STAT 535 10/11/22

Lecture 4

HW 4 · t.b. posted
today (web)

Q1 - Oct 18

Material:
to end of
Lecture I

Zoom recordings
t.b. found

Tutorial - today !!

Nearest Neighbors
Kernel predictors

Lecture Notes I – Examples of Predictors

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

September 29, 2022

Prediction problems by the type of output

The "learning" paradigm and vocabulary

The Nearest-Neigbor and kernel predictors

Linear predictors

- Least squares regression
- Linear Discriminant Analysis (LDA)
- QDA (Quadratic Discriminant Analysis)
- Logistic Regression
- The PERCEPTRON algorithm

Classification and regression tree(s) (CART)

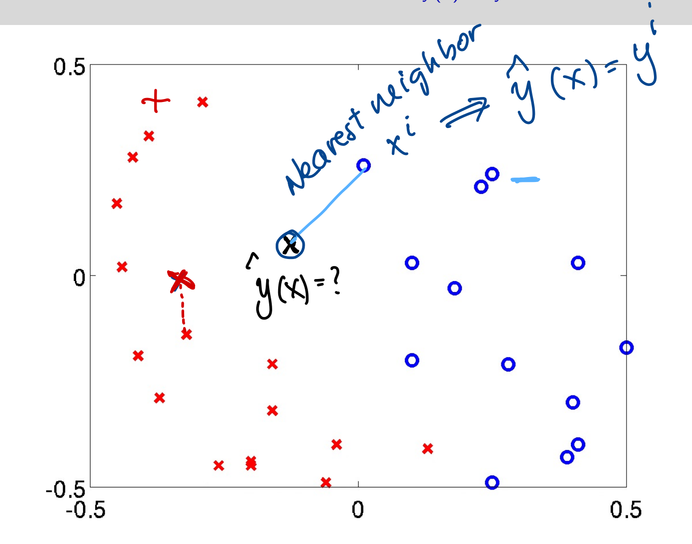
The Naive Bayes classifier

Reading HTF Ch.: 2.3.1 Linear regression, 2.3.2 Nearest neighbor, 4.1–4 Linear classification, 6.1–3. Kernel regression, 6.6.2 kernel classifiers, 6.6.3 Naive Bayes, 9.2 CART, 11.3 Neural networks, Murphy Ch.: 1.4.2 nearest neighbors, 1.4.4 linear regression, 1.4.5 logistic regression, 3.5 and 10.2.1 Naive Bayes,4.2.1–3 linear and quadratic discriminant, 14.7.3– kernel regression, locally weighted regression, 16.2.1–4 CART, (16.5 neural nets), Bach Ch.:

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.

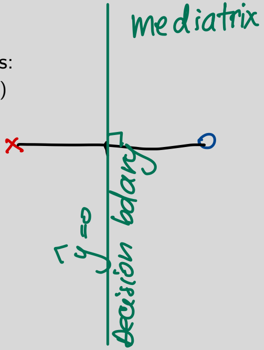
$$\hat{y}(x) = y^i$$

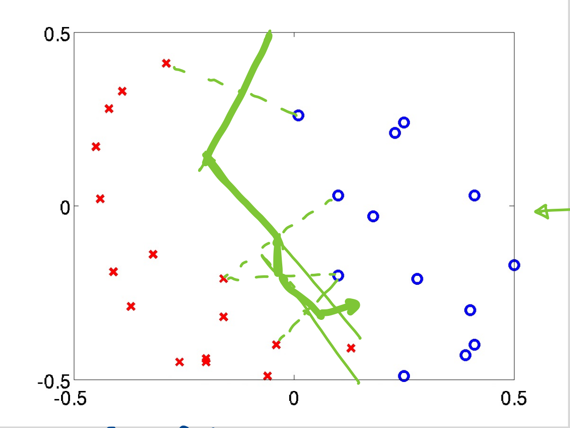


The Nearest-Neighbor predictor

- **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.

