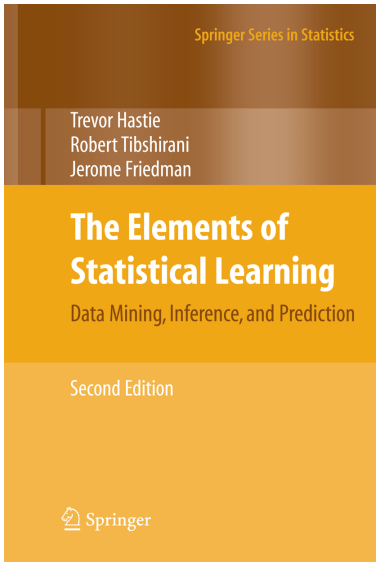
Machine Learning

Marta Vomlelová

marta@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~marta

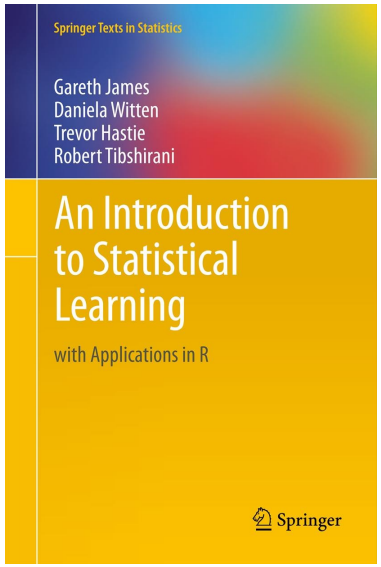
February 21, 2019

Literature



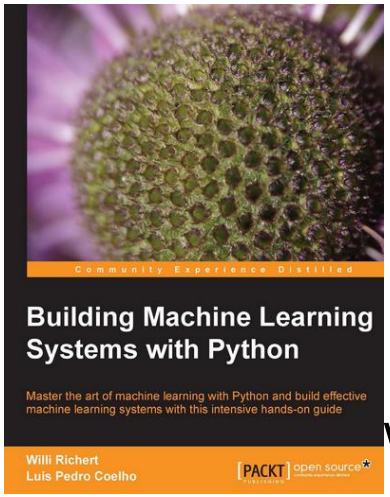
T. Hastie, R. Tishirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer Series in Statistics. Springer, (web)

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani: *An Introduction to Statistical Learning with Applications in R* (2013)



Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

Further reading



Willi Richert, Luis Pedro Coelho: *Building Machine Learning Systems with Python* 2013

- I.H.Witten and E.Frank. Data Mining - Practical machine learning tools and techniques with Java implementation. Accademic Press Pub., USA, 1999.
- P. Berka. Dobývání znalostí z databází. Academia, 2003.
- T. Mitchell. Machine Learning. McGraw Hill, New York, 1997.
- S. Russel and P. Norwig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2003.

Exam

- oral exam on topics coverd by lectures
- most of it is covered by T. Hastie, R. Tishirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer Series in Statistics.

Email Spam example

Supervised classification example

- 4601 email messages annotated email or spam
- features: relative frequencies of 57 of the most commonly occuring words and punctation marks.

	george	you	your	free	!
spam	0.00	2.26	1.38	0.52	0.51
email	1.27	1.27	0.44	0.07	0.11

- Our learning method has to decide which features to use and how: for example, we might use a rule like:
  - if (%george $<$ 0.6) & (%you$>$ 1.5) then email else spam
  - if (%you * 0.2) - (%george * 0.3)$>$0 then spam else email.
  - and of course, more complex models.
- For this problem not all errors are equal
  - we want to avoid filtering our good email.

Prostate Cancer

Supervised regression example

- The goal is to predict the log of PSA lpsa prostate specific antigen
- from measurements including log-cancer-volume lcavol and other clinical measurements.
- It is a regression task.
- **Scatterplot matrix** in the figure.

