CSE 480/680

1/13/26

Lecture 3

K-NN for regression
Real-valued $f(x)$ for classification
Train / Test error
Variance (+ Bias-Variance trade-off)

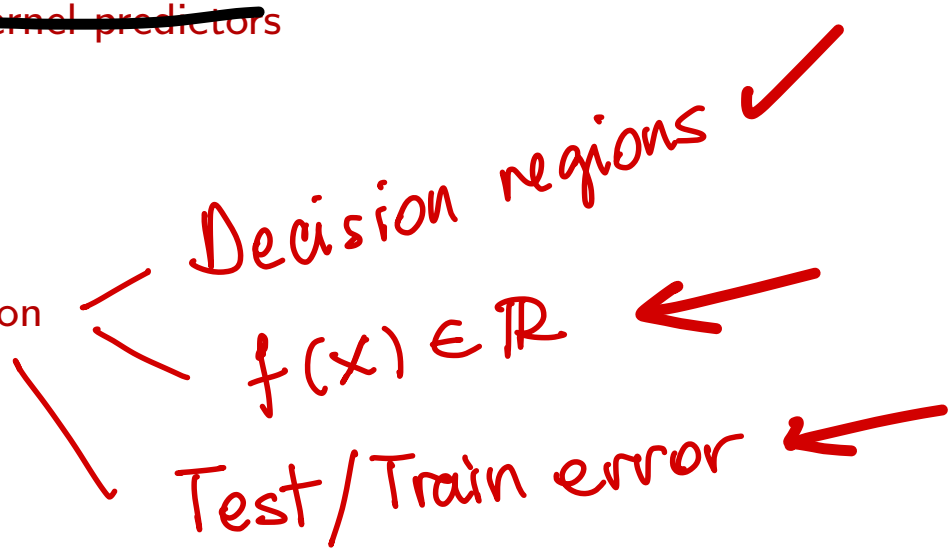
LI-1 posted
LII Linear posted

Refresher Math

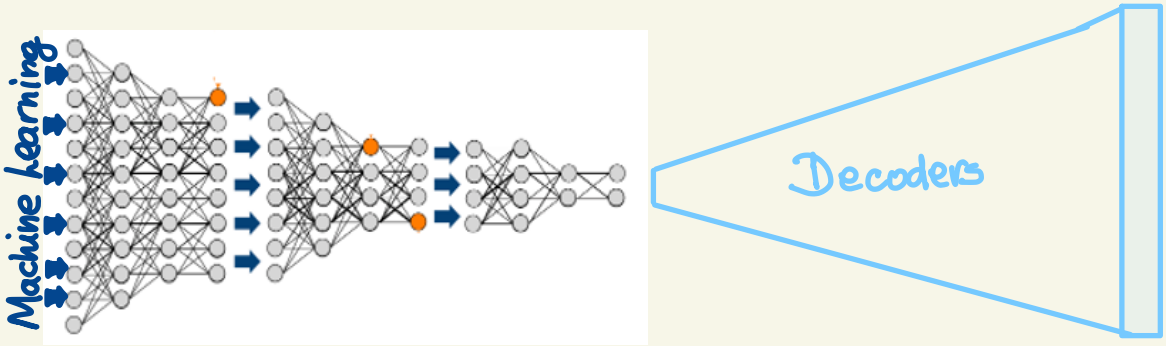
Prediction problems by the type of output

The Nearest-Neighbor and ~~kernel predictors~~

Some concepts in Classification



Reading HTF Ch.: 2.3.2 Nearest neighbor, 6.1–3. Kernel regression, 6.6.2 kernel classifiers,, Murphy Ch.: , Bach Ch.:


