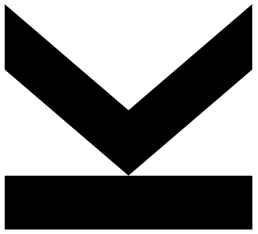
UNIT 2



Basics of Supervised Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

QUESTIONS WE NEED TO ADDRESS

- Does learning help in the future, i.e. does experience from previously observed examples help us to solve a future task?
- What is a good model? How do we assess the quality of a model?
- Will a given model be helpful in the future?

BASIC SETUP: INPUTS

Assume we want to learn something about objects from a set/space X. Most often, these objects are represented by vectors of feature values, i.e.

$$\mathbf{x} = (x_1, \ldots, x_d) \in \underbrace{X_1 \times \cdots \times X_d}_{=X}$$

For simplicity, we will not distinguish between the objects and the feature vector in the following.

If X_j is a finite set of labels, we speak of a *categorical variable/feature*. If $X_j = \mathbb{R}$, a real interval, etc., we speak of a *numerical variable/feature*.

BASIC SETUP: INPUTS (cont'd)

Assume we are given l objects $\mathbf{x}^1, \ldots, \mathbf{x}^l$ that have been observed in the past—the so-called *training set*. Each of these objects is characterized by its feature vector:

$$\mathbf{x}^i = (x_1^i, \ldots, x_d^i)$$

We can write this conveniently in matrix notation (called *matrix of feature vectors*):

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^l \end{pmatrix} = \begin{pmatrix} x_1^1 & \ldots & x_d^1 \\ \vdots & \ddots & \vdots \\ x_1^l & \ldots & x_d^l \end{pmatrix}$$

BASIC SETUP: INPUTS VS. OUTPUTS

Further assume that we know a target value $y^i \in \mathbb{R}$ for each training sample $\mathbf{x}^i$. All these values constitute the *target/label vector*:

$$\mathbf{y} = (y^1, \ldots, y^l)^T$$

The *training data matrix* is then defined as follows:

$$\mathbf{Z} = (\mathbf{X} \mid \mathbf{y}) = \begin{pmatrix} \mathbf{x}^1 & y^1 \\ \vdots & \vdots \\ \mathbf{x}^l & y^l \end{pmatrix} = \begin{pmatrix} x_1^1 & \ldots & x_d^1 & y^1 \\ \vdots & \ddots & \vdots & \vdots \\ x_1^l & \ldots & x_d^l & y^l \end{pmatrix}$$

In the following, we denote $Z = X \times \mathbb{R}$.

CLASSIFICATION VS. REGRESSION

Classification: the target/label values are categorical, i.e. from a finite set of labels; we will often consider *binary classification*, i.e. where we have two classes; in this case, unless indicated otherwise, we will use the labels -1 (*negative class*) and +1 (*positive class*)

Regression: the target/label values are numerical

THE PROBABILISTIC FRAMEWORK FOR SUPERVISED ML (1/3)

The quality of a model can only be judged on the basis of its performance on future data. So assume that future data are generated according to some *joint distribution of inputs and outputs*, the *joint density* of which we denote as

$$p(\mathbf{z}) = p(\mathbf{x}, y)$$

If we have only finitely many possible data samples, $p(\mathbf{z}) = p(\mathbf{x}, y)$ is the probability to observe the datum $\mathbf{z} = (\mathbf{x}, y)$.

THE PROBABILISTIC FRAMEWORK FOR SUPERVISED ML (2/3)

Marginal distributions: $p(\mathbf{x})$ is the density/probability of observing input vector $\mathbf{x}$ (regardless of its target value); $p(y)$ is the density/probability of observing target value y

Conditional distributions: $p(\mathbf{x} \mid y)$ is the density of input values for a given target value y; $p(y \mid \mathbf{x})$ is the density/probability to observe a target value y for a given input $\mathbf{x}$

THE PROBABILISTIC FRAMEWORK FOR SUPERVISED ML (3/3)

In case of binary classification, we will use the following notations to make things a bit clearer:

$p(y = -1)$	probability to observe a negative sample
$p(y = +1)$	probability to observe a positive sample
$p(\mathbf{x} \mid y = -1)$	distribution density of negative class
$p(\mathbf{x} \mid y = +1)$	distribution density of positive class
$p(y = -1 \mid \mathbf{x})$	probability that $\mathbf{x}$ belongs to negative class
$p(y = +1 \mid \mathbf{x})$	probability that $\mathbf{x}$ belongs to positive class

SOME BASIC CORRESPONDENCES

Using definitions:

$$p(\mathbf{x}, y) = p(\mathbf{x} \mid y) \cdot p(y) \qquad p(\mathbf{x}, y) = p(y \mid \mathbf{x}) \cdot p(\mathbf{x})$$

Bayes' Theorem:

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y) \cdot p(y)}{p(\mathbf{x})} \qquad p(\mathbf{x} \mid y) = \frac{p(y \mid \mathbf{x}) \cdot p(\mathbf{x})}{p(y)}$$

Getting marginal densities by integrating out:

$$p(\mathbf{x}) = \int_{\mathbb{R}} p(\mathbf{x}, y) dy = \int_{\mathbb{R}} p(\mathbf{x} \mid y) \cdot p(y) dy$$

$$p(y) = \int_{X} p(\mathbf{x}, y) d\mathbf{x} = \int_{X} p(y \mid \mathbf{x}) \cdot p(\mathbf{x}) d\mathbf{x}$$

SOME BASIC CORRESPONDENCES (cont'd)

In the case of binary classification:

$$p(y = -1) + p(y = +1) = 1$$

$$p(y = -1 \mid \mathbf{x}) + p(y = +1 \mid \mathbf{x}) = 1 \quad \text{for all } \mathbf{x}$$

$$\begin{aligned} p(\mathbf{x}) &= p(\mathbf{x}, y = -1) + p(\mathbf{x}, y = +1) \\ &= p(\mathbf{x} \mid y = -1) \cdot p(y = -1) + p(\mathbf{x} \mid y = +1) \cdot p(y = +1) \end{aligned}$$

LOSS FUNCTIONS

Assume that the mapping g corresponds to our model class (parametric model) in the sense that

$$g(\mathbf{x}; \mathbf{w})$$

maps the input vector $\mathbf{x}$ to the predicted output value using the parameter vector $\mathbf{w}$ (i.e. $\mathbf{w}$ determines the model).
Then a *loss function*

$$L(y, g(\mathbf{x}; \mathbf{w}))$$

measures the loss/cost that incurs for a given data sample $\mathbf{z} = (\mathbf{x}, y)$ (i.e. with real output value y).

EXAMPLES OF LOSS FUNCTIONS

Zero-one loss:

$$L_{\mathbf{zo}}(y, g(\mathbf{x}; \mathbf{w})) = \begin{cases} 0 & y = g(\mathbf{x}; \mathbf{w}) \\ 1 & y \neq g(\mathbf{x}; \mathbf{w}) \end{cases}$$

Quadratic loss:

$$L_{\mathbf{q}}(y, g(\mathbf{x}; \mathbf{w})) = (y - g(\mathbf{x}; \mathbf{w}))^2$$

Clearly, the zero-one loss function makes little sense for regression.
For binary classification tasks, we have $L_{\mathbf{q}} = 4L_{\mathbf{zo}}$.

GENERALIZATION ERROR/RISK

The *generalization error* (or *risk*) is the expected loss on future data for a given model $g(.; \mathbf{w})$:

$$R(g(.; \mathbf{w})) = \mathrm{E}_{\mathbf{z}}\big(L(y, g(\mathbf{x}; \mathbf{w}))\big) = \int_Z L(y, g(\mathbf{x}; \mathbf{w})) \cdot p(\mathbf{z}) d\mathbf{z}$$

$$= \int_X \int_{\mathbb{R}} L(y, g(\mathbf{x}; \mathbf{w})) \cdot p(\mathbf{x}, y) dy d\mathbf{x}$$

$$= \int_X p(\mathbf{x}) \underbrace{\int_{\mathbb{R}} L(y, g(\mathbf{x}; \mathbf{w})) \cdot p(y \mid \mathbf{x}) dy}_{= R(g(\mathbf{x}; \mathbf{w})) = \mathrm{E}_{y|\mathbf{x}}(L(y, g(\mathbf{x}; \mathbf{w})))} d\mathbf{x}$$

Obviously, $R(g(\mathbf{x}; \mathbf{w}))$ denotes the expected loss for input $\mathbf{x}$.

The risk for the quadratic loss is called *mean squared error (MSE).*

GENERALIZATION ERROR FOR A NOISY FUNCTION

Assume that y is a function of $\mathbf{x}$ perturbed by some noise:

$$y = f(\mathbf{x}) + \varepsilon$$

Assume further that ε is distributed according to some *noise distribution* $p_n(\varepsilon)$. Then we can infer

$$p(y \mid \mathbf{x}) = p_n(y - f(\mathbf{x})),$$

which implies

$$p(\mathbf{z}) = p(y \mid \mathbf{x}) \cdot p(\mathbf{x}) = p(\mathbf{x}) \cdot p_n(y - f(\mathbf{x})).$$

GENERALIZATION ERROR FOR A NOISY FUNCTION (cont'd)

Then we obtain

$$R(g(.;\mathbf{w})) = \int_Z L(y, g(\mathbf{x};\mathbf{w})) \cdot p(\mathbf{z}) d\mathbf{z}$$

$$= \int_X p(\mathbf{x}) \int_{\mathbb{R}} L(y, g(\mathbf{x};\mathbf{w})) \cdot p_n(y - f(\mathbf{x})) dy d\mathbf{x}.$$

In the noise-free case, we get

$$R(g(.;\mathbf{w})) = \int_X p(\mathbf{x}) \cdot L(f(\mathbf{x}), g(\mathbf{x};\mathbf{w})) d\mathbf{x},$$

which can be understood as *"modeling error".*

GENERALIZATION ERROR FOR BINARY CLASSIFICATION (1/3)

For the zero-one loss, we obtain

$$R(g(.;\mathbf{w})) = \int_X \int_{\mathbb{R}} p(\mathbf{x}, y \neq g(\mathbf{x};\mathbf{w}))dyd\mathbf{x},$$

i.e. the *misclassification probability*. With the notations

$$X_{-1} = \{\mathbf{x} \in X \mid g(\mathbf{x};\mathbf{w}) < 0\}, \quad X_{+1} = \{\mathbf{x} \in X \mid g(\mathbf{x};\mathbf{w}) > 0\},$$

we can conclude further:

$$R(g(.;\mathbf{w})) = \int_{X_{-1}} p(\mathbf{x}, y = +1)d\mathbf{x} + \int_{X_{+1}} p(\mathbf{x}, y = -1)d\mathbf{x}$$

GENERALIZATION ERROR FOR BINARY CLASSIFICATION (2/3)

So, we get:

$$R(g(.;\mathbf{w})) = \int_{X_{-1}} p(y=+1 \mid \mathbf{x}) \cdot p(\mathbf{x}) d\mathbf{x} + \int_{X_{+1}} p(y=-1 \mid \mathbf{x}) \cdot p(\mathbf{x}) d\mathbf{x}$$

$$= \int_{X} \left\{ \begin{array}{ll} p(y=-1 \mid \mathbf{x}) & \text{if } g(\mathbf{x};\mathbf{w}) = +1 \\ p(y=+1 \mid \mathbf{x}) & \text{if } g(\mathbf{x};\mathbf{w}) = -1 \end{array} \right\} \cdot p(\mathbf{x}) d\mathbf{x}$$

Hence, we can infer an optimal classification function, the so-called *Bayes-optimal classifier*:

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } p(y=+1 \mid \mathbf{x}) > p(y=-1 \mid \mathbf{x}) \\ -1 & \text{if } p(y=-1 \mid \mathbf{x}) > p(y=+1 \mid \mathbf{x}) \end{cases}$$

$$= \text{sign}(p(y=+1 \mid \mathbf{x}) - p(y=-1 \mid \mathbf{x})) \quad (1)$$

GENERALIZATION ERROR FOR BINARY CLASSIFICATION (3/3)

The resulting minimal risk is

$$R_{\min} = \int_X \min(p(\mathbf{x}, y = -1), p(\mathbf{x}, y = +1))d\mathbf{x}$$

$$= \int_X \min(p(y = -1 \mid \mathbf{x}), p(y = +1 \mid \mathbf{x})) \cdot p(\mathbf{x})d\mathbf{x}$$

Obviously, for non-overlapping classes, i.e. $\min(p(y = -1 \mid \mathbf{x}), p(y = +1 \mid \mathbf{x})) = 0$, the minimal risk is zero and the optimal classification function is

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } p(y = +1 \mid \mathbf{x}) > 0, \\ -1 & \text{if } p(y = -1 \mid \mathbf{x}) > 0. \end{cases}$$

MINIMIZING RISK FOR A GAUSSIAN CLASSIFICATION TASK (1/4)

Assume that both negative and positive class are distributed according to d-variate normal distributions, i.e., $p(\mathbf{x} \mid y = -1)$ is $N(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{-1})$-distributed and $p(\mathbf{x} \mid y = +1)$ is $N(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1})$-distributed.

Note that the distribution density of a d-variate $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$-distributed random variable is given as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \cdot \sqrt{\det \boldsymbol{\Sigma}}} \cdot \exp\left(-\frac{1}{2} \cdot (\mathbf{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})^T\right)$$

MINIMIZING RISK FOR A GAUSSIAN CLASSIFICATION TASK (2/4)

Using (1), we can infer

$$g(\mathbf{x}) = \text{sign}(\bar{g}(\mathbf{x})) = \text{sign}(\hat{g}(\mathbf{x}))$$

where

$$\begin{aligned}\bar{g}(\mathbf{x}) &= p(y=+1 \mid \mathbf{x}) - p(y=-1 \mid \mathbf{x}) \\ &= \frac{1}{p(\mathbf{x})} \cdot \Big(p(\mathbf{x} \mid y=+1) \cdot p(y=+1) \\ &\qquad - p(\mathbf{x} \mid y=-1) \cdot p(y=-1) \Big)\end{aligned}$$

$$\hat{g}(\mathbf{x}) = \ln p(\mathbf{x} \mid y=+1) - \ln p(\mathbf{x} \mid y=-1) + \ln p(y=+1) - \ln p(y=-1)$$

$\bar{g}$ and $\hat{g}$ are called *discriminant functions*.

MINIMIZING RISK FOR A GAUSSIAN CLASSIFICATION TASK (3/4)

Determining an optimal discriminant function:

$$
\begin{aligned}
\hat{g}(\mathbf{x}) &= -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{+1})\boldsymbol{\Sigma}_{+1}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{+1})^T - \tfrac{d}{2}\ln 2\pi - \tfrac{1}{2}\ln\det\boldsymbol{\Sigma}_{+1} + \ln p(y=+1) \\
&\quad +\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{-1})\boldsymbol{\Sigma}_{-1}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{-1})^T + \tfrac{d}{2}\ln 2\pi + \tfrac{1}{2}\ln\det\boldsymbol{\Sigma}_{-1} - \ln p(y=-1) \\
&= -\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{+1})\boldsymbol{\Sigma}_{+1}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{+1})^T - \tfrac{1}{2}\ln\det\boldsymbol{\Sigma}_{+1} + \ln p(y=+1) \\
&\quad +\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{-1})\boldsymbol{\Sigma}_{-1}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{-1})^T + \tfrac{1}{2}\ln\det\boldsymbol{\Sigma}_{-1} - \ln p(y=-1) \\
&= -\tfrac{1}{2}\mathbf{x}\overbrace{(\boldsymbol{\Sigma}_{+1}^{-1}-\boldsymbol{\Sigma}_{-1}^{-1})}^{=\mathbf{A}}\mathbf{x}^T + \overbrace{(\boldsymbol{\mu}_{+1}\boldsymbol{\Sigma}_{+1}^{-1}-\boldsymbol{\mu}_{-1}\boldsymbol{\Sigma}_{-1}^{-1})}^{=\mathbf{b}}\mathbf{x}^T \\
&\quad \left.\begin{aligned} &-\tfrac{1}{2}\boldsymbol{\mu}_{+1}\boldsymbol{\Sigma}_{+1}^{-1}\boldsymbol{\mu}_{+1}^T + \tfrac{1}{2}\boldsymbol{\mu}_{-1}\boldsymbol{\Sigma}_{-1}^{-1}\boldsymbol{\mu}_{-1}^T \\ &-\tfrac{1}{2}\ln\det\boldsymbol{\Sigma}_{+1} + \tfrac{1}{2}\ln\det\boldsymbol{\Sigma}_{-1} + \ln p(y=+1) - \ln p(y=-1) \end{aligned}\right\} = c \\
&= -\frac{1}{2}\mathbf{x}\mathbf{A}\mathbf{x}^T + \mathbf{b}\mathbf{x}^T + c
\end{aligned}
$$

MINIMIZING RISK FOR A GAUSSIAN CLASSIFICATION TASK (4/4)

Thus, the optimal classification border $\hat{g}(\mathbf{x}) = 0$ is a d-dimensional hyper-quadric $-\frac{1}{2}\mathbf{x}\mathbf{A}\mathbf{x}^T + \mathbf{b}\mathbf{x}^T + c = 0$.

In the special case $\mathbf{\Sigma}_{-1} = \mathbf{\Sigma}_{+1}$, we obtain $\mathbf{A} = 0$, i.e. the optimal classification border is a linear hyperplane (a separating line in the case $d = 2$).

GAUSSIAN CLASSIFICATION EXAMPLE #1 (1/6)

The data shown on Slide 9 were created according to the following distributions:

- $p(\mathbf{x} \mid y = +1)$ corresponds to a two-variate normal distribution with parameters

$$\boldsymbol{\mu}_{+1} = (0.3, 0.7) \qquad \boldsymbol{\Sigma}_{+1} = \begin{pmatrix} 0.011875 & 0.016238 \\ 0.016238 & 0.030625 \end{pmatrix}$$

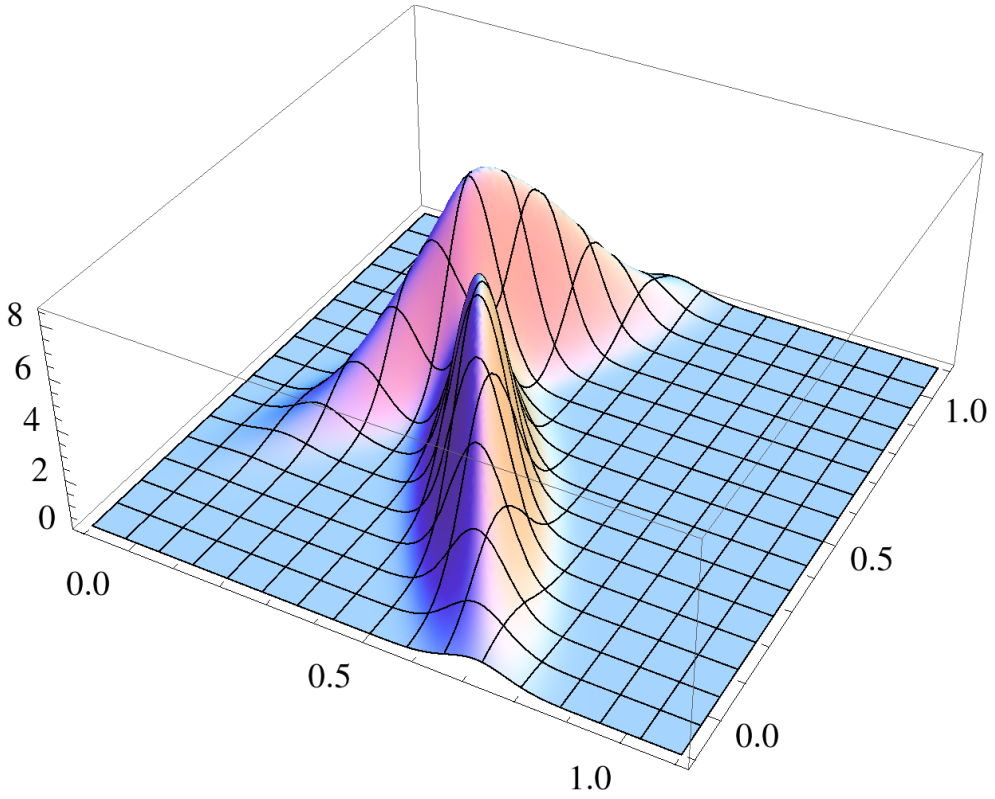
- $p(\mathbf{x} \mid y = -1)$ corresponds to a two-variate normal distribution with parameters

$$\boldsymbol{\mu}_{-1} = (0.5, 0.3) \qquad \boldsymbol{\Sigma}_{-1} = \begin{pmatrix} 0.011875 & -0.016238 \\ -0.016238 & 0.030625 \end{pmatrix}$$

- $p(y = +1) = \frac{55}{120} = 0.45833$, $p(y = -1) = \frac{65}{120} = 0.54167$

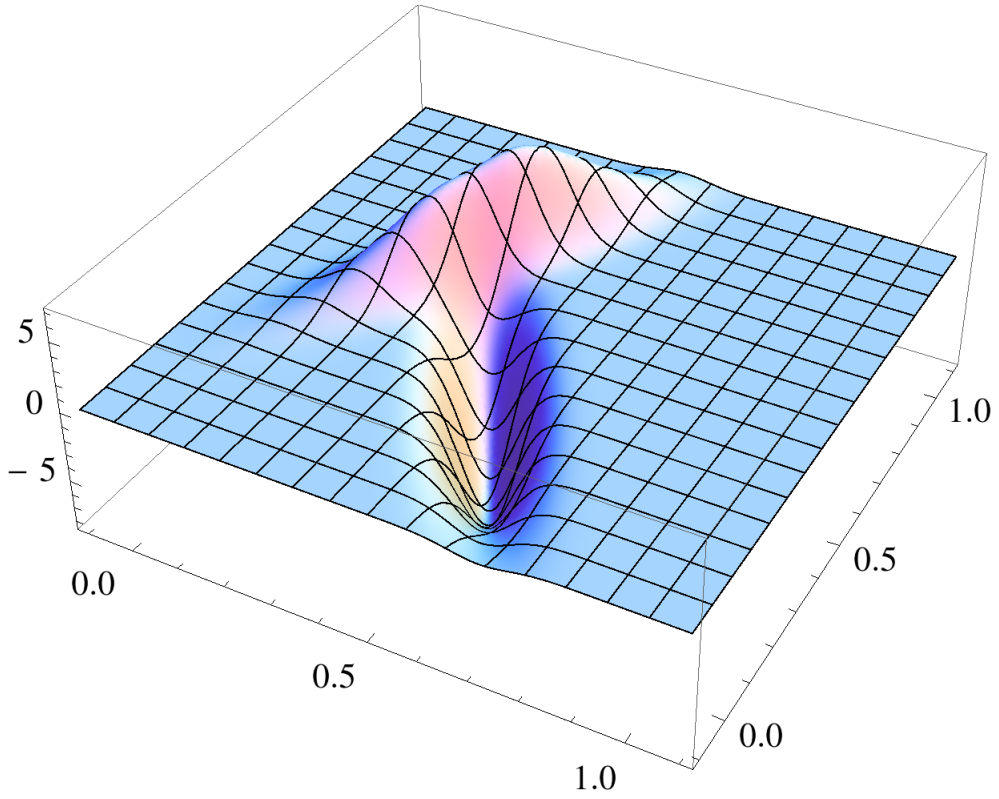
GAUSSIAN CLASSIFICATION EXAMPLE #1 (2/6)

$p(\mathbf{x}) = p(\mathbf{x} \mid y = -1) \cdot p(y = -1) + p(\mathbf{x} \mid y = +1) \cdot p(y = +1)$:



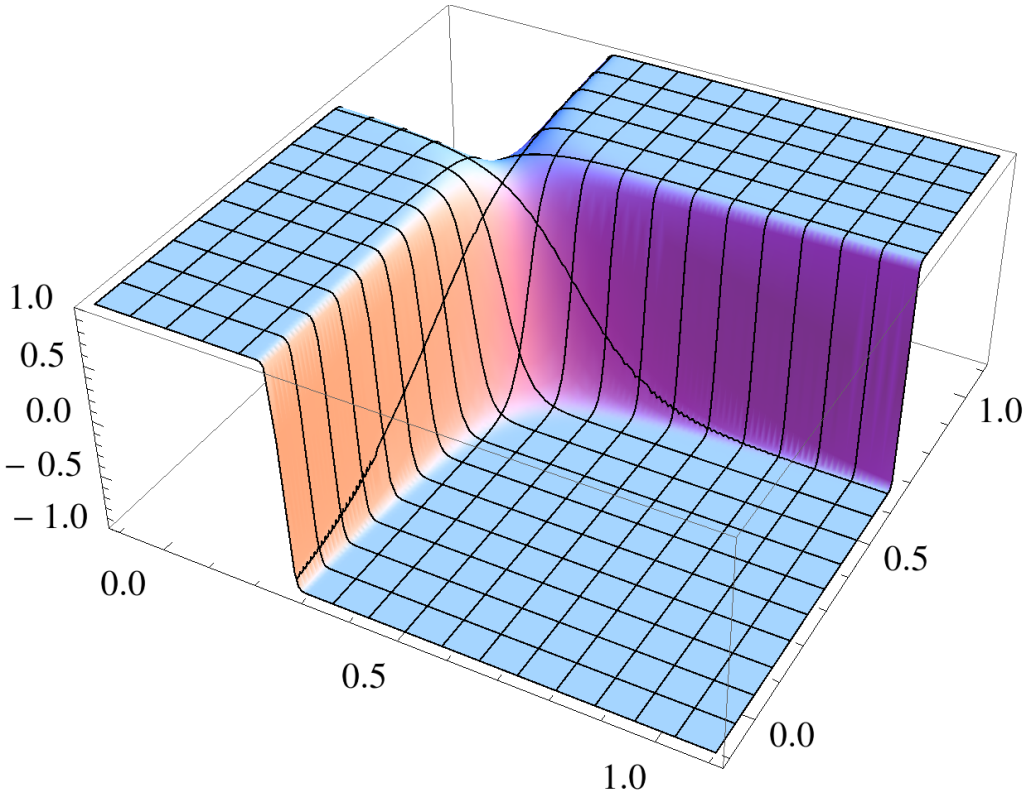
GAUSSIAN CLASSIFICATION EXAMPLE #1 (3/6)

$\tilde{g}(\mathbf{x}) = p(\mathbf{x} \mid y = +1) \cdot p(y = +1) - p(\mathbf{x} \mid y = -1) \cdot p(y = -1)$:



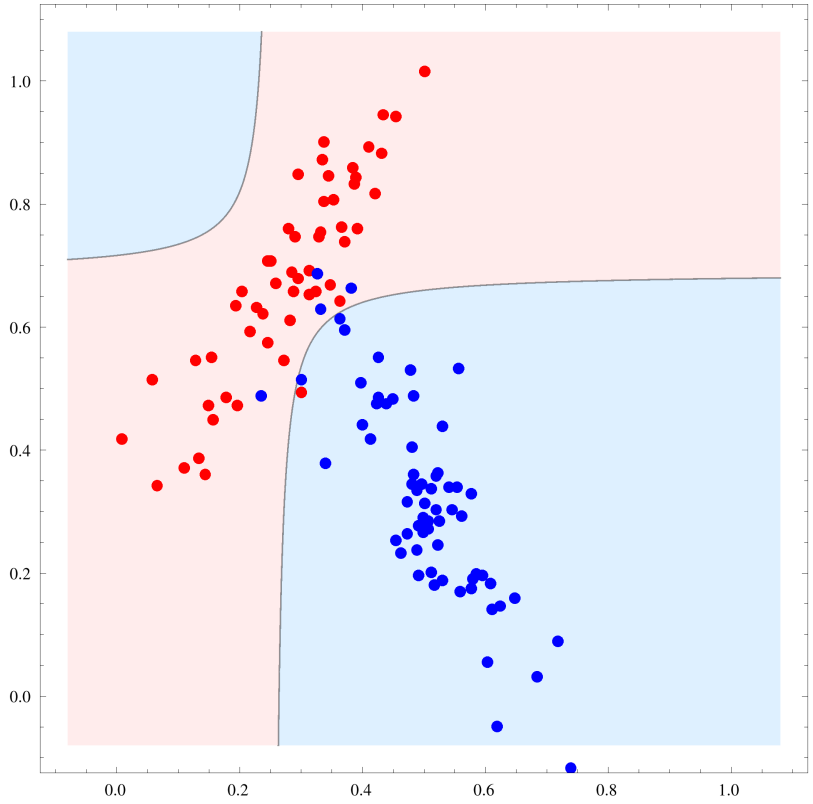
GAUSSIAN CLASSIFICATION EXAMPLE #1 (4/6)

Discriminant Function $\bar{g}(\mathbf{x}) = \tilde{g}(\mathbf{x})/p(\mathbf{x})$:



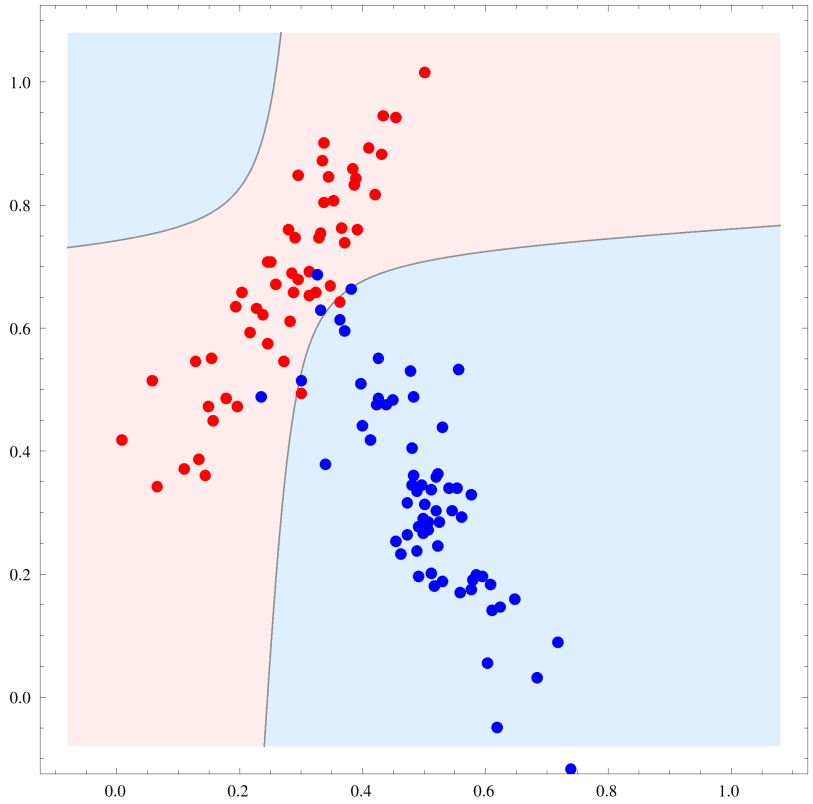
GAUSSIAN CLASSIFICATION EXAMPLE #1 (5/6)