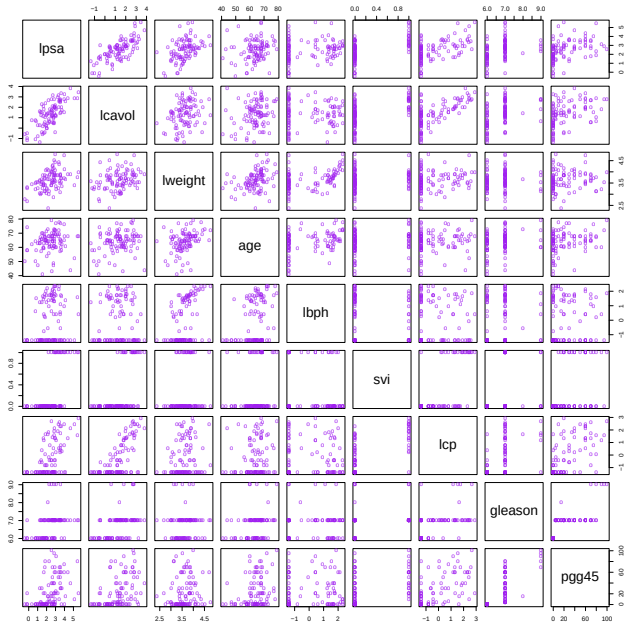
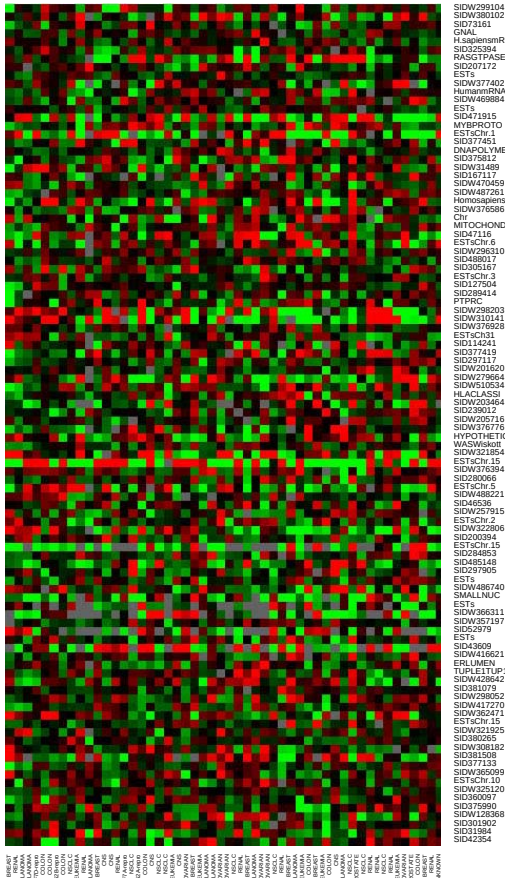


FIGURE 1.1. *Scatterplot matrix of the prostate cancer data. The first row shows the response against each of the predictors in turn. Two of the predictors,* `svi` *and* `gleason`*, are categorical.*

DNA Expression Microarrays

Unsuprevized example

- 64 samples (columnns) from 64 tumor patients, 6830 genes (rows), only 100 shown
- head map from green (negative) to red (positive)
- Which samples are most similar to each other, in terms of their expression profiles across samples?
- Which genes are most similar to each other, in terms of their expression profiles across samples?
- Do certain genes show very high (or low) expression for certain cancer samples?



Typical Scenario

- We have the **goal variable** (**cílovou veličinu**, target)
  - **quantitative** Y – stock price, robot position, or
  - **categorical** G – heart attack yes/no,
    - coding by $0, 1$ or $-1, 1$ or **dummy variables**.
  - (or ordered categorical - small, medium, large, without metric)
- We ame to predict it based on **features (příznaky)** X (=predictors, independent variables, variables) sensor measurement, clinical tests.
- We have **training dataset**, where both features and goal variable are known.
- Based on training set we create a **model** $\hat{f}, \hat{g}$ or the (blackbox) prediction of the goal variable **model** $\hat{Y}, \hat{G}$.
- Good model predicts the goal with a **small error**, for example $RSS(data) = \sum_{i \in data} (y_i - \hat{y}_i)^2$.

Notation

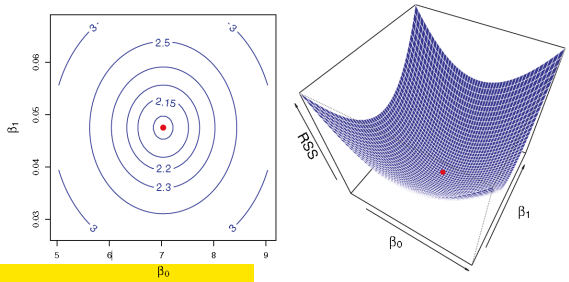
We have

- list of features $X_1, \ldots, X_p$
- numerical goal variable Y
- training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

	A_1	A_j	A_p	Goal attribute
X^T = vector	$\langle X_1$	X_j	$X_p\rangle$	Y or G
$\mathbf{x}_1^T$				
$\mathbf{x}_i^T$ = vector	$\langle x_1$	x_j	$x_p\rangle$	y or g
$\mathbf{x}_N^T$				

  - x and $\mathbf{x}_i$ are p-dimensional column vectors
  - $\mathbf{X}$ is the $N \times p$ matrix
  - $\mathbf{x}_j$ is the N vector consisting of all observations on variable X_j.
- $\mathbf{y} = (y_1, \ldots, y_N)^T$ denotes the vector of training goal data.

The Simplest Model – a Constant



- Assume the goal Y does not depend on X
- What is the best model?
- We take square error loss $L(y, \hat{y}) = (y - \hat{y})^2$
- Find a constant $c \in \mathbb{R}$ that minimizes the error on training data?

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - c)^2$$

- RSS is called **residual sum of squares**
  - **residuum** $r_i = y_i - \hat{y}_i$ the difference between the true and the predicted value.
- Take a derivative, set equal to 0, you get:
- $c = \sum_{i=1}^{n} y_i = \overline{y}$
- that is the average.
- **Training error** of this model is $\frac{RSS}{n} = MSE$ **mean square error**.

