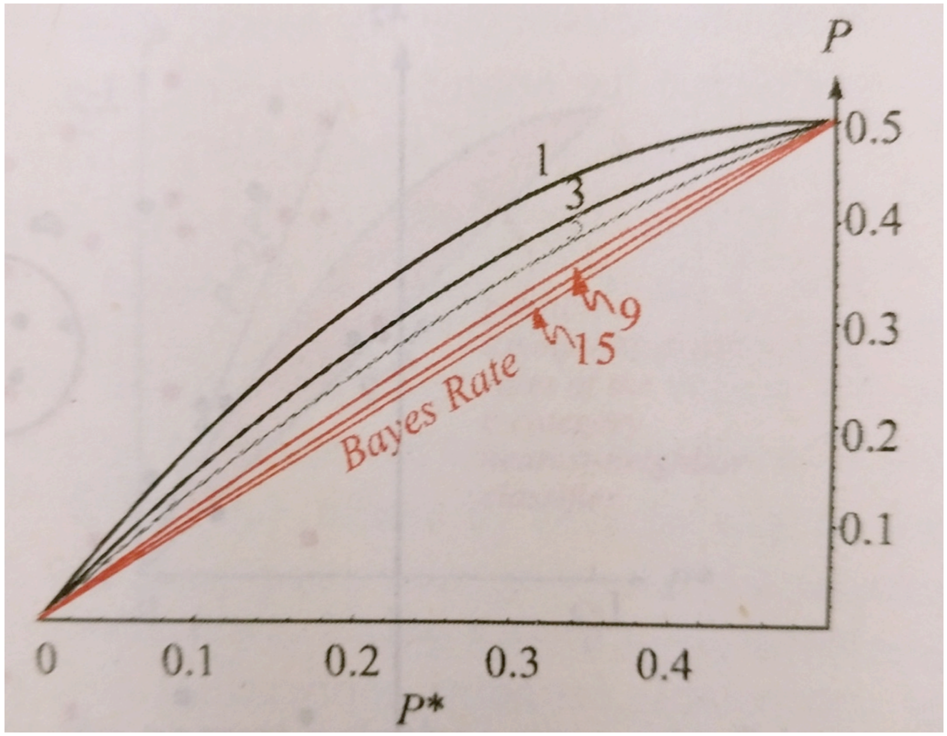
# of classes

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

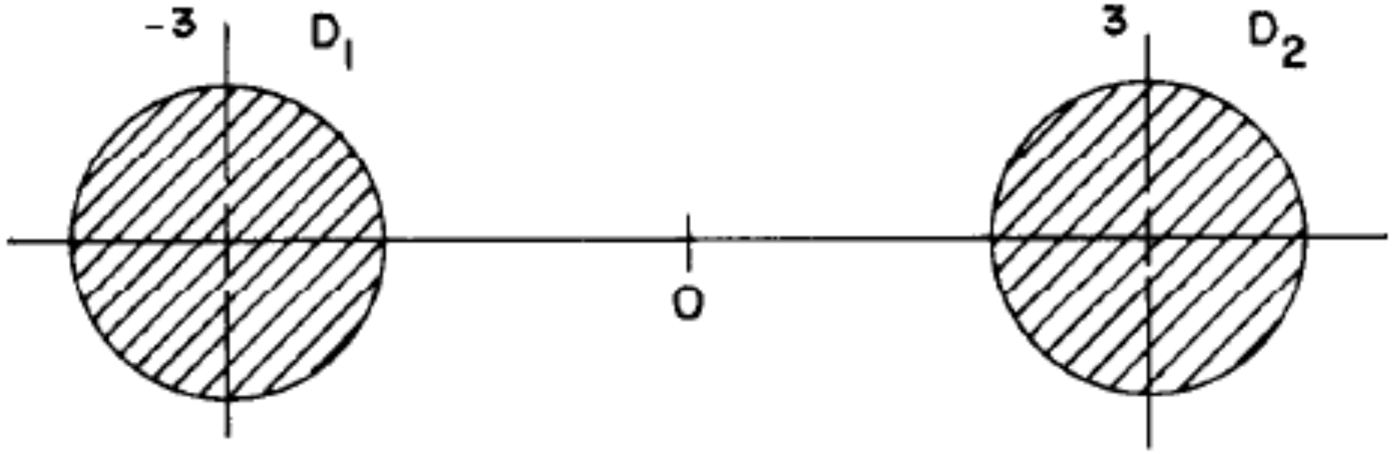


Fig. 1. Admissibility of nearest neighbor rule.