Machine learning
Decoders

Predictors

- K-Nearest-Neighbor

Algorithms

Concepts

- Decision Region, Dec. Boundary

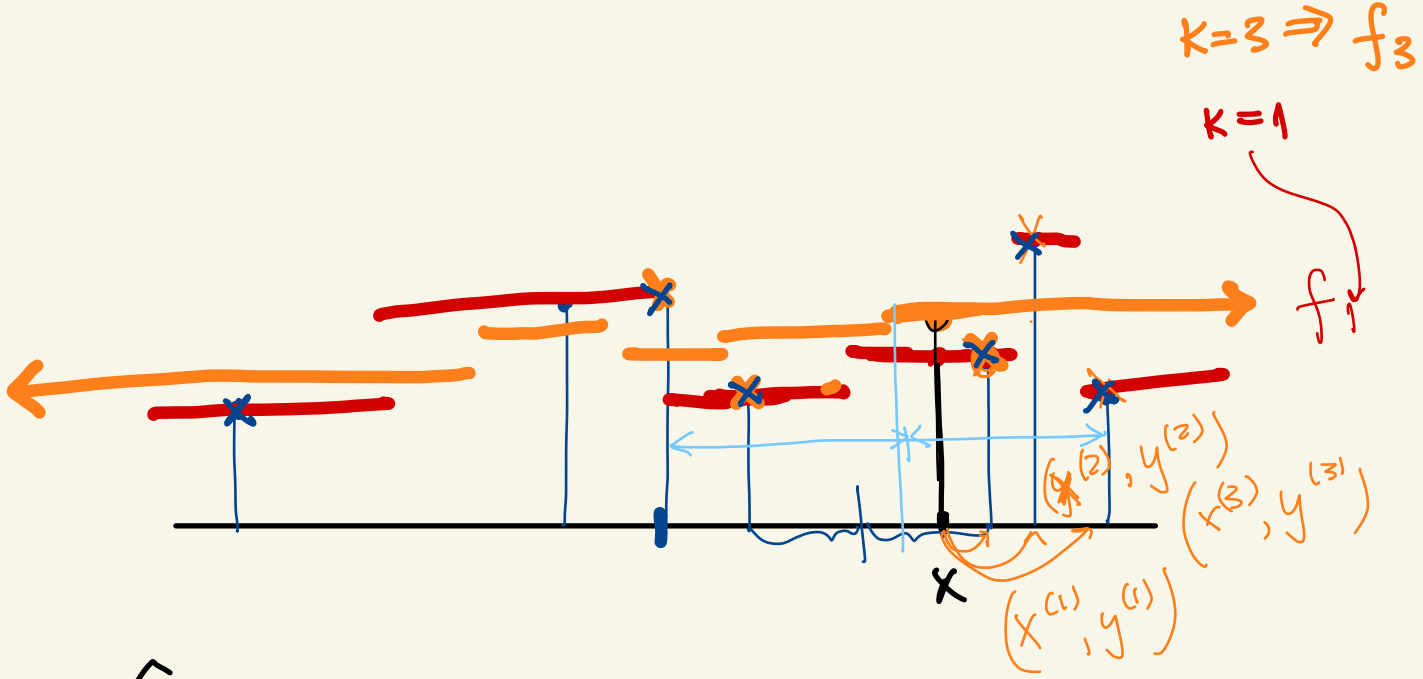
The Nearest-Neighbor predictor

- **1-Nearest Neighbor** The label of a point x is assigned as follows:
  1. find the example x^i that is nearest to x in $\mathcal{D}$ (in Euclidean distance)
  2. assign x the label y^i, i.e.

$$\hat{y}(x) = y^i$$

- **K-Nearest Neighbor** (with $K = 3, 5$ or larger)
  1. find the K nearest neighbors of x in $\mathcal{D}$: $x^{i_1, \cdots i_K}$
  2.
     - for classification $f(x)$ = the most frequent label among the K neighbors (well suited for multiclass)
     - for regression $f(x) = \frac{1}{K} \sum_{i \text{ neighbor of } x} y^i$ = mean of neighbors' labels

- No parameters to estimate!
- No training!
- But all data must be stored (also called memory-based learning)