Linear regression

- Given a vector of inputs $X = (X_1, \ldots, X_p)$ we predict the output Y via the model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

- $\hat{\beta}_0$ is the **intercept, bias**, **(průsečík)**
- We include the constant variable 1 to X, include $\hat{\beta}_0$ in β to get the model in vector form as an inner product

$$\hat{Y} = \sum_{j=0}^{p} X_j \hat{\beta}_j = X^T \beta$$

- The sum $\sum_{j=0}^{p} X_j \hat{\beta}_j$ can be written as $X^T \beta$.

Linear regression from the data

- Let i range over the data samples, $\mathbf{X}$ be an $N \times p$ data matrix, $\mathbf{y}$ is a column vector of the goal variable. We can write:

$$\hat{\mathbf{y}} = \mathbf{X}\beta$$

- We search optimal $\hat{\beta}$ to minimize the residual sum squares RSS:

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\beta)^2 = (\mathbf{y} - \mathbf{X}\beta)^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\beta)$$

- Differentiating w.r.t. β we get *normal equations*

$$\mathbf{X}^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\beta) = \mathbf{0}$$

- If $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ is not singular, then the unique solution is given by

$$\hat{\beta} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

- the estimate $\hat{y}_i$ for a given x_i is $\hat{y}_i = \hat{y}(x_i) = x_i^T\hat{\beta}$
- From a singular $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ we should remove dependent features or filter the data to make it invertible.

Example – Storch brings babies in Europia

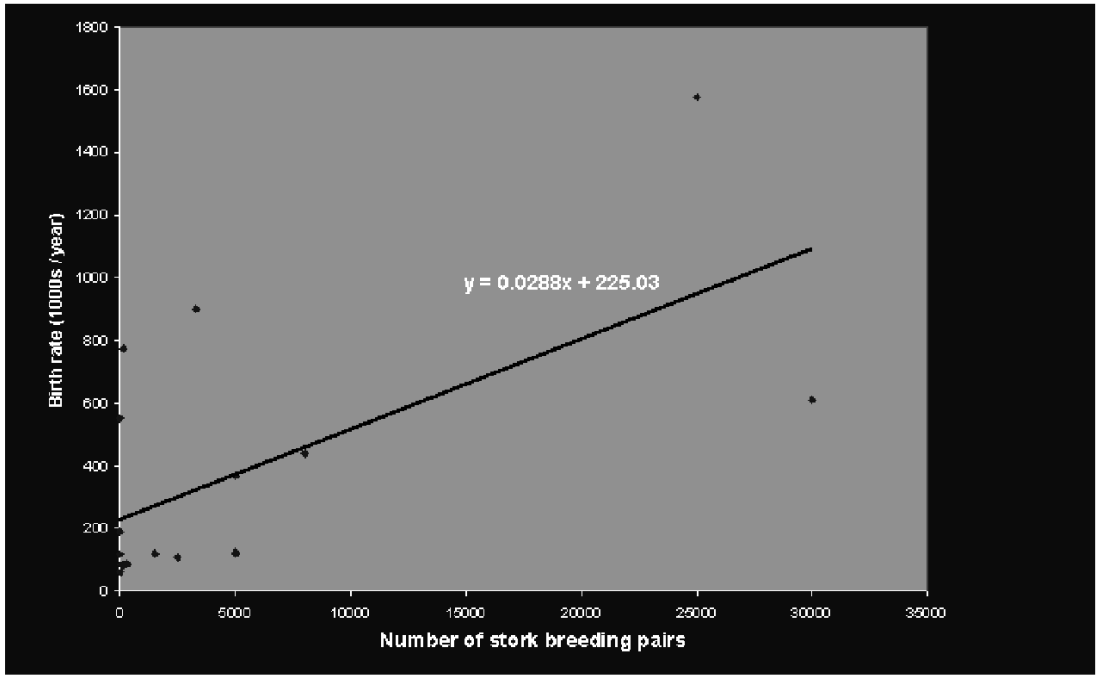


Fig 1. How the number of human births varies with stork populations in 17 European countries.