- 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.

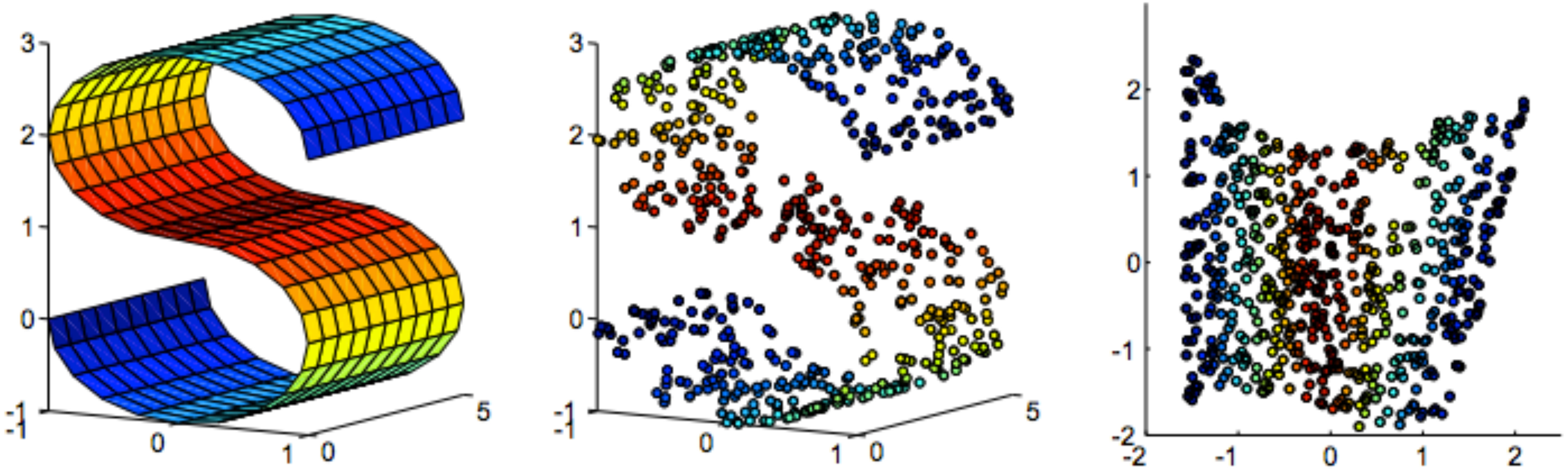
- k-NN is effective when have many training samples

- Dimensionality reduction may be helpful

Outline

- Announcements
- Algorithm
- Theory
- Application

Locally linear embedding (Saul & Roweis'00)