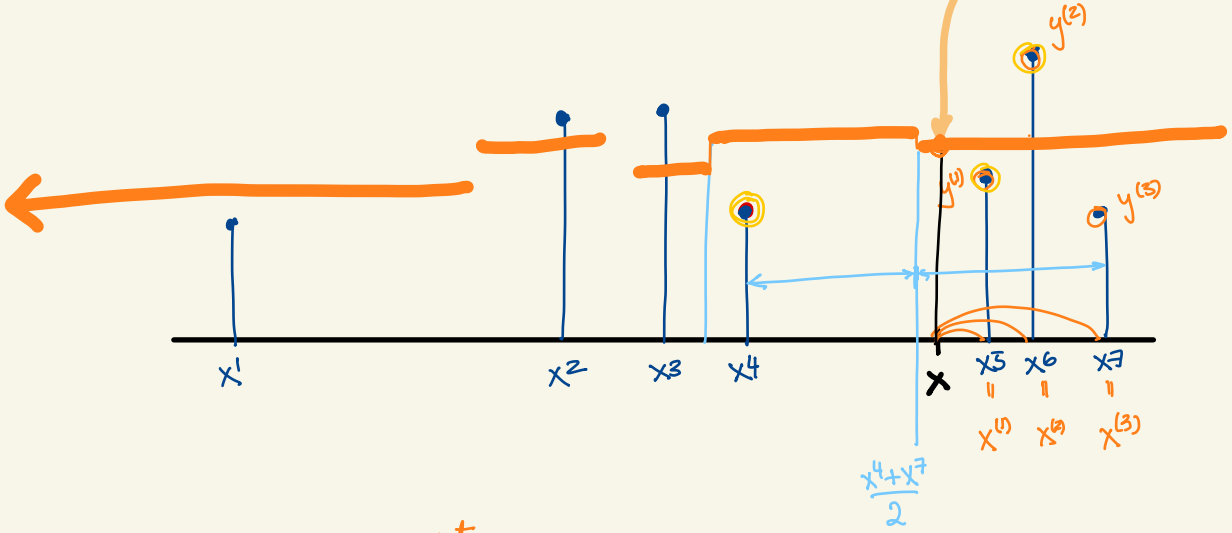


$K=1$: $\hat{y}(x) = ?$

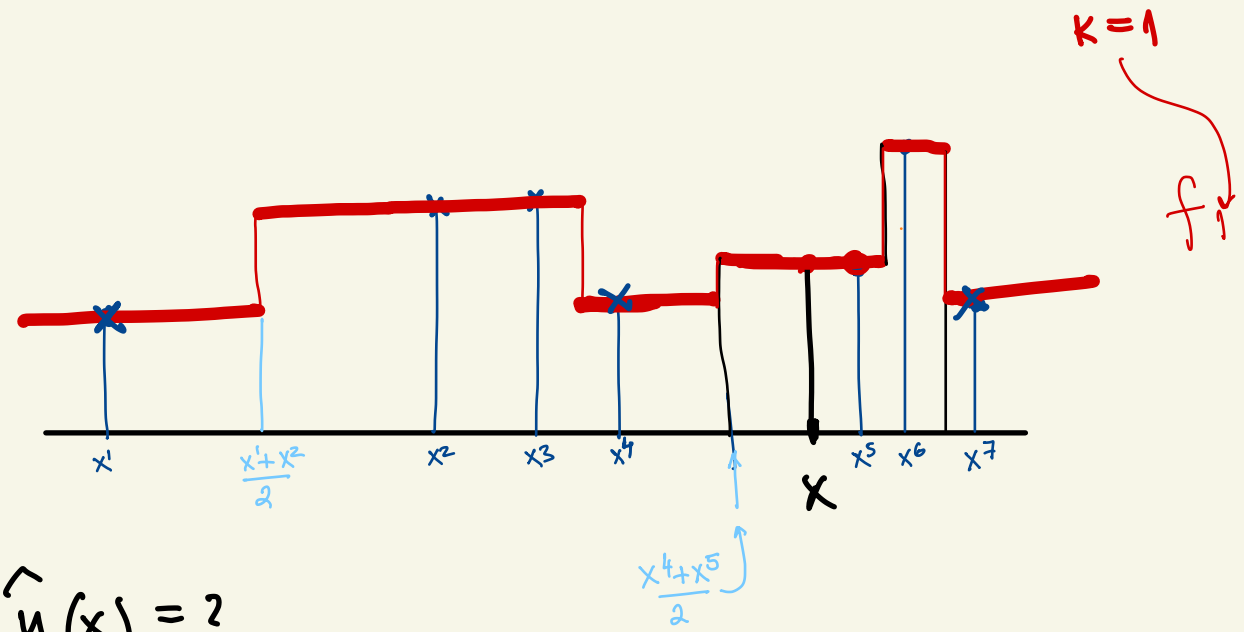
$K=3$ $\hat{y}(x)$

(R) "the k-th"