Teaching Statistics. Volume 22, Number 2, Summer 2000 • **37**

Example – Storch continued

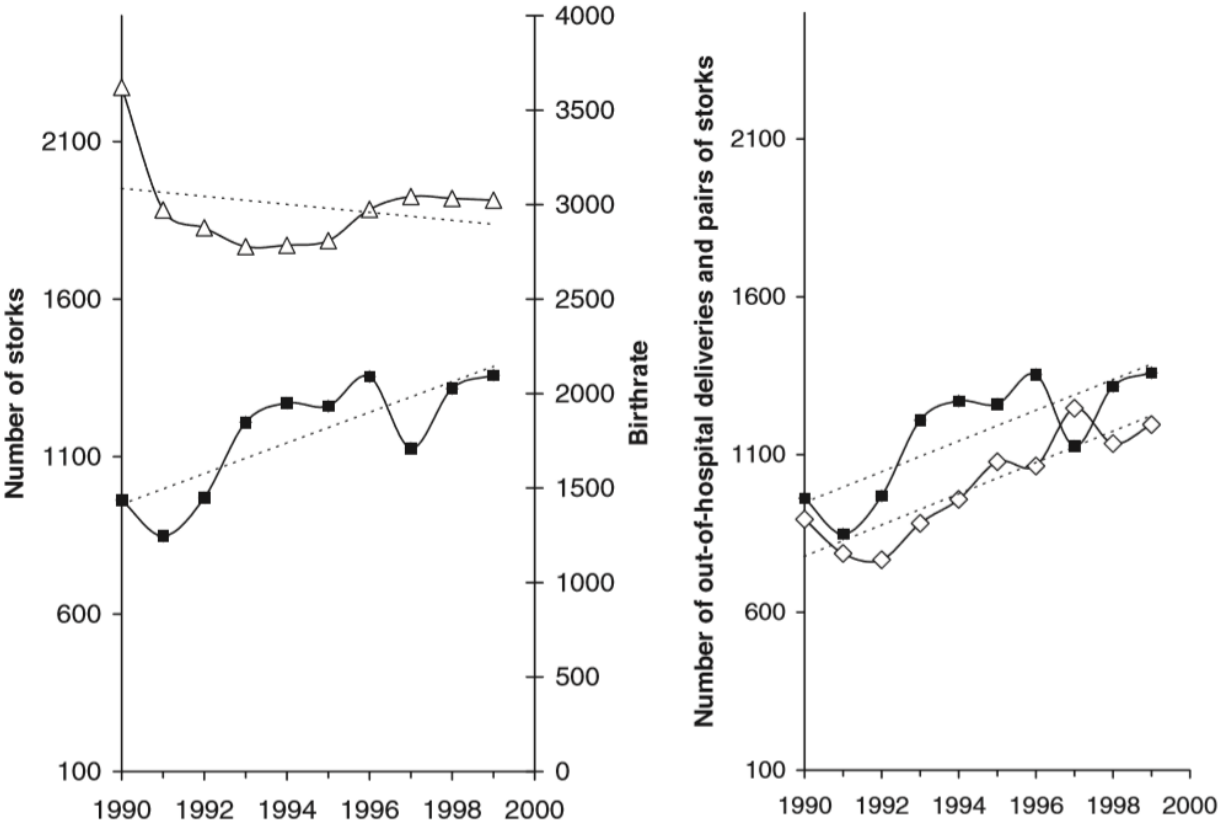


Figure 2. Storks in Brandenburg and the birthrates in Berlin, Germany (1990–99). Open triangles show number of clinical deliveries per year in Berlin. Open diamonds show number of out-of-hospital deliveries per year in Berlin. Number of pairs of storks are shown as full squares. Dotted lines represent linear regression trend ($y = mx + b$). For the convenience of the readers, two figures are presented. Left graph shows clinical deliveries against pairs of storks using two scalings, right graph shows numbers of out-of-hospital deliveries and pairs of storks both on the same scale. In both figures, data are from the years 1990–2000.

Linear methods for classification

- We are given two features X_1, X_2 and the goal *BLUE* or *ORANGE*.
- Later, we will see better ways. For now, we encode *BLUE* $= 0$ a *ORANGE* $= 1$, and find a linear regression model.
- The fitted values $\hat{Y}$ are converted to a fitted class variable $\hat{G}$ as follows:
$$\hat{G} = \left\{ \begin{array}{c} \textit{BLUE} \text{ for } Y \leq 0.5 \\ \textit{ORANGE} \text{ for } Y > 0.5 \end{array} \right\}$$
- The hyperplane $\{x : x^T \beta = 0.5\}$ is called the **decision boundary (rozhodovací hranice)**.

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T \hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

Two Scenarios

- The training data in each class were generated from bivariate Gaussian distribution with uncorrelated components and different means.
- The linear model is (almost) optimal.

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE*.*

- The training data in each class came from a mixture of 10 low-variance Gaussian distributions, with individual means themselves distributed as Gaussians.
- The linear model is **not** optimal.

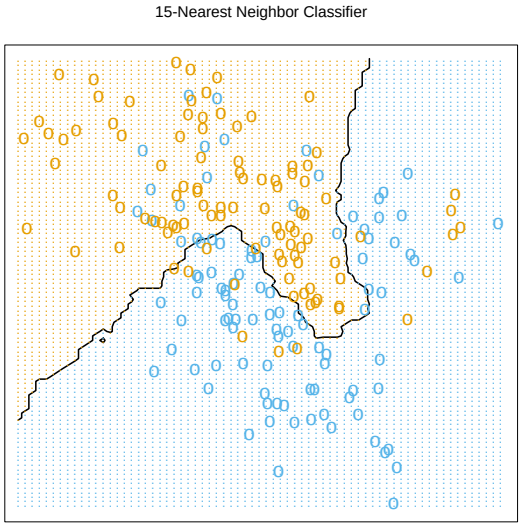


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*) and then fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.*

Nearest-Neighbor Methods

- The nearest-neighbor methods use those observations in the training set $\mathcal{T}$ closest in the input space to x to form $\hat{Y}$.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

- In classification, majority vote is used.
- Figures correspond to 15 nearest neighbor and 1 nearest neighbor respectively.
- Training error (usually) increases with increasing k.
- The **effective number of parameters** of k nearest neighbors is N/k and is generally bigger than p of the linear regression.