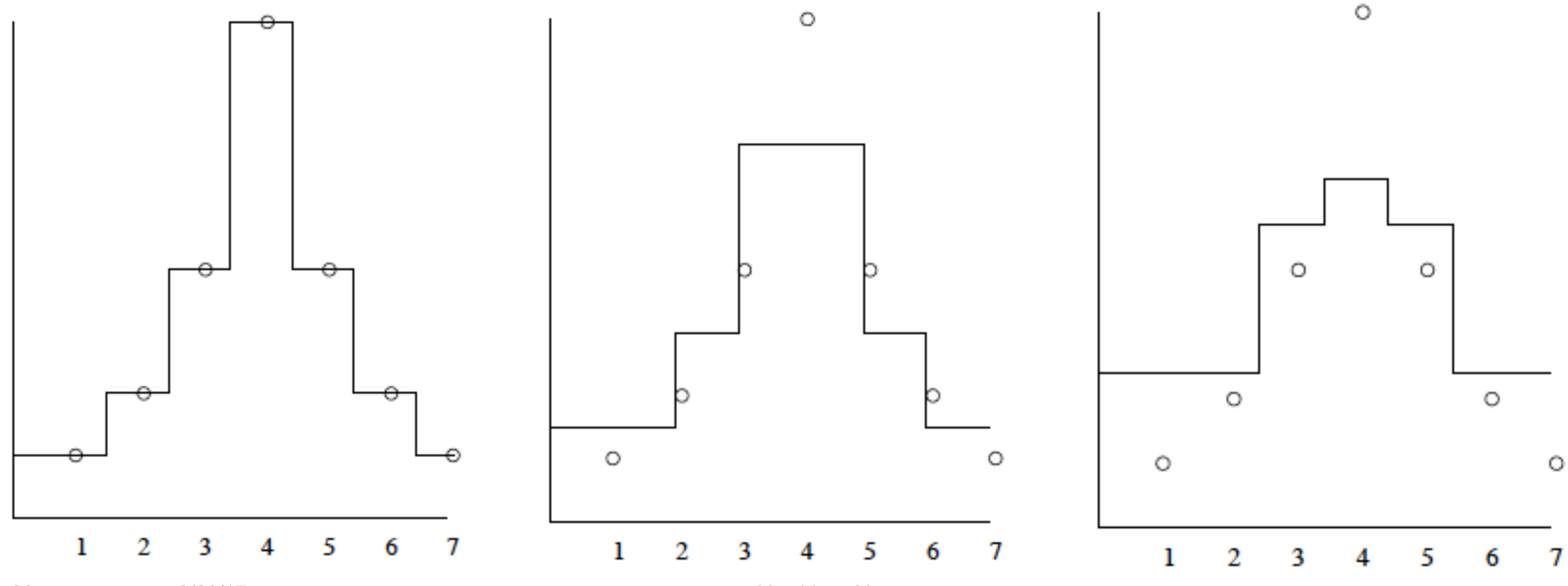


$$\min_{W\mathbf{1}=\mathbf{1}} \;\|\mathbf{x}_i - \sum_j W_{ij}\mathbf{x}_j\|_2^2$$

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

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?

Why 1-NN works

- Let X_1 be the NN for query X
- Assuming $P(Y_1=j \mid X_1) - P(Y=j \mid X) \rightarrow 0$ as $n \rightarrow \inf$
- Thus $P(Y_1 \neq Y \mid X) = \Sigma_{i \neq j} P(Y=i \mid X)\, E[\, P(Y_1=j|X_1) \mid X\,]$

$$\rightarrow \Sigma_{i \neq j} P(Y=i \mid X)\, P(Y=j|X)$$
$$= 1 - \Sigma_i P^2(Y=i \mid X)$$

error of 1-NN

- Two conditions: $\Sigma_i P(Y=i \mid X) = 1$, $\max_i P(Y=i \mid X) = 1\text{-}P^*$

Are all neighbors equal?

$$y' = \frac{y_1 + y_2 + \cdots + y_k}{k} \iff y' = \operatorname*{argmin}_y \sum_{j=1}^n 1_{j \in \mathrm{kNN}(x')} (y - y_j)^2$$

$$\xrightarrow{} \text{Parzen Window}$$

- More generally, can weigh the neighbors

$$y' = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i} \iff y' = \operatorname*{argmin}_y \sum_{i=1}^n w_i (y - y_i)^2$$

For instance $\quad w_i = \exp(-d(x', x_i)/\sigma)$

UNIVERSITY OF WATERLOO

Density estimation

- Given iid samples $X_1, \ldots, X_n$, estimate density function $X \sim p(x)$

- Kernel: function K: R → R with integral 1

- Kernel density estimation

$$\hat{p}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{\underset{\nearrow \text{bandwidth}}{h}}\right)$$

