
# Unit 2

Model Evaluation in Supervised Machine Learning

# 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 $\varepsilon$ 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 the 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 the Risk for a Gaussian Classification Task (2/4)

Using (1), we can infer

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

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 the Risk for a Gaussian Classification Task (3/4)

Determining an optimal discriminant function:

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

# Minimizing the 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: Distributions

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

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

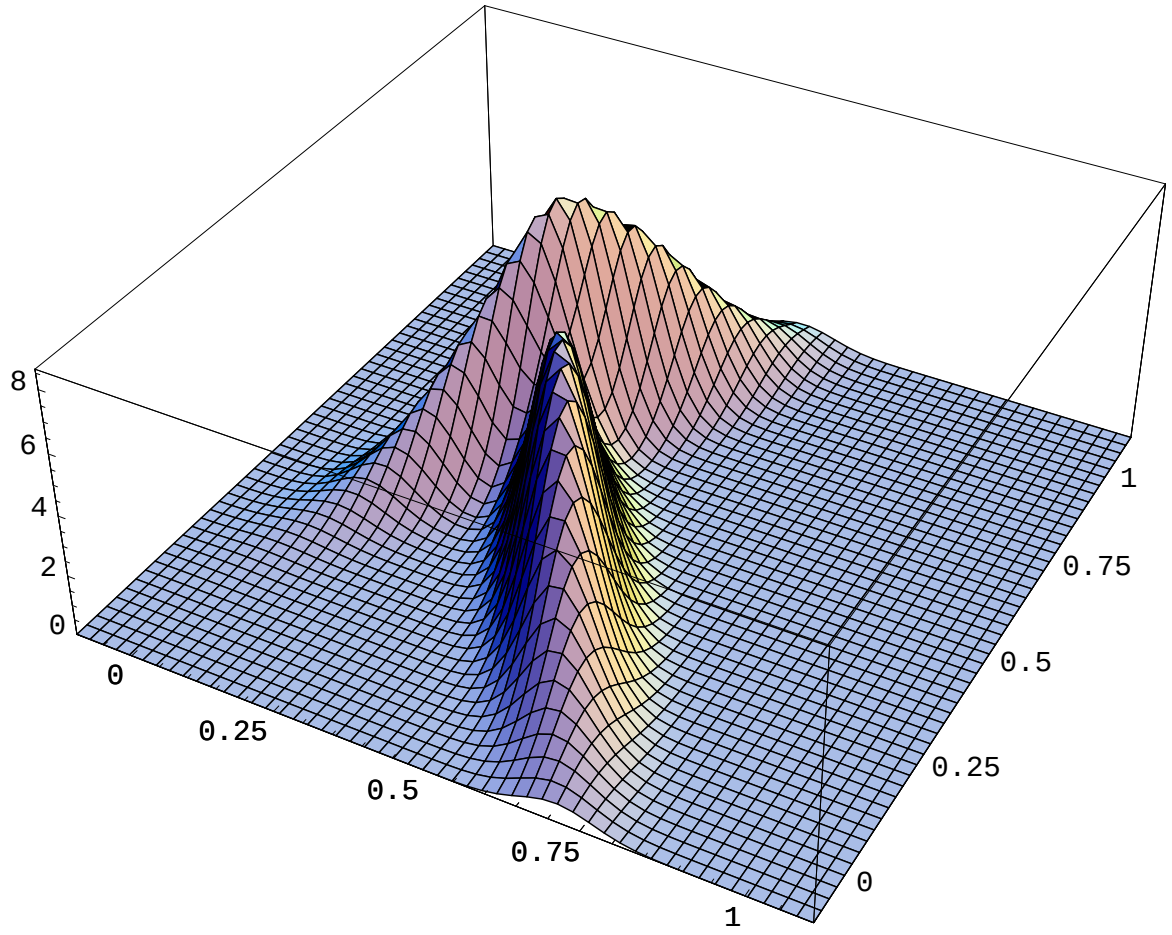
- $p(\mathbf{x} \mid y = -1)$ corresponds to a two-variate normal distribution with parameters

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

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

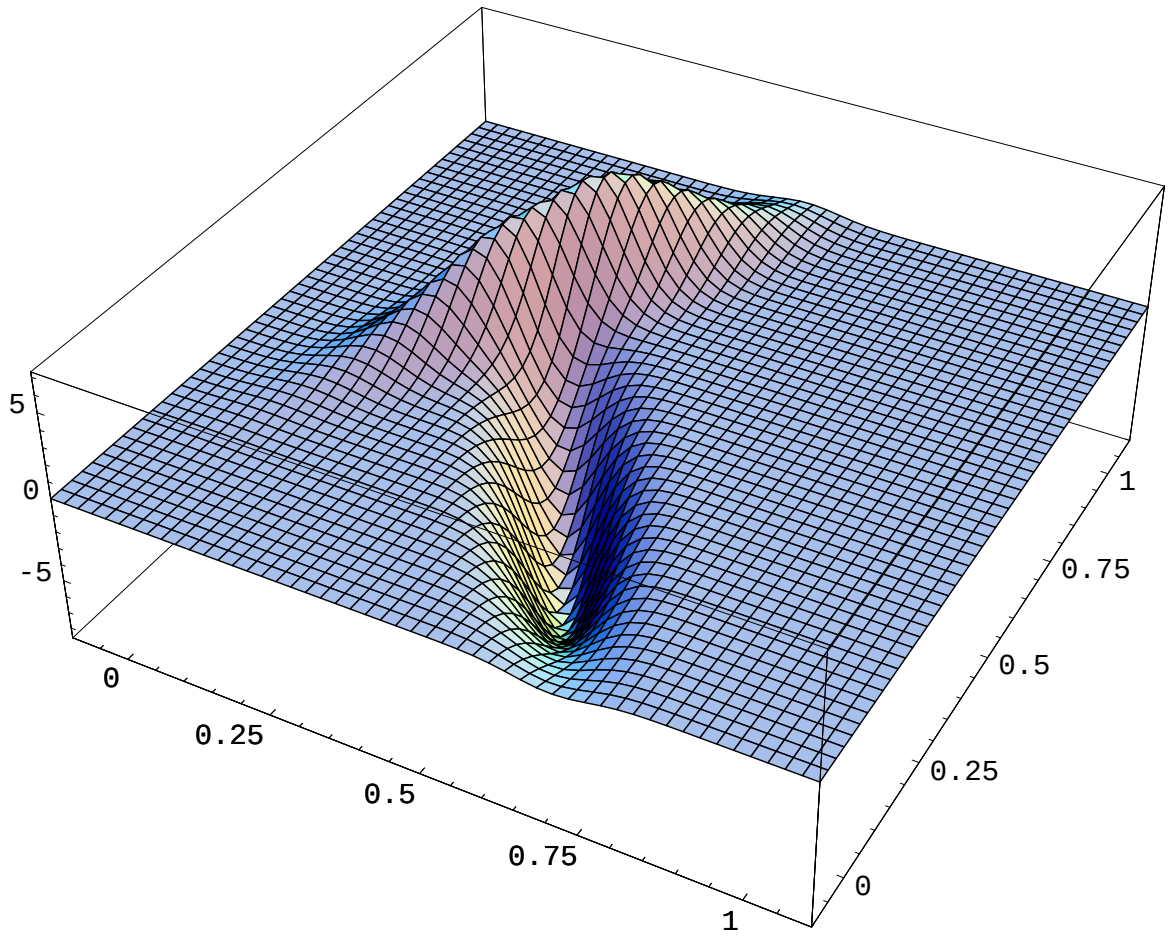
# Gaussian Classification Example #1:

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

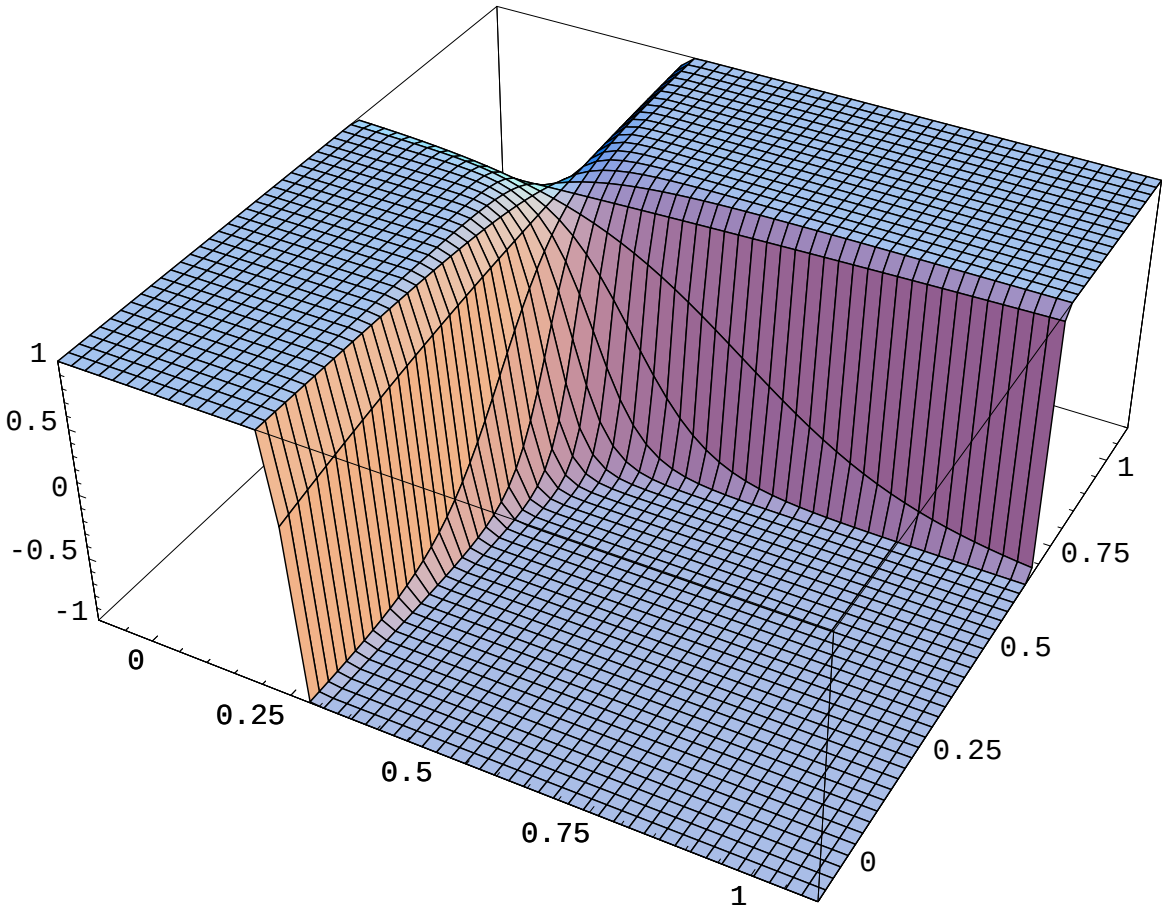


# Gaussian Classification Example #1:

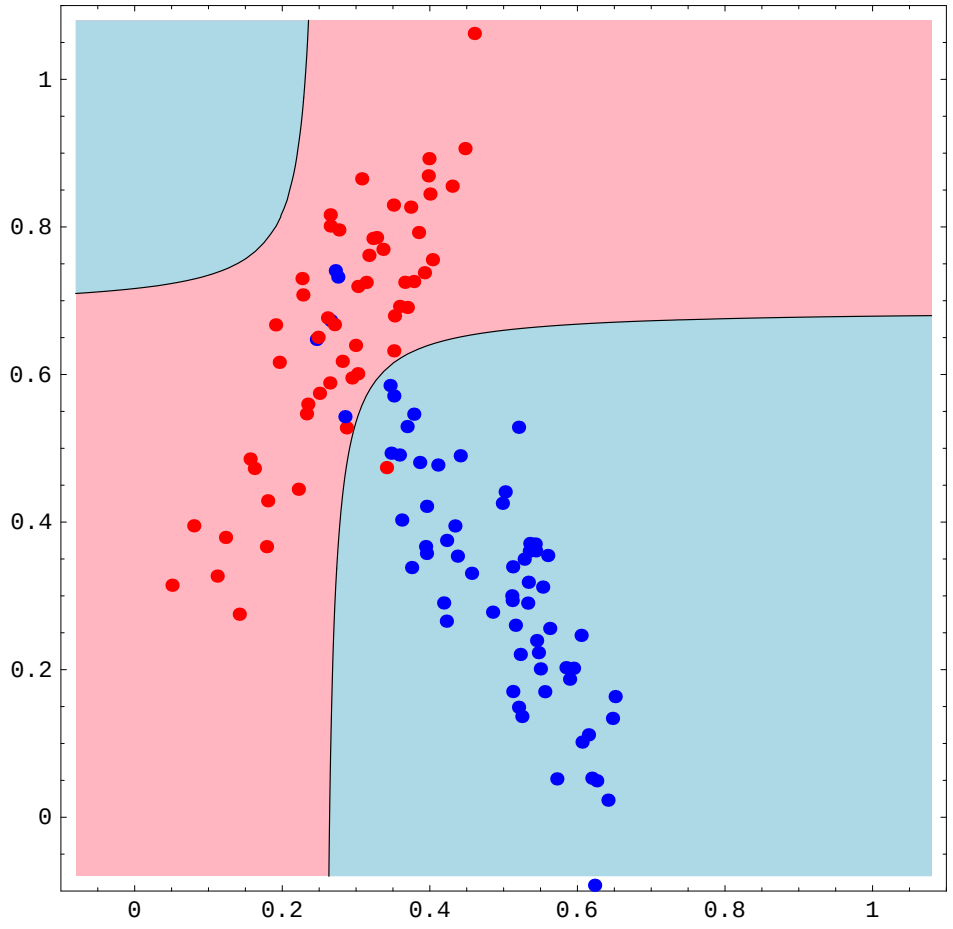
$$\tilde{g}(\mathbf{x}) = p(\mathbf{x} \mid y = +1) \cdot p(y = +1) - p(\mathbf{x} \mid y = -1) \cdot p(y = -1)$$



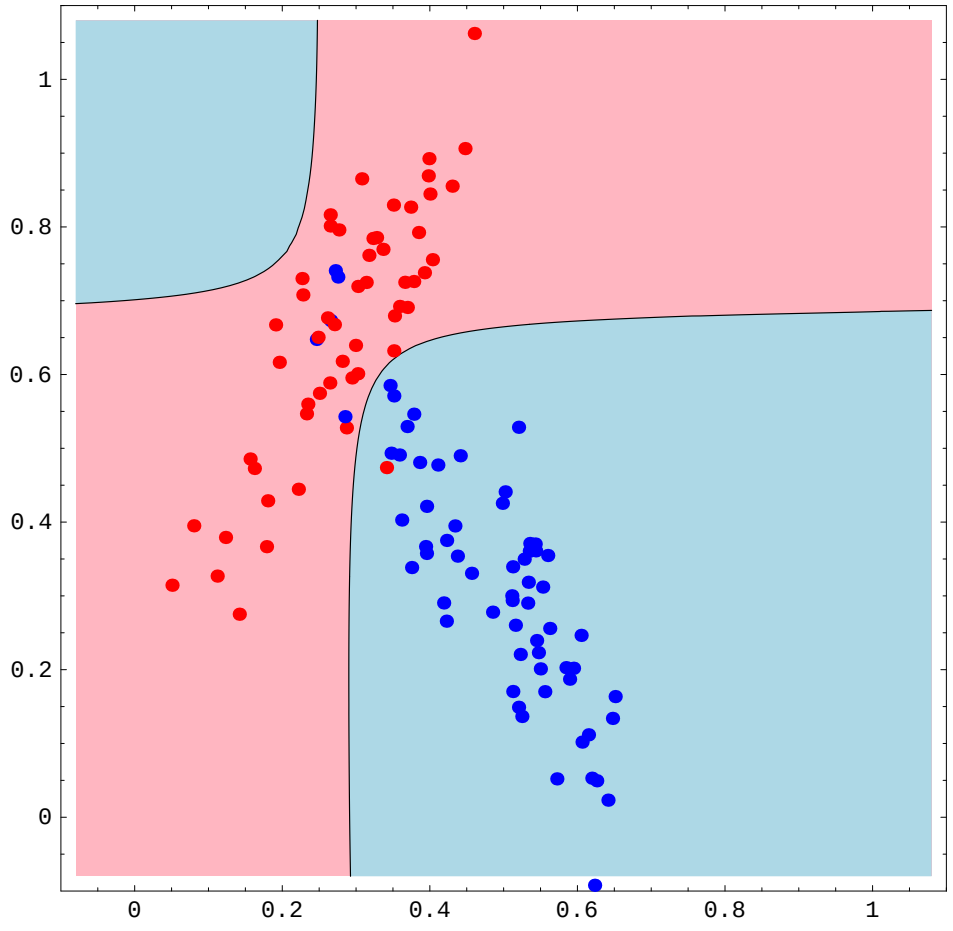
# Gaussian Classification Example #1: Discriminant Function $\bar{g}(\mathbf{x}) = \tilde{g}(\mathbf{x})/p(\mathbf{x})$



# Gaussian Classification Example #1: Data + Optimal Decision Border



# Gaussian Classification Example #1: Data + Estimated Decision Border



# Gaussian Classification Example #2: Distributions

Data set no. 6 (cf. exercises) was created according to the following distributions:

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

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

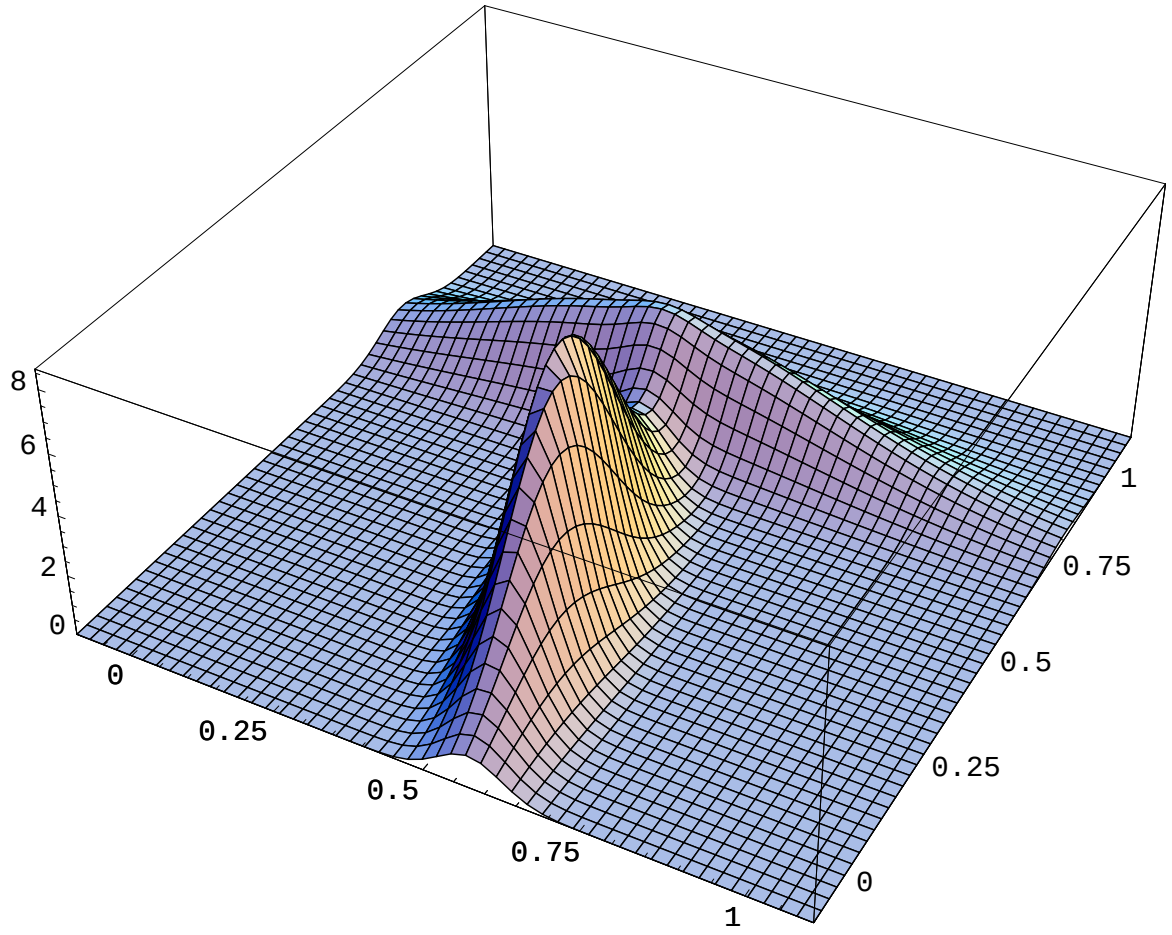
- $p(\mathbf{x} \mid y = -1)$ corresponds to a two-variate normal distribution with parameters