$f_3(x) = \frac{y^{(1)} + y^{(2)} + y^{(3)}}{3}$ $k=3 \Rightarrow f_3$



$x^{(k)}$ = the k-th nearest neighbor of x



$K=1: \hat{y}(x) = ?$

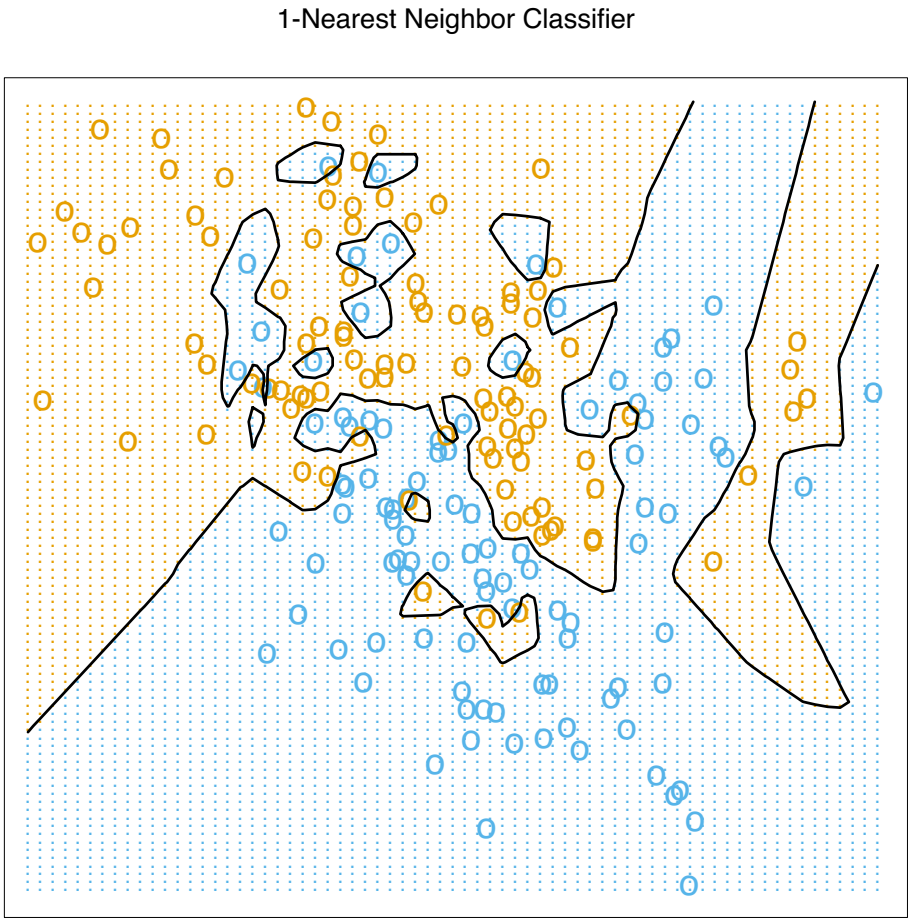


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

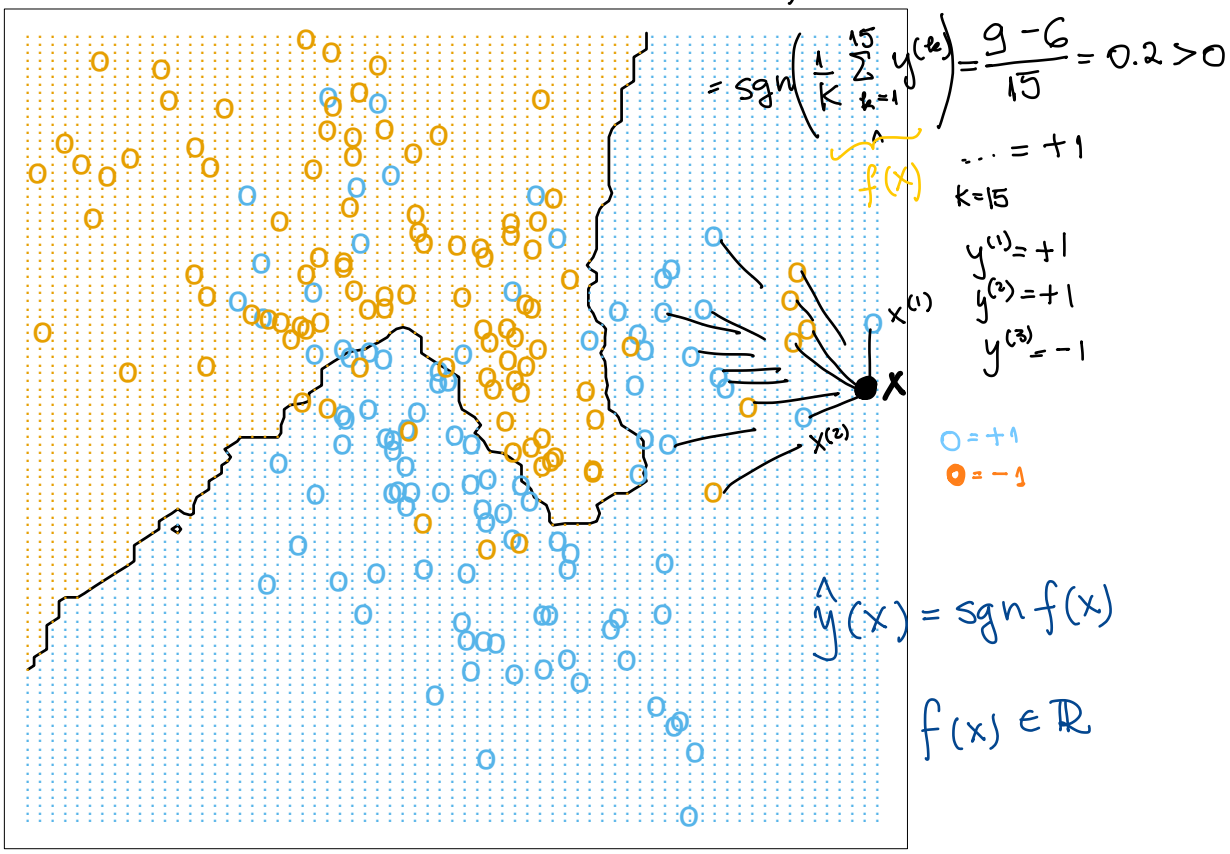


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

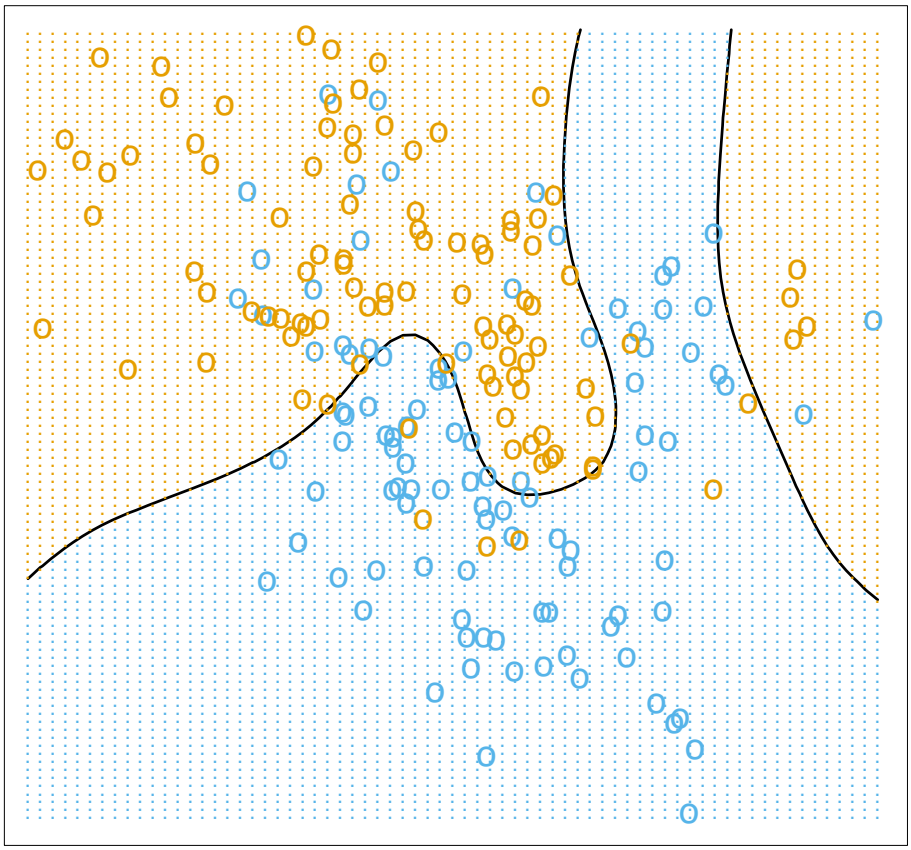


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Classifiers with real-valued output

Binary classification

- Since $y \in \{\pm 1\}$, naturally $f : \mathbf{X} \to \{\pm 1\}$
- But sometimes we prefer a classifier $f : \mathbf{X} \to \mathbb{R}$ (from a predictor class $\mathcal{F}$ of real-valued functions)
- In this case, the prediction $\hat{y}$ is usually

$$\hat{y} = \text{sgn}(f(x)) \tag{7}$$

This is sometimes known as the sign trick.

Examples of real-valued classifiers

- Logistic Regression
- Naive Bayes

  in both of the above, $f(x) = P[Y = 1|X = x] \in [0, 1]$. Hence

$$\hat{y} = \text{sgn}\left(f(x) - \frac{1}{2}\right) \tag{8}$$

- Support Vector Machines
- Kernel classifiers
- Neural Networks

Sign trick

The *sign* function $\text{sgn}(y) = y/|y|$ if $y \neq 0$ and 0 iff $y = 0$ turns a real valued variable Y into a discrete-valued one.