Data + Optimal Decision Border:



GAUSSIAN CLASSIFICATION EXAMPLE #1 (6/6)

Data + Estimated Decision Border:



GAUSSIAN CLASSIFICATION EXAMPLE #2 (1/6)

Let us consider a data set created according to the following distributions:

- $p(\mathbf{x} \mid y = +1)$ corresponds to a two-variate normal distribution with parameters

$$\boldsymbol{\mu}_{+1} = (0.4, 0.8) \qquad \boldsymbol{\Sigma}_{+1} = \begin{pmatrix} 0.09 & 0.0 \\ 0.0 & 0.0049 \end{pmatrix}$$

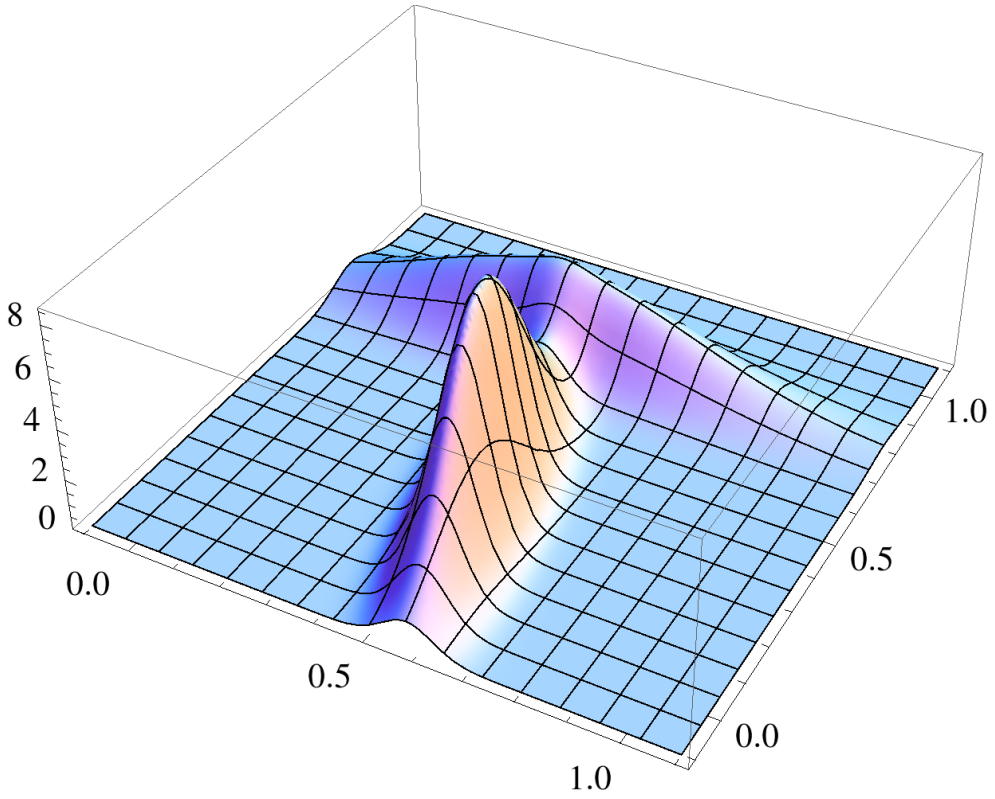
- $p(\mathbf{x} \mid y = -1)$ corresponds to a two-variate normal distribution with parameters

$$\boldsymbol{\mu}_{-1} = (0.5, 0.3) \qquad \boldsymbol{\Sigma}_{-1} = \begin{pmatrix} 0.00398011 & -0.00730159 \\ -0.00730159 & 0.0385199 \end{pmatrix}$$

- $p(y = +1) = \frac{55}{120} = 0.45833$, $p(y = -1) = \frac{65}{120} = 0.54167$

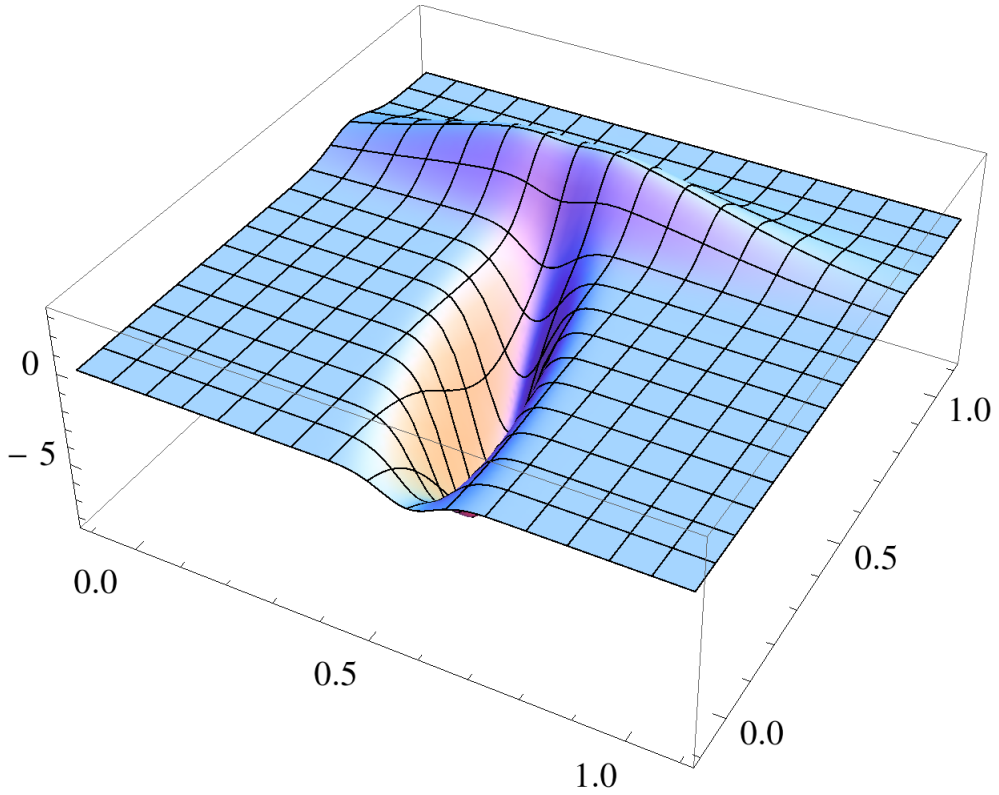
GAUSSIAN CLASSIFICATION EXAMPLE #2 (2/6)

$p(\mathbf{x}) = p(\mathbf{x} \mid y = -1) \cdot p(y = -1) + p(\mathbf{x} \mid y = +1) \cdot p(y = +1)$:



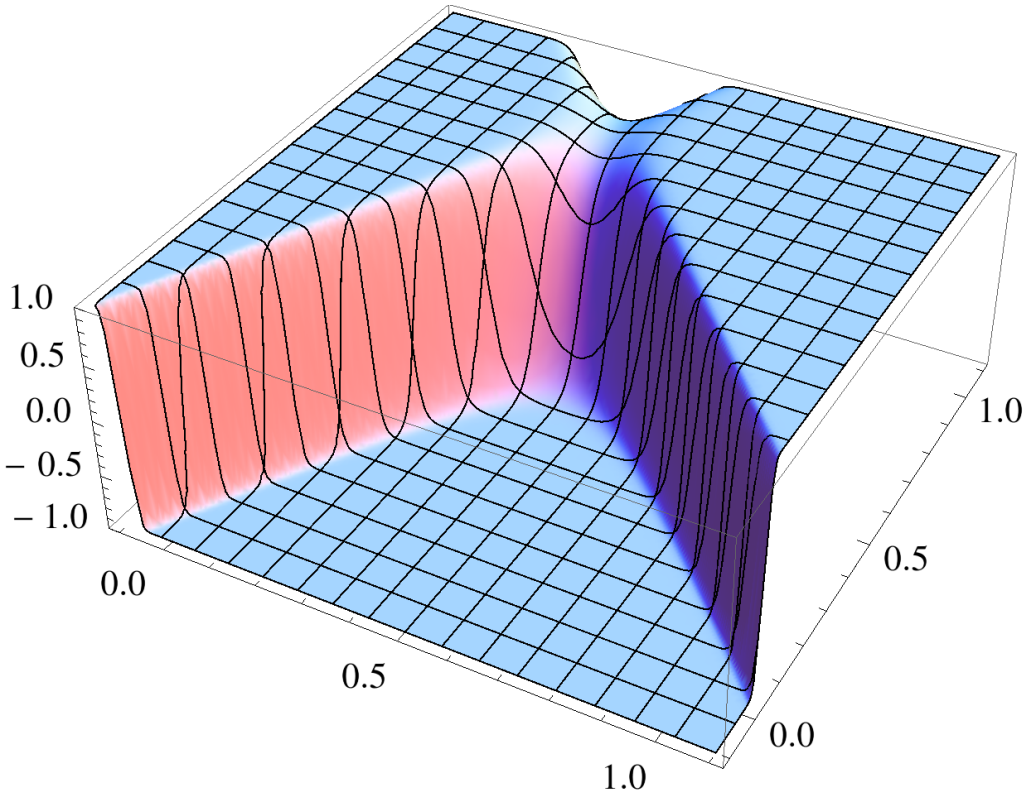
GAUSSIAN CLASSIFICATION EXAMPLE #2 (3/6)

$\tilde{g}(\mathbf{x}) = p(\mathbf{x} \mid y = +1) \cdot p(y = +1) - p(\mathbf{x} \mid y = -1) \cdot p(y = -1)$:



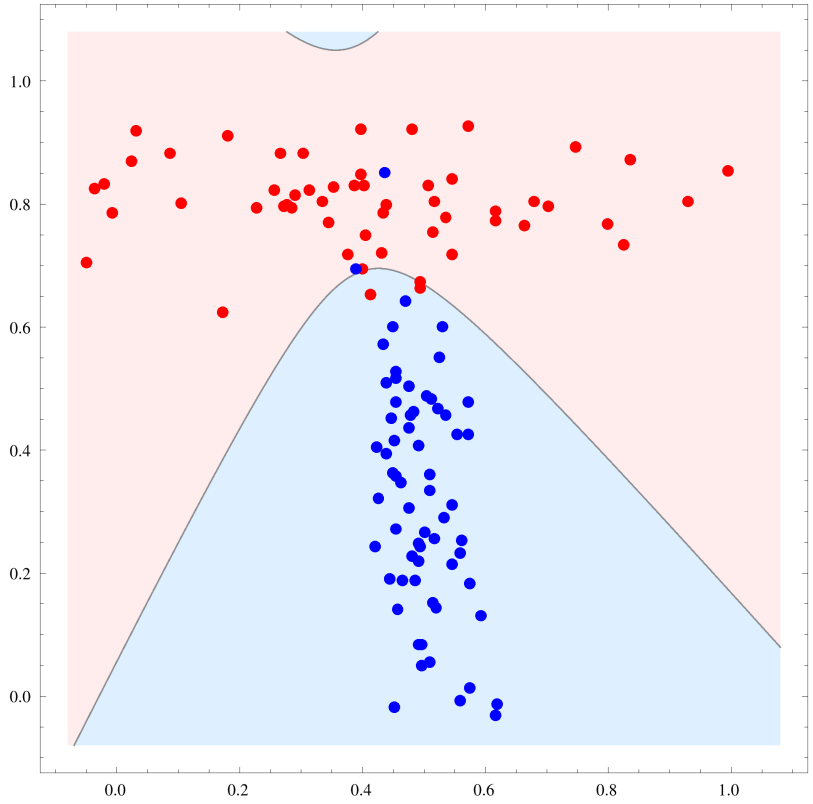
GAUSSIAN CLASSIFICATION EXAMPLE #2 (4/6)

Discriminant Function $\bar{g}(\mathbf{x}) = \tilde{g}(\mathbf{x})/p(\mathbf{x})$:



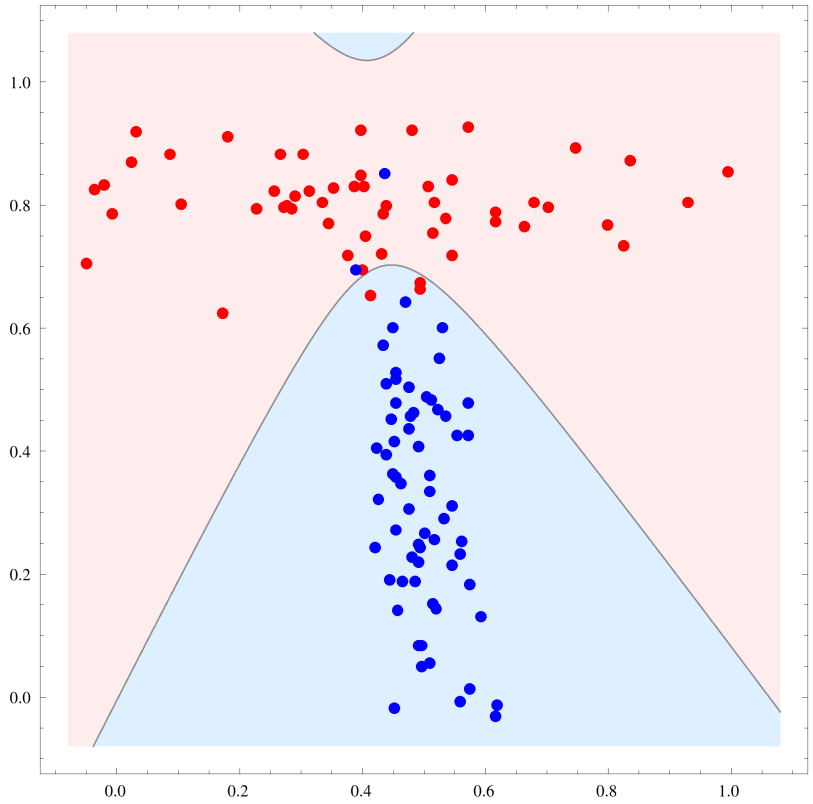
GAUSSIAN CLASSIFICATION EXAMPLE #2 (5/6)

Data + Optimal Decision Border:



GAUSSIAN CLASSIFICATION EXAMPLE #2 (6/6)

Data + Estimated Decision Border:



GAUSSIAN CLASSIFICATION EXAMPLE #3 (1/6)

Let us consider a data set created according to the following distributions:

- $p(\mathbf{x} \mid y = +1)$ corresponds to a two-variate normal distribution with parameters

$$\boldsymbol{\mu}_{+1} = (0.3, 0.7) \qquad \boldsymbol{\Sigma}_{+1} = \begin{pmatrix} 0.0016 & 0.0 \\ 0.0 & 0.0016 \end{pmatrix}$$

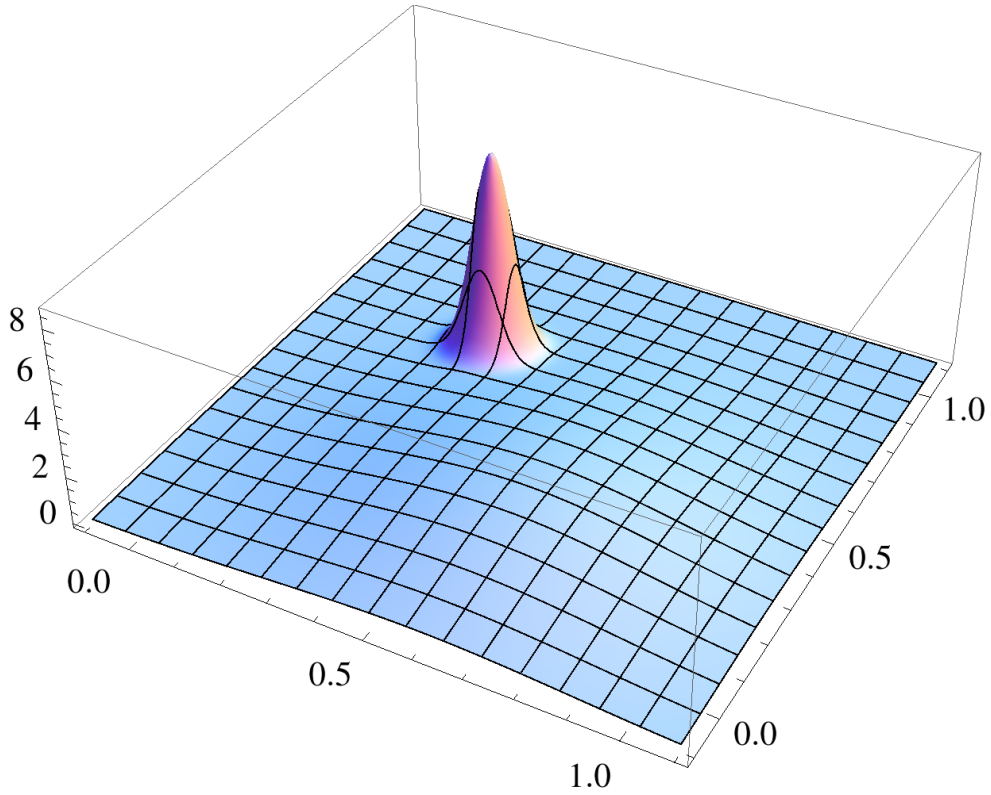
- $p(\mathbf{x} \mid y = -1)$ corresponds to a two-variate normal distribution with parameters

$$\boldsymbol{\mu}_{-1} = (0.6, 0.3) \qquad \boldsymbol{\Sigma}_{-1} = \begin{pmatrix} 0.09 & 0.0 \\ 0.0 & 0.09 \end{pmatrix}$$

- $p(y = +1) = \frac{1}{12} = 0.0833$, $p(y = -1) = \frac{11}{12} = 0.9167$

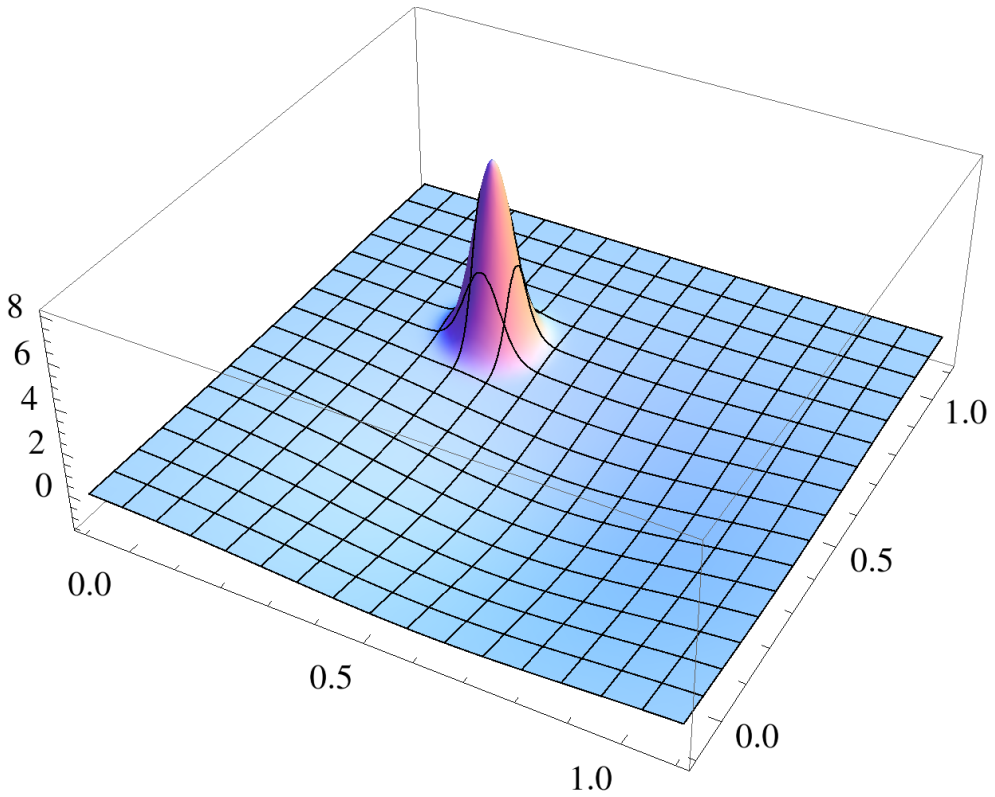
GAUSSIAN CLASSIFICATION EXAMPLE #3 (2/6)

$p(\mathbf{x}) = p(\mathbf{x} \mid y = -1) \cdot p(y = -1) + p(\mathbf{x} \mid y = +1) \cdot p(y = +1)$:



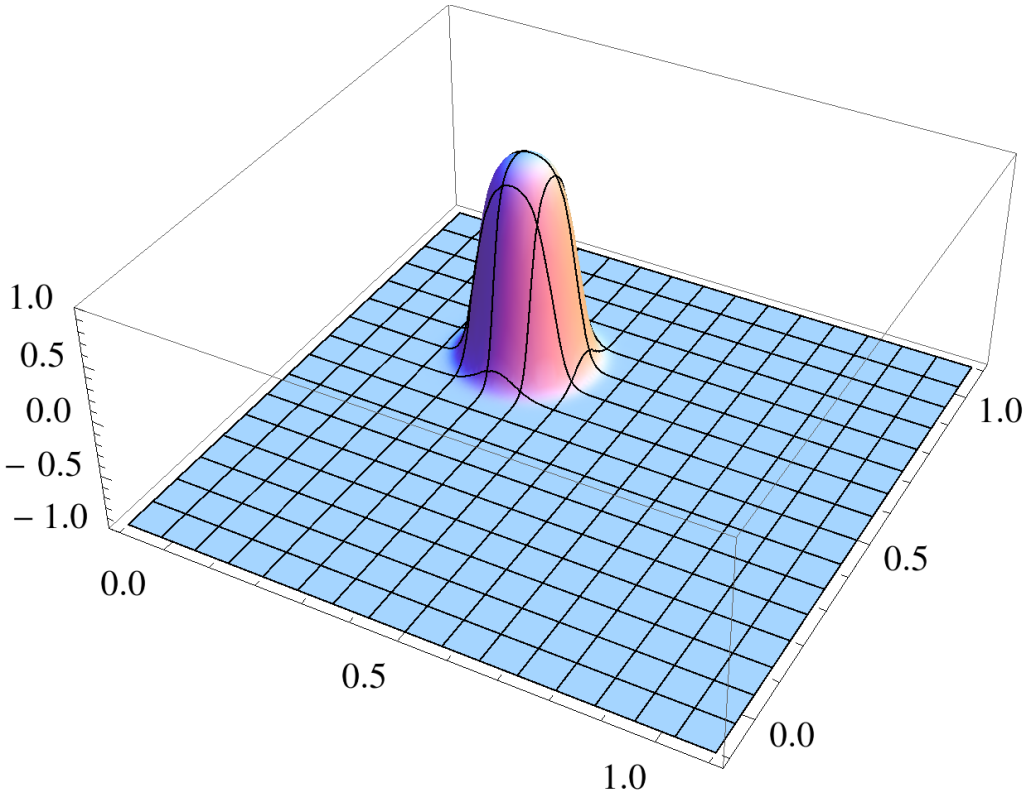
GAUSSIAN CLASSIFICATION EXAMPLE #3 (3/6)

$\tilde{g}(\mathbf{x}) = p(\mathbf{x} \mid y = +1) \cdot p(y = +1) - p(\mathbf{x} \mid y = -1) \cdot p(y = -1)$:



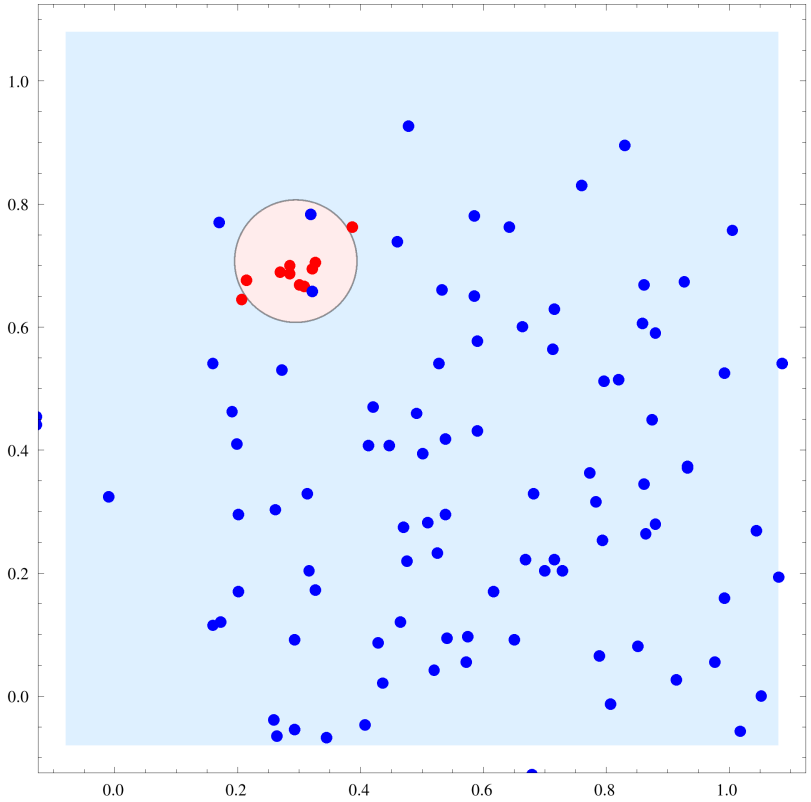
GAUSSIAN CLASSIFICATION EXAMPLE #3 (4/6)

Discriminant Function $\bar{g}(\mathbf{x}) = \tilde{g}(\mathbf{x})/p(\mathbf{x})$:



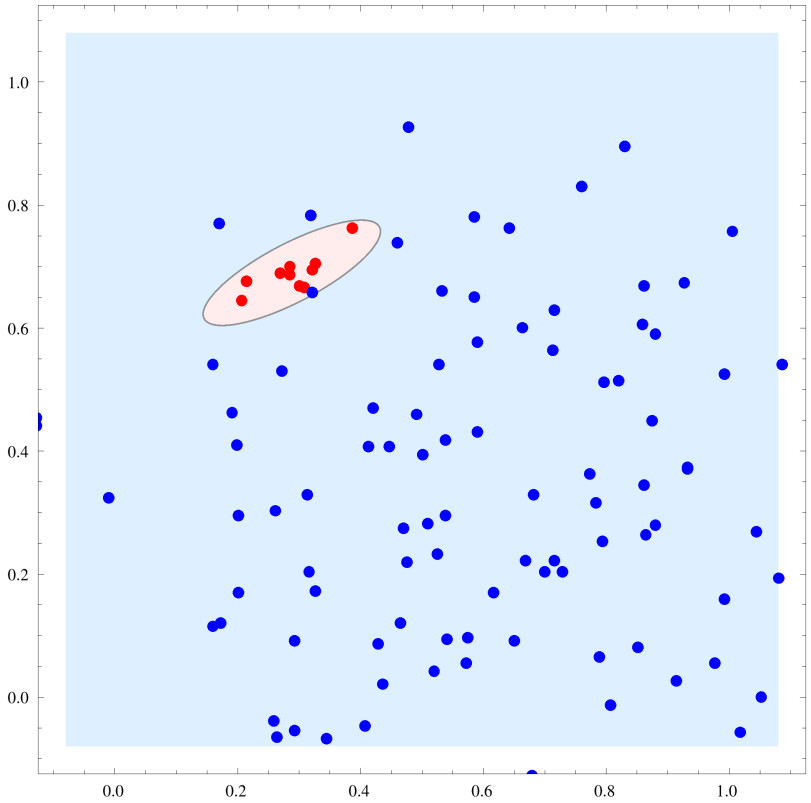
GAUSSIAN CLASSIFICATION EXAMPLE #3 (5/6)

Data + Optimal Decision Border:



GAUSSIAN CLASSIFICATION EXAMPLE #3 (6/6)

Data + Estimated Decision Border:



WHAT ABOUT PRACTICE?

- In practice, we hardly have any knowledge about $p(\mathbf{x}, y)$
- If we had, we could infer optimal prediction functions directly without using any machine learning method.

Therefore,

1. we have to estimate the prediction function with other methods;
2. we have to estimate the generalization error.