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





Decision bdary:
- piecewise linear
Union of subsets of hyperplanes
mediatrices btw $(x^i, x^j), y^i \neq y^j$

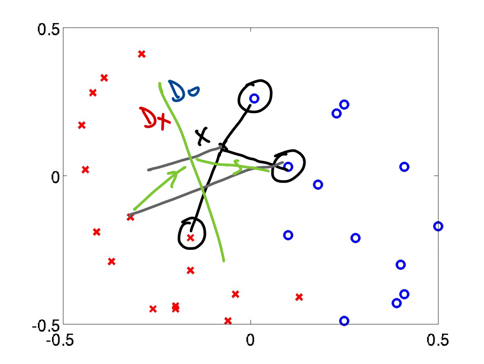
- decision bdary can take arbitrary shape

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 $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  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



K=3

x: 2 o, 1 x ⇒ $\hat{y}(x)$=o

Decision bdary:

- piecewise linear

Exercise

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.

$\Rightarrow$

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

- **K-Nearest Neighbor** (with $K = 3, 5$ or larger)
  1. find the K nearest neighbors of x in $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  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

pcwise constant

2-class or multiclass

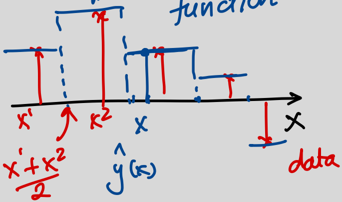
↓

ties possible

$\Rightarrow$ on decision bdary.

1-NN regression

piece wise constant function



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 $\mathcal{D}$: $x^{i_1, \ldots i_K}$
  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

↗ with K

• smoothness parameter

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

– computing $\hat{y}$ → requires search of $\mathcal{D}$!

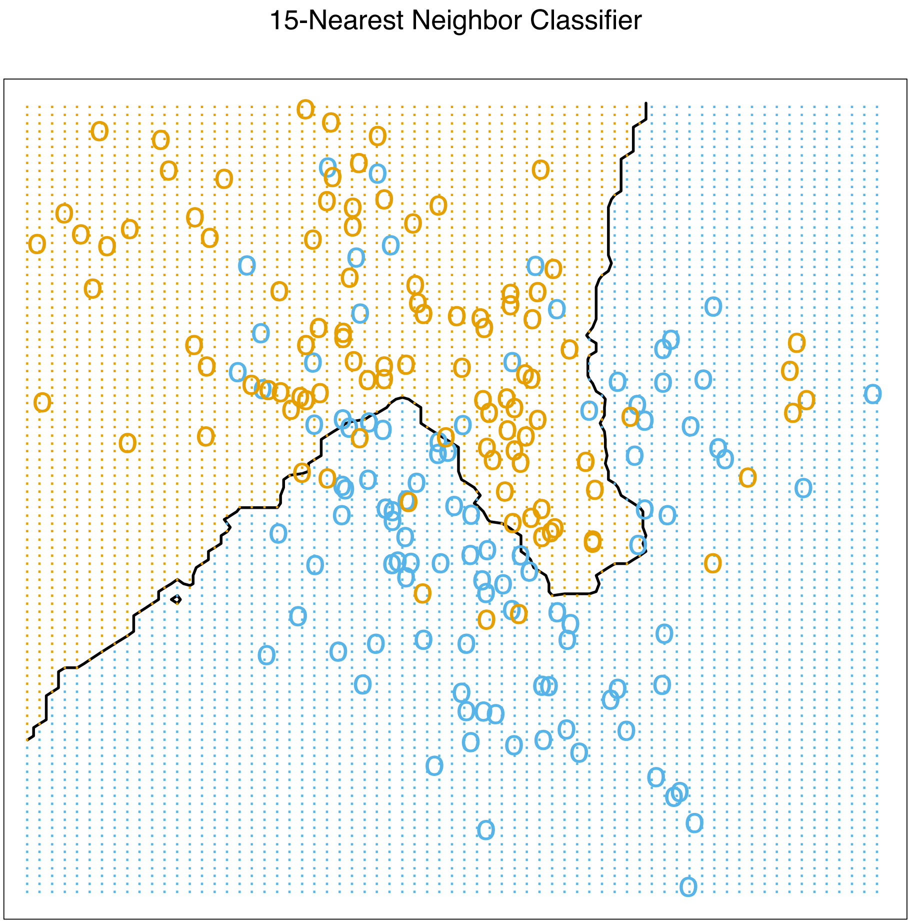


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* ($\texttt{BLUE} = 0, \texttt{ORANGE} = 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest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