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

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

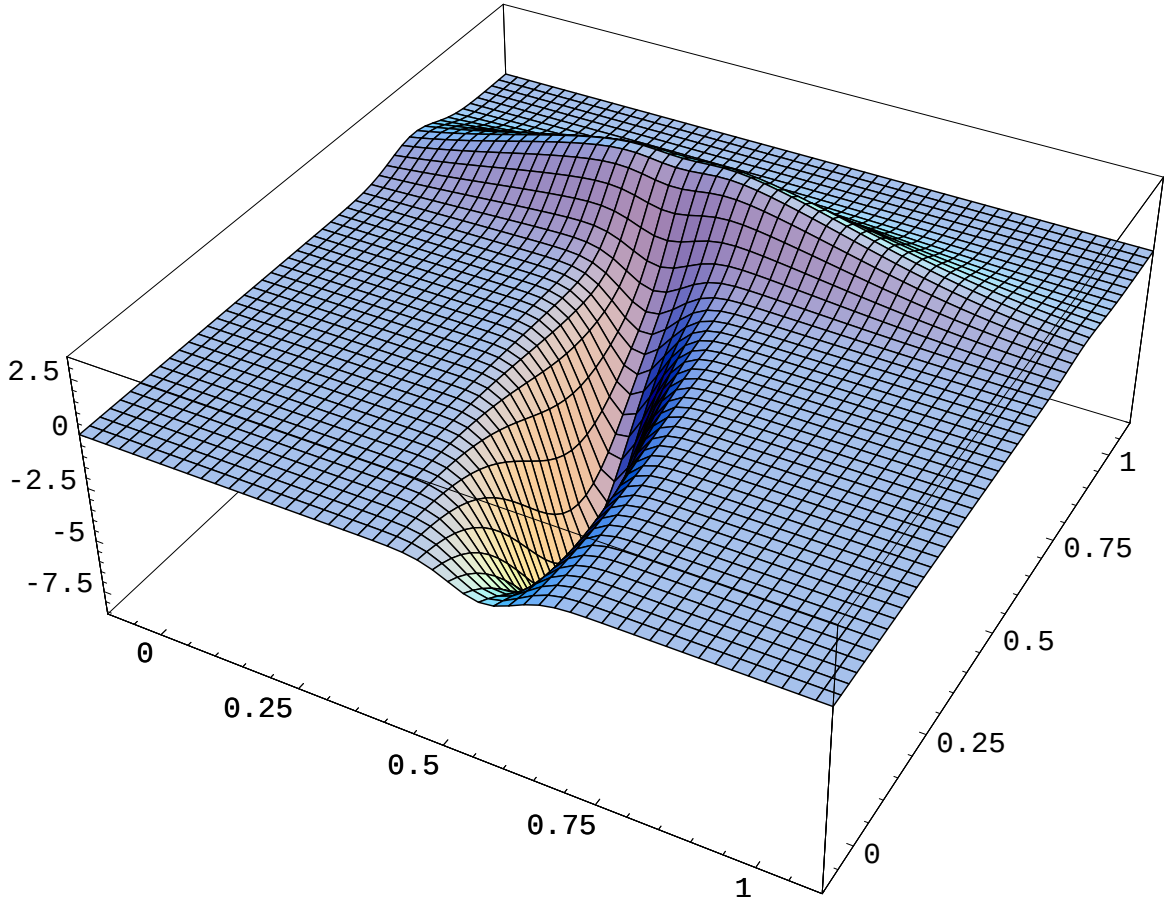
# Gaussian Classification Example #2:

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

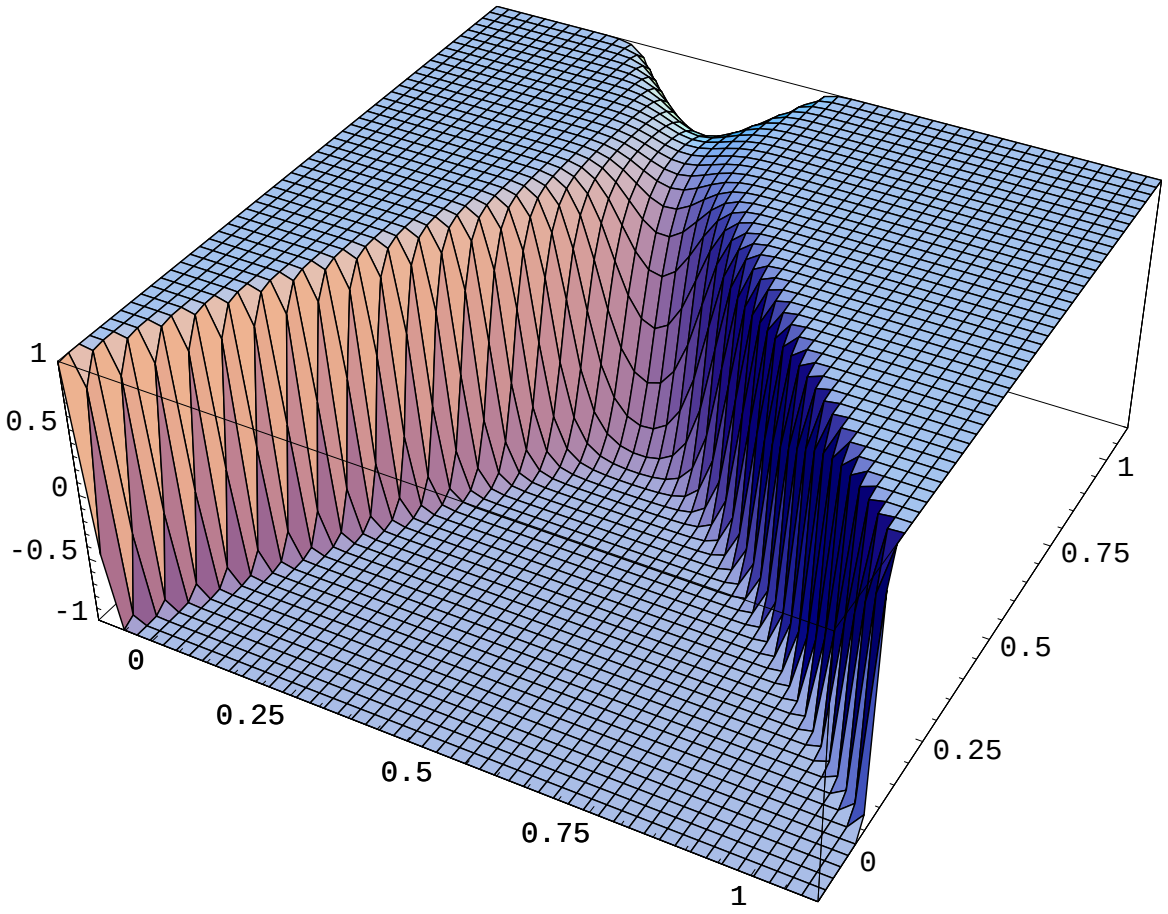


# Gaussian Classification Example #2:

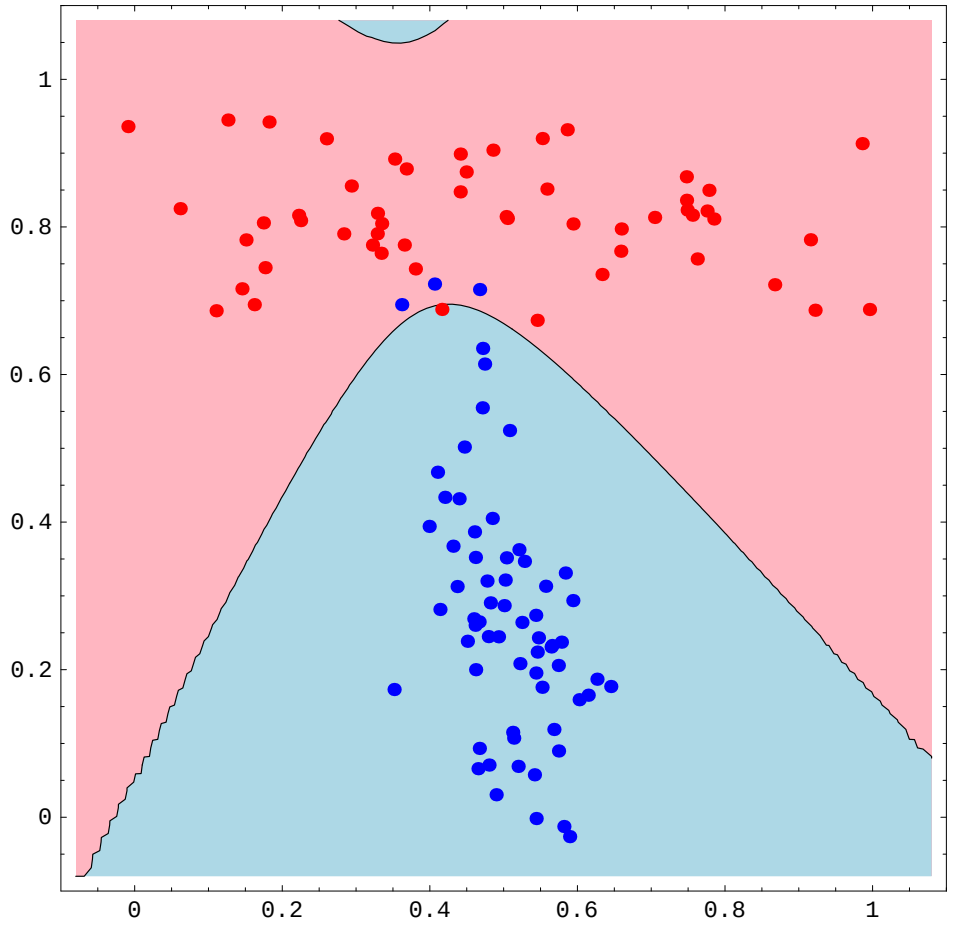
$$\tilde{g}(\mathbf{x}) = p(\mathbf{x} \mid y = +1) \cdot p(y = +1) - p(\mathbf{x} \mid y = -1) \cdot p(y = -1)$$



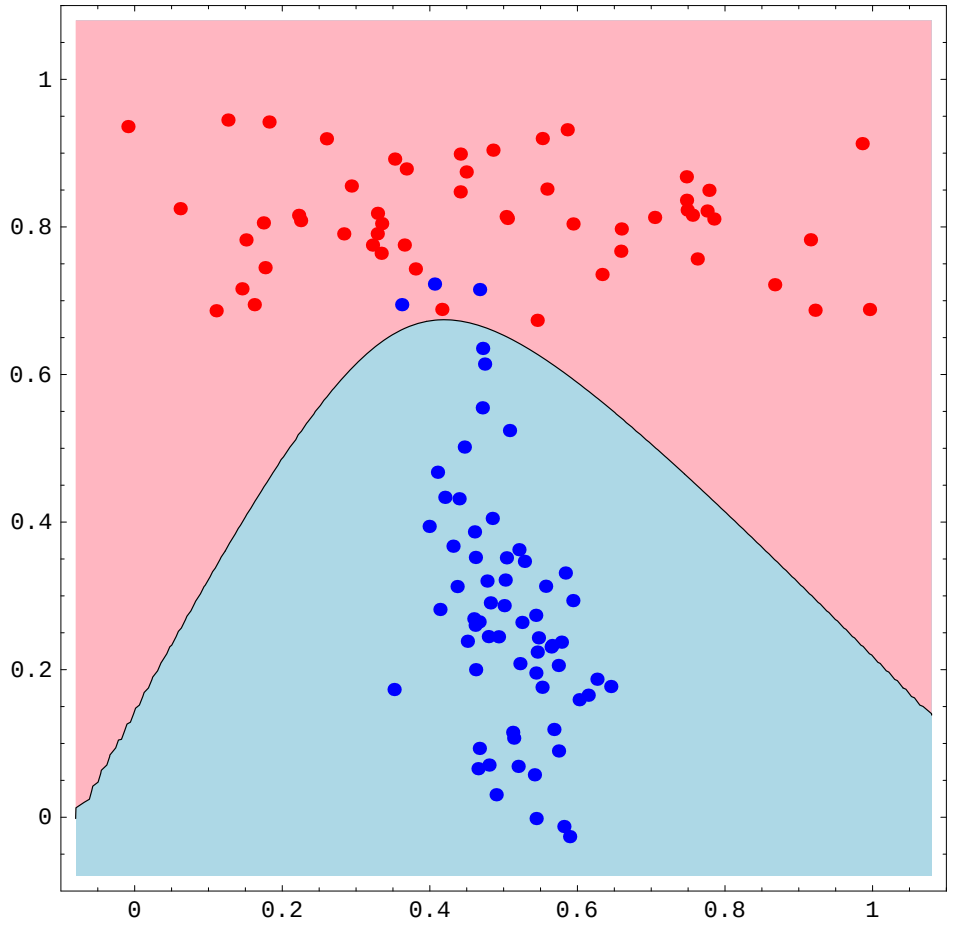
# Gaussian Classification Example #2: Discriminant Function $\bar{g}(\mathbf{x}) = \tilde{g}(\mathbf{x})/p(\mathbf{x})$