A BASIC CLASSIFIER: k-NEAREST NEIGHBOR

Suppose we have a labeled data set $\mathbf{Z}$ and a distance measure on the input space. Then the k-*nearest neighbor classifier* is defined as follows:

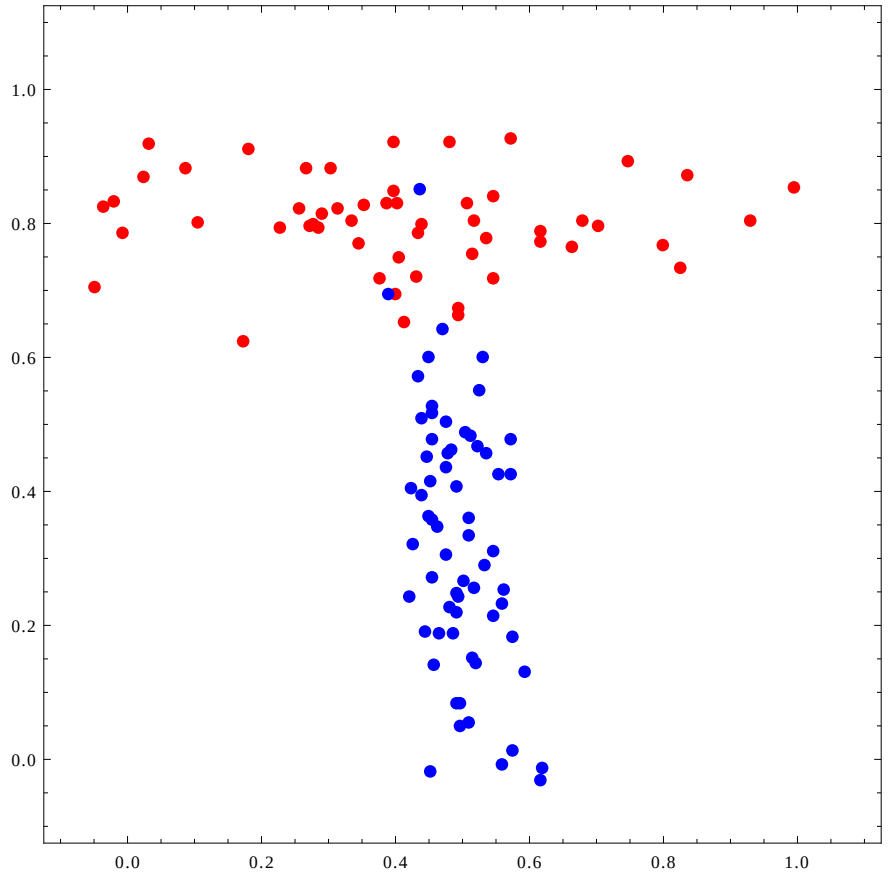
$$g_{k\text{-NN}}(\mathbf{x}; \mathbf{Z}) = \text{class that occurs most often among the } k \text{ samples that are closest to } \mathbf{x}$$

For $k = 1$, we simply call this *nearest neighbor classifier*:

$$g_{\text{NN}}(\mathbf{x}; \mathbf{Z}) = \text{class of the sample that is closest to } \mathbf{x}$$

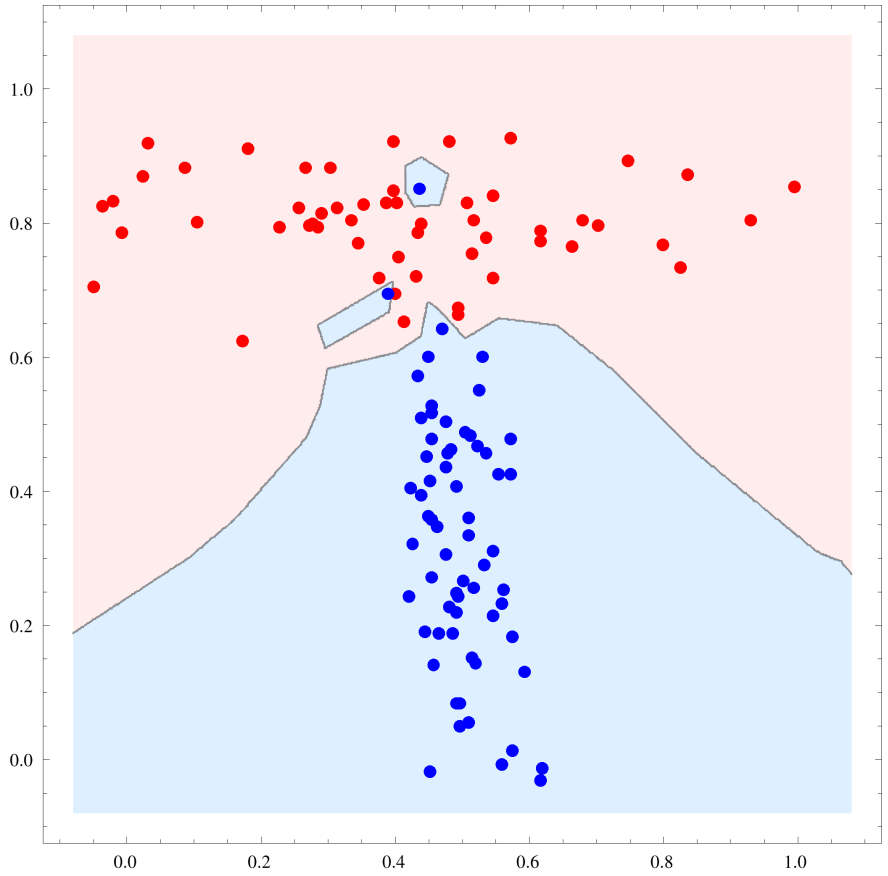
In case of ties, a special strategy has to be employed, e.g. random class assignment or the class with the larger number of samples.

k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #1



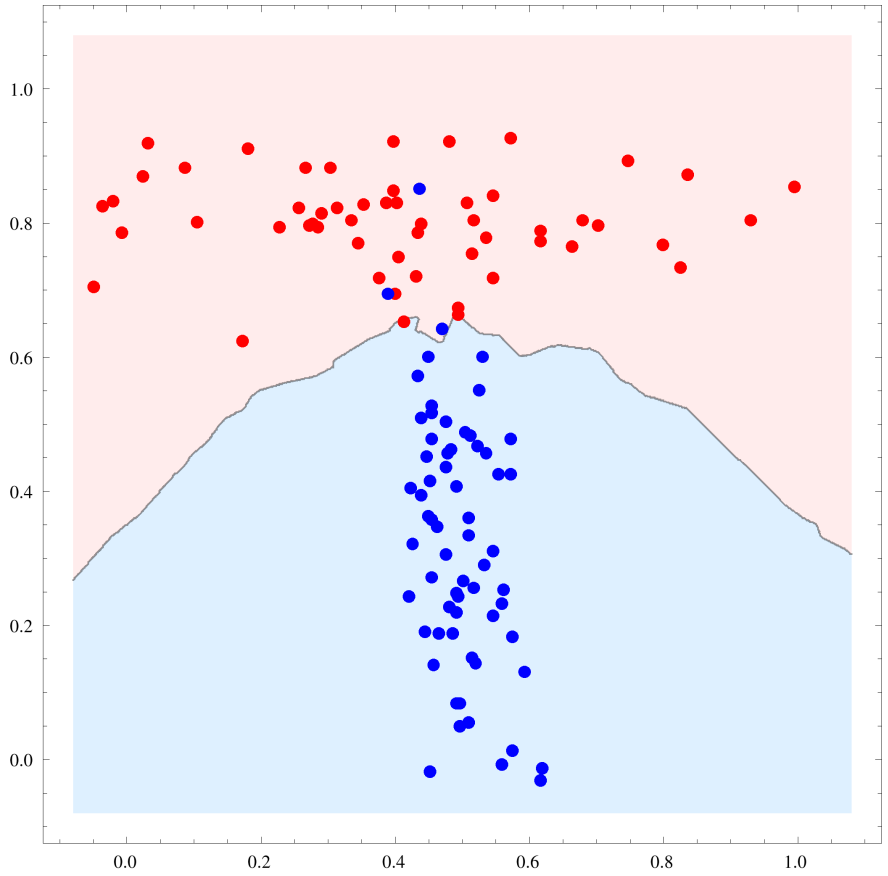
k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #1

$k = 1$:



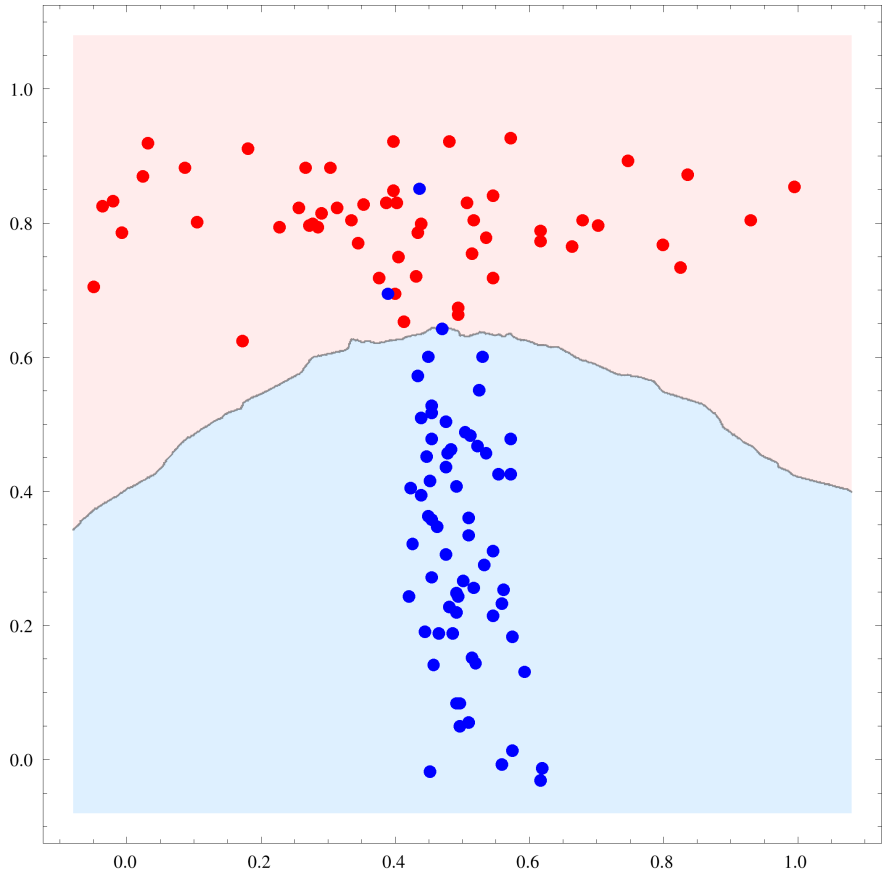
k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #1

$k = 5$:

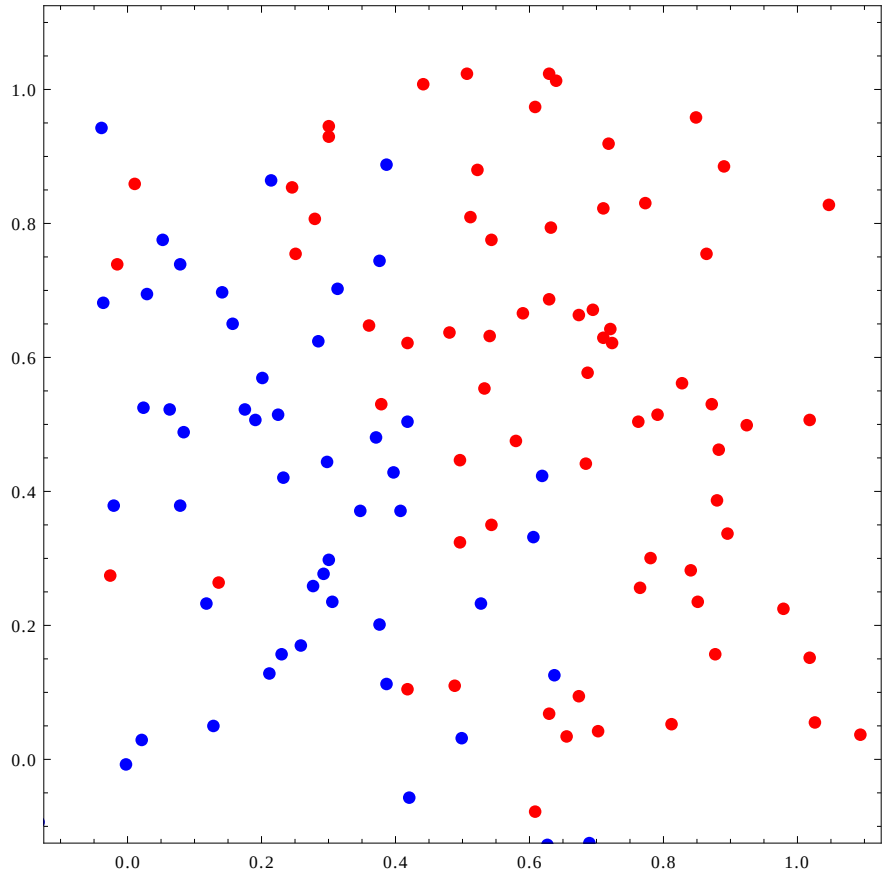


k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #1

$k = 13$:

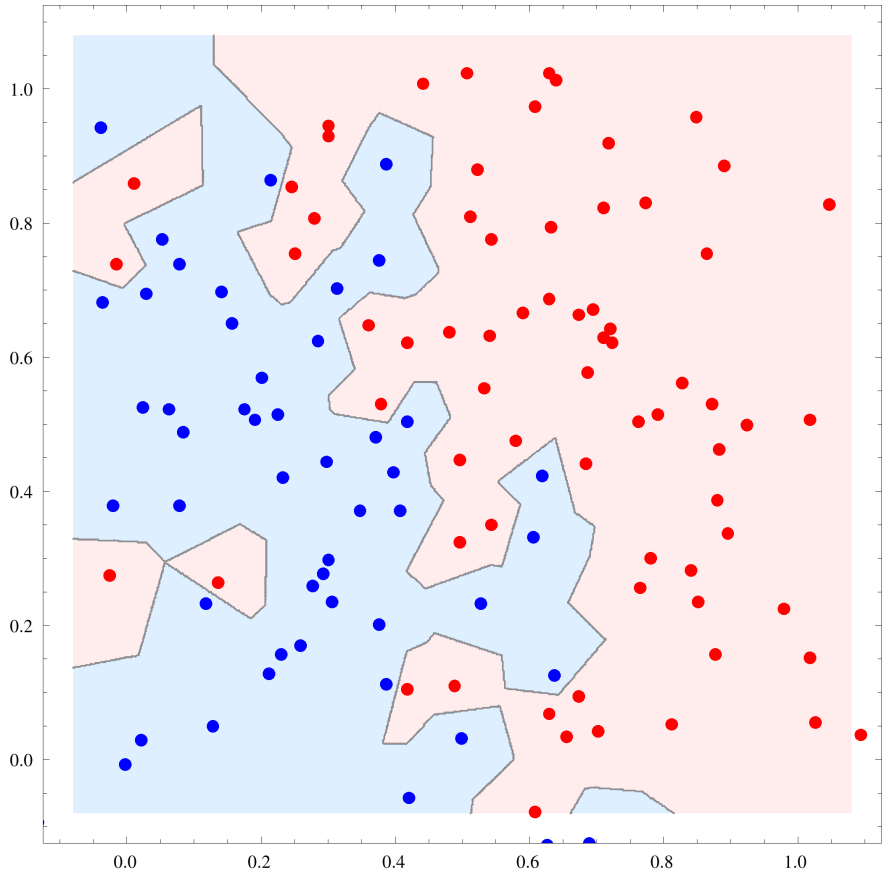


k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #2



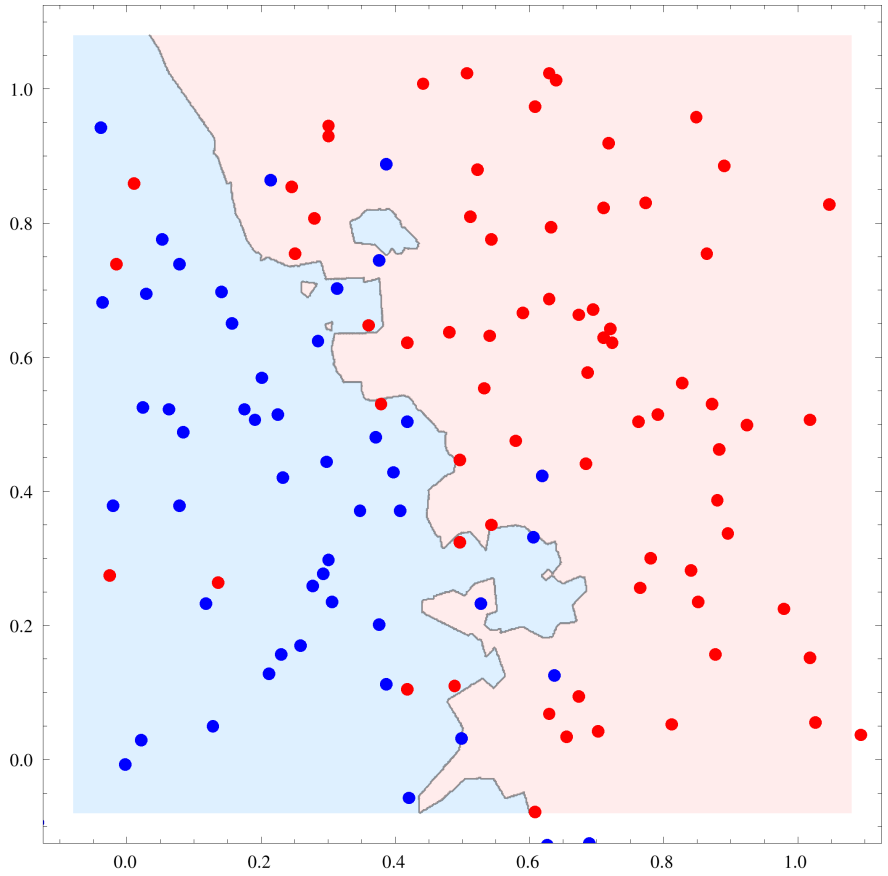
k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #2

$k = 1$:



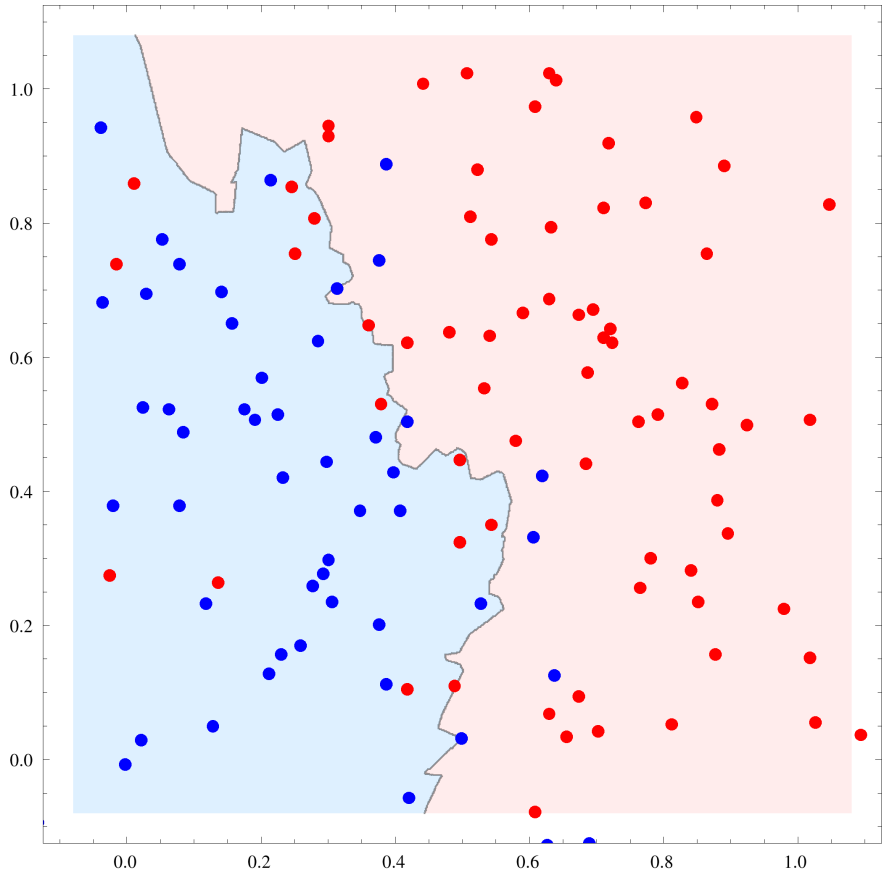
k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #2

$k = 5$:



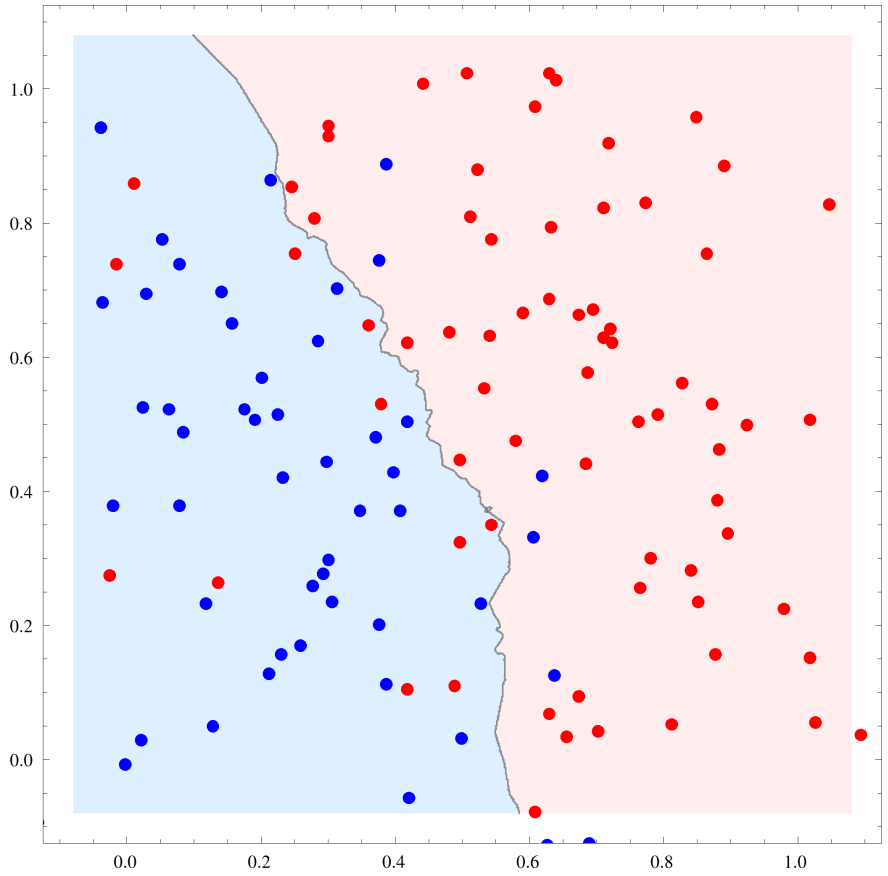
k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #2

$k = 13$:



k-NEAREST NEIGHBOR CLASSIFIER EXAMPLE #2

$k = 25$:



A BASIC NUMERICAL PREDICTOR: 1D LINEAR REGRESSION

Consider a data set $\mathbf{Z} = \{(x^i, y^i) \mid i = 1, \ldots, l\} \subseteq \mathbb{R}^2$ and a linear model

$$y = w_0 + w_1 \cdot x = g\big(x; \underbrace{(w_0, w_1)}_{\mathbf{w}}\big).$$

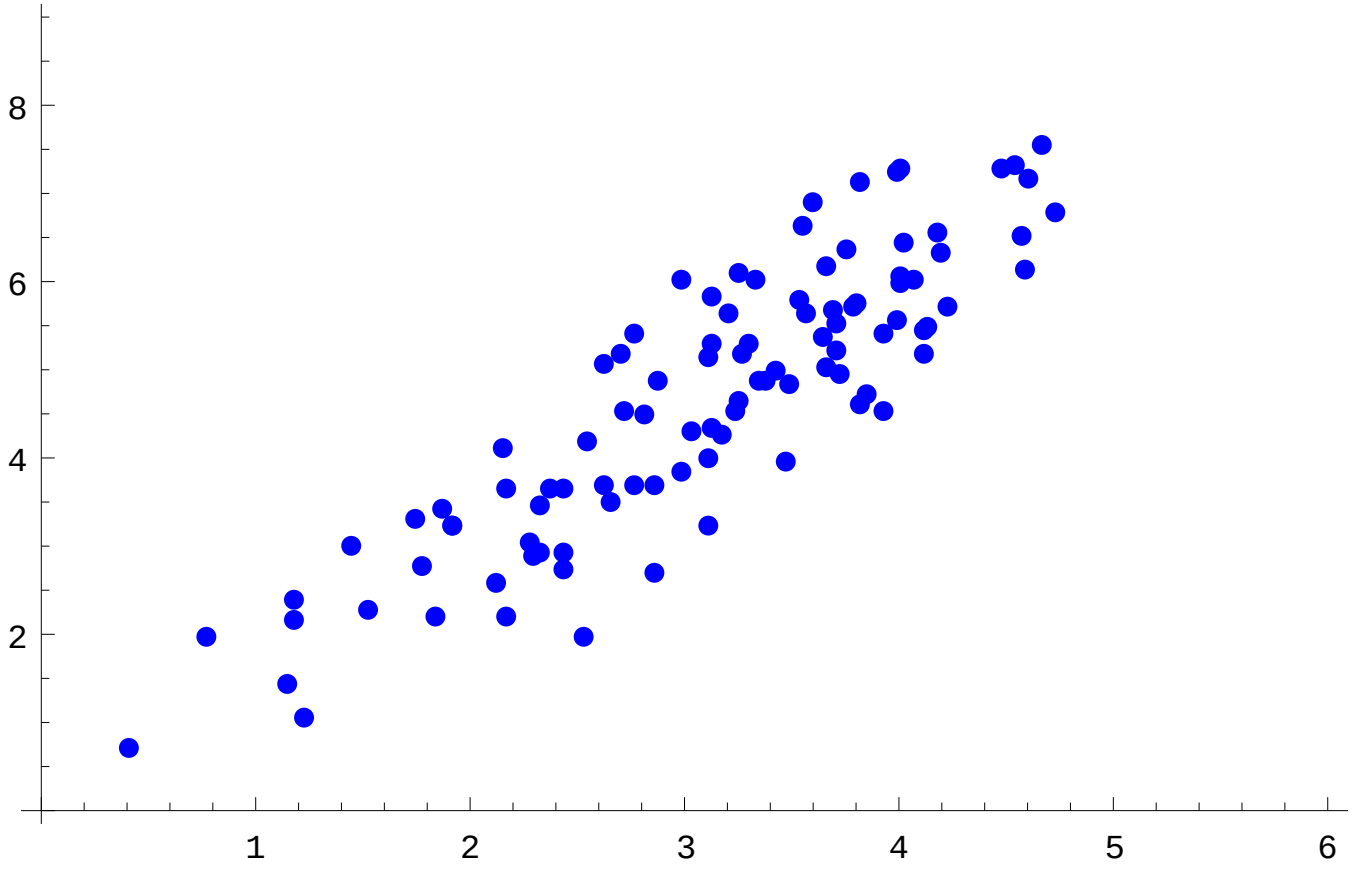
Suppose we want to find (w_0, w_1) such that the average quadratic loss,

$$Q(w_0, w_1) = \frac{1}{l} \sum_{i=1}^{l} \big(w_0 + w_1 \cdot x^i - y^i\big)^2 = \frac{1}{l} \sum_{i=1}^{l} \big(g(x^i; \mathbf{w}) - y^i\big)^2,$$

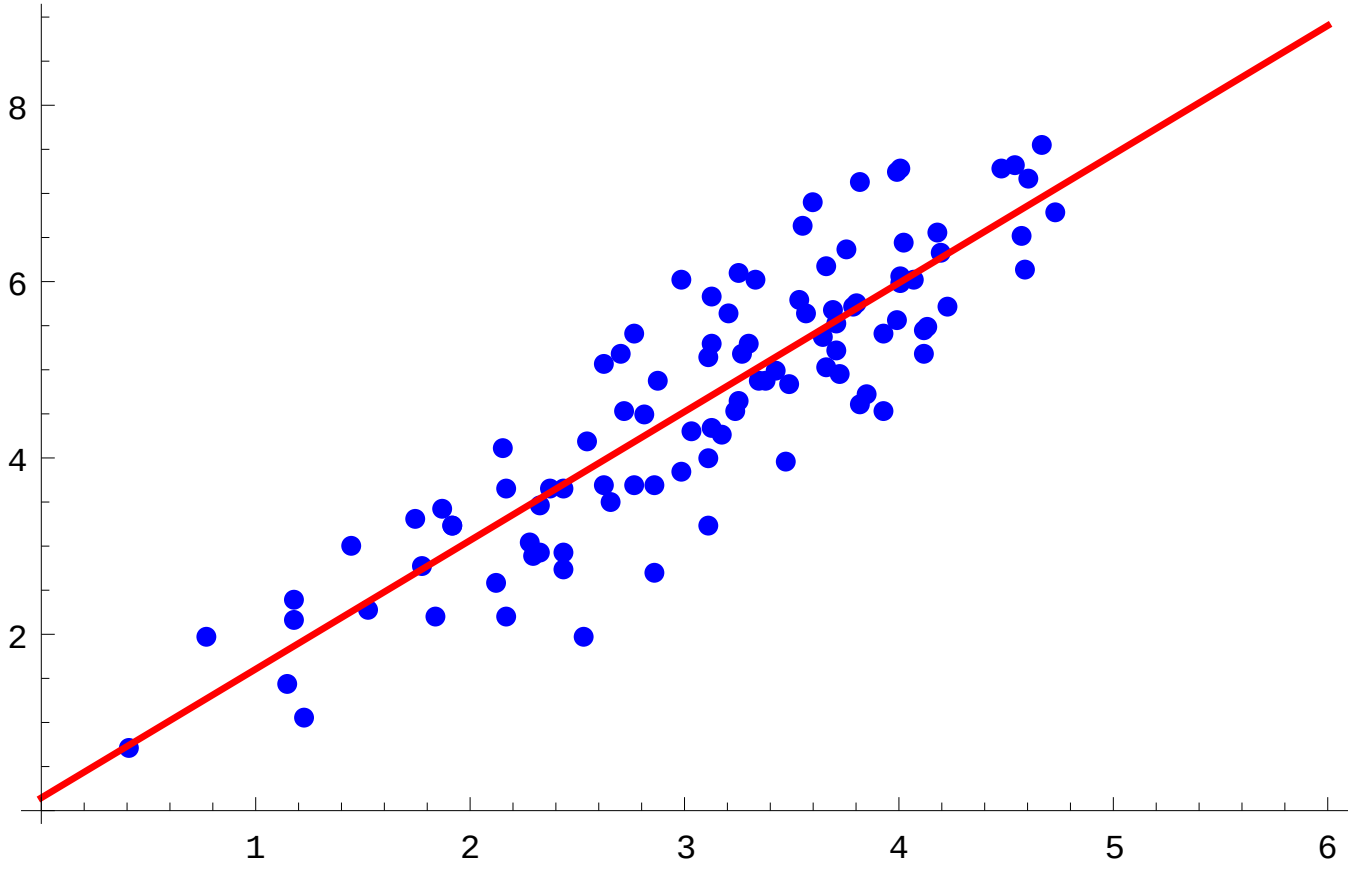
is minimized. Then the unique global solution is given as follows:

$$w_1 = \frac{\mathrm{Cov}(\mathbf{x}, \mathbf{y})}{\mathrm{Var}(\mathbf{x})} \qquad\qquad w_0 = \bar{y} - w_1 \cdot \bar{x}$$

LINEAR REGRESSION EXAMPLE #1



LINEAR REGRESSION EXAMPLE #1



LINEAR REGRESSION FOR MULTIPLE VARIABLES

Consider a data set $\mathbf{Z} = \{(\mathbf{x}^i, y^i) \mid i = 1, \ldots, l\}$ and a linear model