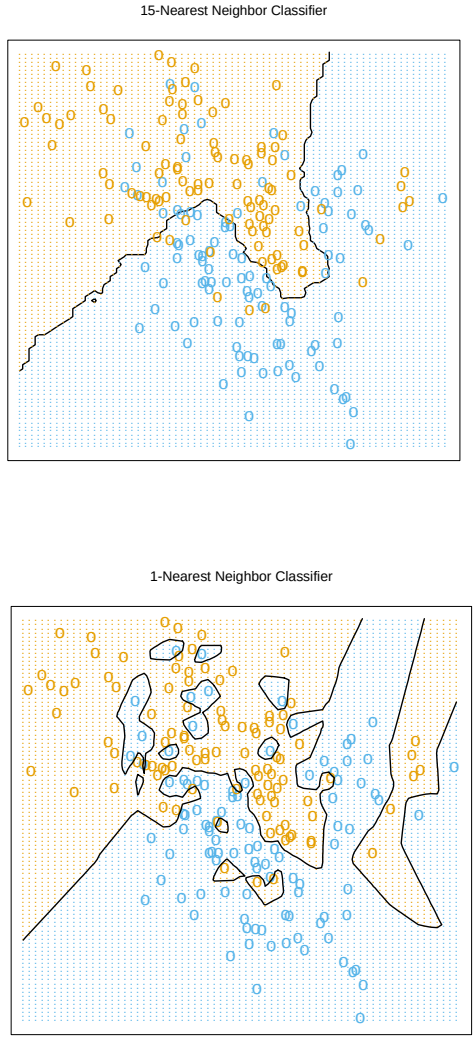
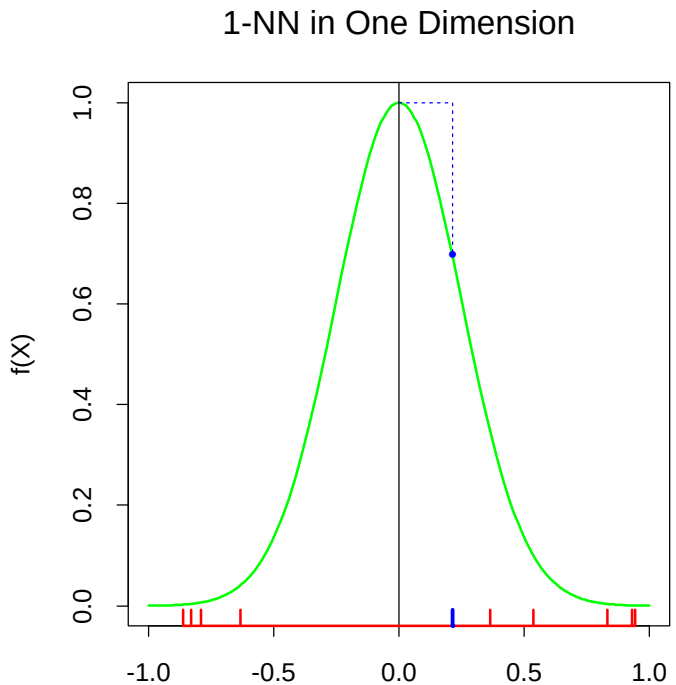


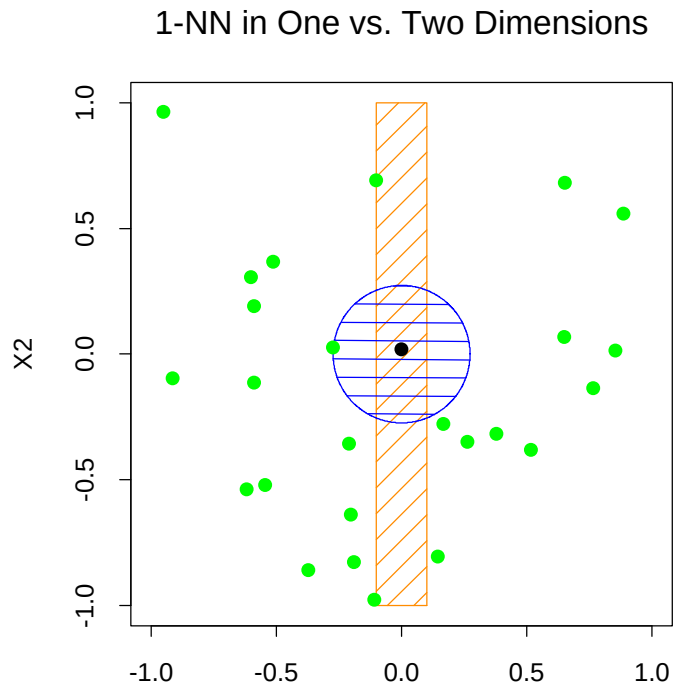
FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Curse of Dimensionality demonstration