Why real valued f?

1) Confidence measure
2) f may be differentiable

- for statistical models $f(x) = P[Y = 1 | X = x]$ Example: Logistic regression
- for non-statistical models, $|f(x)|$ measures **confidence** in prediction $\hat{y}$, with $|f(x)| \approx 0$ meaning low confidence. Example: SVM
- if f is differentiable[1], the gradient ∇f is used in **learning algorithms** Examples: Logistic Regression, neural networks, some forms of linear regression such as Lasso

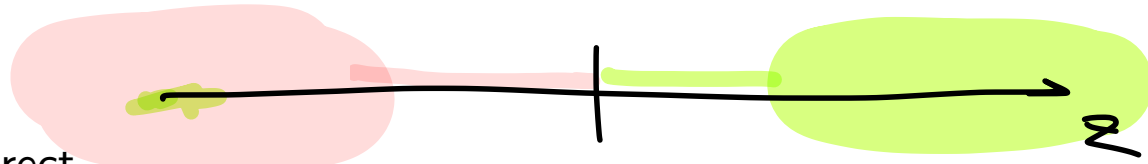
$z > 0 \Leftrightarrow y = +, f(x) > 0$
OR
$y = -, f(x) < 0$
CORRECT!

The margin (assuming $y \in \{\pm 1\}$)

- The **margin** of a classifier f at point $x \in \mathbf{X}$ is defined as

$$z = yf(x). \tag{9}$$

- Note that $\text{sgn}(z) = y\hat{y}$.



- If $z > 0$, $\hat{y} = y$ and $f(x)$ is correct
- If $z \gg 0$, then $f(x)$ is correct, and classifier has high confidence
- If $z < 0$, then $f(x)$ is incorrect, and $|z|$ measures "how wrong" is f on this x
- Note also that $z \approx 0$ means that the classification $\hat{y}$ is not robust, whether correct or not

[1] and ∇f not 0 almost everywhere

Real valued multi-way classifiers

- We train m classifier $f_{1:m} : \mathbf{X} \to \mathbb{R}$. Then (typically)

$$\hat{y} = \operatorname*{argmax}_{c=1:m} f_{1:m}(x). \tag{10}$$

- $\hat{y} = y$ means the classifier is correct
- the training can be done
  - independently for each f_c, $c = 1 : m$ (e.g. generative classifiers – in Lecture II)
  - or at the same time (e.g. neural networks, SVM)

- The **margin** is defined as

$$z(x) = f_y - \max_{c \neq y} f_c(x) \tag{11}$$

In other words

- if $\hat{y} = y$ (correct), then $z = f_{\text{true}} - f_{\text{nextbest}} > 0$
- if $\hat{y} \neq y$ (mistake), then $z = f_{\text{true}} - f_{\hat{y}} < 0$ (since $f_{\hat{y}}(x)$ is the max of $f_c(x)$)

Lecture Notes I-2 – Examples of Predictors. Nearest Neighbor and Kernel Predictors. Bias and Variance

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

January 12, 2026

Reading HTF Ch.: 2.1–5,2.9, 7.1–4 bias-variance tradeoff, Murphy Ch.: 1., 8.6[1], Bach Ch.:

[1] Neither textbook is close to these notes except in a few places; take them as alternative perspectives or related reading

Training and testing error

- Let $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$ be the training set and let the K-NN classifier from $\mathcal{D}$ be f_K
- How "good" is f_K?
- **Training error** $= \frac{1}{n}\#(\text{errors of } f_K \text{ on } \mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[f_K(x^i) \neq y^i]}$

$$\mathbf{1}_{[\text{expression}]} = \begin{cases} 1 & \text{iff exp} = \text{TRUE} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

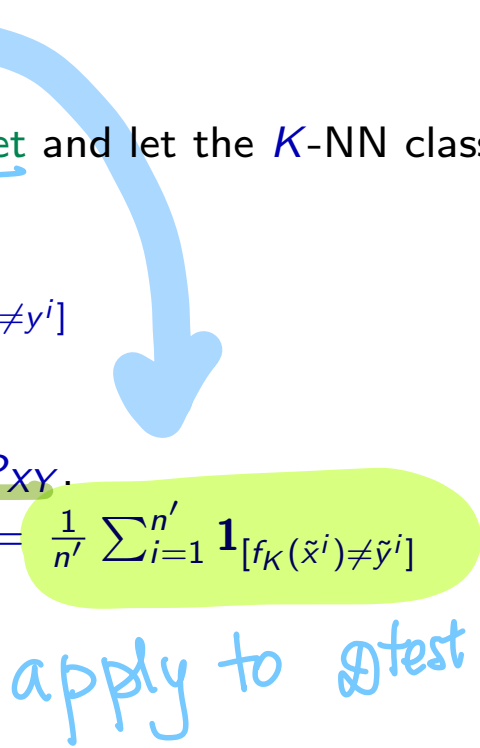
"indicator"