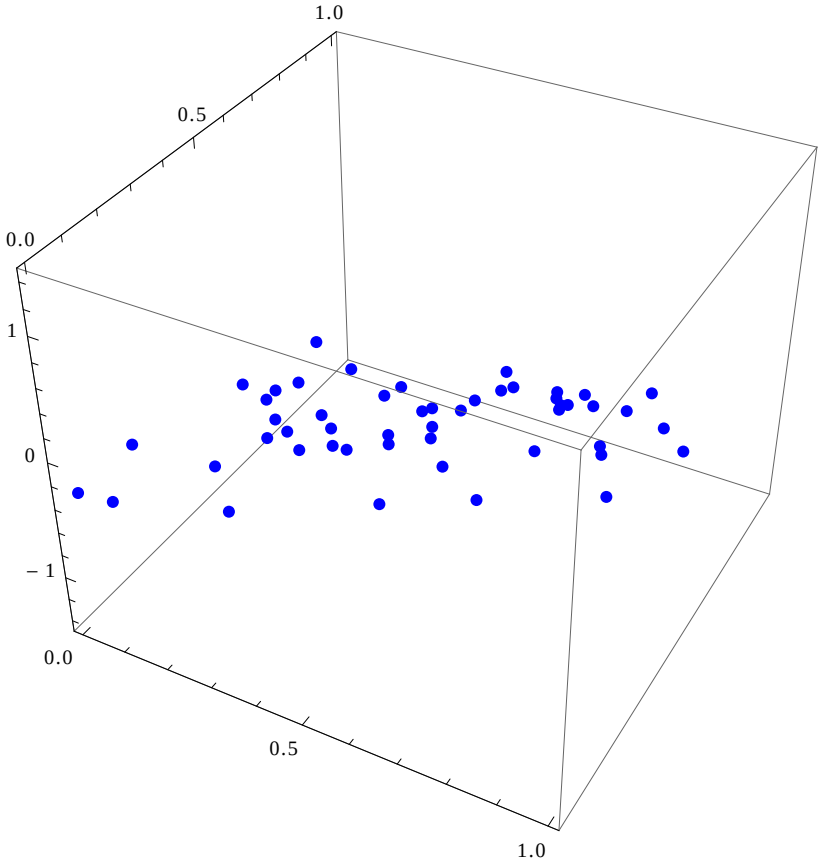
$$y = w_0 + w_1 \cdot x_1 + \cdots + w_d \cdot x_d = (1 \mid \mathbf{x}) \cdot \mathbf{w} = g\big(\mathbf{x}; \underbrace{(w_0, w_1, \ldots, w_d)}_{\mathbf{w}^T}\big).$$

Suppose we want to find $\mathbf{w} = (w_0, w_1, \ldots, w_d)^T$ such that the average quadratic loss is minimized. Then the unique global solution is given as

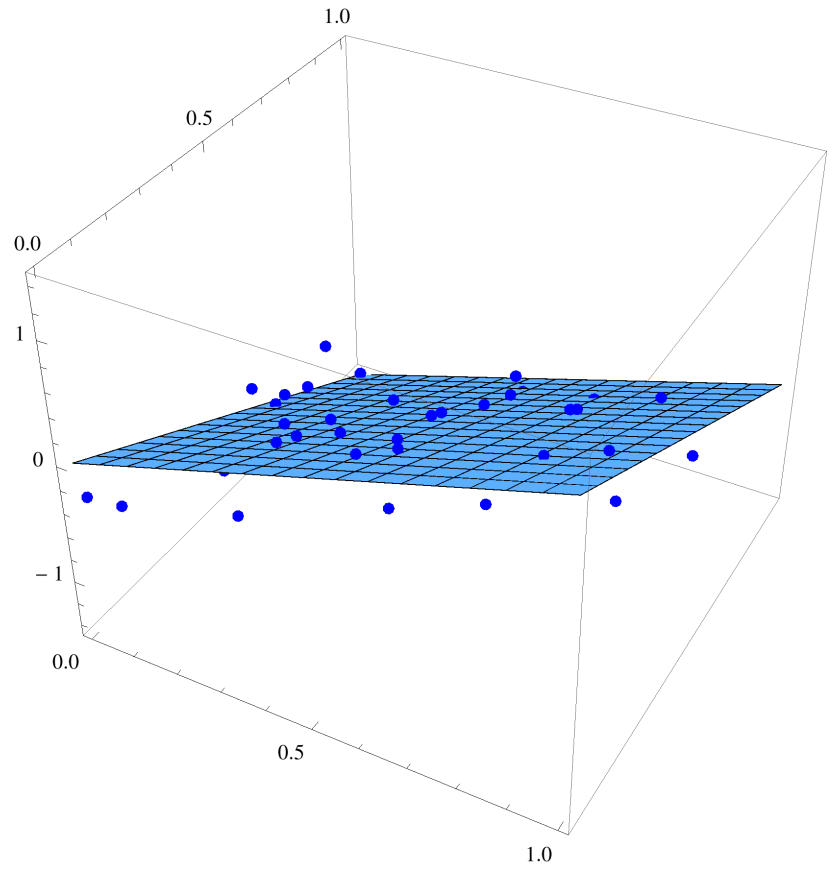
$$\mathbf{w} = \underbrace{\left(\tilde{\mathbf{X}}^T \cdot \tilde{\mathbf{X}}\right)^{-1} \cdot \tilde{\mathbf{X}}^T}_{\tilde{\mathbf{x}}^+} \cdot \mathbf{y},$$

where $\tilde{\mathbf{X}} = (\mathbf{1} \mid \mathbf{X})$.

LINEAR REGRESSION EXAMPLE #2



LINEAR REGRESSION EXAMPLE #2



POLYNOMIAL REGRESSION

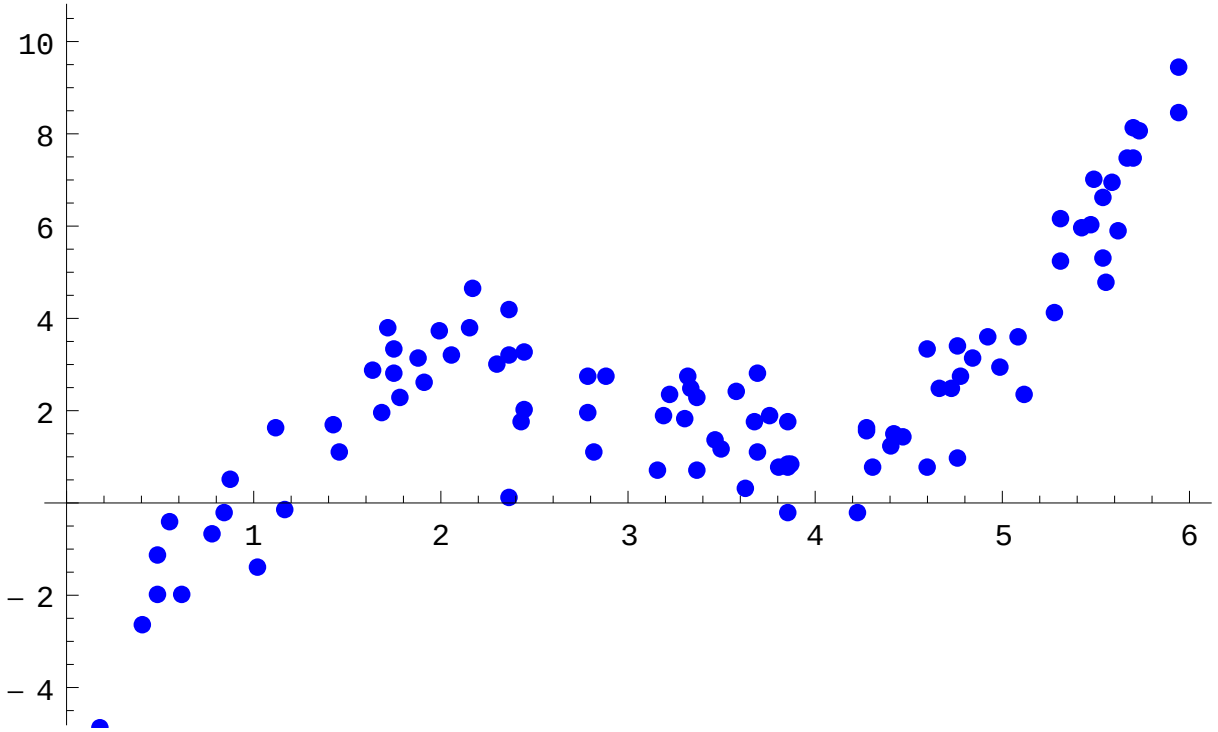
Consider a data set $\mathbf{Z} = \{(x^i, y^i) \mid i = 1, \ldots, l\}$ and a polynomial model of degree n

$$y = w_0 + w_1 \cdot x + w_2 \cdot x^2 + \cdots + w_n \cdot x^n = g\big(x; \underbrace{(w_0, w_1, \ldots, w_n)}_{\mathbf{w}^T}\big).$$

Suppose we want to find $\mathbf{w} = (w_0, w_1, \ldots, w_n)^T$ such that the average quadratic loss is minimized. Then the unique global solution is given as follows:

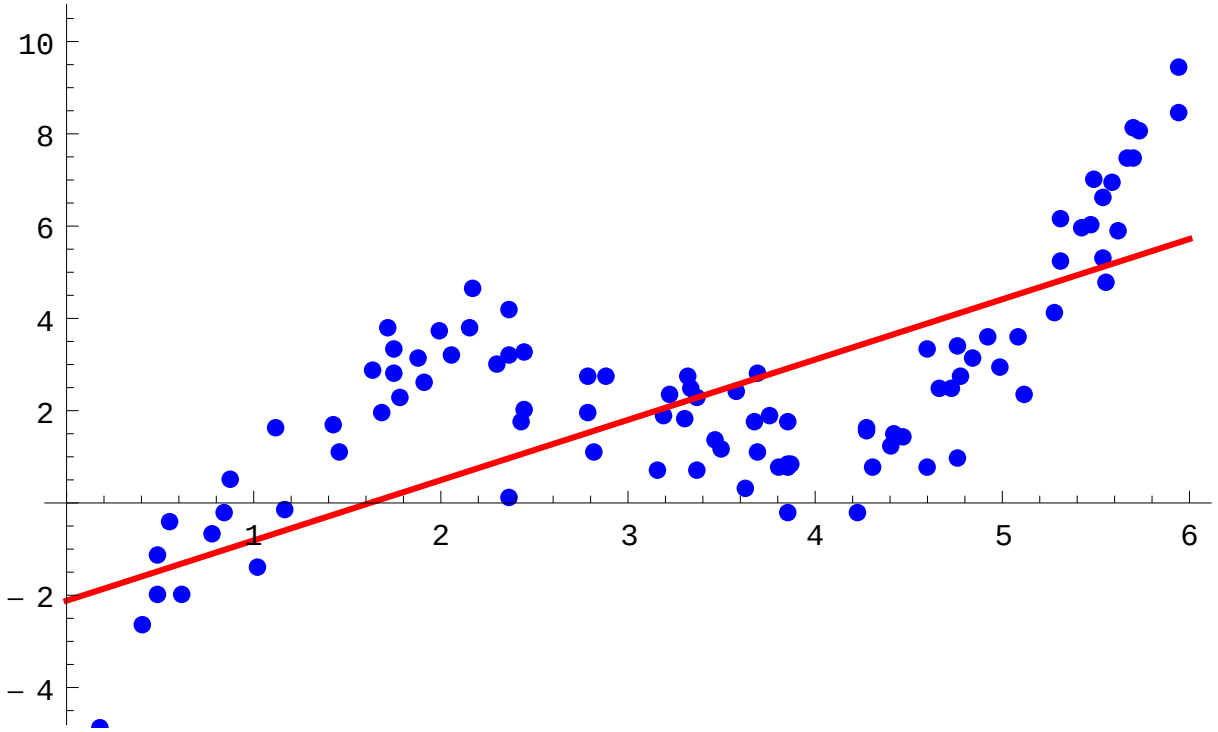
$$\mathbf{w} = \underbrace{\left(\tilde{\mathbf{X}}^T \cdot \tilde{\mathbf{X}}\right)^{-1} \cdot \tilde{\mathbf{X}}^T}_{\tilde{\mathbf{x}}^+} \cdot \mathbf{y} \quad \text{with} \quad \tilde{\mathbf{X}} = (\mathbf{1} \mid \mathbf{x} \mid \mathbf{x}^2 \mid \cdots \mid \mathbf{x}^n)$$

POLYNOMIAL REGRESSION EXAMPLE



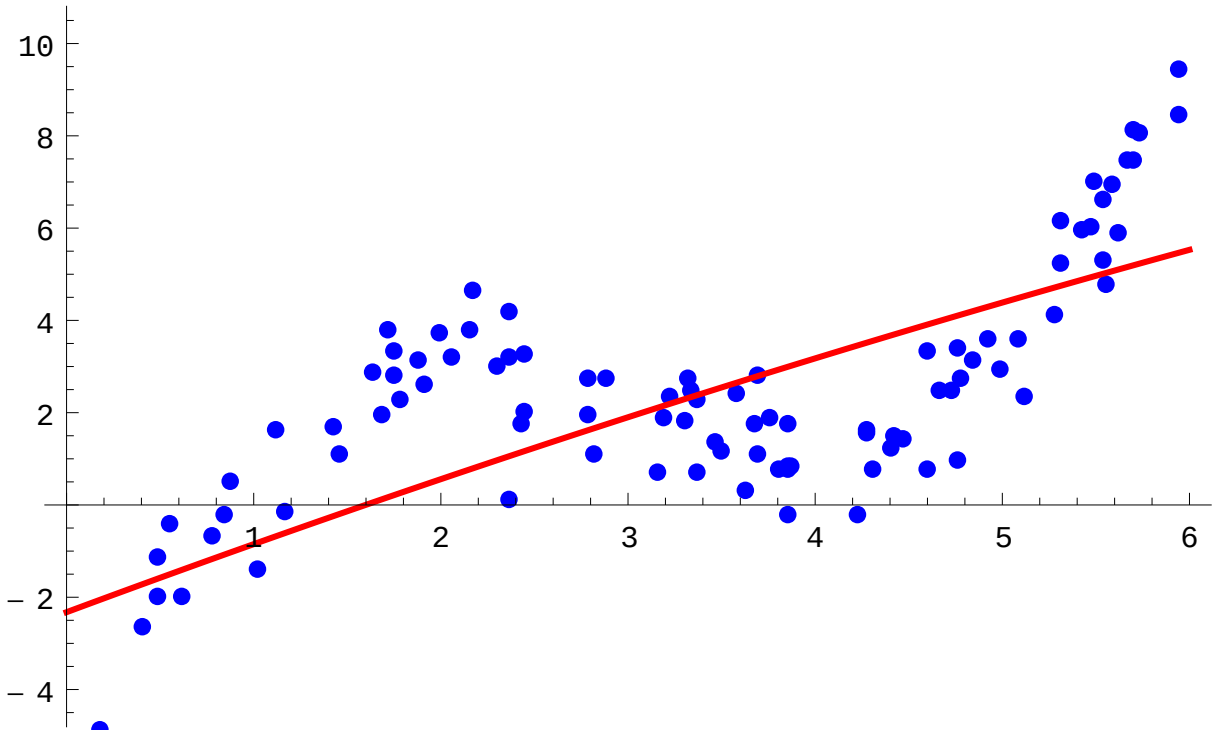
POLYNOMIAL REGRESSION EXAMPLE

$n = 1$:



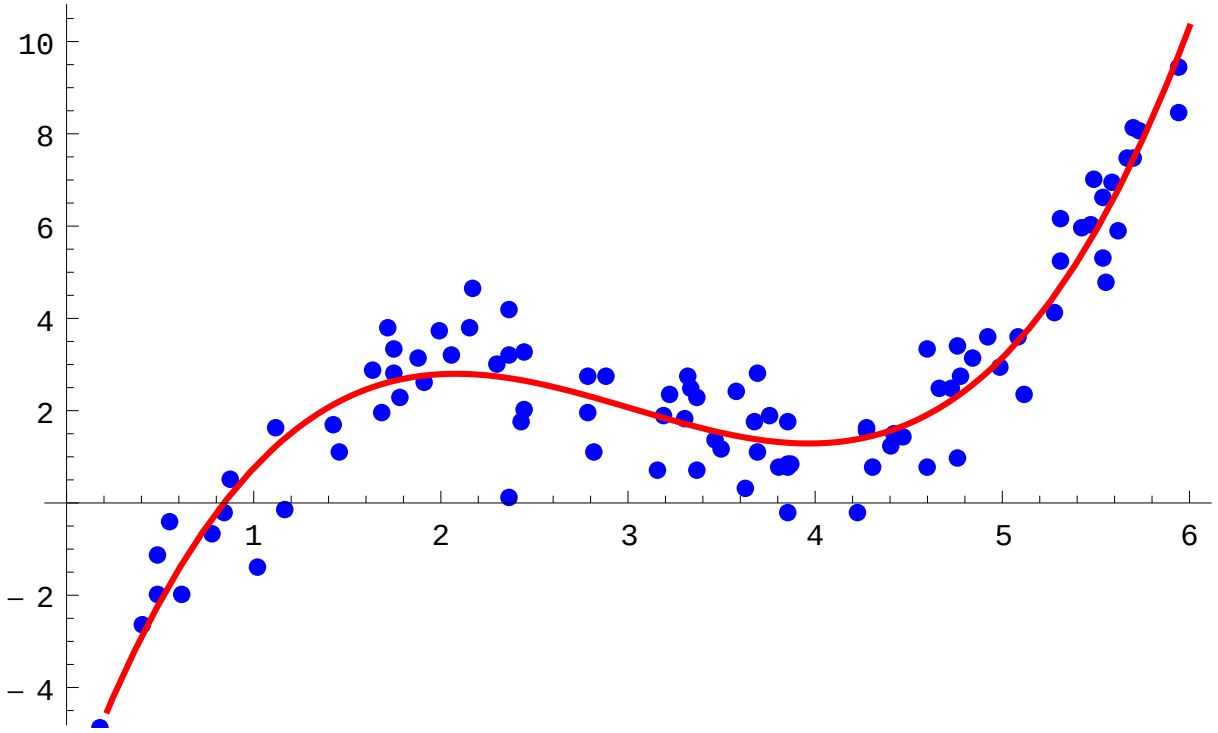
POLYNOMIAL REGRESSION EXAMPLE

$n = 2$:



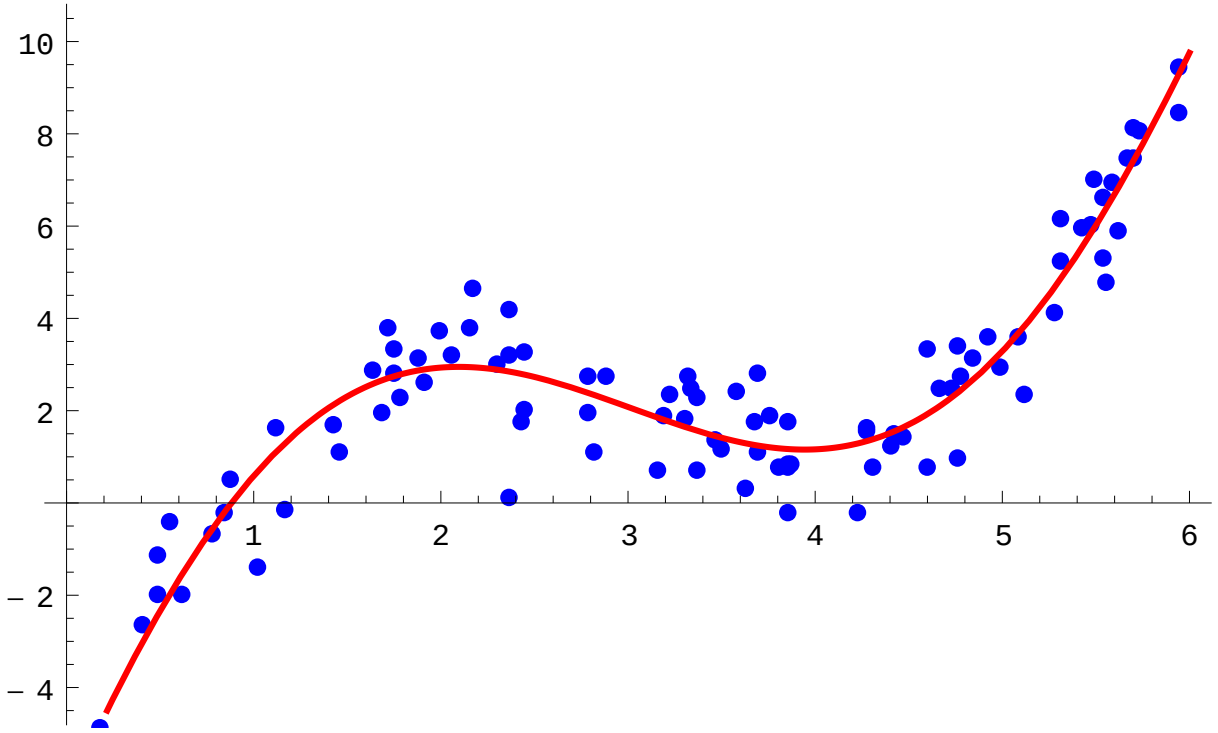
POLYNOMIAL REGRESSION EXAMPLE

$n = 3$:



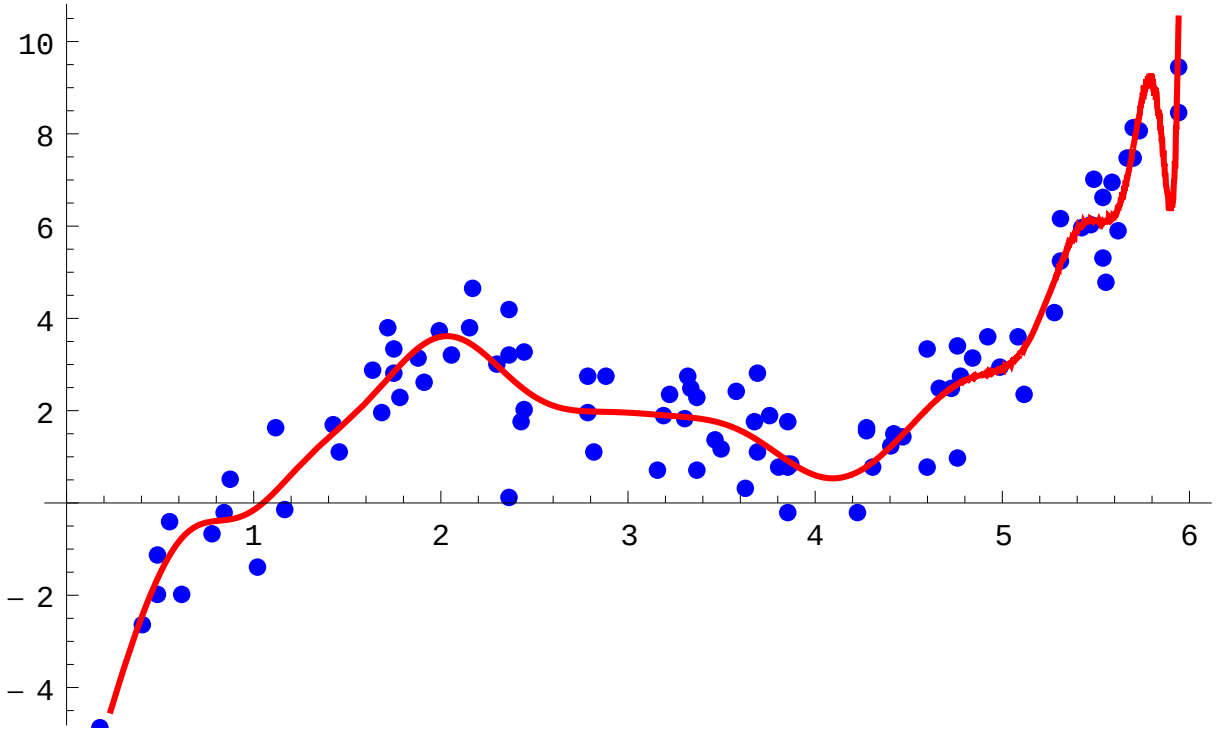
POLYNOMIAL REGRESSION EXAMPLE

$n = 5$:



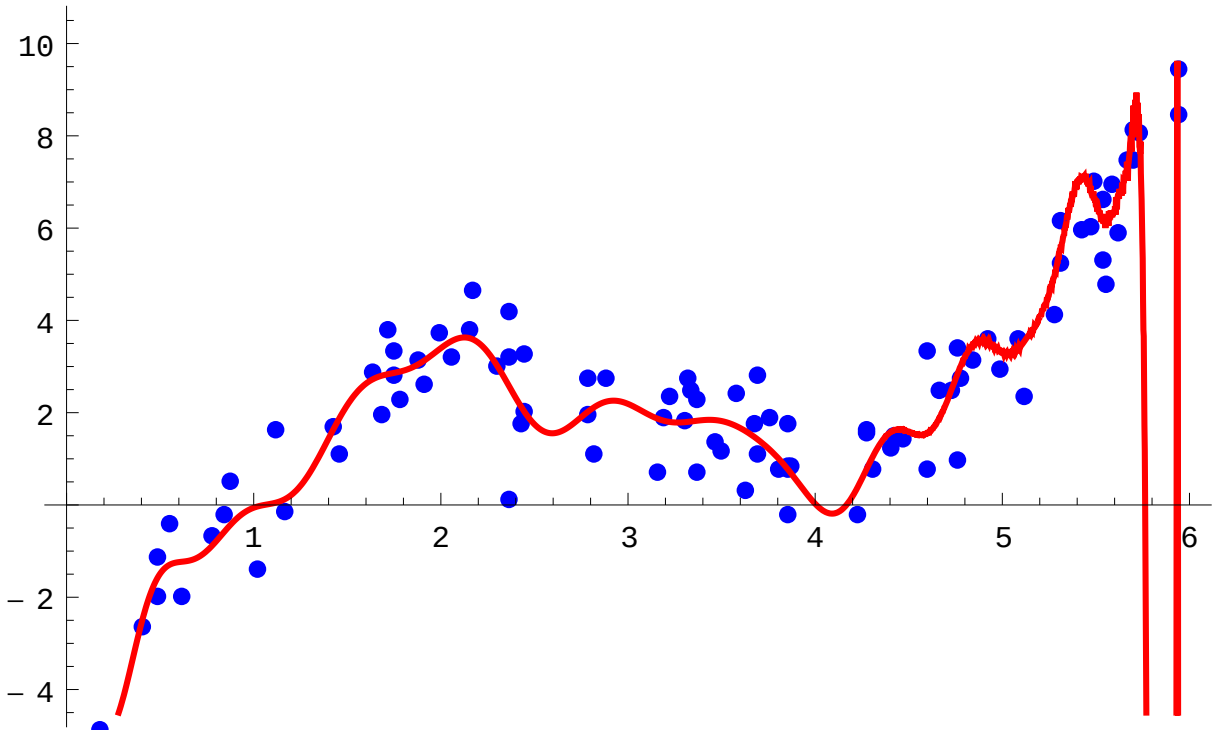
POLYNOMIAL REGRESSION EXAMPLE

$n = 25$:



POLYNOMIAL REGRESSION EXAMPLE

$n = 75$:



EMPIRICAL RISK MINIMIZATION (ERM)

Linear and polynomial regression have been concerned with minimizing the average loss on a given (training) data set. This strategy is called *empirical risk minimization*:

Given a training set $\mathbf{Z}_l$, empirical risk minimization is concerned with finding a parameter setting $\mathbf{w}$ such that the *empirical risk*

$$R_{\text{emp}}(g(.;\mathbf{w}),\mathbf{Z}_l) = \frac{1}{l}\cdot\sum_{i=1}^{l} L(y^i, g(\mathbf{x}^i;\mathbf{w}))$$

is minimal (or at least as small as possible).

ESTIMATING THE RISK: TEST SET METHOD

Assume that we have m more data samples $\mathbf{Z}_m = (\mathbf{z}^{l+1}, \ldots, \mathbf{z}^{l+m})$, the so-called *test set*, that are independently and identically distributed (i.i.d.) according to $p(\mathbf{x}, y)$ (and, therefore, so is $L(y, g(\mathbf{x}, \mathbf{w}))$). Then

$$R_{\text{emp}}(g(.; \mathbf{w}), \mathbf{Z}_m) = \frac{1}{m} \sum_{j=1}^{m} L(y^{l+j}, g(\mathbf{x}^{l+j}; \mathbf{w})) \tag{2}$$

can be considered an estimate for $R(g(.; \mathbf{w}))$. By the (strong) law of large numbers, $R_E(g(.; \mathbf{w}))$ converges to $R(g(.; \mathbf{w}))$ for $m \to \infty$.

TEST SET METHOD: PRACTICAL REALIZATION

The common way of applying the test set method in practice is the following:

1. Split the set of labeled samples into a *training set* of l samples and a *test set* of m samples.
2. Perform model selection, i.e. find a suitable model $\mathbf{w}$, making use *only of the training set* (hence, $\mathbf{w} = \mathbf{w}(\mathbf{Z})$), while *withholding the test set*.
3. Estimate the generalization error by (2) using the test set

This is also called *hold-out method*.

TEST SET METHOD: A WORD OF CAUTION

The model $g(.; \mathbf{w})$ is geared to the training set. Therefore, for training and test samples, the random variables $L(y, g(\mathbf{x}, \mathbf{w}))$ are *not identically distributed*. Hence, the estimate $R_E(g(.; \mathbf{w}))$ becomes *invalid* as soon as a single training sample is being used for estimating the risk; therefore:

- ***Training samples may never be used for "testing", i.e. estimating the generalization error!***
- ***Test samples may never be used for training!***

TEST SET METHOD: PRACTICAL CAVEATS

To avoid the pitfall described above, take the following rules into account:

1. Choose training/test samples *randomly* (unless you can be completely sure that they are already in random order)! If the probabilities for being selected as training or test samples are not equal for all samples, the independence property cannot be guaranteed.
2. Make sure that there is not the slightest influence that test samples have on the selection of the model! Also pre-processing or feature selection steps that use all samples imply that the estimate is biased.

CROSS VALIDATION: MOTIVATION

The following platitudes can be stated about the test set method:

- The more training samples (and the less test samples), the better the model, but the worse the risk estimate.
- The more test samples (and the less training samples), the coarser the model, but the better the risk estimate.

In particular, for small sample sets, the requirement that training and test set must not overlap is painful.