# Gaussian Classification Example #2: Data + Optimal Decision Border



# Gaussian Classification Example #2: Data + Estimated Decision Border



# Gaussian Classification Example #3: Distributions

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

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

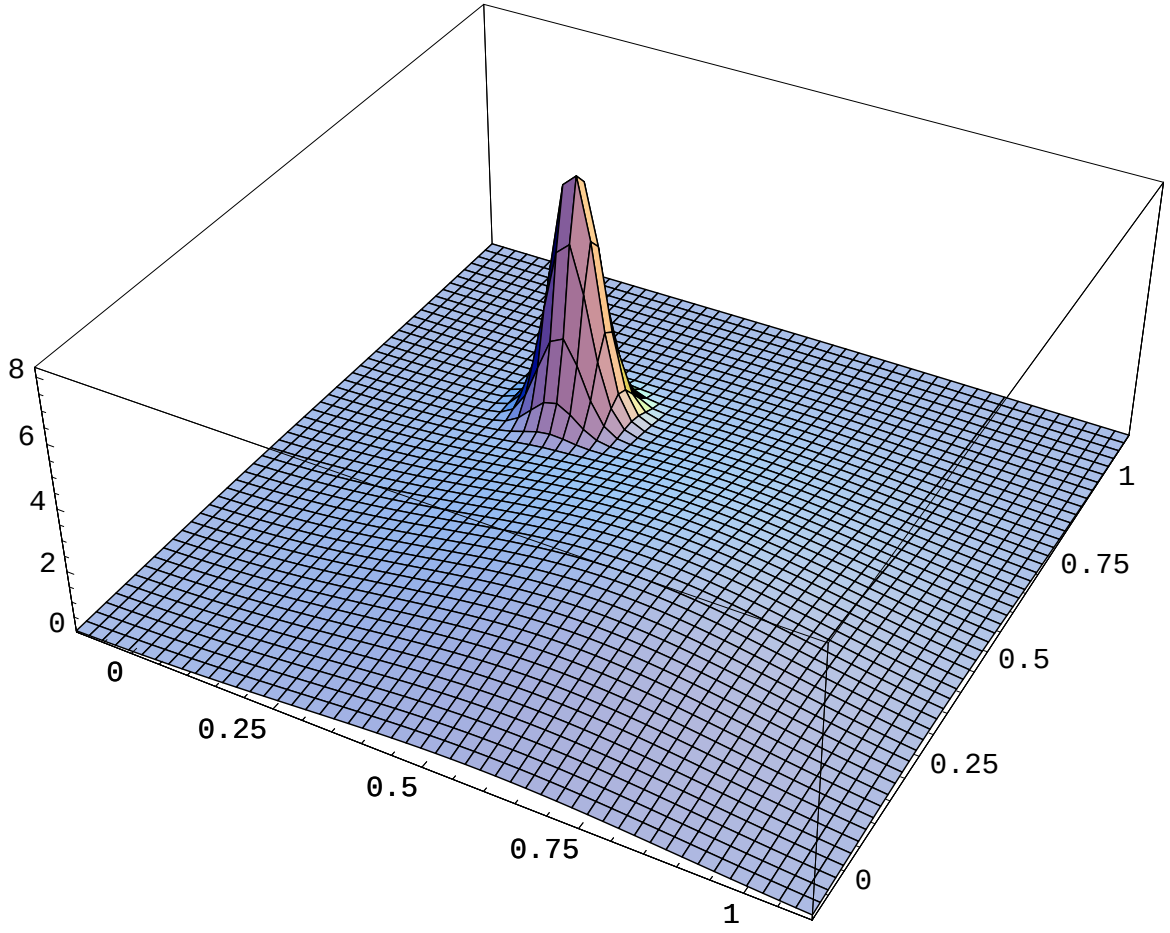
- $p(\mathbf{x} \mid y = -1)$ corresponds to a two-variate normal distribution with parameters

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

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

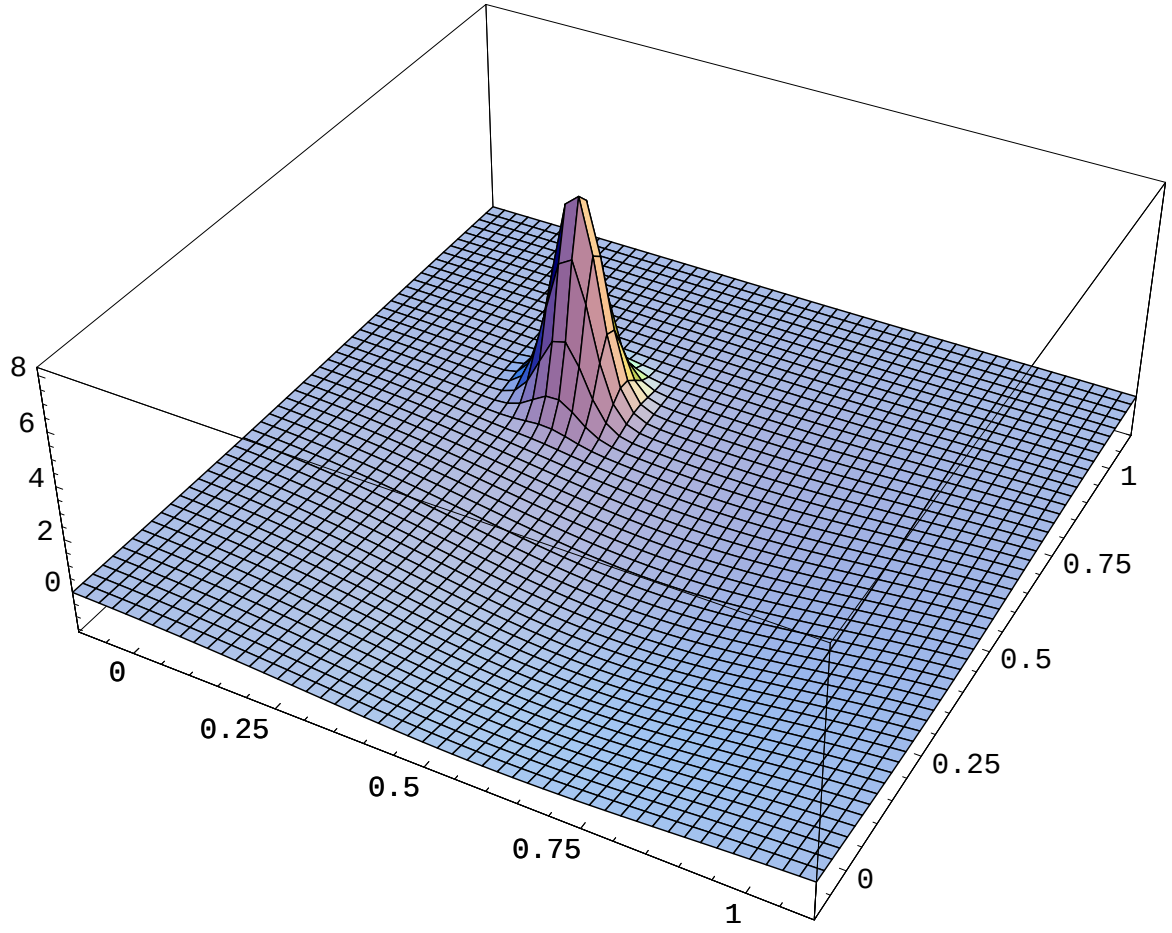
# Gaussian Classification Example #3:

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

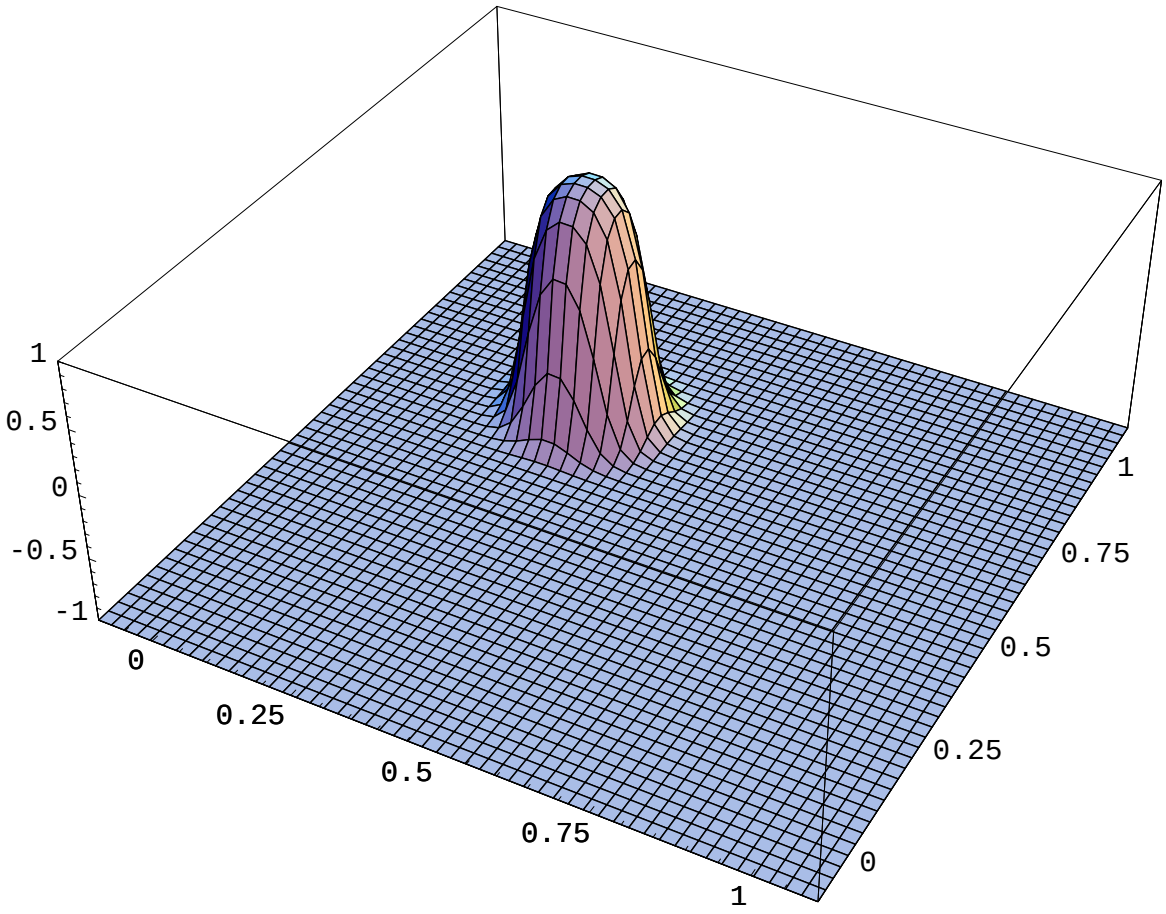


# Gaussian Classification Example #1:

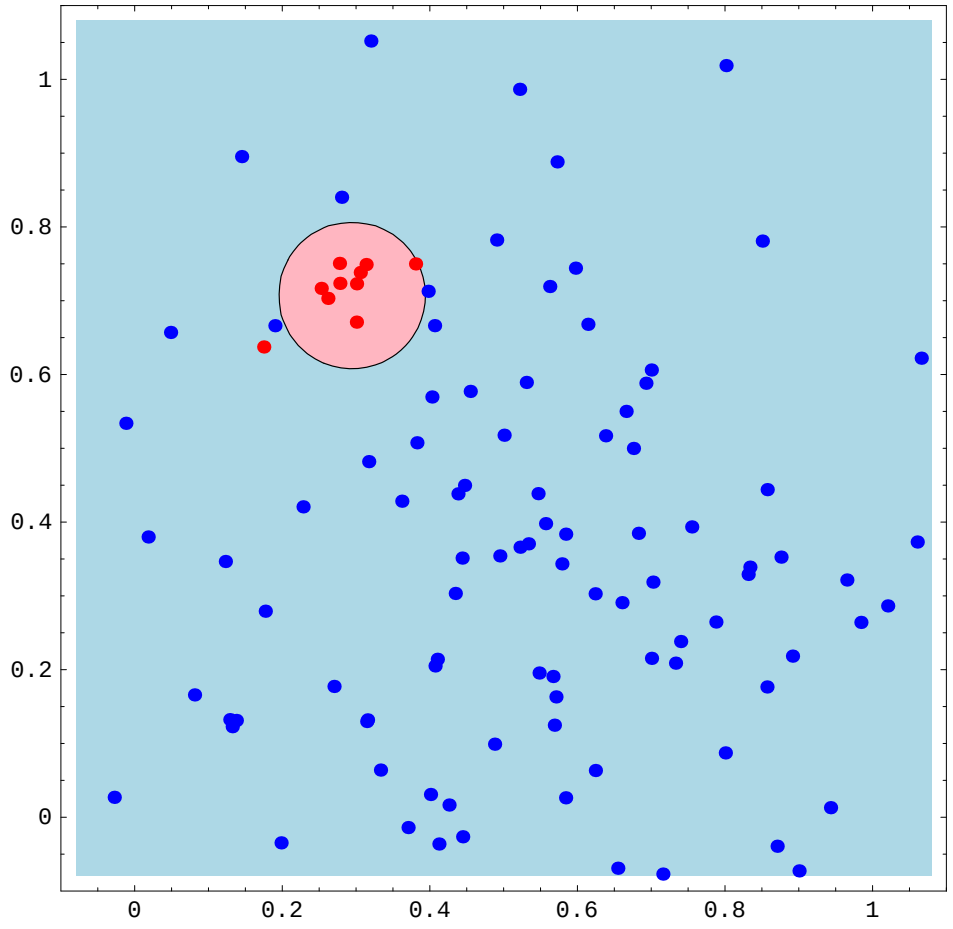
$$\tilde{g}(\mathbf{x}) = p(\mathbf{x} \mid y = +1) \cdot p(y = +1) - p(\mathbf{x} \mid y = -1) \cdot p(y = -1)$$



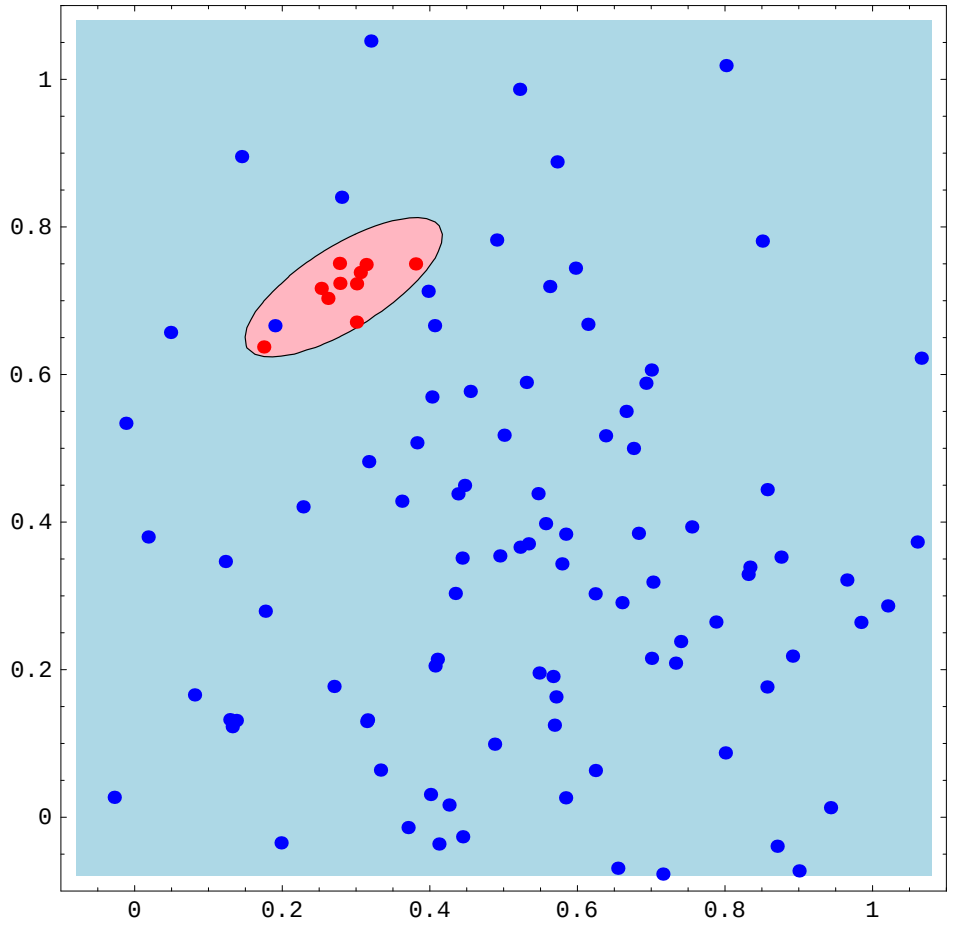
# Gaussian Classification Example #1: Discriminant Function $\bar{g}(\mathbf{x}) = \tilde{g}(\mathbf{x})/p(\mathbf{x})$



# Gaussian Classification Example #3: Data + Optimal Decision Border



# Gaussian Classification Example #3: 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, we can only *estimate* the generalization error

# Estimating the Risk: Test Set Method

Assume that we have $m$ more data samples $(\mathbf{z}^{l+1}, \ldots, \mathbf{z}^{l+m})$, 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

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

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?*

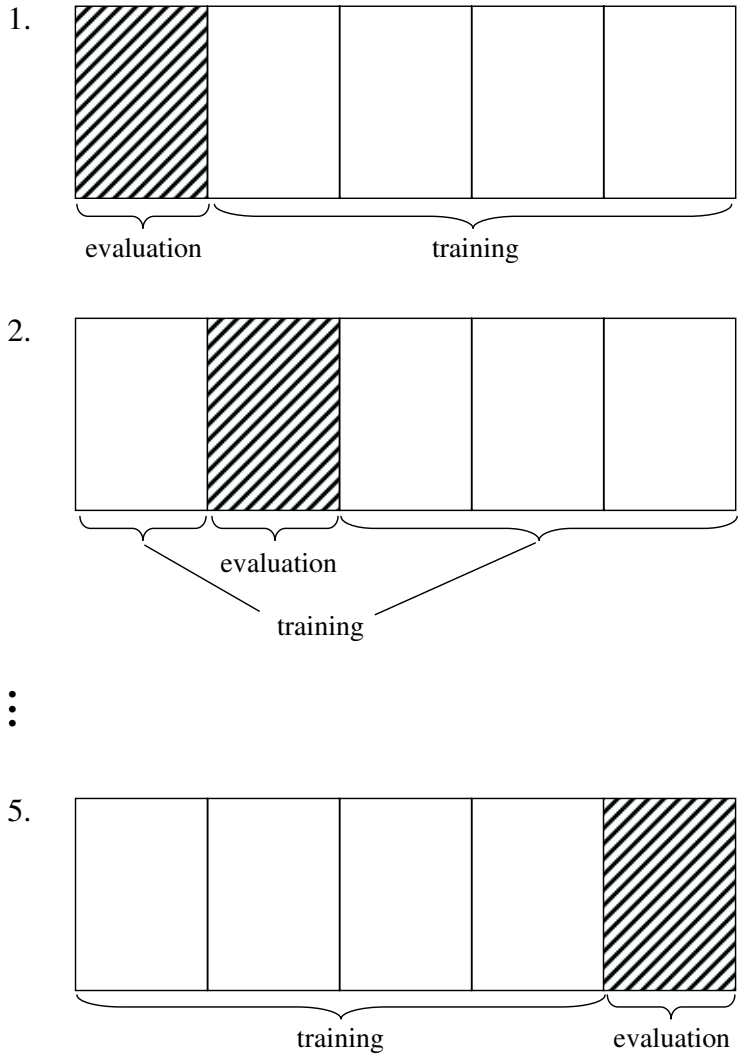
# Cross Validation: Basic Idea

- 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

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

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

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

# 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 hyperparameters*, 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.

# 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 a Binary Classification Task

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

**true positive (TP)** if $y = +1$ and $g(\mathbf{x}; \mathbf{w}) = +1$,

**true negative (TN)** if $y = -1$ and $g(\mathbf{x}; \mathbf{w}) = -1$,

**false positive (FP)** if $y = -1$ and $g(\mathbf{x}; \mathbf{w}) = +1$,

**false negative (FN)** if $y = +1$ and $g(\mathbf{x}; \mathbf{w}) = -1$.

# Confusion Matrix for a Binary Classification Task (cont'd)

Given a test 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 test data set.

# Evaluation Measures Derived from the Confusion Matrix

**Accuracy:** number 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 Especially 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:** (weighted) harmonic mean of precision and recall, i.e.

$$F_1 = 2 \cdot \frac{\text{PREC} \cdot \text{TPR}}{\text{PREC} + \text{TPR}} \qquad F_\beta = (1 + \beta^2) \cdot \frac{\text{PREC} \cdot \text{TPR}}{\beta^2 \cdot \text{PREC} + \text{TPR}}$$

# Confusion Matrix for a Multi-Class Classification Task

Assume that we have a $k$-class classification task. Given a test data set $(\mathbf{z}^1, \ldots, \mathbf{z}^m)$, the *confusion matrix* is defined as follows:

| | | predicted class $g(\mathbf{x}; \mathbf{w})$ | | | | |
|---|---|---|---|---|---|---|
| | | $1$ | $\cdots$ | $j$ | $\cdots$ | $k$ |
| 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 $g(.; \mathbf{w})$.