Training and testing error

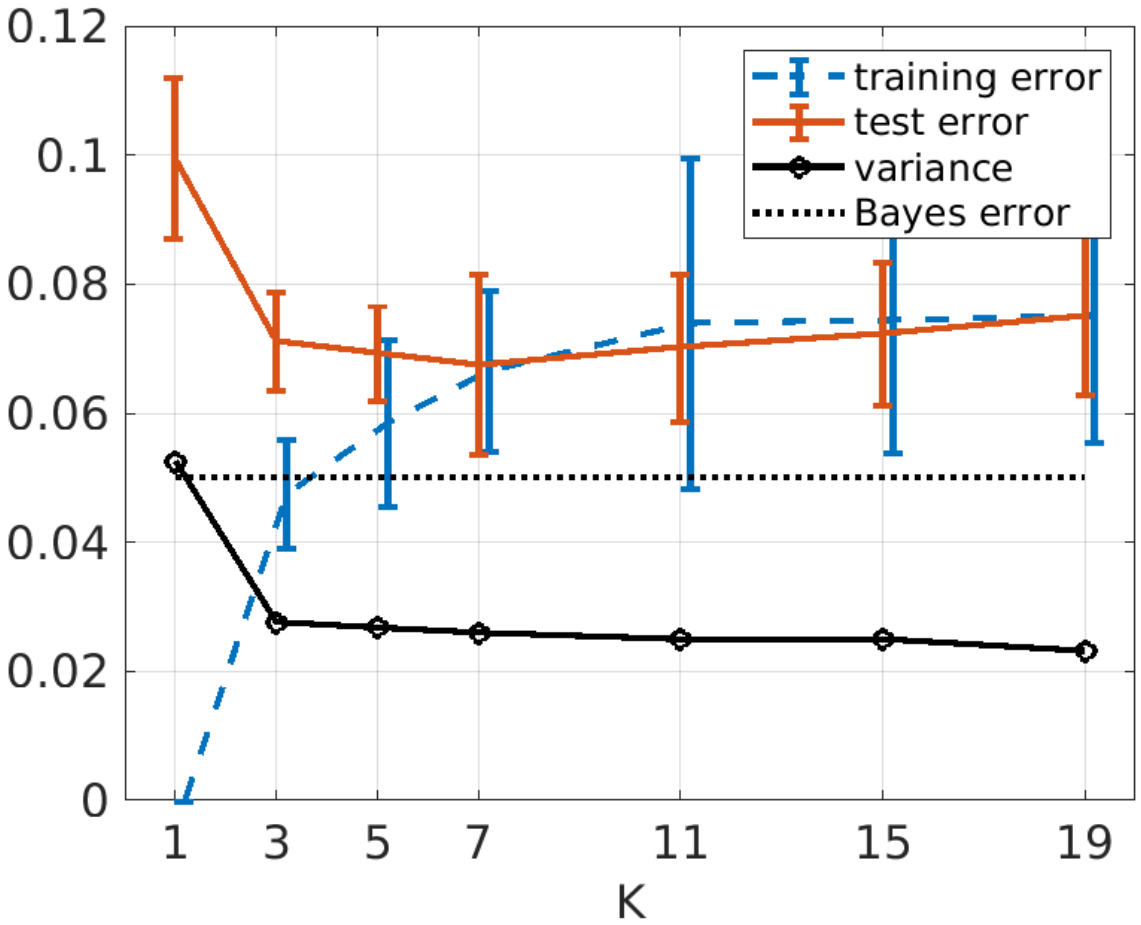
- Let $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$ be the training set and let the K-NN classifier from $\mathcal{D}$ be f_K
- How "good" is f_K?
- **Training error** $= \frac{1}{n}\#(\text{errors of } f_K \text{ on } \mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[f_K(x^i) \neq y^i]}$
- **Test error** $Pr[f_K(x) \neq y]$ for new points $(x, y) \sim P_{XY}$

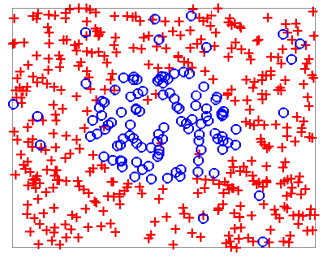
Training and testing error

- Let $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots (x^n, y^n)\}$ be the training set and let the K-NN classifier from $\mathcal{D}$ be f_K
- How "good" is f_K?
- **Training error** $= \frac{1}{n}\#(\text{errors of } f_K \text{ on } \mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[f_K(x^i) \neq y^i]}$
- **Test error** $Pr[f_K(x) \neq y]$ for new points $(x, y) \sim P_{XY}$
- We approximate the test error by using a **test set** $\mathcal{D}^{\text{test}} = \{(\tilde{x}^1, \tilde{y}^1), (\tilde{x}^2, \tilde{y}^2), \ldots (\tilde{x}^{n'}, \tilde{y}^{n'})\}$ from the same P_{XY}.
- Thus, in practice, **Test error** $= \frac{1}{n}\#(\text{errors of } f_K \text{ on } \mathcal{D}^{\text{test}}) = \frac{1}{n'}\sum_{i=1}^{n'} \mathbf{1}_{[f_K(\tilde{x}^i) \neq \tilde{y}^i]}$