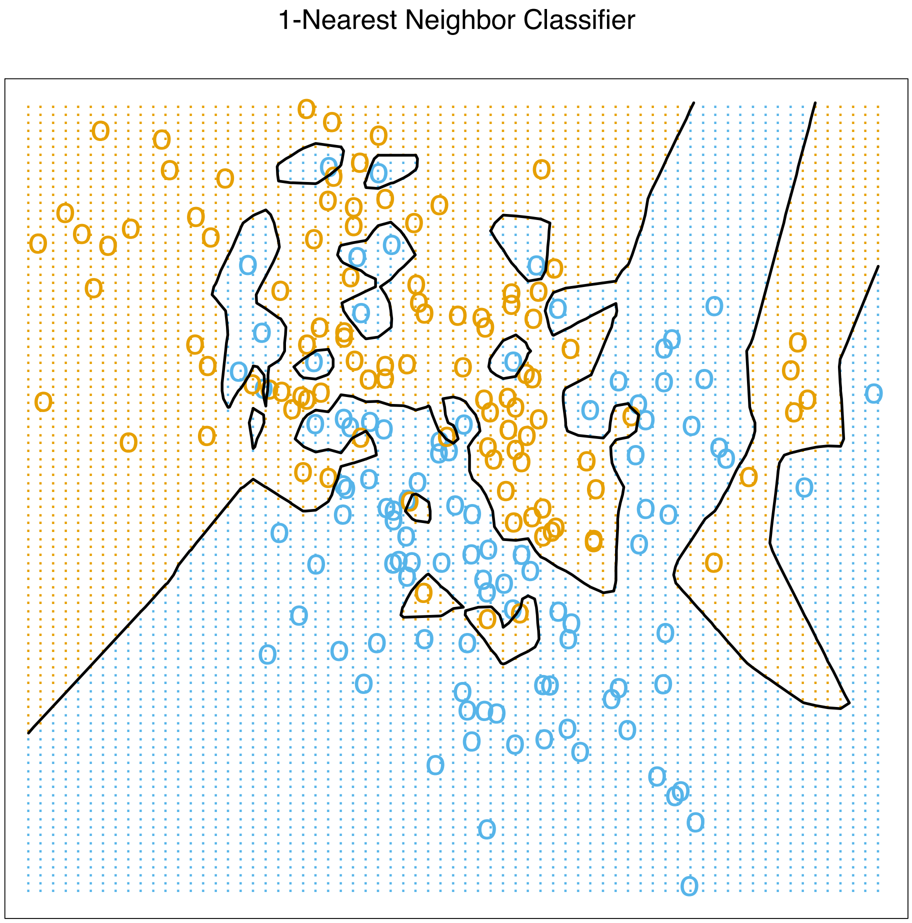
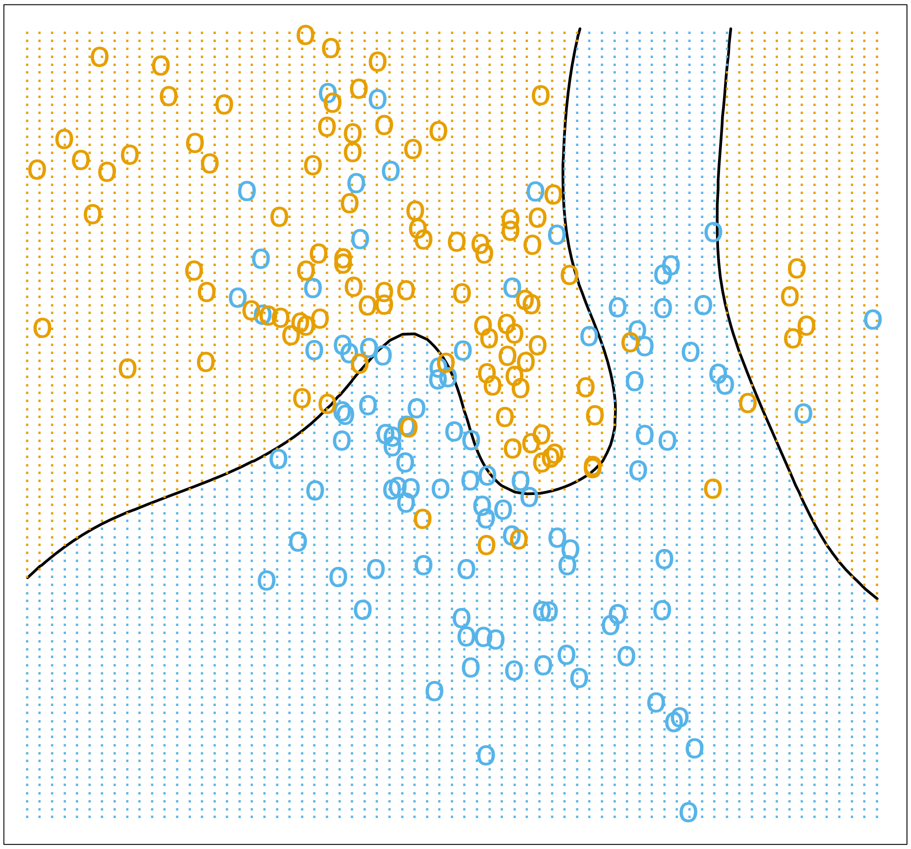