Question: *can we somehow improve the risk estimate without necessarily sacrificing model accuracy?*

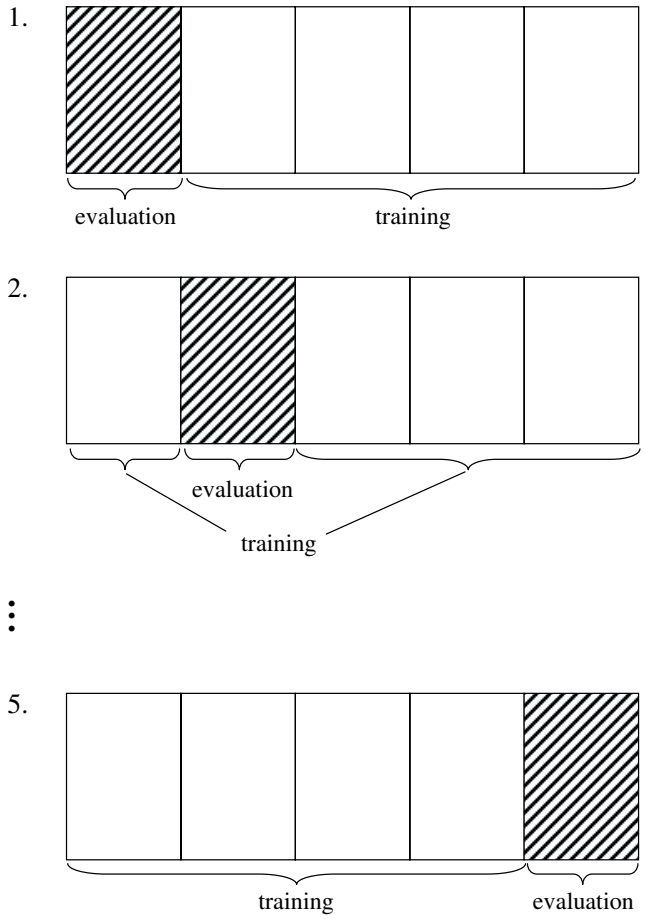
CROSS VALIDATION: BASIC IDEA

- A simple idea would be to perform the splitting into training and set several times and to average the estimates. This is incorrect, as the test sets overlap and, therefore, are not independent anymore.
- *Cross validation* somehow follows this line of thought, but splits the sample set into n *disjoint* fractions[a] (so-called *folds*):
  1. Training is done n times, every time leaving out one fold (i.e. taking the other $n-1$ folds as training set)
  2. The risk estimate is then computed as the average of the risk estimate of the n left-out test folds

  The special case $n = l$ is commonly called *leave-one-out cross validation.*

[a] For simplicity, assume in the following that l is divisible by n.

FIVE-FOLD CROSS VALIDATION VISUALIZED



CROSS VALIDATION: DEFINITION

We denote a given arbitrary sample set with l elements as $\mathbf{Z}_l$ in the following. The j-th fold inside $\mathbf{Z}_l$ is denoted as $\mathbf{Z}_{l/n}^j$ and the sample set corresponding to the remaining $n-1$ folds as $\mathbf{Z}_l \backslash \mathbf{Z}_{l/n}^j$.

Then the risk estimate given by the j-th fold is given as

$$R_{n-\text{cv},j}(\mathbf{Z}_l) = \frac{n}{l} \sum_{\mathbf{z} \in \mathbf{Z}_{l/n}^j} L\Big(y, g(\mathbf{x}; \mathbf{w}_j\left(\mathbf{Z}_l \backslash \mathbf{Z}_{l/n}^j\right))\Big).$$

The n-*fold cross validation risk* is defined as

$$R_{n-\text{cv}}(\mathbf{Z}_l) = \frac{1}{n} \sum_{j=1}^{n} R_{n-\text{cv},j}(\mathbf{Z}_l) = \frac{1}{l} \sum_{j=1}^{n} \sum_{\mathbf{z} \in \mathbf{Z}_{l/n}^j} L\Big(y, g(\mathbf{x}; \mathbf{w}_j\left(\mathbf{Z}_l \backslash \mathbf{Z}_{l/n}^j\right))\Big).$$

CROSS VALIDATION: JUSTIFICATION

Theorem (Luntz & Brailovsky). The cross-validation risk estimate is an “almost unbiased estimator”:

$$\mathrm{E}_{\mathbf{Z}_{l-l/n}}\left(R(g(.;\mathbf{w}(\mathbf{Z}_{l-l/n}))\right) = \mathrm{E}_{\mathbf{Z}_l}\left(R_{n-\mathsf{cv}}(\mathbf{Z}_l)\right)$$

CROSS VALIDATION: MISCELLANEA

Obviously, n different models are computed during n-fold cross validation. Questions:

1. Which of these models should we select finally?
2. Can we get a better model if we manage to average these n models?

Answers:

1. None, as the selection would be biased to a certain fold.
2. It depends on the model class whether this is possible and meaningful (see later).

A good strategy is, once that we know about the generalization abilities of our model, to finally train a model using all l samples.

CROSS VALIDATION: MISCELLANEA (cont'd)

Cross validation is also commonly applied to *finding good choices of hyper-parameters*, i.e. by selecting those hyperparameters for which the smallest cross validation risk is obtained.

Note, however, that the obtained risk estimate is then *biased* to the whole training set. If an unbiased estimate for the risk is desired, this can only be done by a combination of the test set method and cross validation:

1. Split the sample set into training set and test set first.
2. Apply cross validation on the training set (completely withholding the test set) to find the best hyperparameter choice.
3. Finally, compute the risk estimate using the test set.

ERM IN PRACTICE

- As said, ERM is concerned with minimizing the *training error*.
- Our goal, however, is to minimize the *generalization error* (which can be estimated by the *test error*).
- In other words, ERM does not use the correct objective!

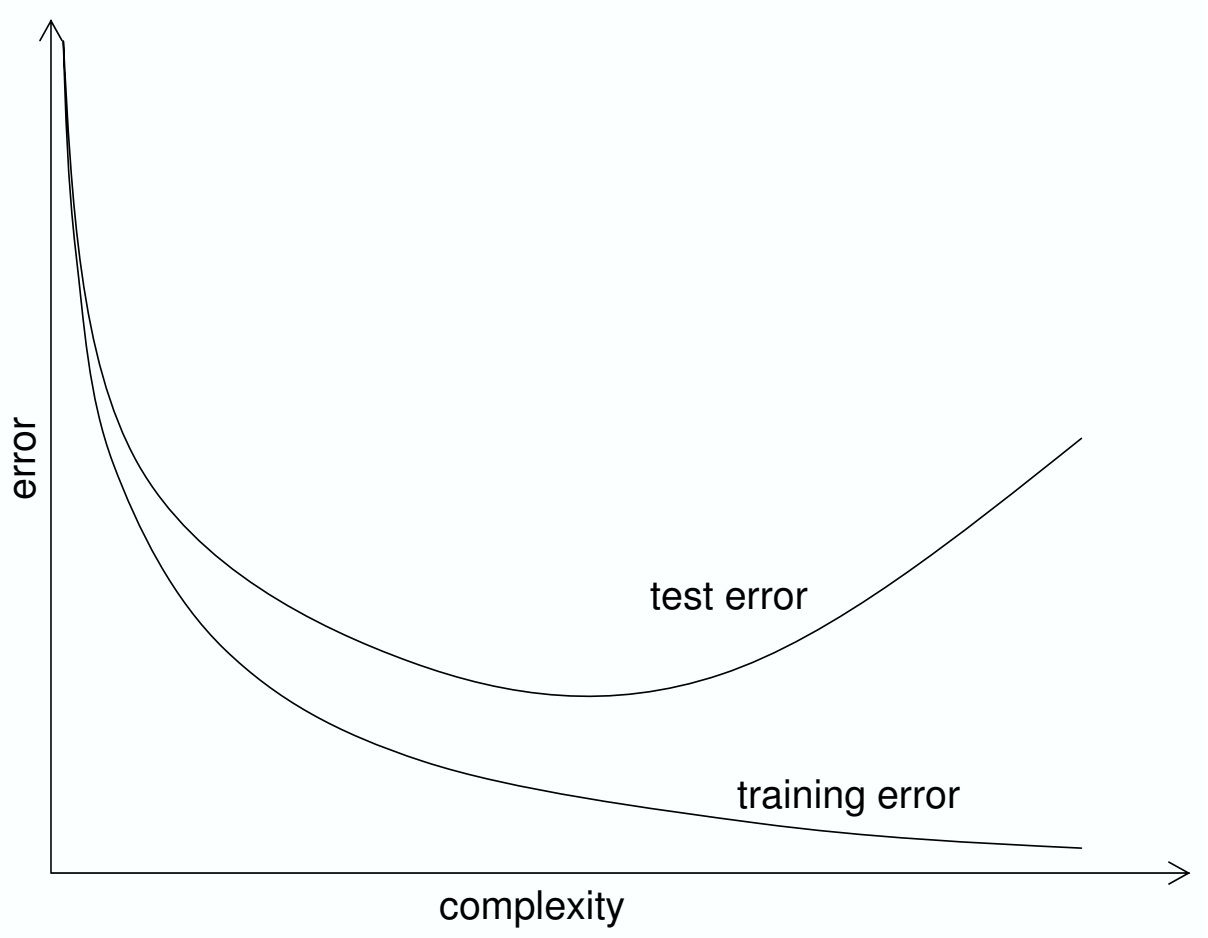
Question: What can go wrong because of using the wrong objective?

UNDERFITTING AND OVERFITTING

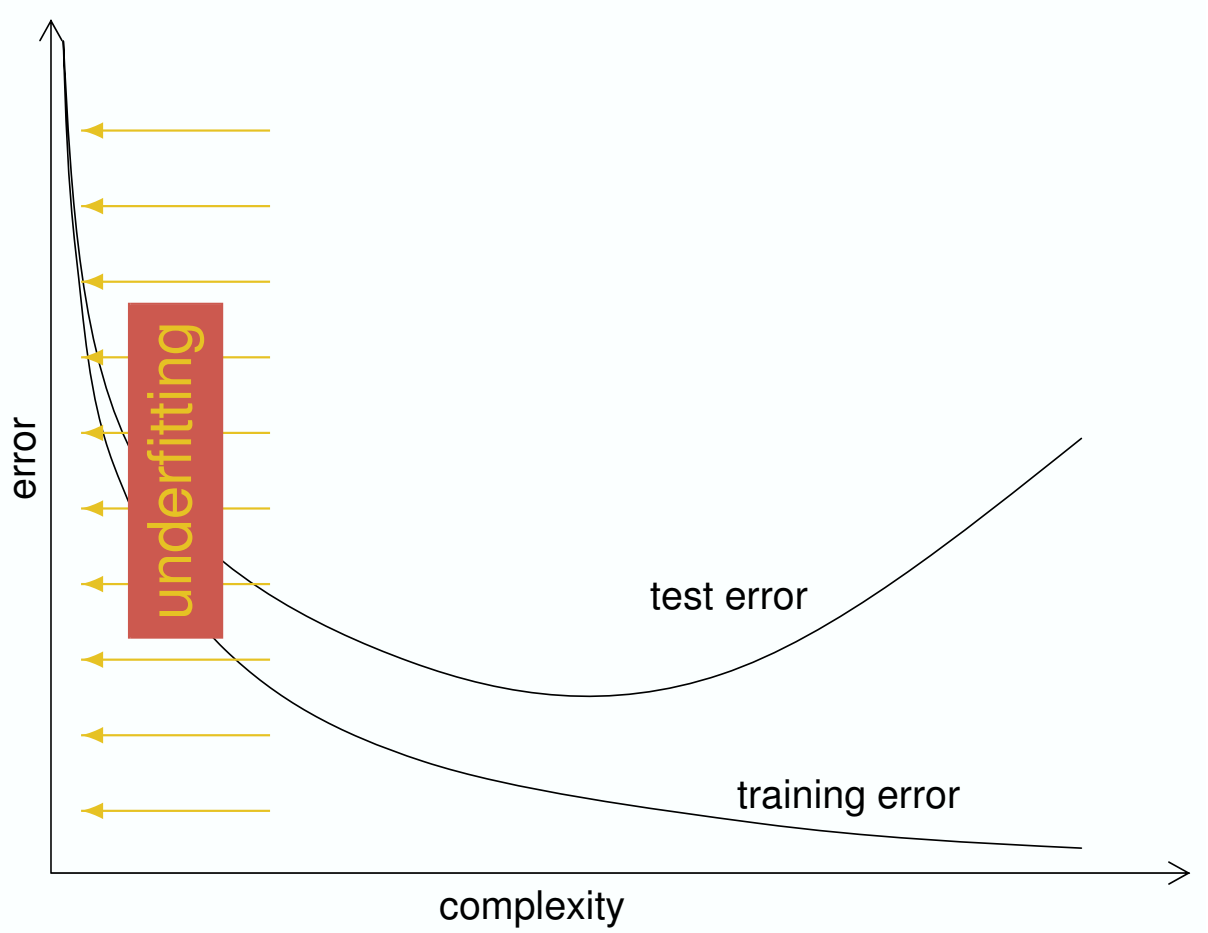
Underfitting: our model is too coarse to fit the data (neither training nor test data); this is usually the result of too restrictive model assumptions (i.e. *too low complexity of model*).

Overfitting: our model works very well on training data, but generalizes poorly to future/test data; this is usually the result of *too high model complexity*.

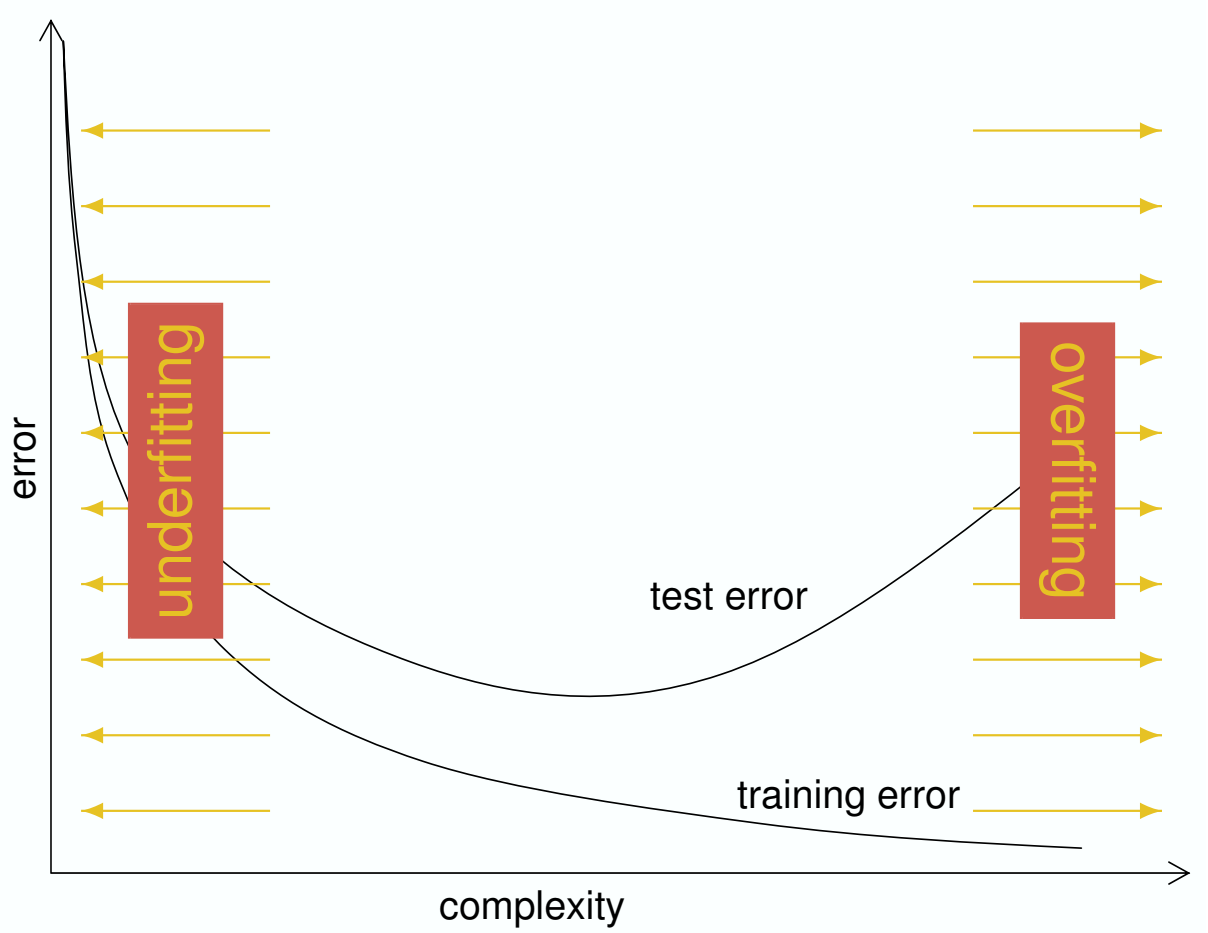
NOTORIOUS SITUATION IN PRACTICE



NOTORIOUS SITUATION IN PRACTICE



NOTORIOUS SITUATION IN PRACTICE



BIAS-VARIANCE DECOMPOSITION FOR QUADRATIC LOSS (1/4)

We are interested in the expected prediction error for a given $\mathbf{x}_0 \in X$ (assuming that the size of the training set is fixed to l examples):

$$\begin{aligned}\text{EPE}(\mathbf{x}_0) &= \mathrm{E}_{y|\mathbf{x}_0,\mathbf{Z}_l}\left(L_{\mathbf{q}}(y, g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))\right) \\ &= \mathrm{E}_{y|\mathbf{x}_0,\mathbf{Z}_l}\left((y - g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))^2\right)\end{aligned}$$

Since $y \mid \mathbf{x}_0$ and the selection of training samples are independent (or at least this should be assumed to be the case), we can infer the following:

$$\text{EPE}(\mathbf{x}_0) = \mathrm{E}_{y|\mathbf{x}_0}\left(\mathrm{E}_{\mathbf{Z}_l}\left((y - g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))^2\right)\right)$$

BIAS-VARIANCE DECOMPOSITION FOR QUADRATIC LOSS (2/4)

Using basic properties of expected values, we can infer the following representation:

$$\begin{aligned}\text{EPE}(\mathbf{x}_0) =& \text{Var}(y \mid \mathbf{x}_0) \\ &+ \Big(\text{E}(y \mid \mathbf{x}_0) - \text{E}_{\mathbf{Z}_l}\left(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l))\right)\Big)^2 \\ &+ \text{E}_{\mathbf{Z}_l}\Big(\left(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)) - \text{E}_{\mathbf{Z}_l}(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))\right)^2\Big)\end{aligned}$$

BIAS-VARIANCE DECOMPOSITION FOR QUADRATIC LOSS (3/4)

1. The first term, $\mathrm{Var}(y \mid \mathbf{x}_0)$ is nothing else but the average amount to which the label y varies at $\mathbf{x}_0$. This is often termed *unavoidable error.*
2. The second term,

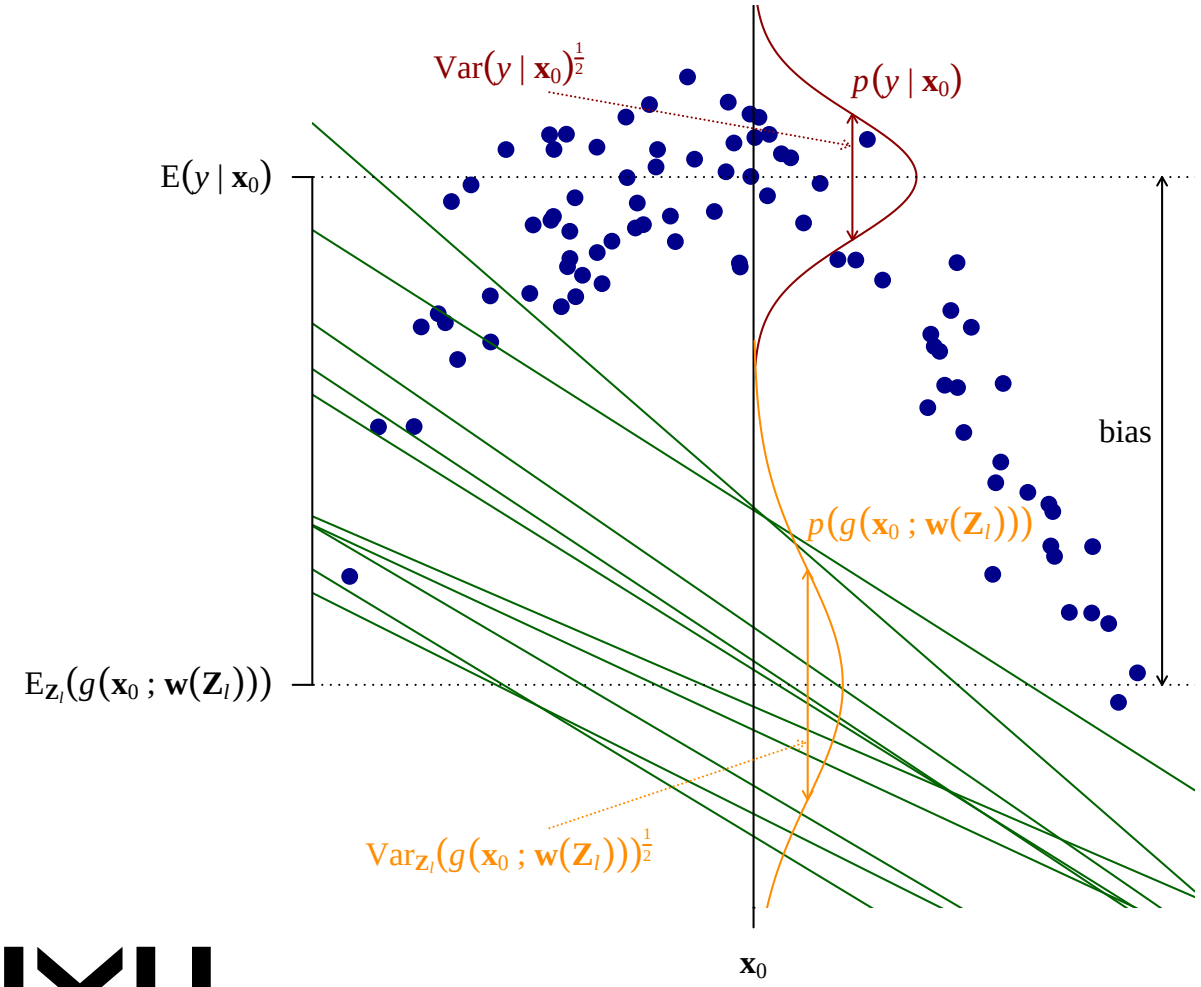
$$\text{bias}^2 = \Big(\mathrm{E}(y \mid \mathbf{x}_0) - \mathrm{E}_{\mathbf{Z}_l}\left(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l))\right)\Big)^2$$

   measures how close the model in average approximates the average target y at $\mathbf{x}_0$; thus, it is nothing else but the *squared bias*.
3. The third term,

$$\text{variance} = \mathrm{E}_{\mathbf{Z}_l}\left(\left(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)) - \mathrm{E}_{\mathbf{Z}_l}(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))\right)^2\right)$$

   is nothing else but the *variance* of models at $\mathbf{x}_0$, i.e. $\mathrm{Var}_{\mathbf{Z}_l}(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))$.

BIAS-VARIANCE DECOMPOSITION FOR QUADRATIC LOSS (4/4)



BIAS-VARIANCE DECOMPOSITION: SIMPLIFICATIONS

- Assume that $y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon$ holds, where f is a deterministic function and ε is a random variable that has mean zero and variance σ_ε^2 and is independent of $\mathbf{x}$. Then we can infer the following:

$$\begin{aligned} \mathrm{Var}(y \mid \mathbf{x}_0) &= \sigma_\varepsilon^2, \\ \mathrm{E}(y \mid \mathbf{x}_0) &= f(\mathbf{x}_0), \\ \text{bias}^2 &= \Big(f(\mathbf{x}_0) - \mathrm{E}_{\mathbf{Z}_l} \big(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)) \big) \Big)^2. \end{aligned}$$

- In the noise-free case ($\sigma_\varepsilon = 0$), consequently, we get $\mathrm{Var}(y \mid \mathbf{x}_0) = 0$, i.e. the unavoidable error vanishes and the rest stays the same.

BIAS-VARIANCE DECOMPOSITION FOR BINARY CLASSIFICATION

Now assume that we are given a binary classification task, i.e. $y \in \{-1, +1\}$ and $g(\mathbf{x}; \mathbf{w}) \in \{-1, +1\}$. Since $L_{\mathbf{zo}} = \frac{1}{4} L_{\mathbf{q}}$ holds, we can infer the following:

$$\begin{aligned}
\text{EPE}(\mathbf{x}_0) &= \mathrm{E}_{y|\mathbf{x}_0, \mathbf{Z}_l} \left(L_{\mathbf{zo}}(y, g(\mathbf{x}_0; \mathbf{w})) \right) \\
&= \frac{1}{4} \cdot \mathrm{E}_{y|\mathbf{x}_0} \left(\mathrm{E}_{\mathbf{Z}_l} \left((y - g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)))^2 \right) \right) \\
&= \frac{1}{4} \cdot \left(\mathrm{Var}(y \mid \mathbf{x}_0) + \text{bias}^2 + \text{variance} \right)
\end{aligned}$$

Note that, in these calculations, g is the final binary classification function and *not* an arbitrary discriminant function. If the latter is the case, the above representation is *not valid!* (see literature)

BIAS-VARIANCE DECOMPOSITION FOR BINARY CLASSIF. (cont'd)

With the notations $p_R = p(y = +1 \mid \mathbf{x_0})$ and

$$p_O = p_{\mathbf{Z}_l}(g(\mathbf{x_0}; \mathbf{w}(\mathbf{Z}_l)) = +1),$$

we can infer further

$$\begin{aligned} \mathrm{Var}(y \mid \mathbf{x}_0) &= 4 \cdot p_R \cdot (1 - p_R), \\ \text{bias}^2 &= 4 \cdot (p_R - p_O)^2, \\ \text{variance} &= 4 \cdot p_O \cdot (1 - p_O), \end{aligned}$$

hence, we obtain

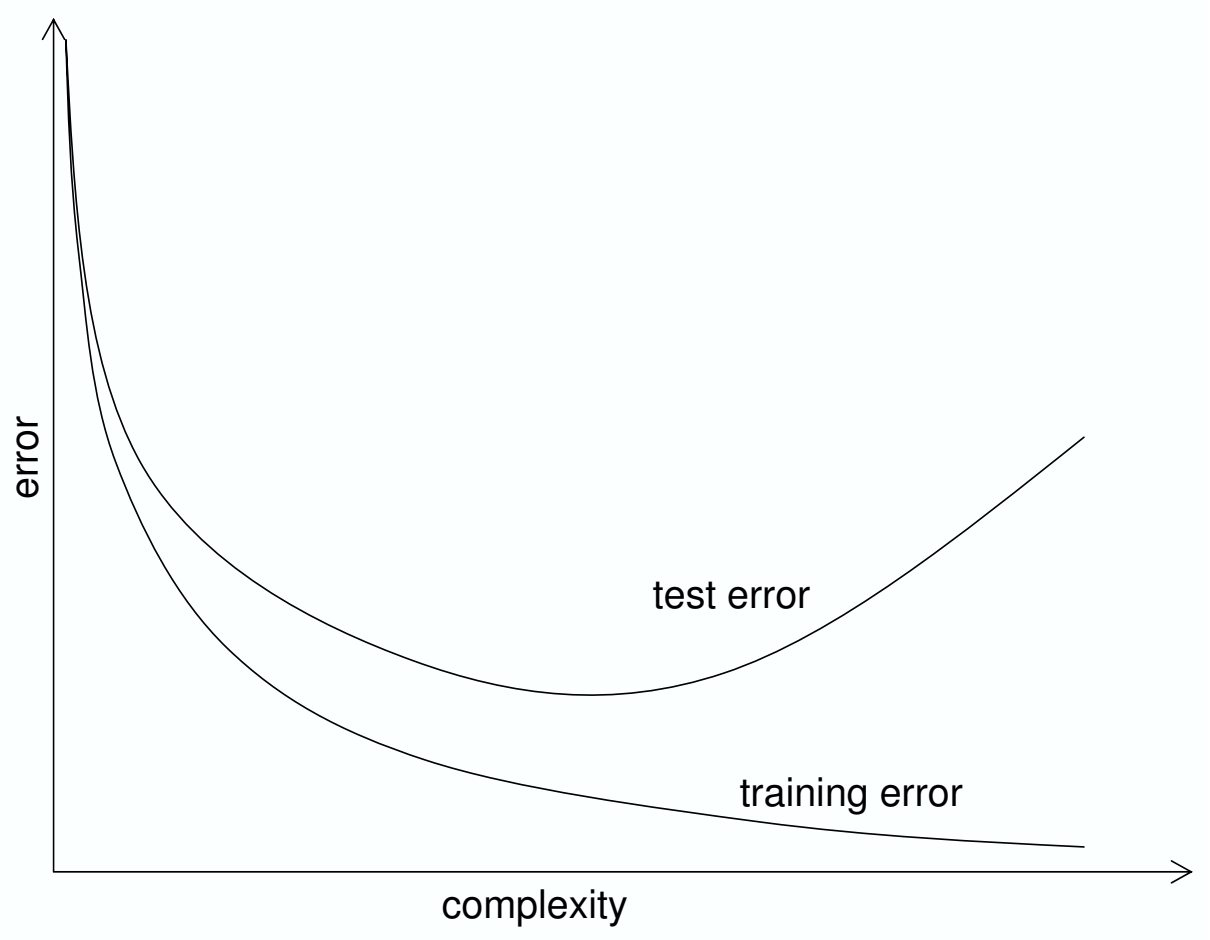
$$\text{EPE}(\mathbf{x}_0) = \underbrace{p_R \cdot (1 - p_R)}_{\text{unavoidable error}} + \underbrace{(p_R - p_O)^2}_{\text{squared bias}} + \underbrace{p_O \cdot (1 - p_O)}_{\text{variance}}.$$

THE BIAS-VARIANCE TRADE-OFF

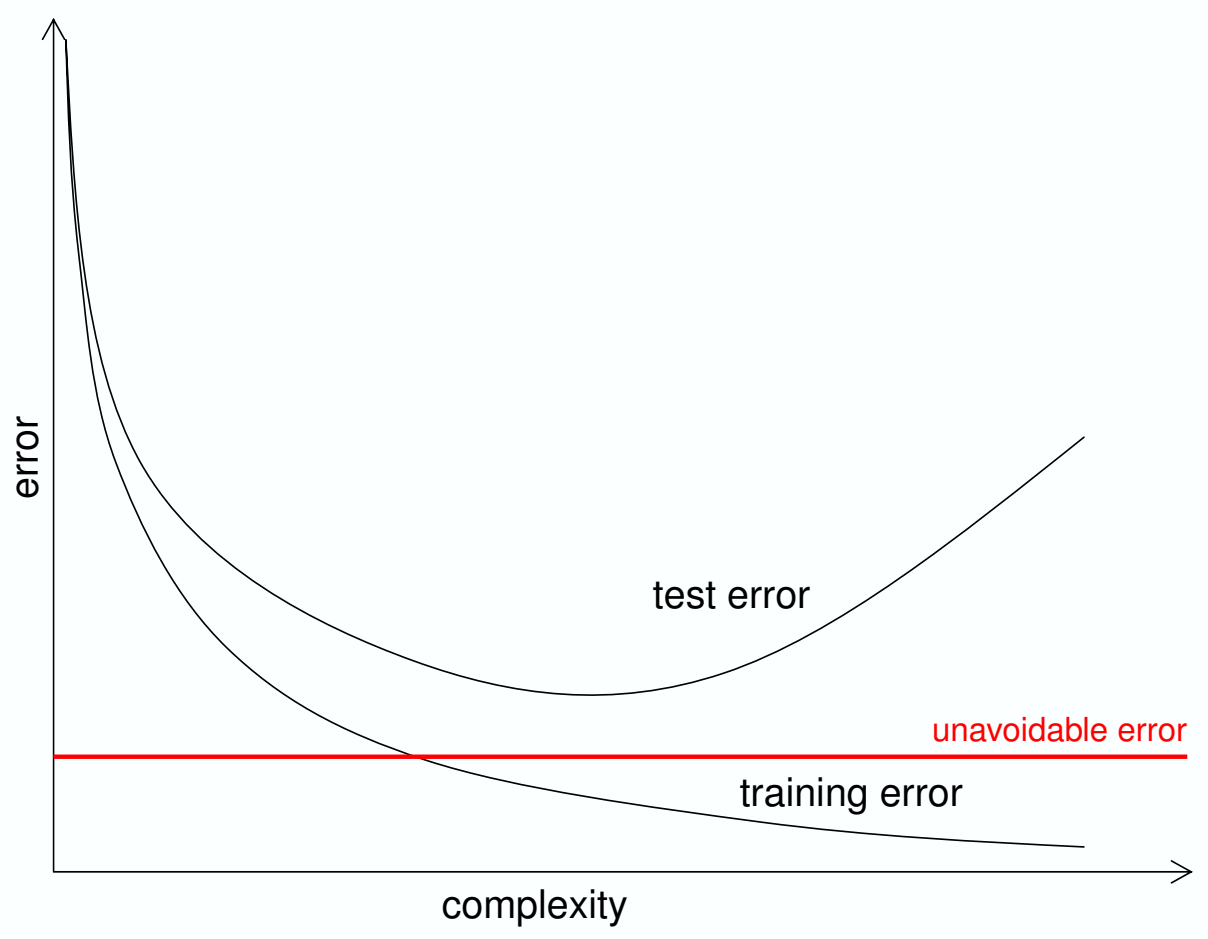
- It seems intuitively reasonable that the bias decreases with model complexity.
  Rationale: the more degrees of freedom we allow, the easier we can fit the actual function/relationship.
- It also seems intuitively clear that the variance increases with model complexity.
  Rationale: the more degrees of freedom we allow, the higher the risk to fit to noise.