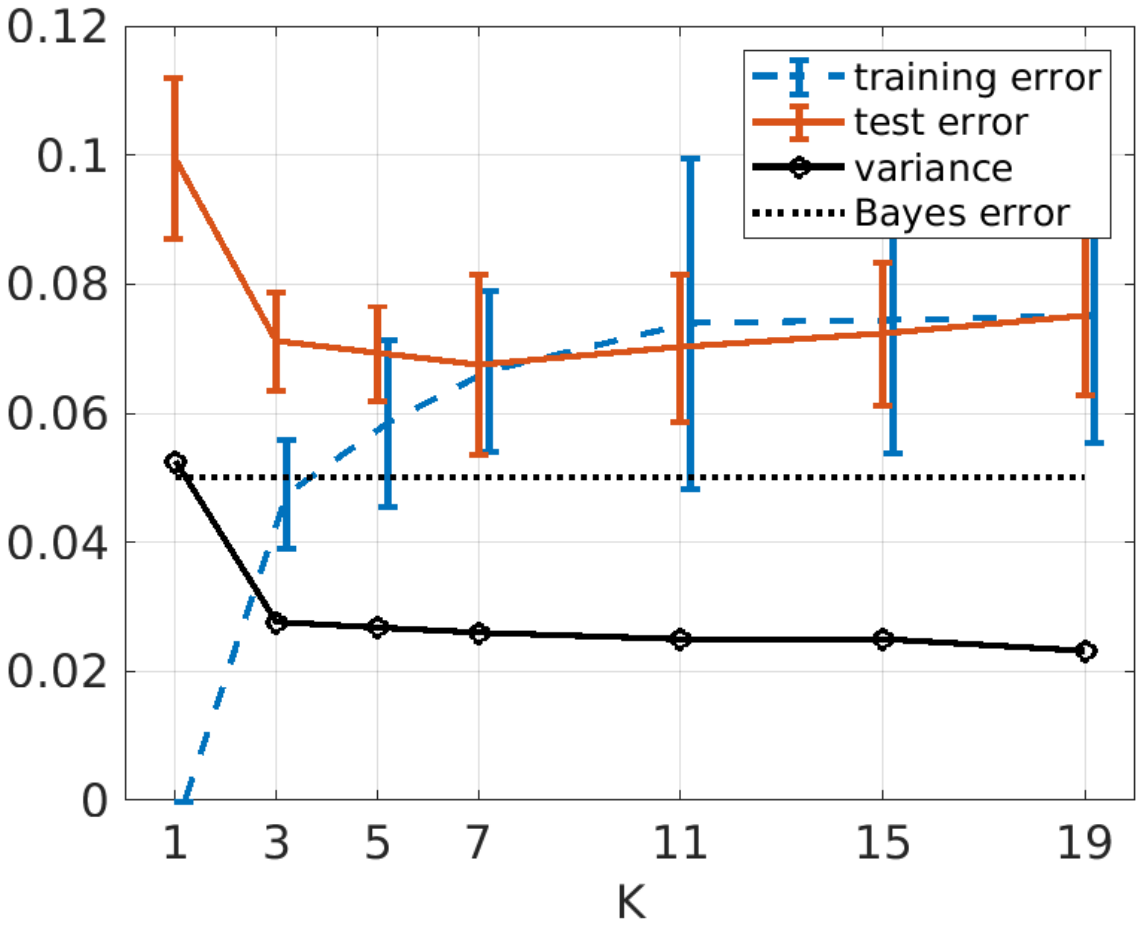
Training and testing error for K-NN

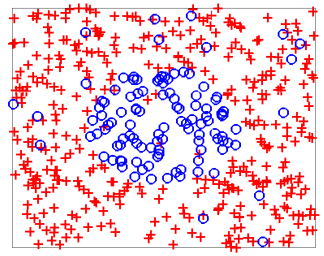




Ignore the "variance" and "Bayes error" for now

Training and testing error for K-NN

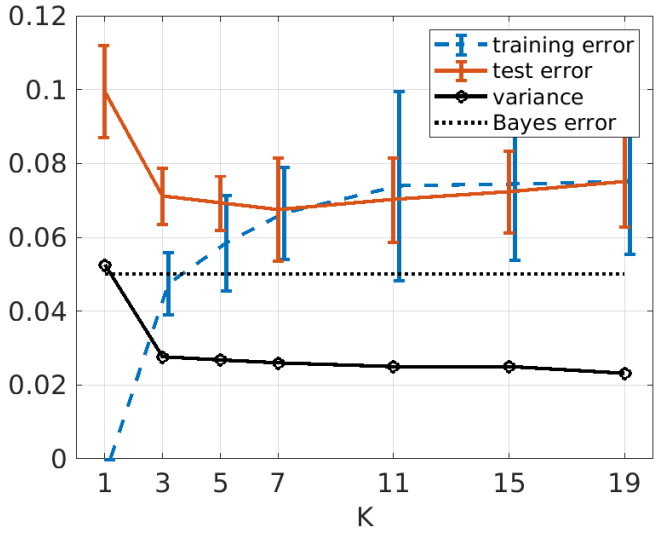




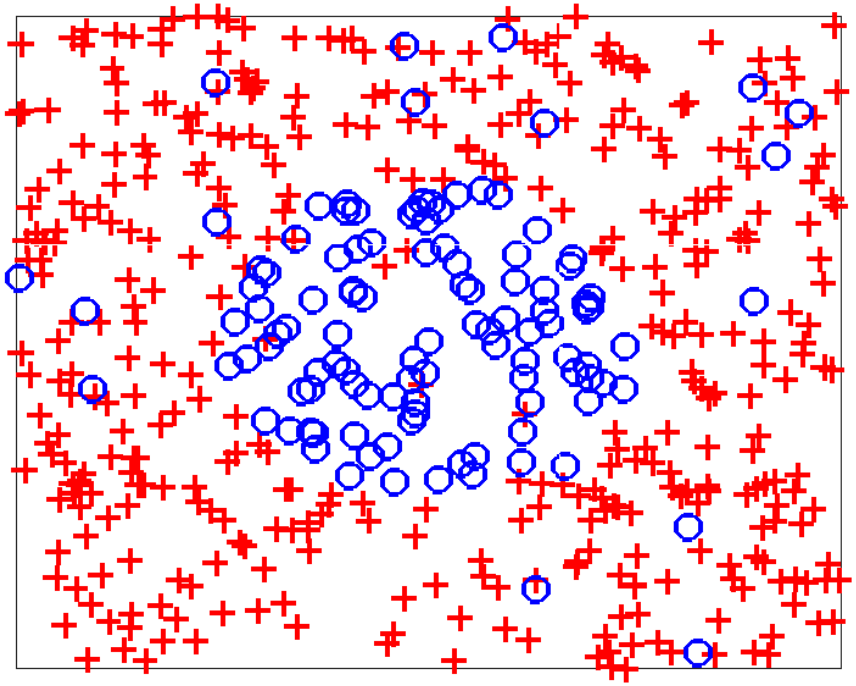
Ignore the "variance" and "Bayes error" for now