Prediction

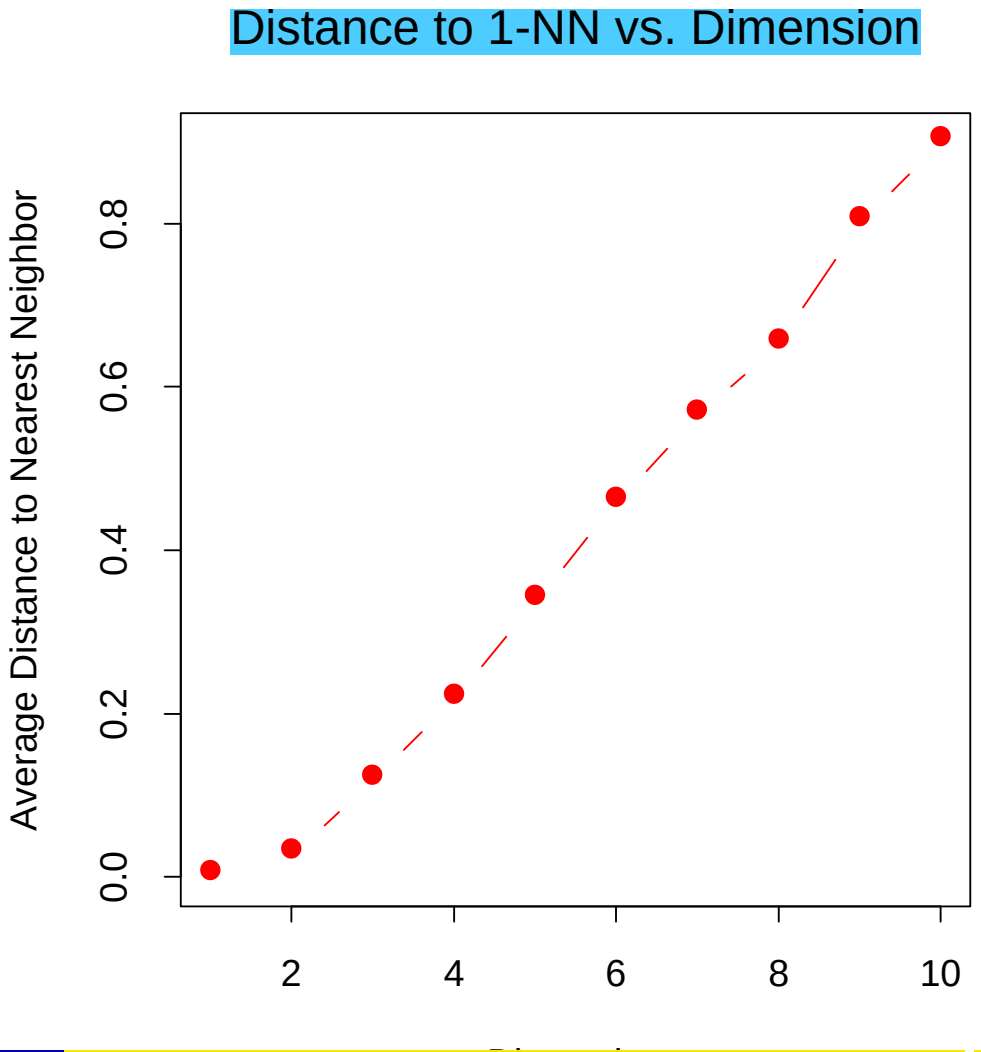
- Assume x_i uniformly generated form the interval $\langle -1, 1 \rangle^p$
- We have $Y = f(X) = e^{-8\|X\|^2}$, without any noise, for x_i we know exactly $f(x_i)$.
- We use 1–NN to estimate $f(0)$ based on 1000 data sample.
- Predicted value $x = \langle 0, \ldots, 0 \rangle$ is lower that 1 and in high dimensions p it goes to 0.
- Increasing k in k–NN does not help here.





Empirical Nearest Neighbor Distance

- Assume x_i uniformly generated form the interval $\langle -1, 1 \rangle^p$
- We use 1–NN to estimate $f(0)$ based on 1000 data sample.



Local Methods in High Dimensions

- Having enough data, k-NN is a good choice since it is less biased than other models.
- **Curse of dimensionality** In high dimensions we never have enough data.

Example 1:

- Consider p–dimensional space, where instances are uniformly distributed in the unit hypercube.
- The size of the neighborhood containing $r \times 100\%$ instances has the volume r, that is the edge length $e_n(r) = r^{\frac{1}{p}}$.
- $e_{10}(0.01) = 0.63$ and $e_{10}(0.1) = 0.8$, $e_{6830}(\frac{1}{64}) = 0.9993913$.

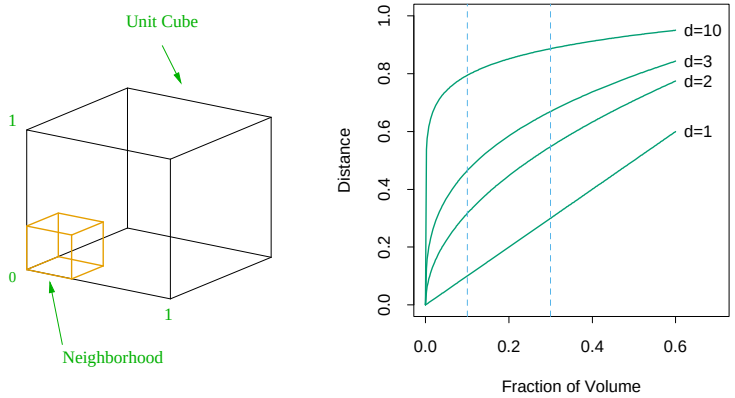


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover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

- To cover 1% instances in 10-dimensional space we need 63% values of any feature. It is not local any more.
- Decreasing r, we get fewer instances and the average is no more robust.