# Accuracy for a Multi-Class Classification Task

For a multi-class classification task (with the notations as on the previous slide), the *accuracy* of a classifier $g(.; \mathbf{w})$ 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 classifier samples*. The other evaluation measures cannot be generalized to the multi-class case in a straightforward way.

# Other Performance Measures for Multi-Class Classification Task

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 Revisited: The Asymmetric Case (1/4)

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 Revisited: The Asymmetric Case (2/4)

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 Revisited: The Asymmetric Case (3/4)

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 $\theta$ (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 Curve Algorithm

**Input:** list of class labels of test samples $(\tilde{y}^1, \ldots, \tilde{y}^m)$ that were first sorted in ascending order according to their discriminant value; no. of positive samples in test set: $\#p$, no. of negative samples: $\#n$;

$p := 0;$
$n := 0;$
$(x', y') := (0, 0);$

**FOR** $i := m$ **TO** $1$ **STEP** $-1$ **DO**
**BEGIN**
  **IF** $\tilde{y}^i > 0$ **THEN**
    $p := p + 1;$
  **ELSE**
    $n := n + 1;$

  $(x, y) = (\frac{n}{\#n}, \frac{p}{\#p});$
  draw line from $(x', y')$ to $(x, y)$;

  $(x', y') := (x, y);$
**END**

# 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 Algorithm: Direct Variant

**Input:** analogous to above;

$p := 0;$
$n := 0;$
AUC $:= 0$

**FOR** $i := m$ **TO** $1$ **STEP** $-1$ **DO**
  **IF** $\tilde{y}^i > 0$ **THEN**
    $p := p + 1;$
  **ELSE**
    AUC $:=$ AUC $+ \frac{p}{\#p \cdot \#n};$

# AUC Algorithm: Indirect Variant

It can be proved that 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* applied to the set of positive examples.

# ROC Curve + Direct AUC Algorithm

**Input:** analogous to above;

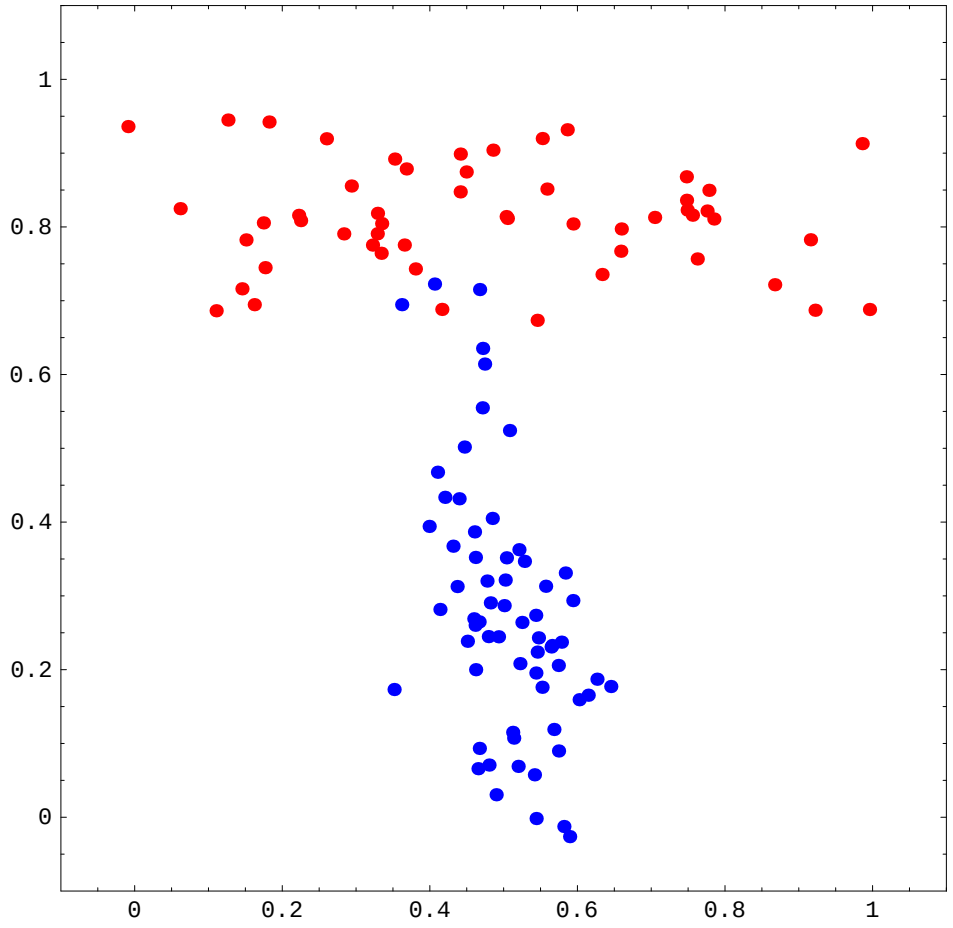
$p := 0;$
$n := 0;$
AUC $:= 0$
$(x', y') := (0, 0);$

**FOR** $i := m$ **TO** $1$ **STEP** $-1$ **DO**
**BEGIN**
  **IF** $\tilde{y}^i > 0$ **THEN**
    $p := p + 1;$
  **ELSE**
  **BEGIN**
    $n := n + 1;$
    AUC $:=$ AUC $+ \frac{p}{\#p \cdot \#n};$
  **END**

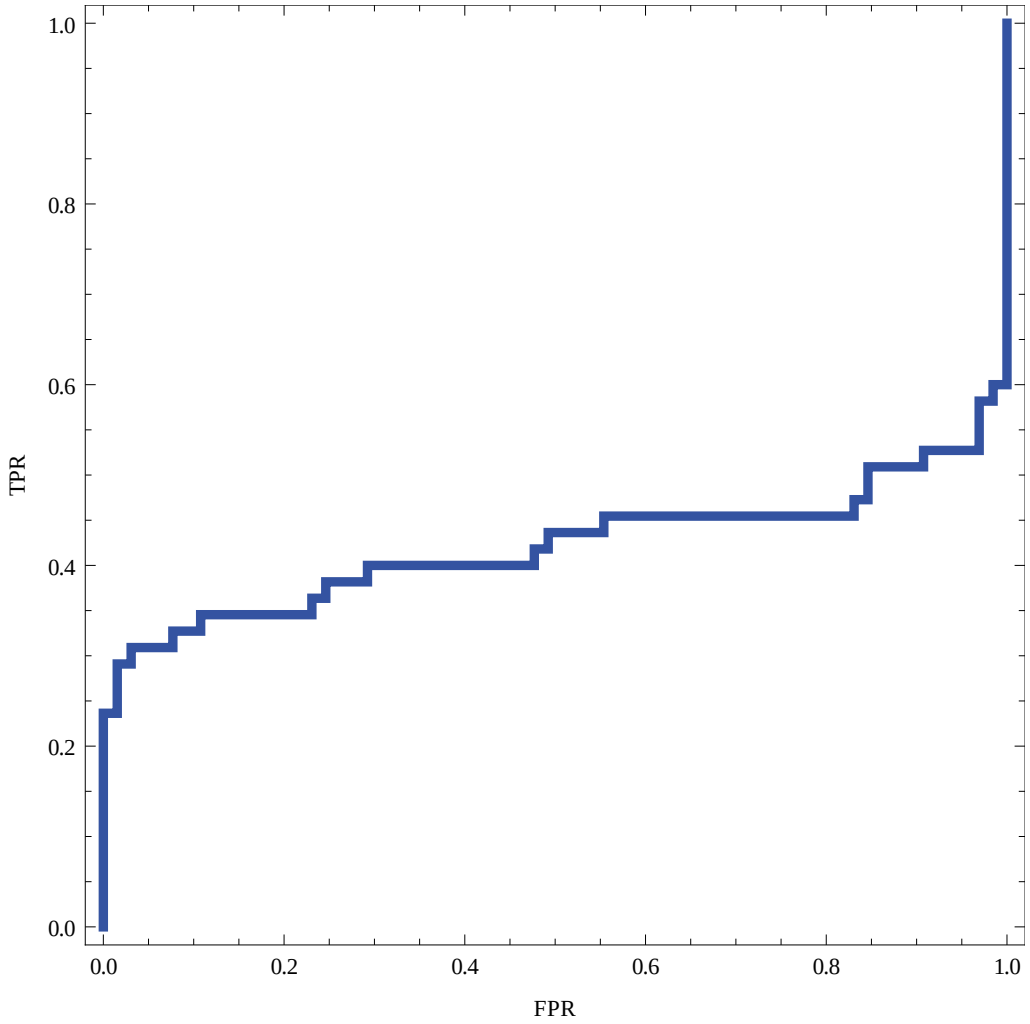
  $(x, y) = (\frac{n}{\#n}, \frac{p}{\#p});$
  draw line from $(x', y')$ to $(x, y);$

  $(x', y') := (x, y);$
**END**

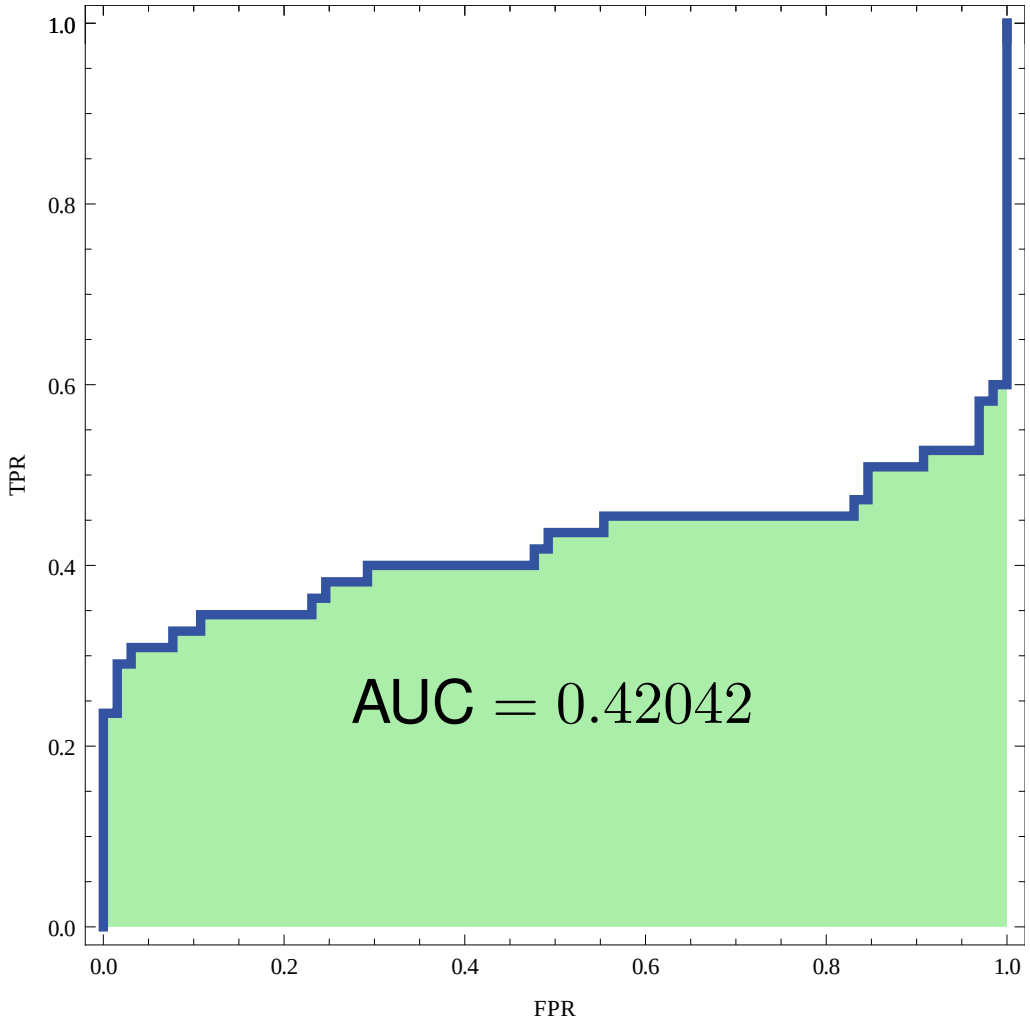
# ROC Example: Data Set 6 (Exercises)