- So, what's happening? For $K = 1$, training error=0 but test error is large
- As K increases, test error decreases at first, then increases again

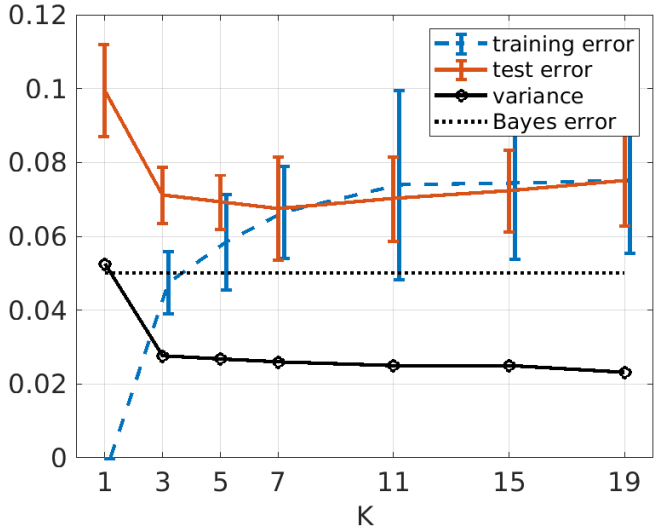
The case $K = 1$: Variance

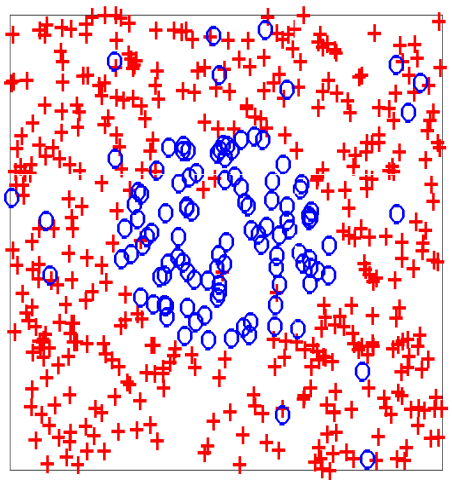


- $\mathcal{D} \sim P_{XY} \Rightarrow \mathcal{D}$ is **random**



The case $K = 1$: Variance





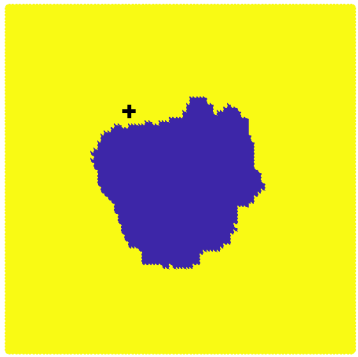
$(K = 1)$

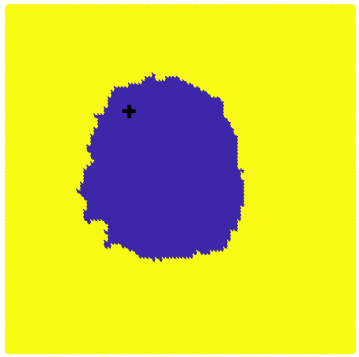
- $\mathcal{D} \sim P_{XY} \Rightarrow \mathcal{D}$ is **random**
- Hence any function f_K we estimate from $\mathcal{D}$ is also random

- Formally, for any fixed x, $f_K(x)$ is a random variable, hence it has a variance.
- In this course, we do not explicitly calculate the variance, but we want to know what increases or decreases it.

The case of K large: Bias

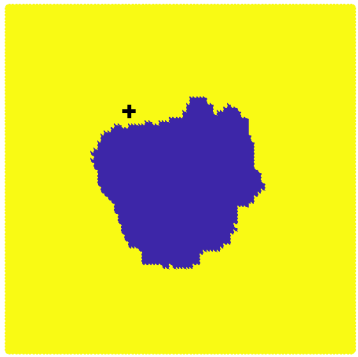
($K = 11$)

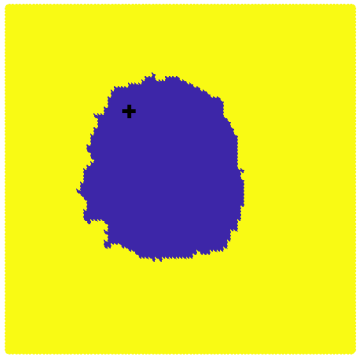




The case of K large: Bias

($K = 11$)





- **Bias** means to let one's own prior beliefs override the evidence.
- In data science/ML/statistics **every model/prediction** is a combination of **prior belief** and data

- **prior** = before seeing the data
- (usually) **prior belief** = prior **knowledge**, e.g. from previous experiments

- Bias can take many forms – in this course you will encounter several
- We do not explicitly calculate bias, but we want to identify where it is coming from, and what increases/decreases it
- One way to look for bias: if a predictor f cannot exactly/accurately predict a training set, "whatever is causing this" is bias.

The Bias-Variance trade-off

- When bias ↗, variance ↘

- When data set size n ↗, variance ↘