This is usually referred to as the *bias-variance trade-off*. sometimes even *bias-variance "dilemma".*

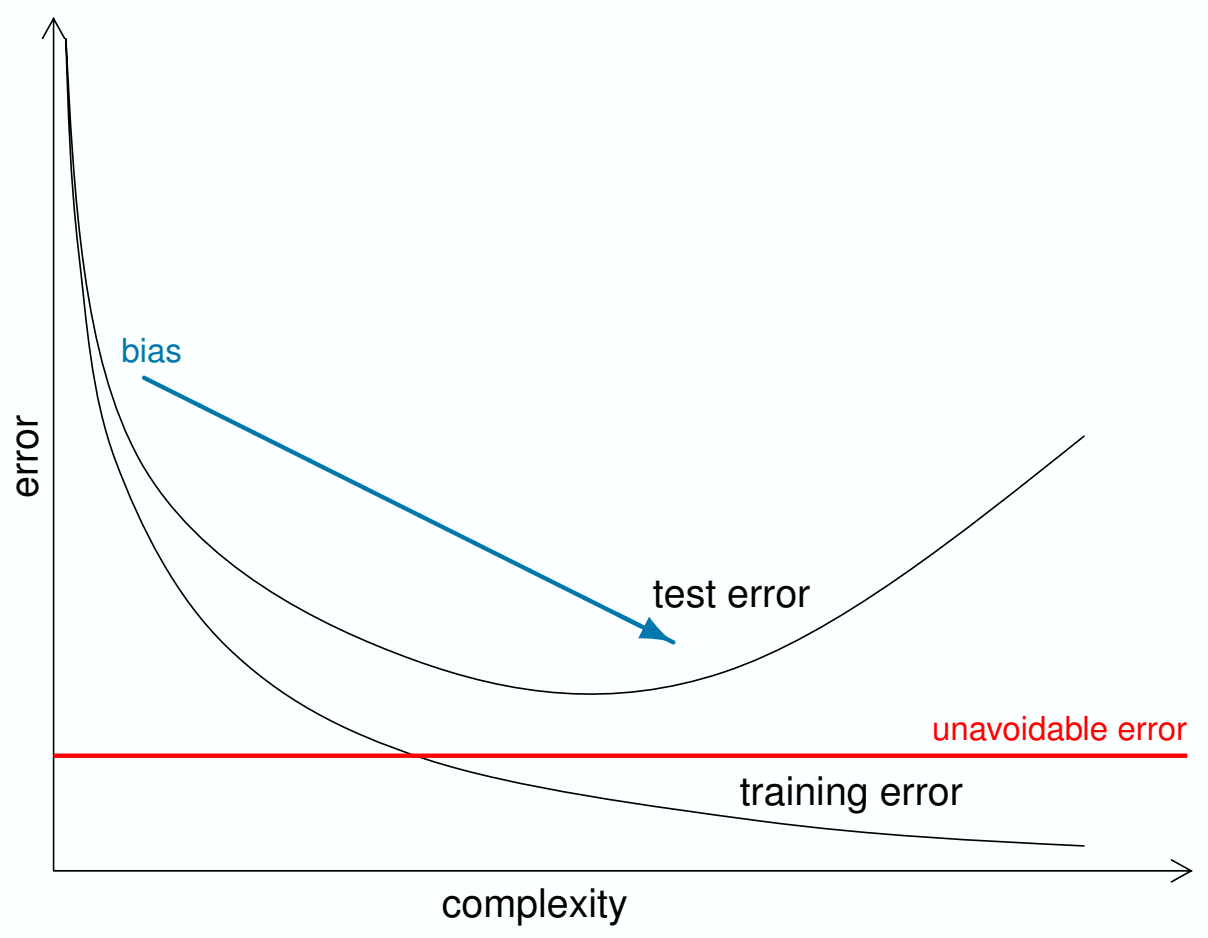
THE BIAS-VARIANCE TRADE-OFF (cont'd)



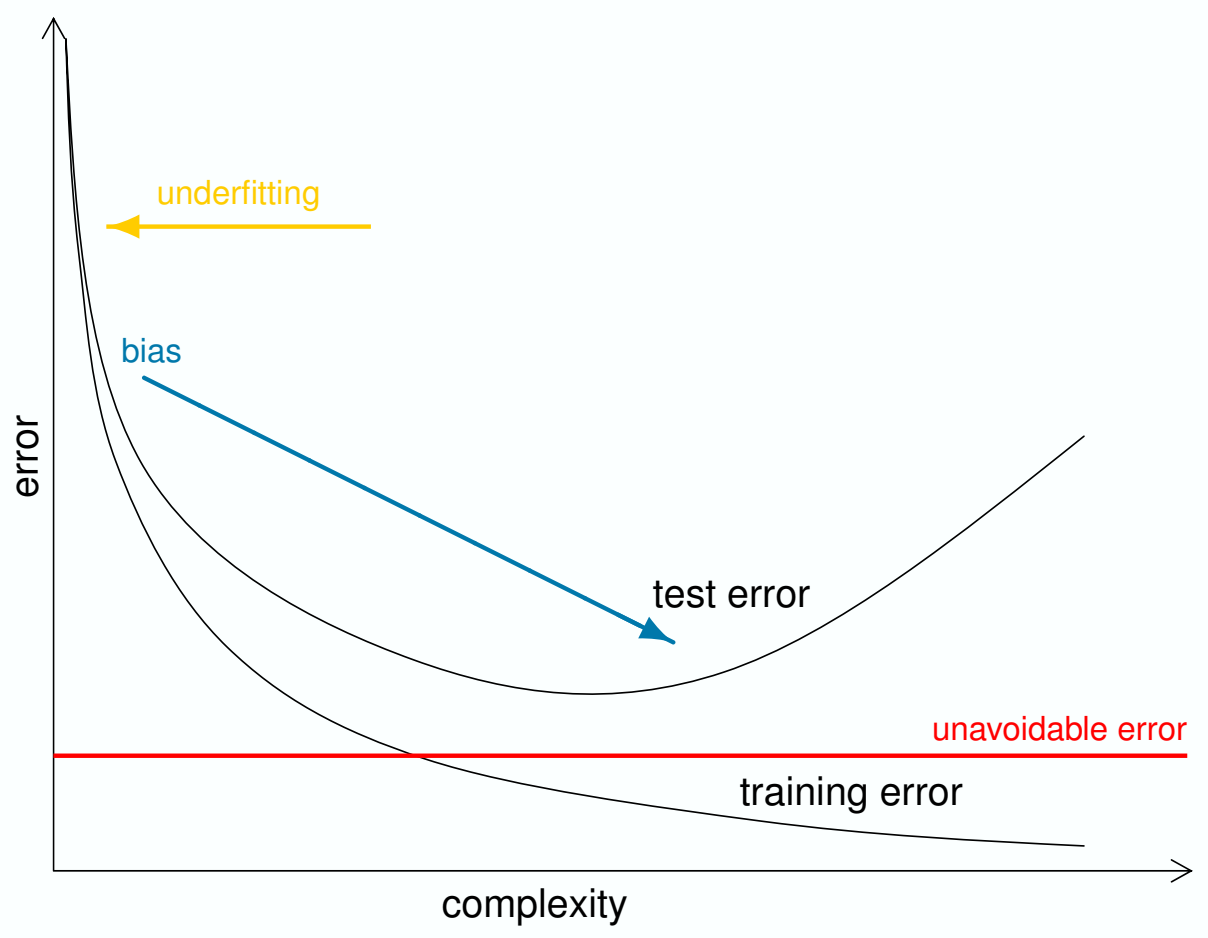
THE BIAS-VARIANCE TRADE-OFF (cont'd)



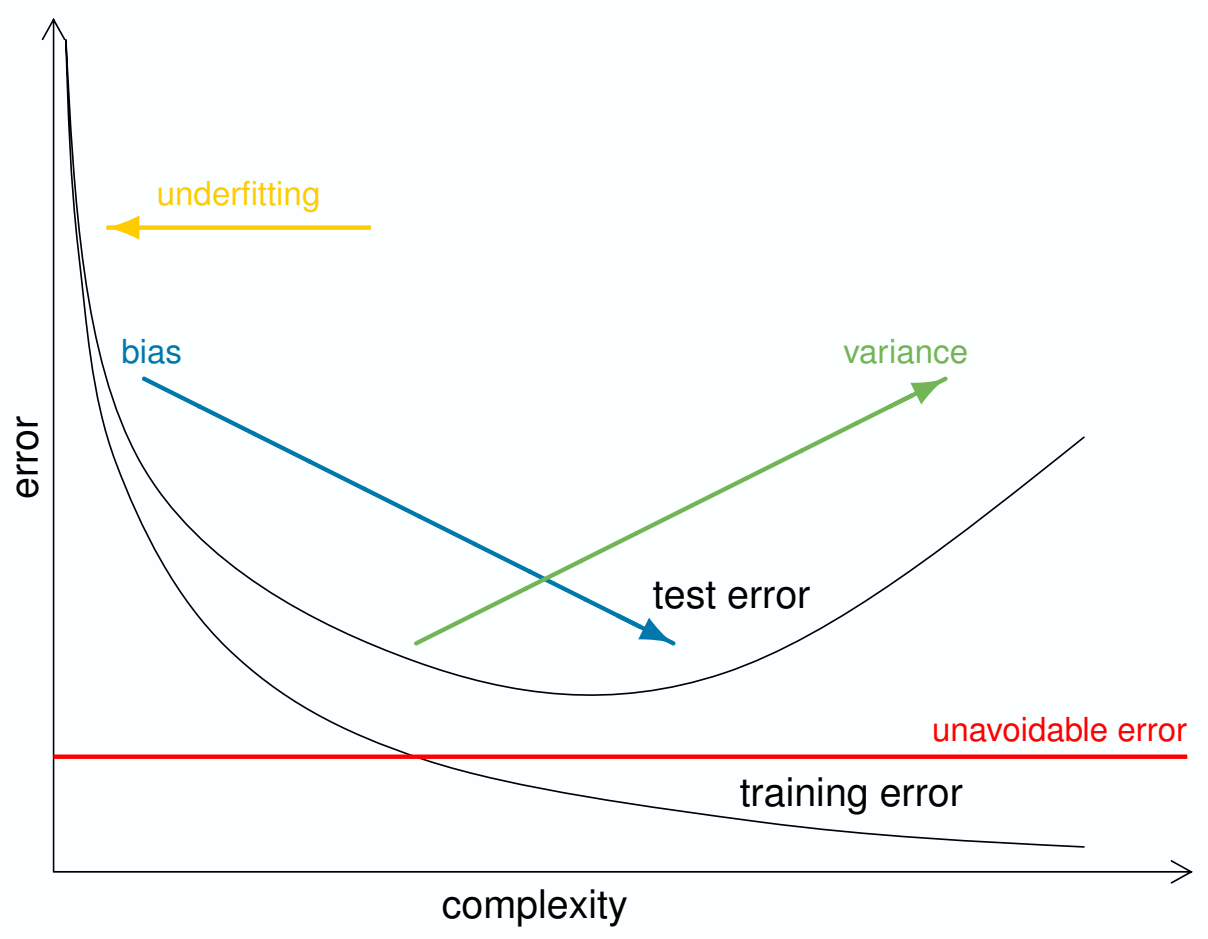
THE BIAS-VARIANCE TRADE-OFF (cont'd)



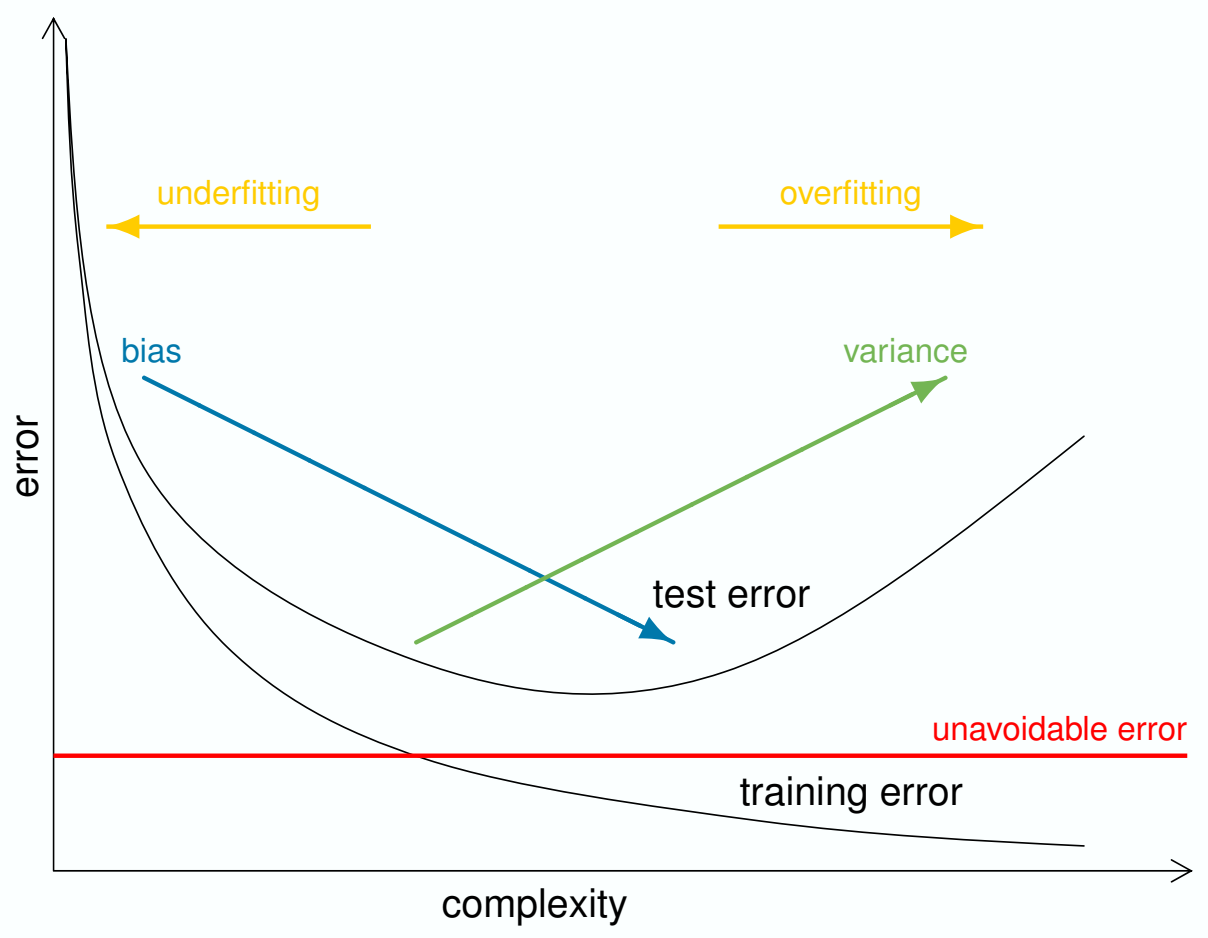
THE BIAS-VARIANCE TRADE-OFF (cont'd)



THE BIAS-VARIANCE TRADE-OFF (cont'd)



THE BIAS-VARIANCE TRADE-OFF (cont'd)



BIAS-VARIANCE DECOMPOSITION: SUMMARY

- We can state that minimizing the generalization error (learning) is concerned with optimizing bias and variance simultaneously.
- Underfitting $=$ high bias $=$ too simple model
- Overfitting $=$ high variance $=$ too complex model
- It is clear that empirical risk minimization itself does not include any mechanism to assess bias and variance independently (how should it?); more specifically, if we do not care about model complexity (in particular, if we allow highly or even arbitrarily complex models), ERM has a high risk to produce over-fitted models.

HOW TO EVALUATE CLASSIFIERS?

So far, the only measure we have considered for assessing the performance of a classifier was the generalization error based on the zero-one loss.

- What if the data set is unbalanced?
- What if the misclassification cost depends on the sample's class?
- Can we define a general performance measure independent class distributions and misclassification costs?

In order to answer these questions, we need to introduce confusion matrices first.

CONFUSION MATRIX FOR BINARY CLASSIFICATION

Let us introduce the following terminology (for a given sample $(\mathbf{x}, y)$ and a classifier $g(.)$): $(\mathbf{x}, y)$ is a

true positive (TP) if $y = +1$ and $g(\mathbf{x}) = +1$,
true negative (TN) if $y = -1$ and $g(\mathbf{x}) = -1$,
false positive (FP) if $y = -1$ and $g(\mathbf{x}) = +1$,
false negative (FN) if $y = +1$ and $g(\mathbf{x}) = -1$.

CONFUSION MATRIX FOR BINARY CLASSIFICATION (cont'd)

Given a data set $(\mathbf{z}^1, \ldots, \mathbf{z}^m)$, the *confusion matrix* is defined as follows:

		predicted value $g(\mathbf{x}; \mathbf{w})$	
		+1	-1
actual value y	+1	#TP	#FN
	-1	#FP	#TN

In this table, the entries #TP, #FP, #FN and #TN denote the numbers of true positives, . . . , respectively, for the given data set.

EVALUATION MEASURES DERIVED FROM CONFUSION MATRIX

Accuracy: proportion of correctly classified items, i.e.

$$\text{ACC} = \frac{\#\text{TP} + \#\text{TN}}{\#\text{TP} + \#\text{FN} + \#\text{FP} + \#\text{TN}}.$$

True Positive Rate (aka recall/sensitivity): proportion of correctly identified positives, i.e.

$$\text{TPR} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}}.$$

False Positive Rate: proportion of negative examples that were incorrectly classified as positives, i.e.

$$\text{FPR} = \frac{\#\text{FP}}{\#\text{FP} + \#\text{TN}}.$$

Precision: proportion of predicted positive examples that were correct, i.e.

$$\text{PREC} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}.$$

True Negative Rate (aka specificity): proportion of correctly identified negatives, i.e.

$$\text{TNR} = \frac{\#\text{TN}}{\#\text{FP} + \#\text{TN}}.$$

False Negative Rate: proportion of positive examples that were incorrectly classified as negatives, i.e.

$$\text{FNR} = \frac{\#\text{FN}}{\#\text{TP} + \#\text{FN}}.$$

EVALUATION MEASURES DESIGNED FOR UNBALANCED DATA

Balanced Accuracy: mean of true positive and true negative rate, i.e.

$$\text{BACC} = \frac{\text{TPR} + \text{TNR}}{2}$$

Matthews Correlation Coefficient: measure of non-randomness of classification; defined as normalized determinant of confusion matrix, i.e.

$$\text{MCC} = \frac{\#\text{TP} \cdot \#\text{TN} - \#\text{FP} \cdot \#\text{FN}}{\sqrt{(\#\text{TP} + \#\text{FP})(\#\text{TP} + \#\text{FN})(\#\text{TN} + \#\text{FP})(\#\text{TN} + \#\text{FN})}}$$

F-score: harmonic mean of precision and recall, i.e.

$$F_1 = 2 \cdot \frac{\text{PREC} \cdot \text{TPR}}{\text{PREC} + \text{TPR}}$$

CONFUSION MATRIX FOR MULTI-CLASS CLASSIFICATION

Assume that we have a k-class classification task. Given a data set, the *confusion matrix* is defined as follows:

		predicted class $g(\mathbf{x})$				
		1	$\cdots$	j	$\cdots$	k
actual value y	1	C_{11}	$\cdots$	C_{1j}	$\cdots$	C_{1k}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	i	C_{i1}	$\cdots$	C_{ij}	$\cdots$	C_{ik}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	k	C_{k1}	$\cdots$	C_{kj}	$\cdots$	C_{kk}

The entries C_{ij} correspond to the numbers of test samples that actually belong to class i and have been classified as j by the classifier $g(.)$.

ACCURACY FOR MULTI-CLASS CLASSIFICATION

For a multi-class classification task (with the notations as on the previous slide), the *accuracy* of a classifier $g(.)$ is defined as

$$\text{ACC} = \frac{\sum_{i=1}^{k} C_{ii}}{\sum_{i,j=1}^{k} C_{ij}} = \frac{1}{m} \cdot \sum_{i=1}^{k} C_{ii},$$

i.e., not at all surprisingly, as the *proportion of correctly classified samples*. The other evaluation measures cannot be generalized to the multi-class case in a straightforward way.

OTHER PERFORMANCE MEASURES FOR MULTI-CLASS CLASSIFICATION

Beside accuracy, the other evaluation measures cannot be generalized to the multi-class case in a direct way, but we can easily define them for each class separately. Given a class j, we can define the confusion matrix of class j as follows:

		predicted value $g(\mathbf{x}; \mathbf{w})$	
		$= j$	$\neq j$
actual value y	$= j$	#TP$_j$	#FN$_j$
	$\neq j$	#FP$_j$	#TN$_j$

From this confusion matrix, we can easily define all previously known evaluation measures (for class j).

RISK FOR BINARY CLASSIFICATION: ASYMMETRIC CASE (1/3)

Consider the following loss function (with $l_{\mathbf{FP}}, l_{\mathbf{FN}} > 0$):

$$L_{\mathbf{as}}(y, g(\mathbf{x}; \mathbf{w})) = \begin{cases} 0 & y = g(\mathbf{x}; \mathbf{w}) \\ l_{\mathbf{FP}} & y = -1 \text{ and } g(\mathbf{x}; \mathbf{w}) = +1 \\ l_{\mathbf{FN}} & y = +1 \text{ and } g(\mathbf{x}; \mathbf{w}) = -1 \end{cases}$$

Then we obtain the following:

$$\begin{aligned} R(g(.; \mathbf{w})) &= \int_{X_{-1}} l_{\mathbf{FN}} \cdot p(\mathbf{x}, y = +1) d\mathbf{x} + \int_{X_{+1}} l_{\mathbf{FP}} \cdot p(\mathbf{x}, y = -1) d\mathbf{x} \\ &= \int_{X} \left\{ \begin{array}{ll} l_{\mathbf{FP}} \cdot p(y = -1 \mid \mathbf{x}) & \text{if } g(\mathbf{x}; \mathbf{w}) = +1 \\ l_{\mathbf{FN}} \cdot p(y = +1 \mid \mathbf{x}) & \text{if } g(\mathbf{x}; \mathbf{w}) = -1 \end{array} \right\} \cdot p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

RISK FOR BINARY CLASSIFICATION: ASYMMETRIC CASE (2/3)

We can infer the following optimal classification function:

$$g(\mathbf{x}) = \begin{cases} +1 & \text{if } l_{\mathbf{FN}} \cdot p(y=+1 \mid \mathbf{x}) > l_{\mathbf{FP}} \cdot p(y=-1 \mid \mathbf{x}) \\ -1 & \text{if } l_{\mathbf{FP}} \cdot p(y=-1 \mid \mathbf{x}) > l_{\mathbf{FN}} \cdot p(y=+1 \mid \mathbf{x}) \end{cases}$$
$$= \text{sign}(l_{\mathbf{FN}} \cdot p(y=+1 \mid \mathbf{x}) - l_{\mathbf{FP}} \cdot p(y=-1 \mid \mathbf{x})) \quad (3)$$

The resulting minimal risk is

$$R_{\text{min}} = \int_X \min(l_{\mathbf{FP}} \cdot p(\mathbf{x}, y=-1), l_{\mathbf{FN}} \cdot p(\mathbf{x}, y=+1)) d\mathbf{x}$$
$$= \int_X \min(l_{\mathbf{FP}} \cdot p(y=-1 \mid \mathbf{x}), l_{\mathbf{FN}} \cdot p(y=+1 \mid \mathbf{x})) \cdot p(\mathbf{x}) d\mathbf{x}$$

RISK FOR BINARY CLASSIFICATION: ASYMMETRIC CASE (3/3)

Since

$$l_{\mathbf{FN}} \cdot p(y = +1 \mid \mathbf{x}) > l_{\mathbf{FP}} \cdot p(y = -1 \mid \mathbf{x})$$

if and only if (with the convention $1/0 = \infty$)

$$\frac{p(y = +1 \mid \mathbf{x})}{p(y = -1 \mid \mathbf{x})} > \frac{l_{\mathbf{FP}}}{l_{\mathbf{FN}}},$$

we can rewrite (3) as follows:

$$g(\mathbf{x}) = \text{sign}\left(\frac{p(y = +1 \mid \mathbf{x})}{p(y = -1 \mid \mathbf{x})} - \frac{l_{\mathbf{FP}}}{l_{\mathbf{FN}}}\right)$$

Hence, the optimal classification function only depends on the ratio of $l_{\mathbf{FP}}$ and $l_{\mathbf{FN}}$.

GENERAL PERFORMANCE OF DISCRIMINANT FUNCTION

If we have a general discriminant function $\bar{g}$ that maps objects to real values, we can adjust to different asymmetric/unbalanced situations by varying the classification threshold θ (which is by default 0):

$$g(\mathbf{x}) = \text{sign}(\bar{g}(\mathbf{x}) - \theta)$$

Question: can we assess the general performance of a classifier without choosing a particular discrimination threshold?

ROC CURVES

- ROC stands for *Receiver Operator Characteristic*. The concept has been introduced in signal detection theory.
- ROC curves are a simple means for evaluating the performance of a binary classifier *independent of class distributions and misclassification costs*.
- The basic idea of ROC curves is to plot the true positive rate (TPR) vs. the false positive rate (FPR) while varying the classification threshold.

ROC CURVES: PRACTICAL REALIZATION

- Sort samples descendingly according to $\bar{g}$.
- Divide horizontal axis into as many bins as there are negative samples; divide vertical axis into as many bins as there are positive samples.
- Start curve at $(0, 0)$.
- Iterate over all possible thresholds, i.e. all possible "slots" between two discriminant function values. Every positive sample is a step up, every negative sample is a step to the right.
- In case of ties (equal discriminant function values), process them at once (which results in a ramp in the curve).
- Finally, end curve in $(1, 1)$.

AREA UNDER THE ROC CURVE (AUC)

- The *area under the ROC curve (AUC)* is a common measure for assessing the general performance of a classifier $g(.; \mathbf{w})$.
- The lowest possible value is 0, the highest possible value is 1. Obviously, *the higher the better*.
- An AUC of 1 means that there exists a threshold which perfectly separates the test samples.
- A random classifier produces an AUC of 0.5 in average; hence, an AUC smaller than 0.5 corresponds to a classification that is *worse than random* and an AUC greater than 0.5 corresponds to a classification that is *better than random*.