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$), *and then predicted by* 1*-nearest-neighbor classification.*



= Knowing true data distribution

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).*

Kernel regression and classification

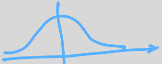
Kernel Predictors

- Like the K-nearest neighbor but with "smoothed" neighborhoods
- The predictor

$$f(x) = \sum_{i=1}^{n} \overbrace{\beta_i}^{\text{weights}} \underbrace{b(x, x^i)}_{\text{kernel}} \underbrace{y^i}_{\text{data}} \tag{1}$$

$\propto$ weighted avg of y^i of neighbors

where β_i are coefficients



Kernel $b(z)$

- $b(z) \geq 0$ for all z
- $\int_{-\infty}^{\infty} b(z)dz < \infty$

$\left[\int_{-\infty}^{\infty} b(z)dz = 1 \text{ in KDE} \right]$ w.l.o.g (without loss of generality)

Typically:

- $b(z) = b(-z)$
- smooth $\rightarrow C^{\infty}$ $\quad b', b'', \ldots$ exist a.e. (almost everywhere)
- compact support $\Leftrightarrow b(z) = 0$ for $|z| > R \Rightarrow \overline{\text{supp}\, b} = \bar{B}_R(0) = \{z \mid \|z\| \leq R\}$ ball

$\text{supp} f = \{z \mid f(z) \neq 0\}$

Kernel regression and classification

$x \in \mathbb{R}^d$

- Like the K-nearest neighbor but with "smoothed" neighborhoods
- The predictor

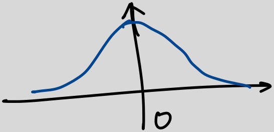
$$f(x) = \sum_{i=1}^{n} \beta_i \underline{b(x, x^i)} y^i \tag{1}$$

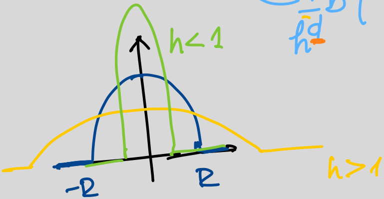
$$\frac{1}{h^d} b\left(\frac{\|x - x^i\|}{h} \right)$$

kernel bandwidth

where β_i are coefficients

Example:





Gaussian

$$b(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

Quadratic (compact support)

- h is smoothing parameter

$$b : \mathbb{R} \to [0, \infty)$$

Kernel regression and classification

- Like the K-nearest neighbor but with "smoothed" neighborhoods
- The predictor

$$f(x) = \sum_{i=1}^{n} \beta_i b(x, x^i) y^i \tag{1}$$

  where β_i are coefficients
- Intuition: center a "bell-shaped" *kernel* function b on each data point, and obtain the prediction $f(x)$ as a weighted sum of the values y^i, where the weights are $\beta_i b(x, x^i)$
- Requirements for a kernel function $b(x, x')$
  1. non-negativity
  2. symmetry in the arguments x, x'
  3. optional: radial symmetry, bounded support, smoothness
- A typical kernel function is the **Gaussian kernel** (or **Radial Basis Function (RBF)**)

$$b(z) \propto e^{-z^2/2} \tag{2}$$

$$b_h(x, x') \propto e^{-\frac{||x-x'||^2}{2h^2}} \quad \text{with} \quad h = \textbf{the kernel width} \tag{3}$$