UNIVERSITY OF WATERLOO

Nonparametric regression

- Recall the regression function

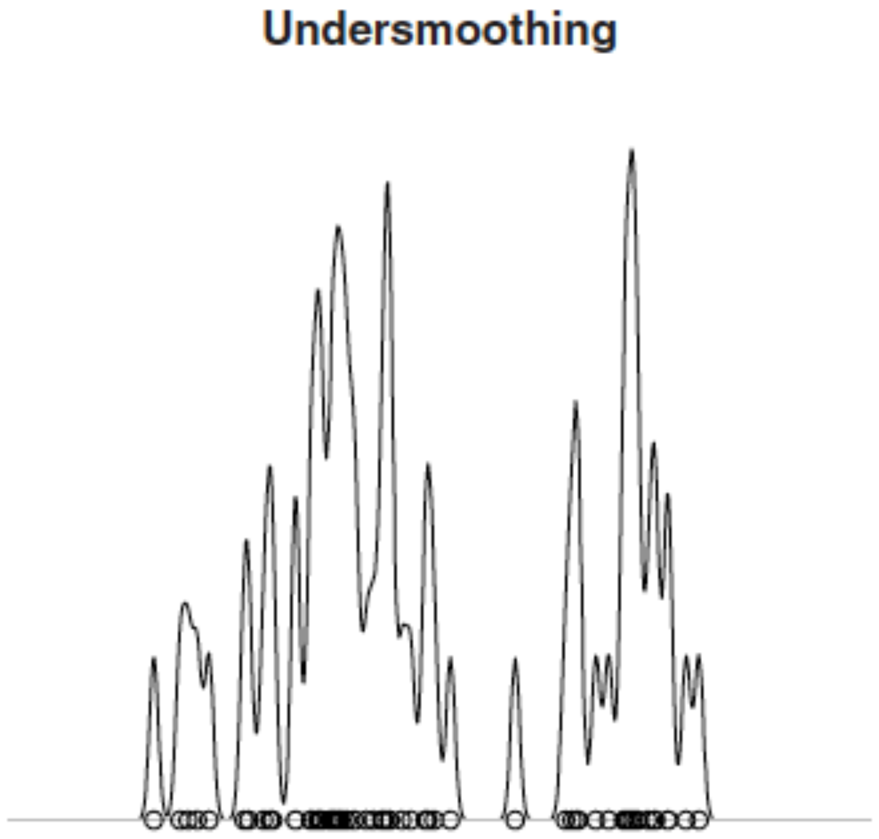
$$m(x) = \mathbf{E}(Y|X) = \int y \frac{p(x,y)}{p(x)} \mathrm{d}y$$

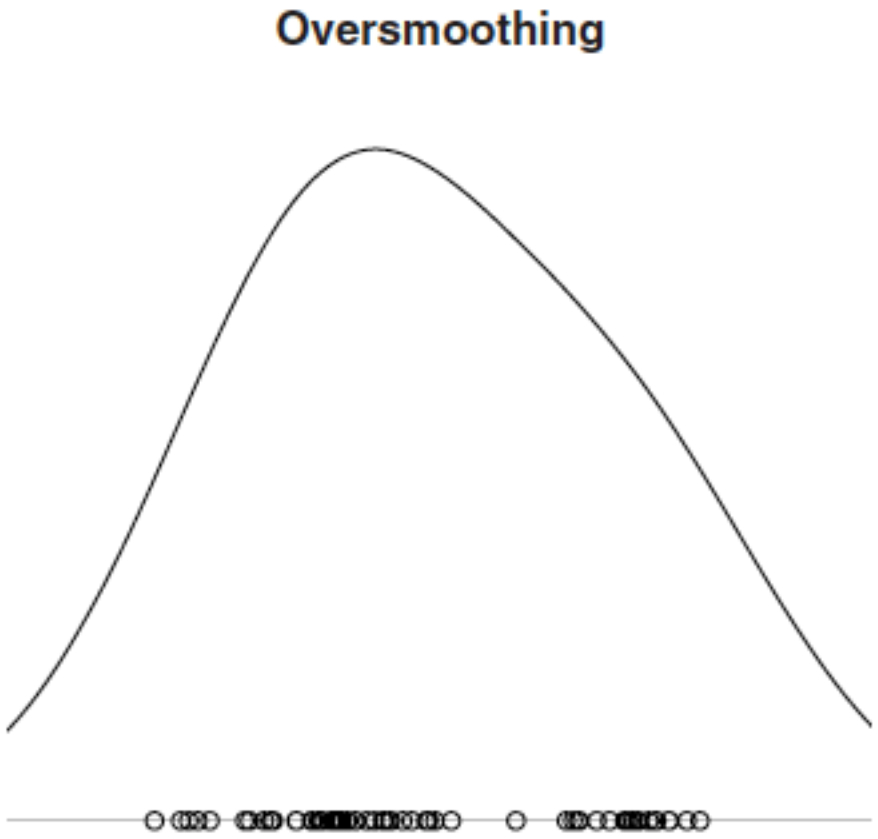
- Plugin estimator

$$\hat{p}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)$$

$$\hat{p}(x,y) = \frac{1}{nh^2} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) K\left(\frac{Y_i - y}{h}\right)$$

Bandwidth effect


Undersmoothing


Oversmoothing