Curse of dimensionality

Example 2: Most points are close to the border

- Consider N instances uniformly distributed in a p–dimensional unit ball.
- Median distance of the nearest neighbor from the center is:

$$d(p, N) = \left(1 - \frac{1}{2}^{\frac{1}{N}}\right)^{\frac{1}{p}}$$

- For $N = 500$, $p = 10$, we get $d(p, N) \approx 0.52$, that is more than a half way to the border.
- Close to the border, we must **extrapolate**, what is more difficult than interpolation.

Overfitting

- Our goal is the minimal error on test data (orange).
- Usually, **overfitting** appears for complex models - an increase of the test error despite the decrease of the training error.
- This is the reason for other models then nearest neighbor model.
- Possible improvements:
  - Kernel methods
  - different weights for dimensions
  - local regression fits
  - linear models fit to a basis expansion
  - sums of non-linearly transformed linear models.

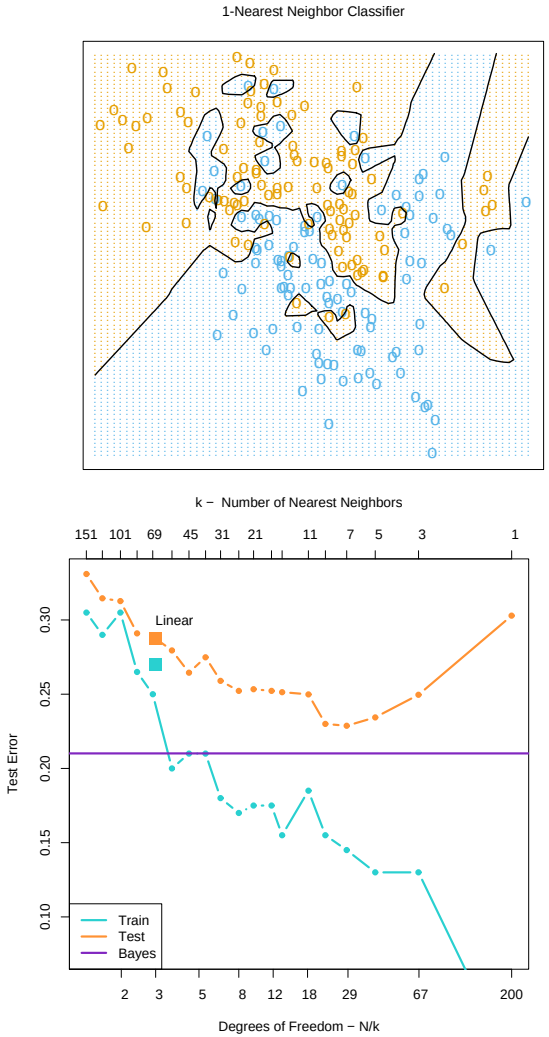


FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample*

Statistical Decision Theory

- Let $X \in \mathbb{R}^p$ denote a real valued random input vector, and $Y \in \mathbb{R}$ a real valued random output variable, with joint distribution $P(X, Y)$.
- We seek a function $f(X)$ for prediction Y given values of the input X.
- The theory requires a **loss function (chybovou funkci)** $L(Y, f(X))$ for penalizing errors in predictions.
- The far most common and convenient is **squared error loss (kvadratická chybová funkce)** $L(Y, f(X)) = (Y - f(X))^2$
- this leads us to a criterion for choosing f, the **expected (squared) prediction error (očekávanou chybu)** (EPE),

$$\begin{aligned} EPE(f) &= \mathbb{E}(Y - f(X))^2 \\ &= \int (y - f(x))^2 P(dx, dy) \end{aligned}$$

- by conditioning on X we get

$$EPE(f) = \mathbb{E}_X \mathbb{E}_{Y|X}([Y - f(X)]^2 | X)$$

- and we see that it suffices to minimize EPE poinwise:

$$f(x) = \text{argmin}_c \mathbb{E}_{Y|X}([Y - c]^2 | X = x).$$

- the solution is the conditional expectation also known as the **regression**

k-NN and Conditional Expectation

- We seek the conditional expectation:

$$f(x) \quad = \quad \mathbb{E}(Y|X = x).$$

- Thus the best prediction of Y at any point $X = x$ is the conditional mean, when the best is measured by the average squared error.
- The nearest neighbor methods attempt to directly implement this.
  - Since there are typically at most one observation at any point x, we settle for

$$\hat{f}(x) \quad = \quad mean(y_i|x \in N_k(x)),$$

  - where *mean* denotes average, and $N_k(x)$ is the neighborhood containing k points in $\mathcal{T}$ closest to x.
- Under mild regularity conditions on $P(X, Y)$ one can show as $k, N \to \infty$, such that $\frac{k}{N} \to 0$ then $\hat{f}(x) \to \mathbb{E}(Y|X = x)$.
- **The rate of convergence decreases as the dimension increases**. The problem is the speed of the convergence.