AUC: CORRESPONDENCES

Suppose that $\#p$ and $\#n$ are the numbers of positive and negative samples, respectively, and further assume that $\tilde{\mathbf{y}}$ is the label vector if the samples are sorted according to the discriminant function value $g(.; \mathbf{w})$. Then the following holds:

$$\text{AUC} = \frac{1}{\#p \cdot \#n} \cdot \Big(\underbrace{\big(\sum_{\tilde{y}^i > 0} i \big)}_{=R} - \frac{\#p \cdot (\#p + 1)}{2} \Big)$$

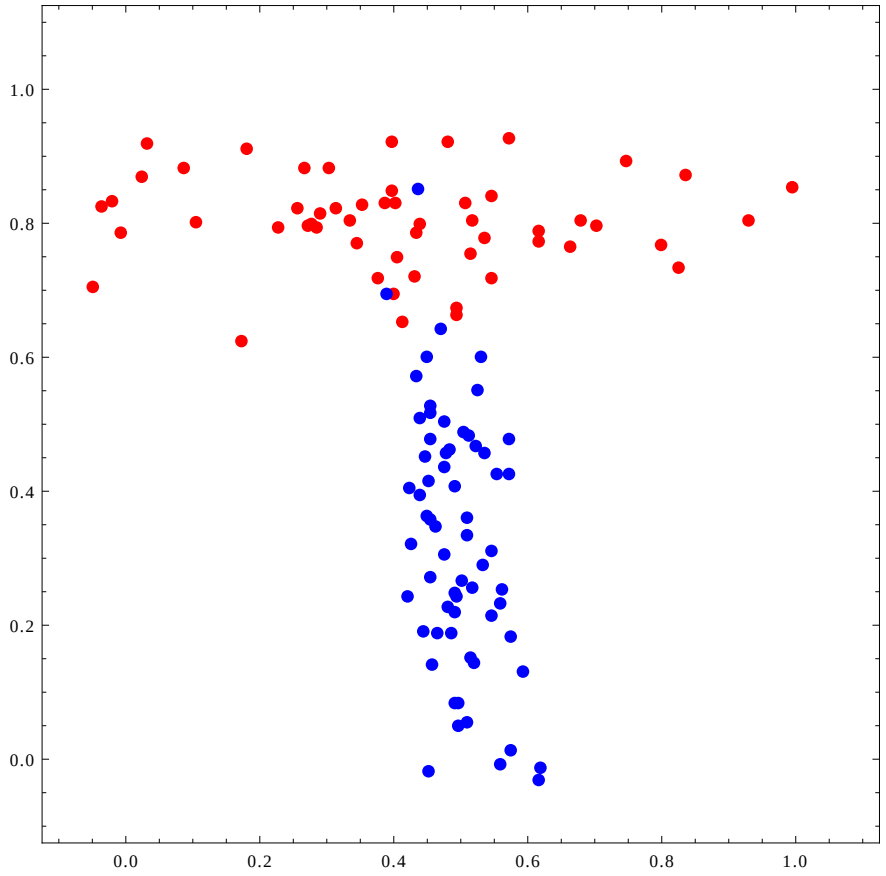
Obviously, R is the sum of ranks of positive examples. Note that

$$U = R - \frac{\#p \cdot (\#p + 1)}{2}$$

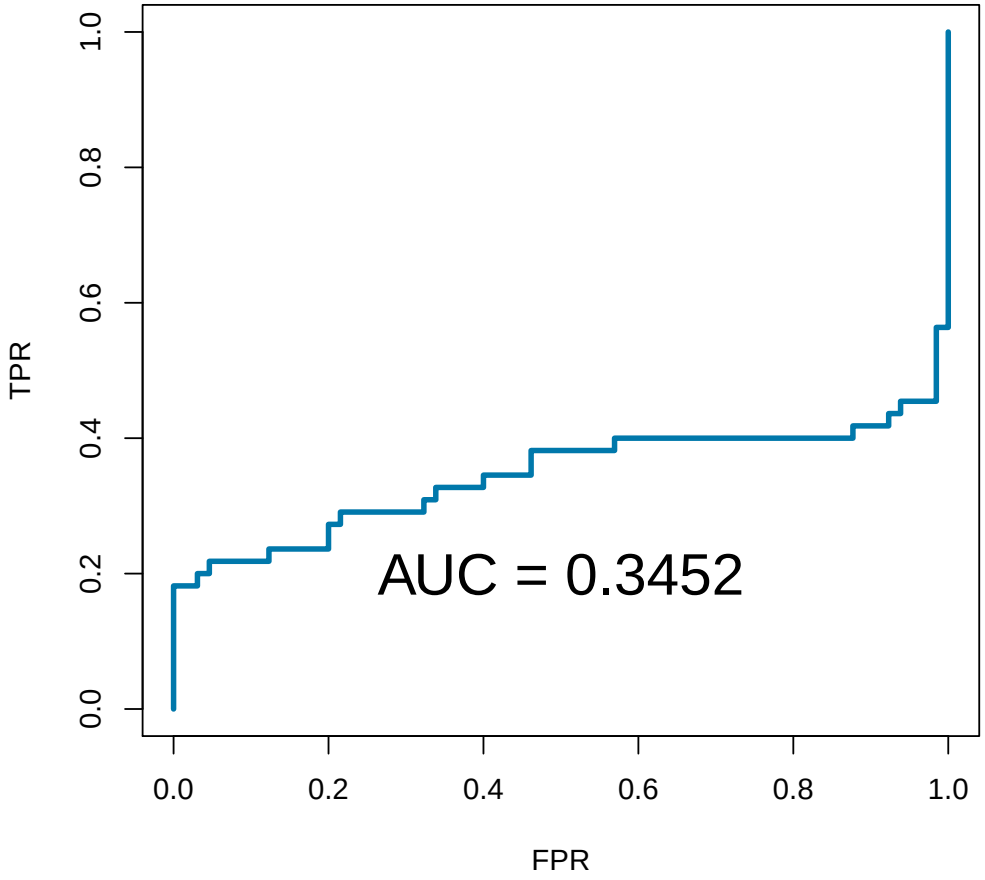
is nothing else but the *Mann-Whitney-Wilcoxon statistic*.

ROC EXAMPLE:

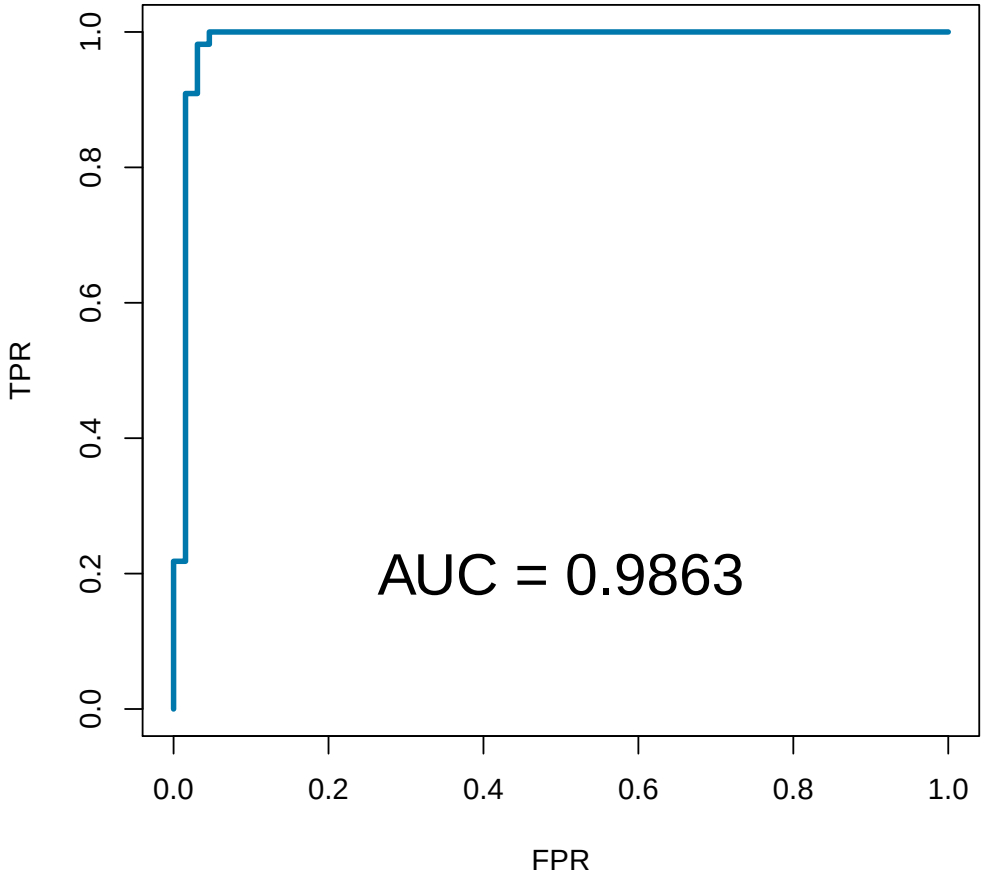
GAUSSIAN CLASSIF. EXAMPLE #2 REVISITED



ROC CURVE FOR $\bar{g}((x_1, x_2)) = x_1$

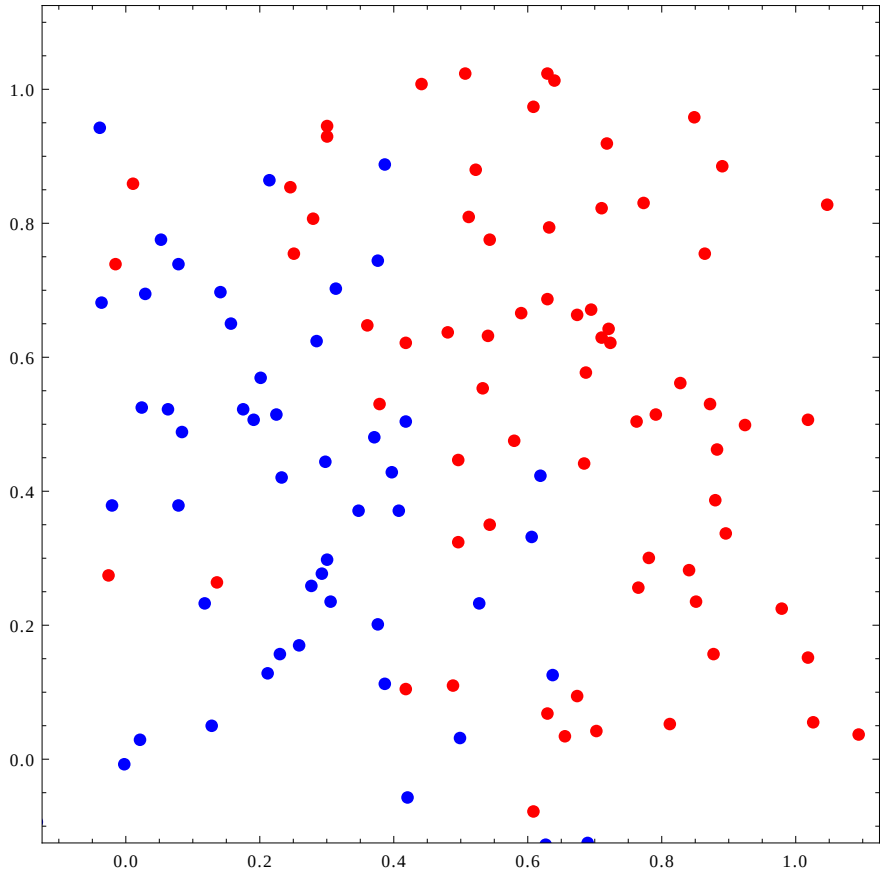


ROC CURVE FOR $\bar{g}((x_1, x_2)) = x_2$



ROC EXAMPLE:

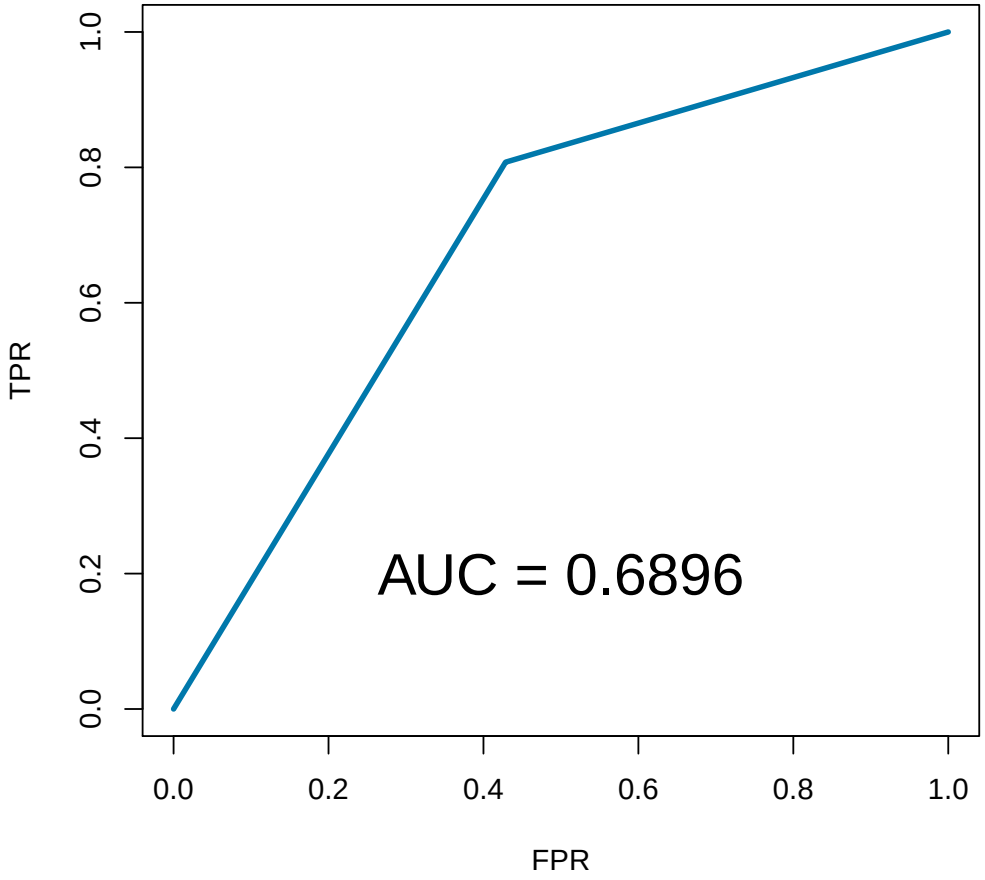
k-NN EXAMPLE #2 REVISITED



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

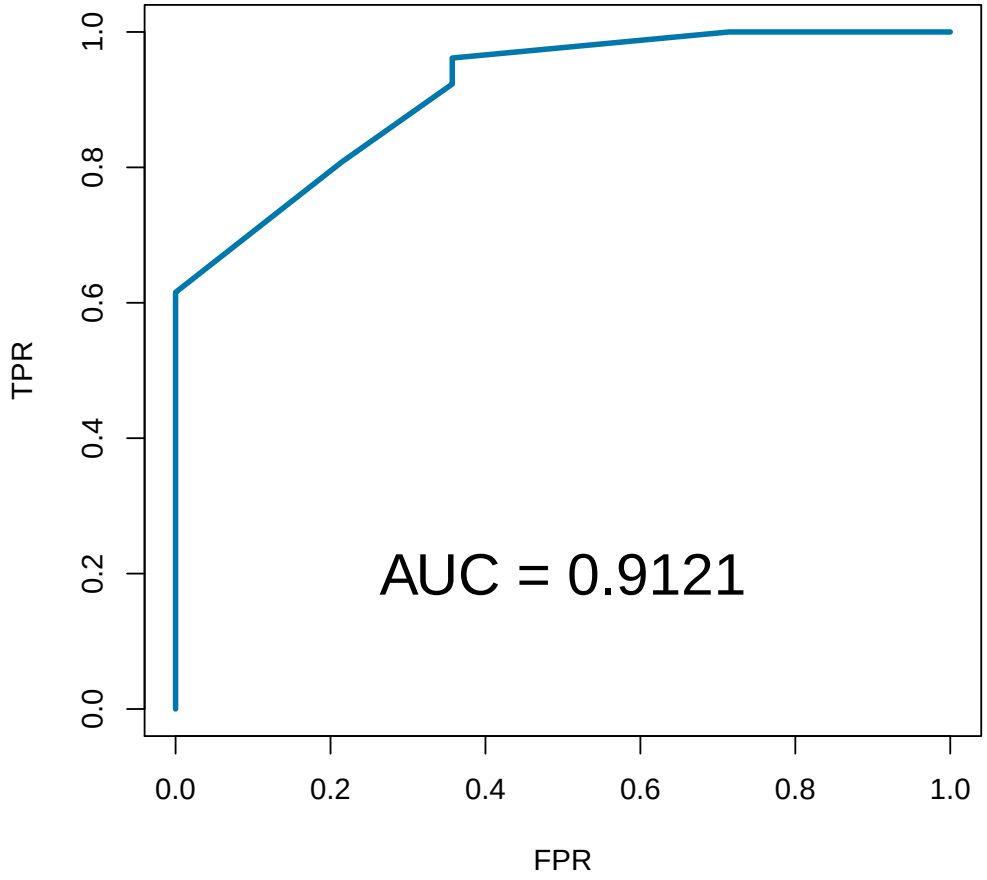
$k = 1$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

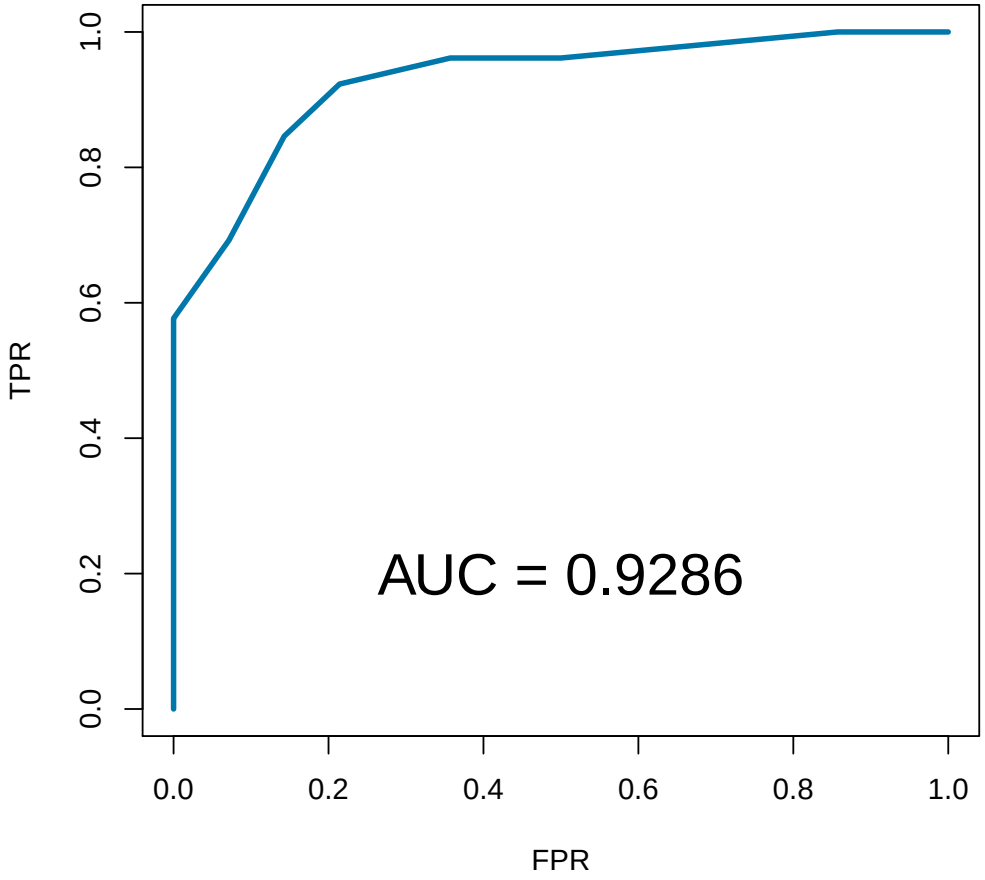
$k = 5$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

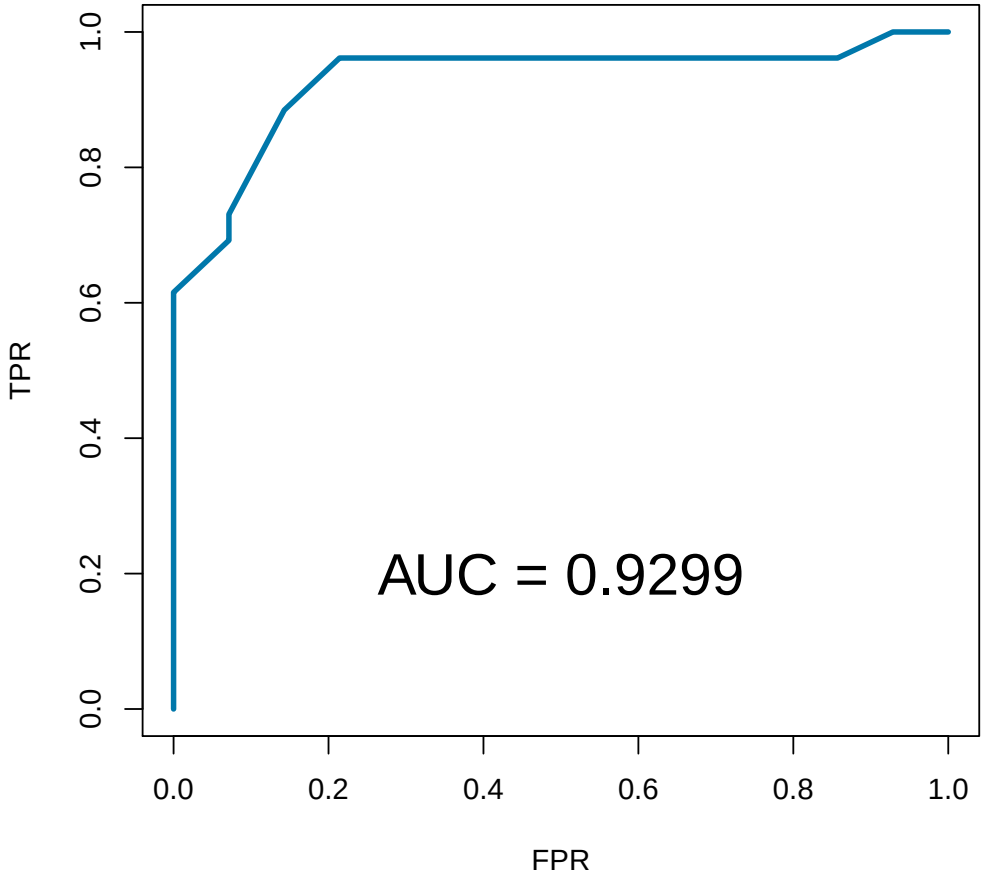
$k = 9$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

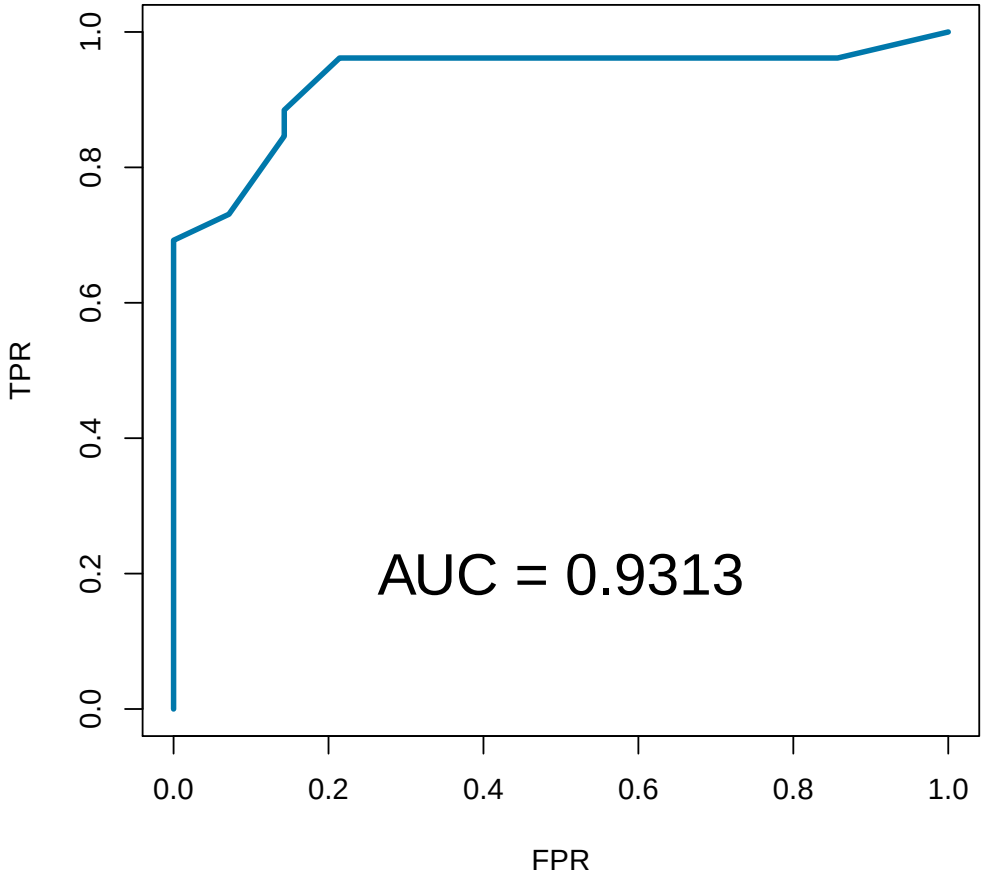
$k = 13$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

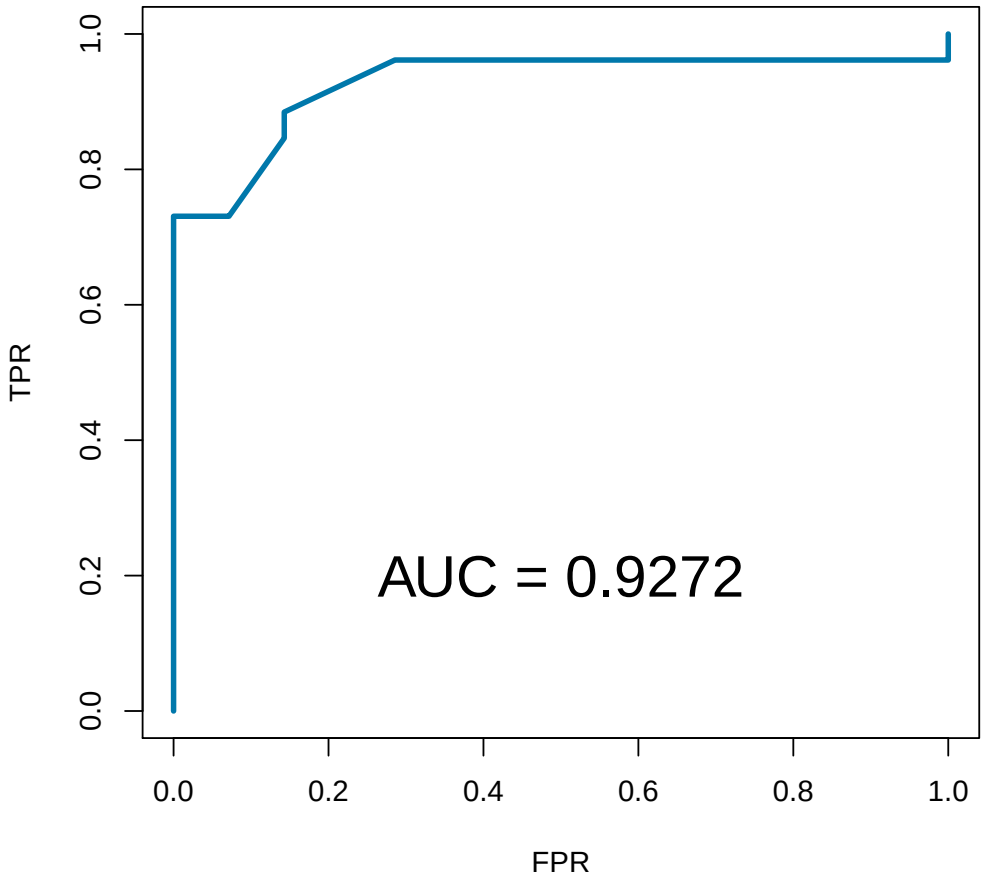
$k = 17$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

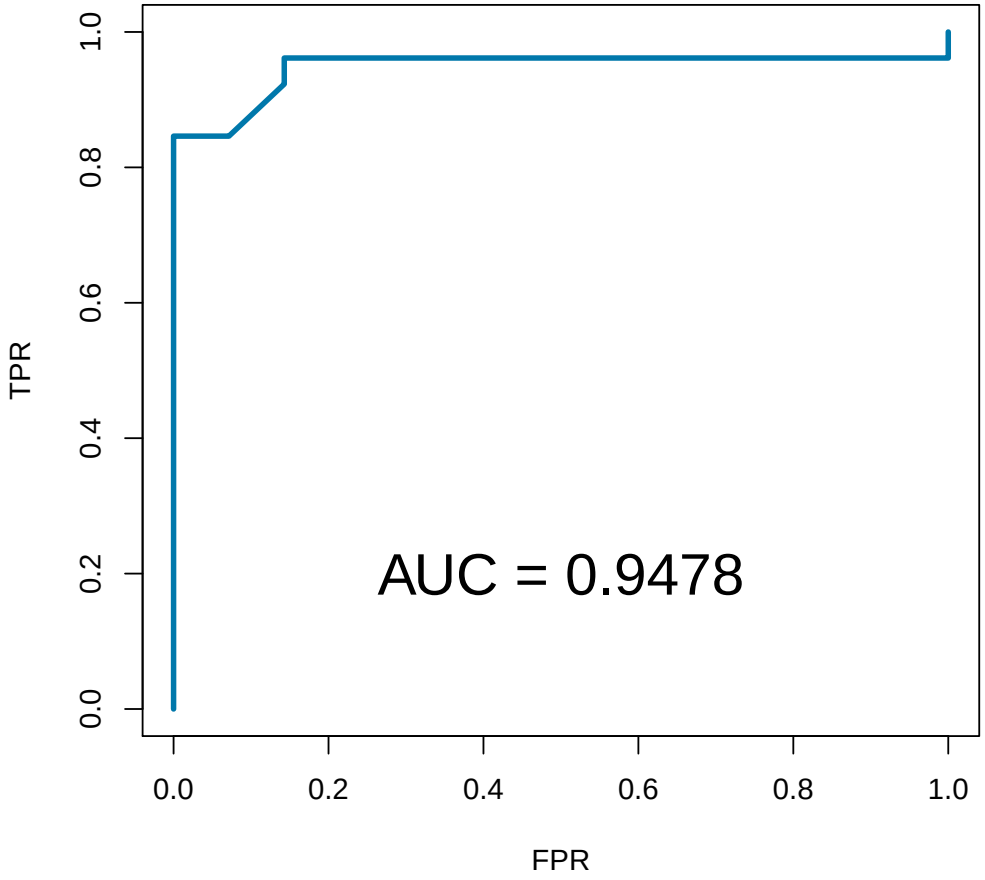
$k = 21$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