Regression example

A special case in wide use is the Nadaraya-Watson regressor

• x = query point

$$f(x) = \frac{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right) y^i}{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right)}. \quad (4)$$

$$= \sum_{i=1}^{n} \frac{\beta}{h^d}\, b\left(\frac{||x-x^i||}{h}\right) y^i$$

normalizing constant (x)

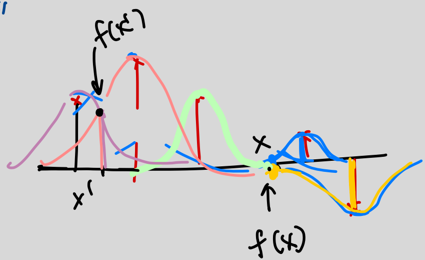
In this regressor, $f(x)$ is always a convex combination of the y^i's, and the weigths are proportional to $b_h(x, x^i)$.
The Nadaraya-Watson regressor is biased if the density of P_X varies around x.

$$\beta = \frac{1}{\sum_{i=1}^{n} b\left(\frac{||x-x^i||}{h}\right) \cdot \frac{1}{h^d}}$$

$$f(x) = \sum w_i y^i \Leftarrow$$ convex combination of y^i's

$$\sum_{i=1}^{n} w_i(x,h) = 1$$

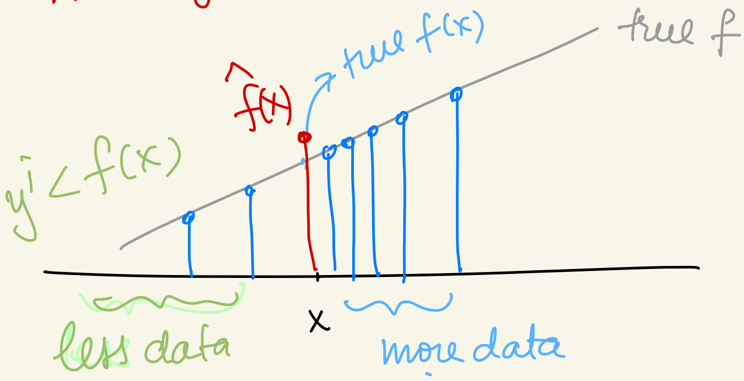
• $\sum \alpha_i y_i$ = convex combination
if $\alpha_i \geq 0$ for all i
and $\sum_i \alpha_i = 1$



Nadaraya Watson bias $\rightarrow$ defined later

(FYI bias $\propto \nabla f(x) \cdot \nabla p(x)$)
function density

[Without proof]



$$\hat{f}(x) = \sum w_i \, y^i$$

Local Linear Regression

To correct for the bias (to first order) one can estimate a regression line around x.

1. Given **query point** x
2. Compute kernel $b_h(x, x^i) = w_i$ for all $i = 1, \ldots N$
3. Solve weighted regression $\min_{\beta,\beta_0} \sum_{i=1}^{d} w_i \left(y^i - \beta^T x^i - \beta_0\right)^2$ to obtain β, β_0 (β, β_0 depend on x through w_i)
4. Calculate $f(x) = \beta^T x + \beta_0$ ← evaluated at 1 point!

Exercise Show that Nadaraya-Watson solves a local linear regression with fixed $\beta = 0$

Idea

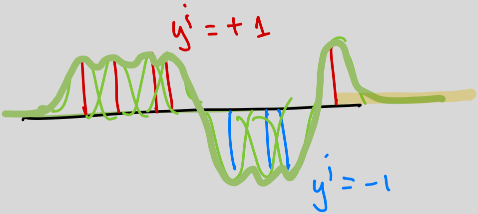
N-Watson : $f^{NW}(x) = \sum w_i y^i$ → best constant fction fit to weighted data

LLR : $f^{LLR}(x) =$ best linear fction fit to weighted data

Kernel binary classifiers

- Obtained from Nadaraya-Watson by setting y^i to ± 1.
- Note that the classifier can be written as the difference of two non-negative functions

$$f(x) \propto \sum_{i:y^i=1} b\left(\frac{||x-x^i||}{h}\right) - \sum_{i:y^i=-1} b\left(\frac{||x-x^i||}{h}\right). \qquad (5)$$



$\hat{y} = \text{sign } f(x)$