Linear Regression and Expected Prediction Error

- If we assume that the regression function is approximately linear $f(x) \approx x^T \beta$,
- w plug it into the EPE formula

$$EPE(f) = \mathbb{E}(Y - X^T \beta)^2$$

- we differrentiate w.r.t. β and we get

$$\beta = [\mathbb{E}(XX^T)]^{-1}\mathbb{E}(XY).$$

- We have not conditioned on X, rather we have used our knowledge of the functional relationship to pool over values of X.

Model Assumptions before minimizing EPE

- Least squares assumes $f(x)$ is well approximated by a globally linear function.
- k-nearest neighbors assumes $f(x)$ is well approxiamted by a locally constatn function.
- **additive models** assume that

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} f_j(x_j)$$

- each coordinate is approximated by an arbitrary function f_j
  - usually k-nearest neighbors to approximate univariate conditional expectations
  - simultaneously for each of the coordinate functions.
- the additivity of the linear model retains.
- β_j is hidden in f_j, it is not needed separately.

Structural Regression Models

- **penalized methods, bayesian methods**

$$PRSS(f;\lambda) = RSS(f) + \lambda \cdot complexity(f)$$

  - Lasso, Ridge regression, smoothing spline
- **kernal methods, local regression**

$$K_\lambda(x_0, x) = \frac{1}{\lambda} e^{-\frac{\|x-x_0\|^2}{2\lambda}}$$

- **Dictionary Methods, Basis Functions**

$$f_\theta(x) = \Sigma_{m=1}^{M} \theta_m h_m(x)$$

  - spline, MARS.

Other Loss Functions

Regression

- There are other loss functions then the most popular $L_2 : \mathbb{E}(Y - f(X))^2$.
- Taking the absolute loss function: $L_1 : \mathbb{E}(|Y - f(X)|)$ leads to conditional median

$$\hat{f}(x) = median(Y|X = x)$$

- its estimates are more robust
- L_1 have discontinuities in their derivatives, which have hindered teir widespread use.
- We will meed other loss functions later (exponential, Hubber).

Predicting probabilities

- A common loss functions for **probability** prediction is the **cross entropy** **(vzájemná entropie)** deviance - compares this and saturated model

$$H(p, \hat{p}) = \mathbb{E}_p[-log_2(\hat{p})] \approx -\sum p_i log_2 \hat{p}_i$$

  - p_i frequency in the test data, $\hat{p}_i$ estimate from train data.

Loss function for Categorical Goal

- Let us have the goal variable G with K classes.
- Our loss function can be represented by a $K \times K$ matrix,
  - L will be zero on the diagonal and nonnegative elsewhere
  - $L(k, \ell)$ is the price paid for classifying an observation belonging to G_k as G_ℓ.
- Most often we use the zero-one 0-1 loss function, where all missclassifications are charged a single unit.
- The expected prediction error is $EPE = \mathbb{E}[L(G, \hat{G}(X)]$, the expectation is taken with respect to the joint distribution $P(G, X)$.
- We condition: $EPE = \mathbb{E}_X \sum_{k=1}^{K} L(G, \hat{G}(X)) \cdot P(G_k|X)$
- and it suffices to minimize EPE pointwise:

$$\hat{G}(X) = \text{argmin}_{g \in G} \sum_{k=1}^{K} L(G, g) \cdot P(G_k|X).$$

- With 0–1 loss function this simplifies to

$$\hat{G}(X) = \text{argmin}_{g \in G}[1 - P(g|X)].$$

- or simply, we classify x as G_k if

$$P(G_k|X = x) = max_{g \in G} P(g|X = x).$$

Bayes rate (minimální chyba)

- We classify x as G_k if

$$P(G_k|X = x) = max_{g \in G} P(g|X = x).$$

- that is we classify to the mosts probable class $P(G|X)$; this solution is called **Bayes classifier**.
- The error rate of the Bayes classifier is called the **Bayes rate**.
- k-NN directly approximates Bayes classifier. It takes the majority vote in a nearest neighborhood.
- Suppose for a two-class problem we had taken the dummy-variable approach and coded G via binary Y 0 and 1, followed by squared error loss estimation. Then $\hat{f}(X) = \mathbb{E}(Y|X) = P(G = G_1|X)$.
- For more classes, better methods exist.

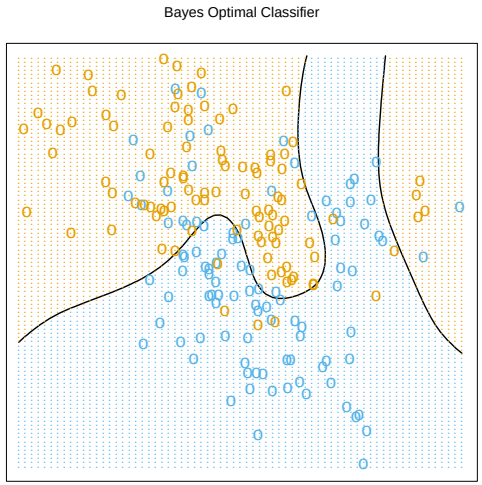


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*