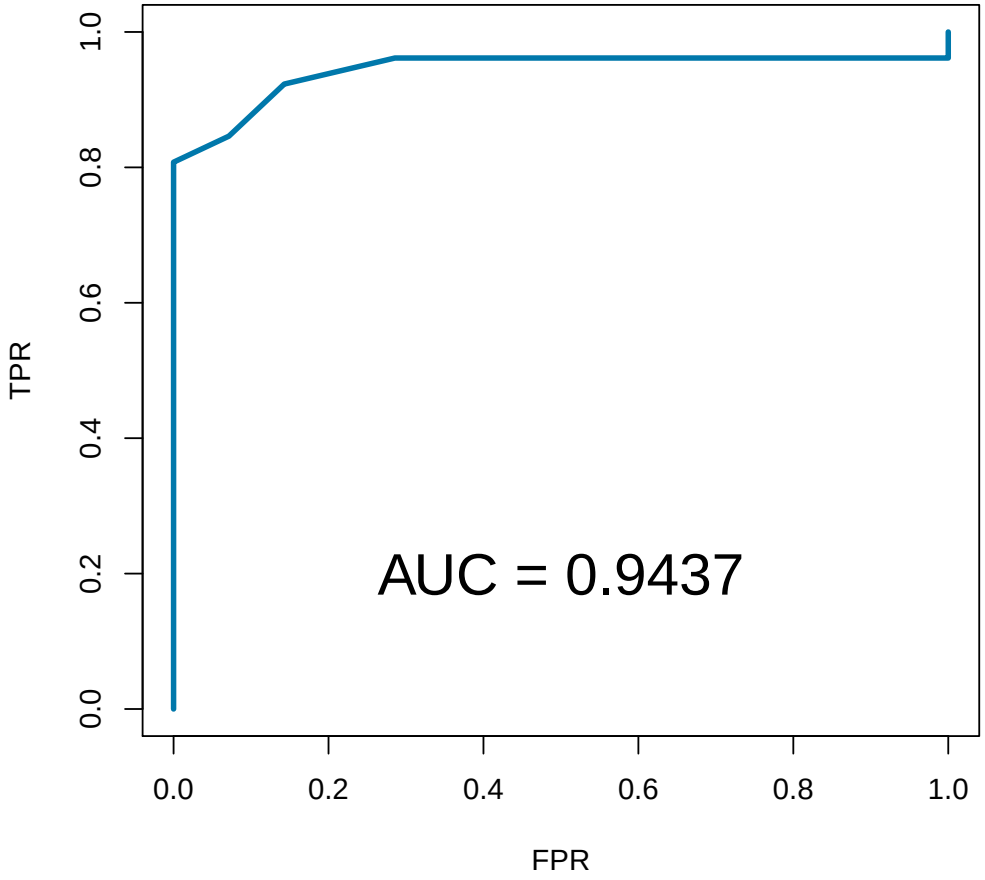
$k = 25$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

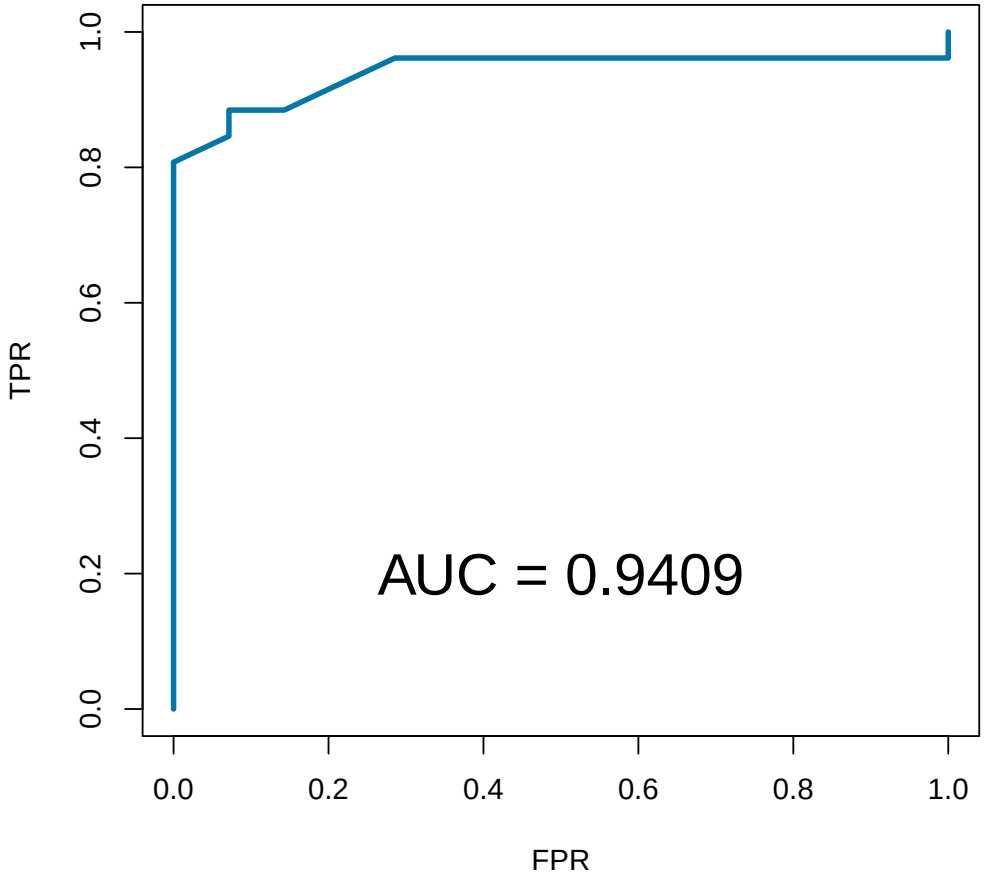
$k = 29$:



ROC CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

$k = 33$:



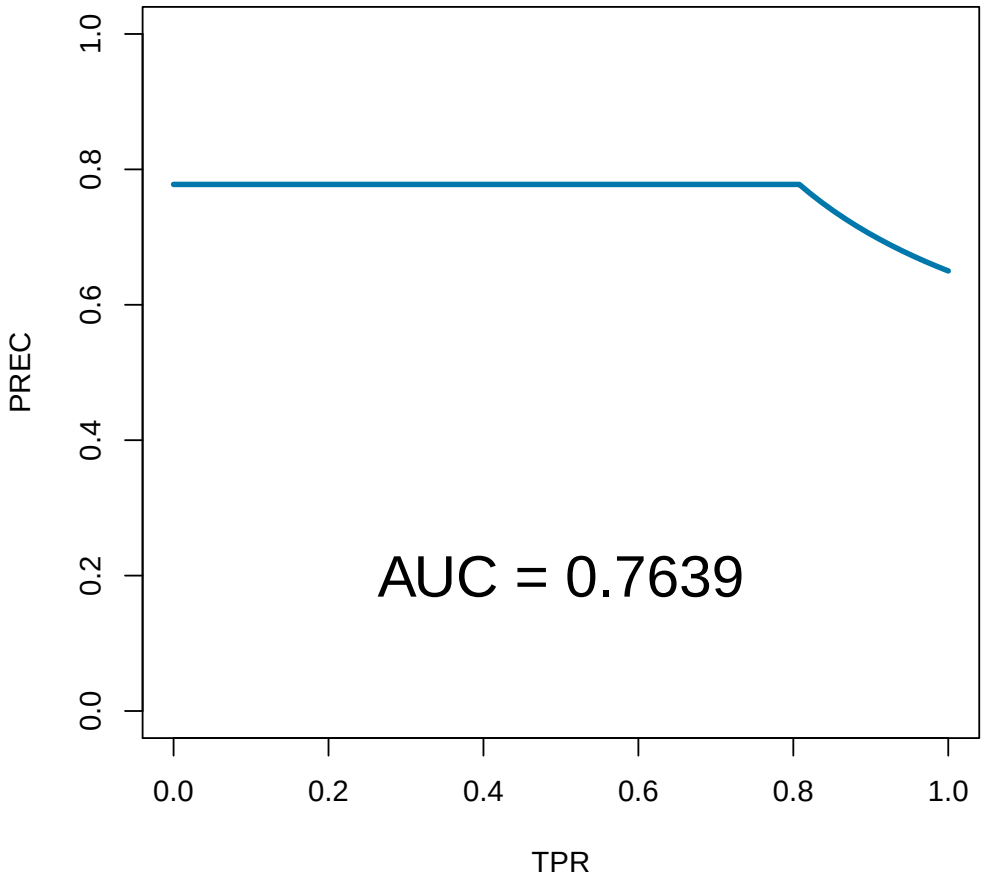
PRECISION-RECALL (PR) CURVES

- For highly unbalanced data sets, in particular, if there are many true negatives, the ROC curves may not necessarily provide a very informative picture.
- For computing a precision-recall curve, similarly to ROC curves, sweep through all possible thresholds, but plot precision (vertical axis) versus recall (horizontal axis)
- The higher the area under the curve, the better the classifier.

PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

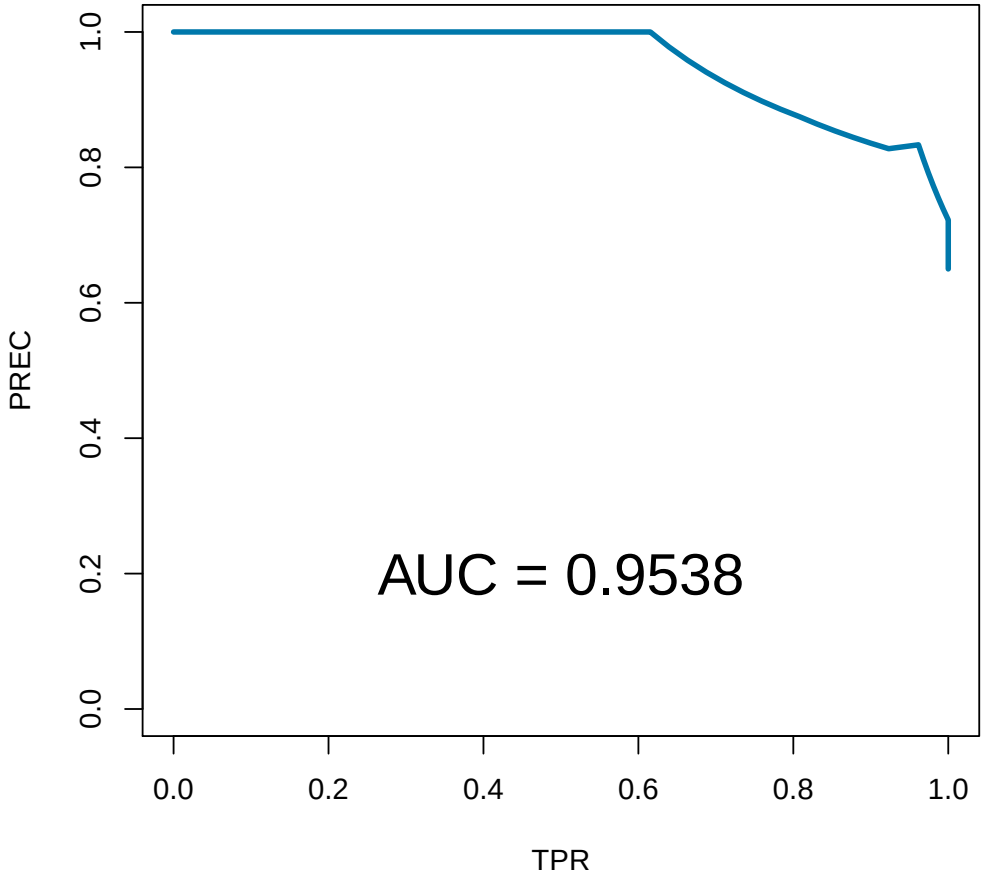
$k = 1$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

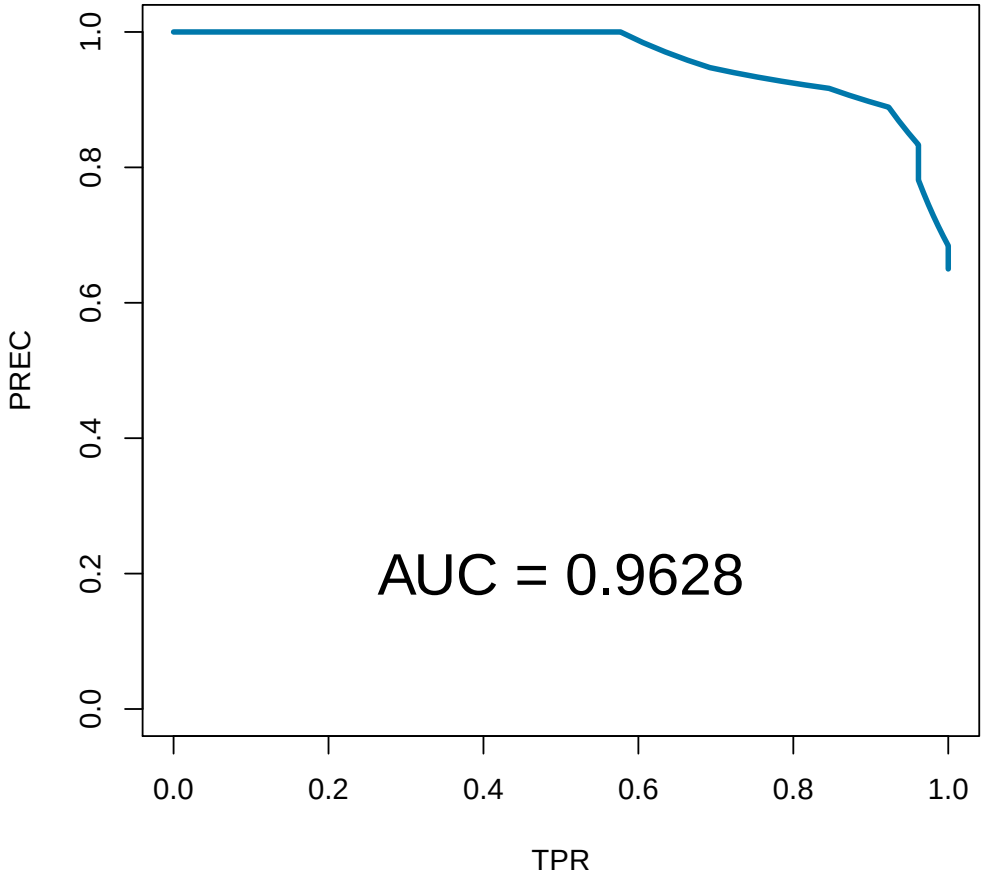
$k = 5$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

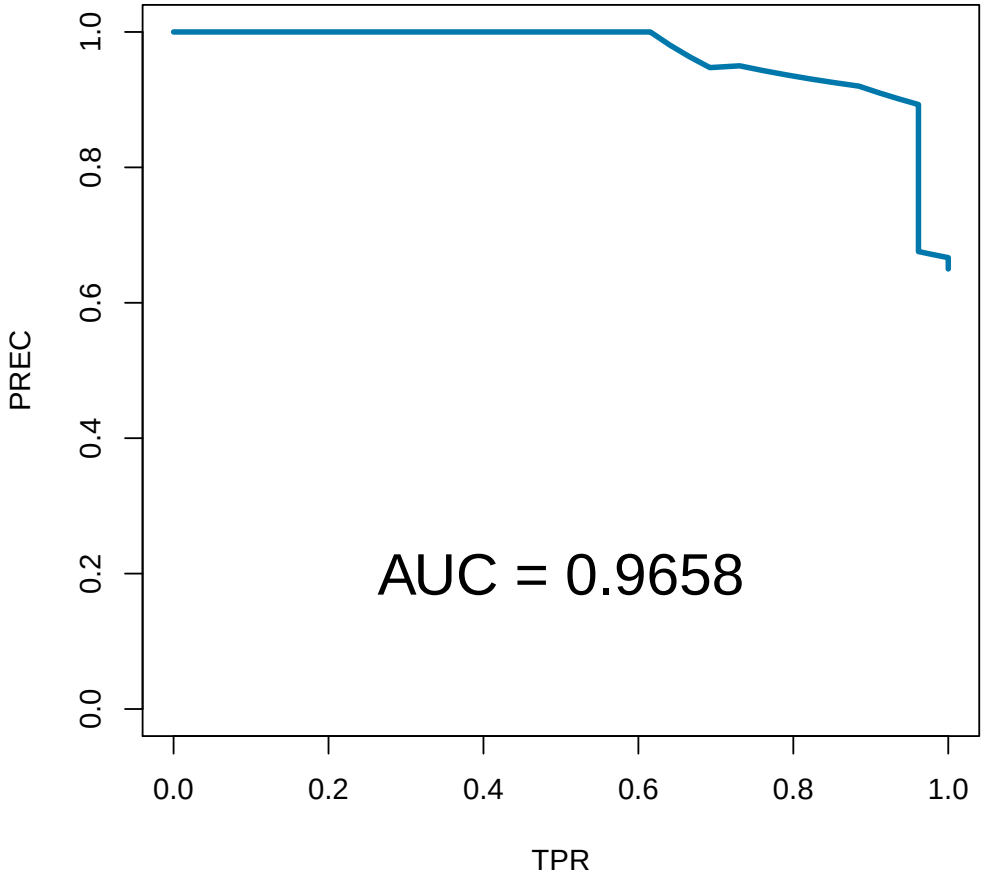
$k = 9$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

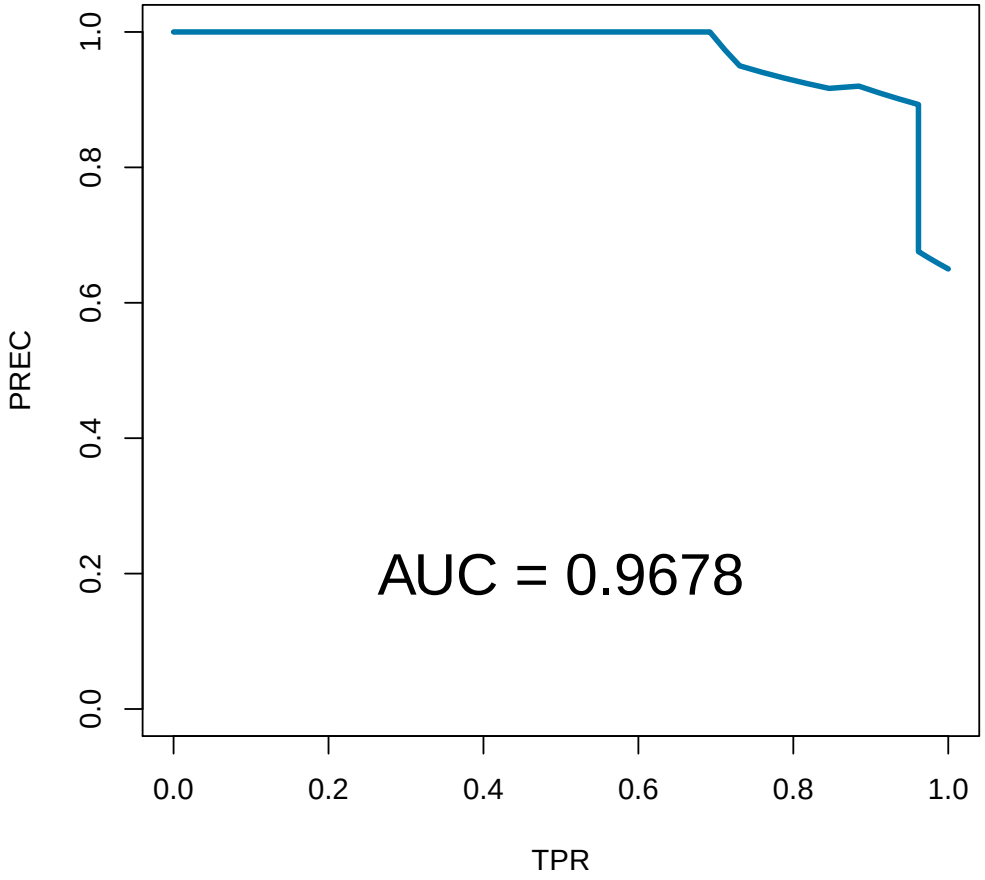
$k = 13$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

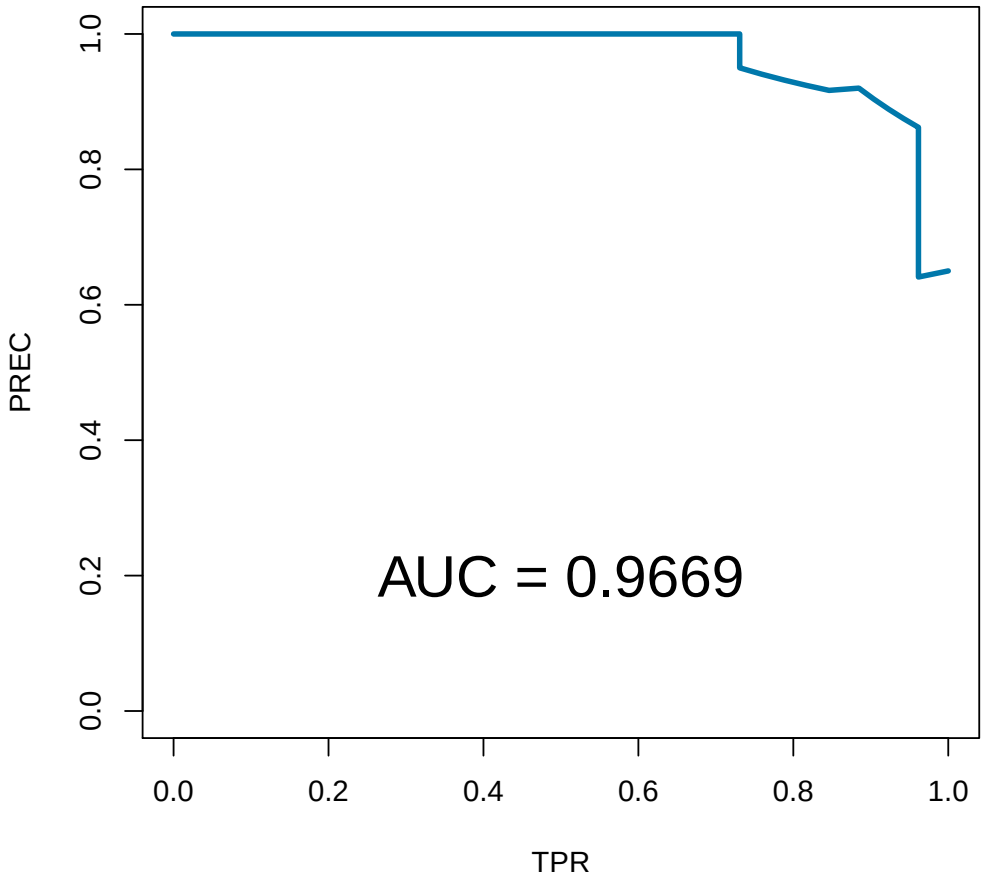
$k = 17$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

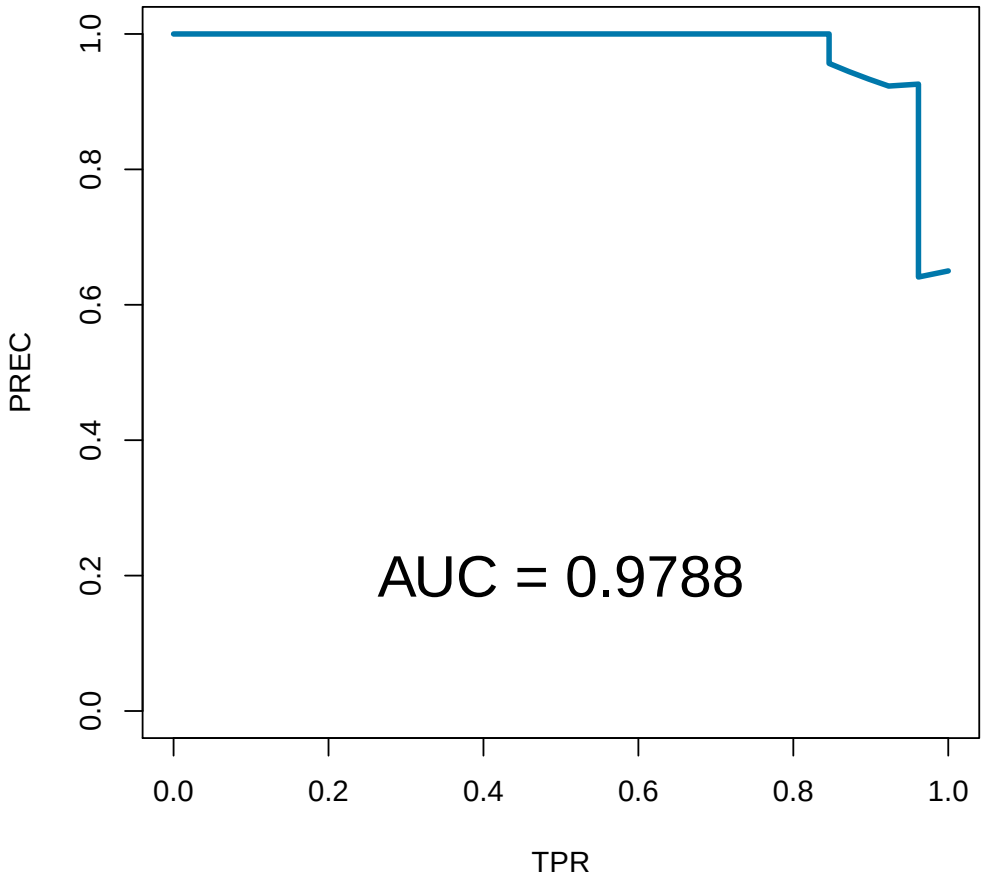
$k = 21$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

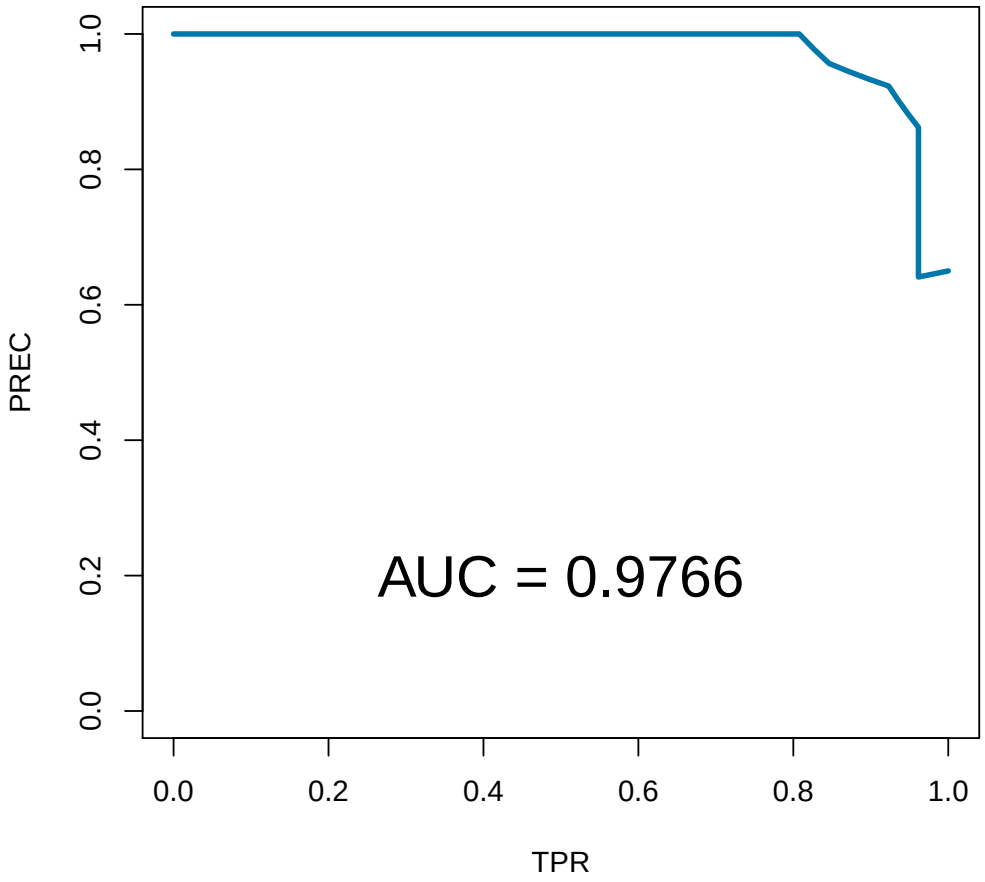
$k = 25$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

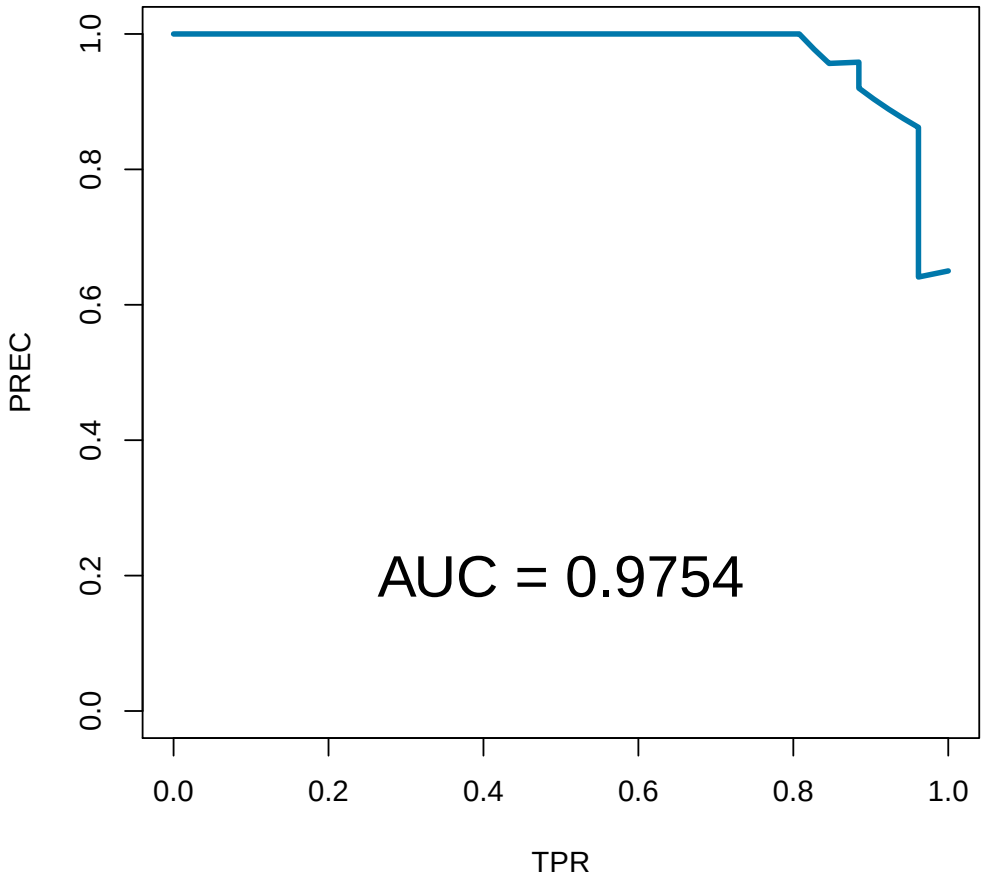
$k = 29$:



PR CURVES FOR k-NN EXAMPLE #2

(75% training, 25% test samples)

$k = 33$:



SUMMARY AND OUTLOOK

In this unit, we have studied the following:

- How to evaluate a given model:
  - Generalization error/risk
  - Estimates via test set method and cross validation
  - Confusion matrices and evaluation measures
  - ROC and PRC analysis
- Simple predictors like k-NN and linear/polynomial regression.
- ERM and the phenomena of underfitting and overtitting.

The following units will be devoted to *state-of-the-art methods for classification and regression.*