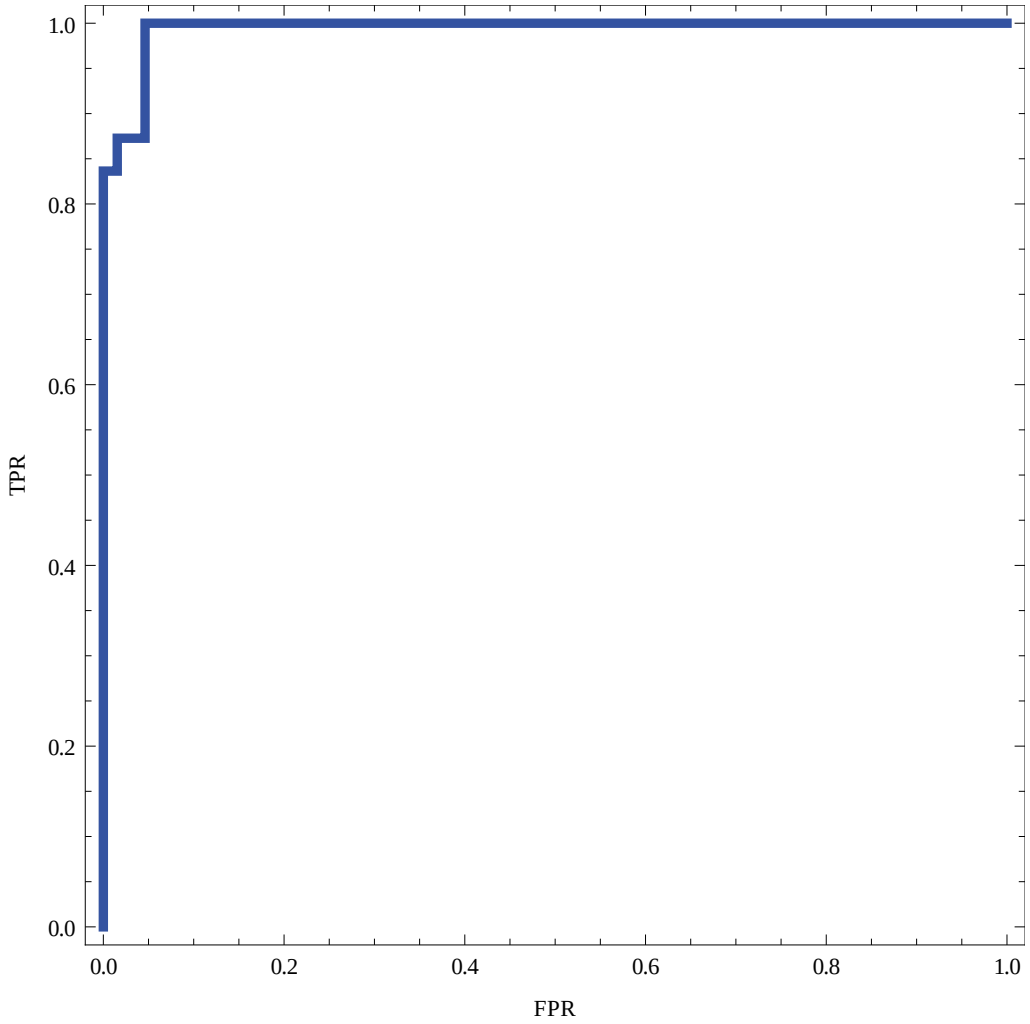
# ROC Example: ROC Curve for $\bar{g}((x_1, x_2)) = x_1$ Using All Samples



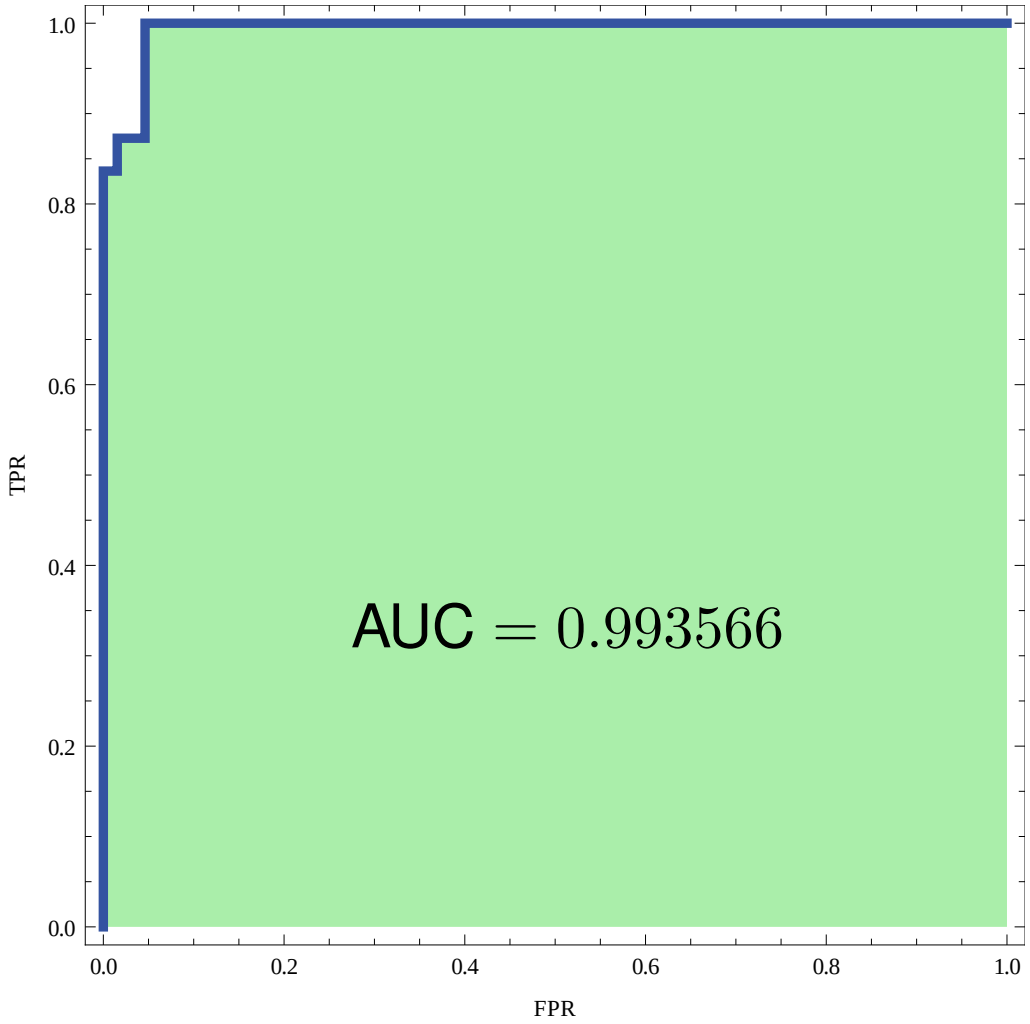
# ROC Example: ROC Curve for $\bar{g}((x_1, x_2)) = x_1$ Using All Samples



# ROC Example: ROC Curve for $\bar{g}((x_1, x_2)) = x_2$ Using All Samples



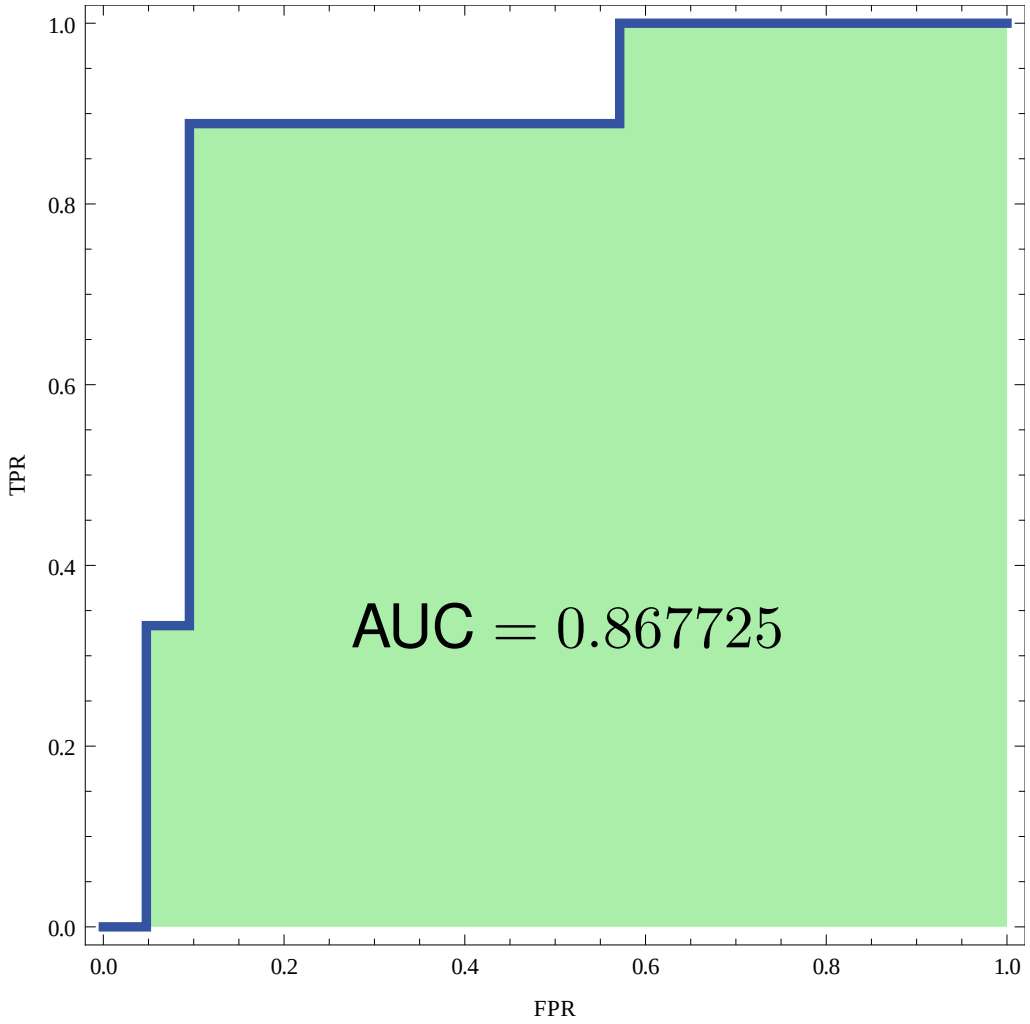
# ROC Example: ROC Curve for $\bar{g}((x_1, x_2)) = x_2$ Using All Samples



# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

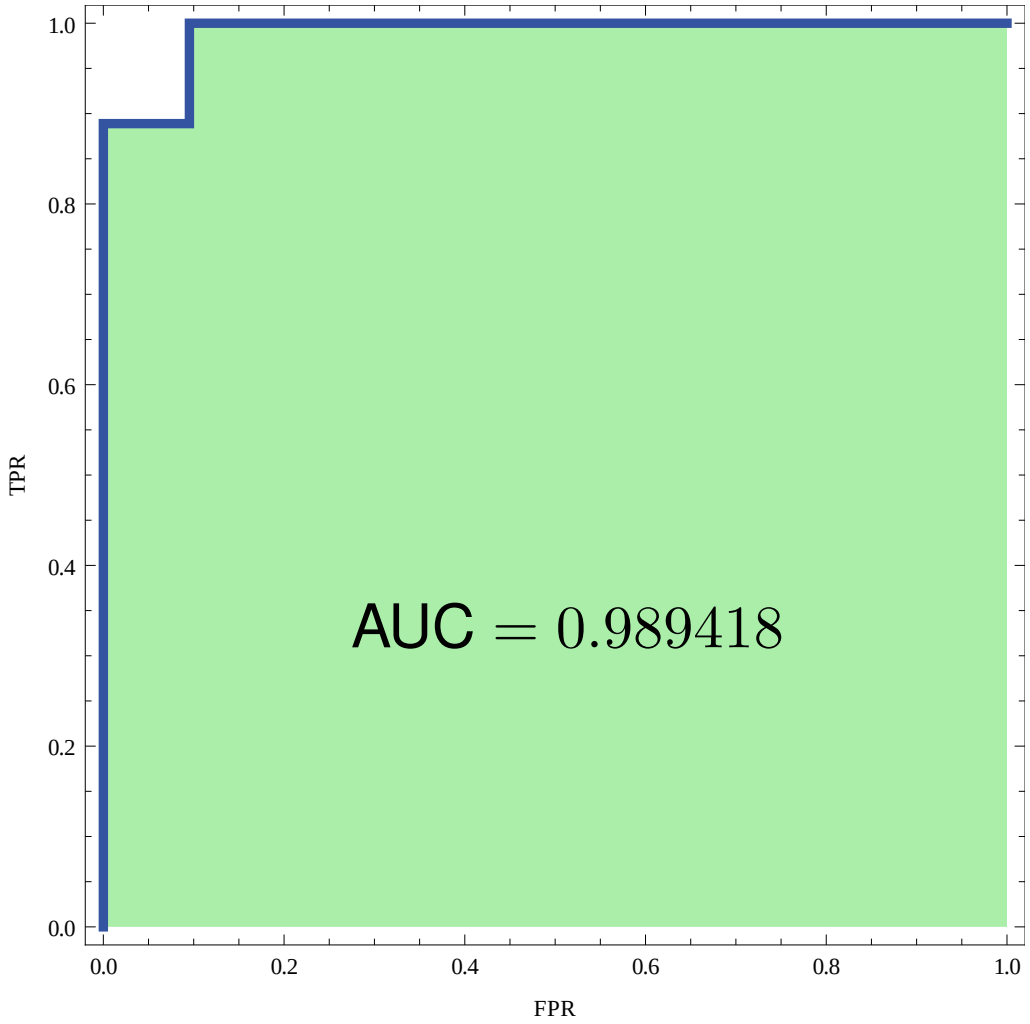
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 1$:



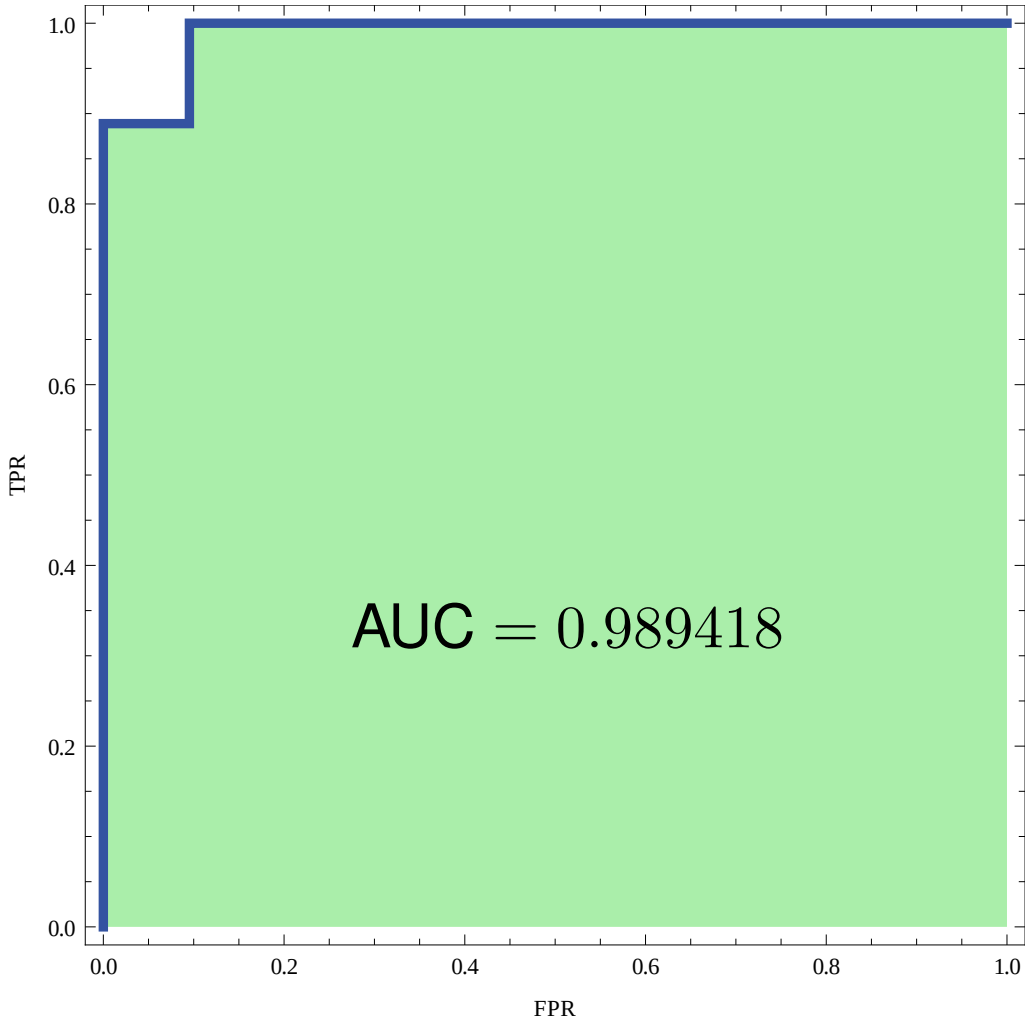
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 3$:



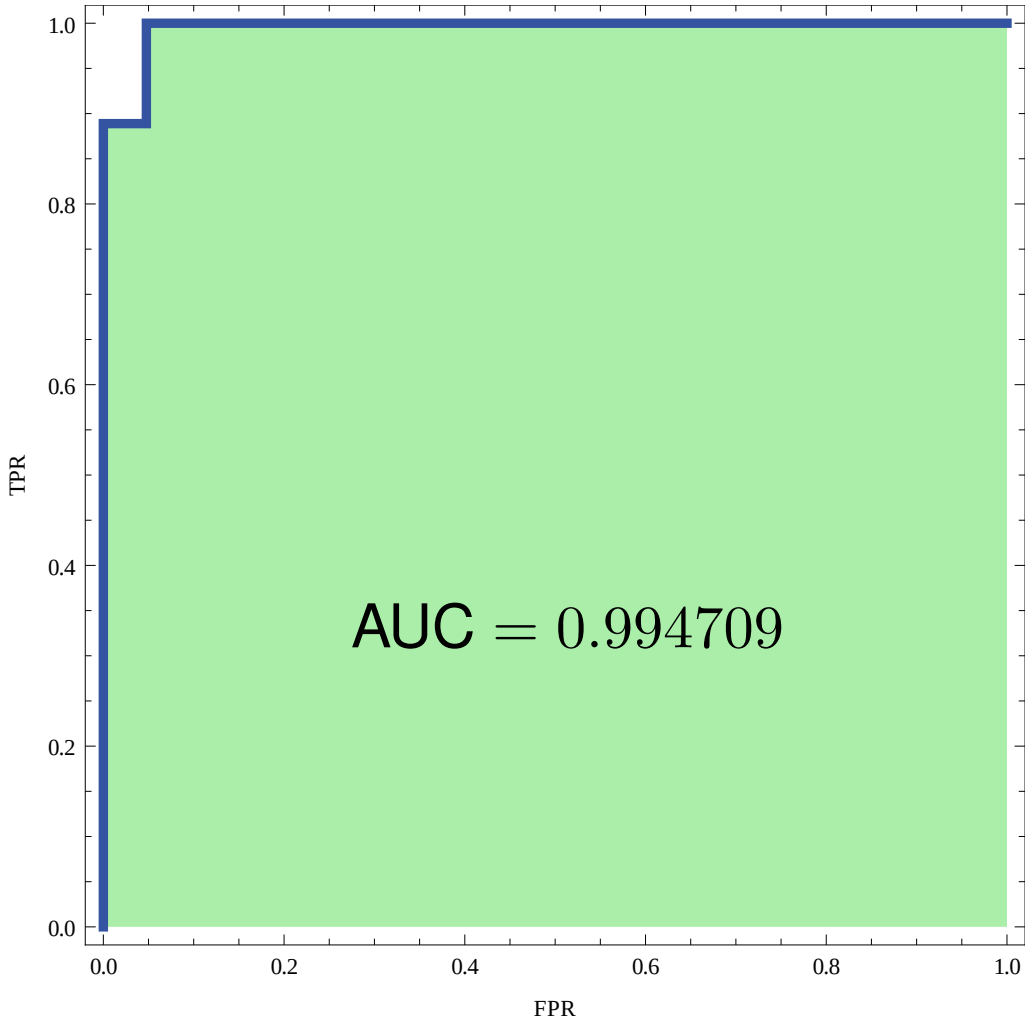
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 5$:



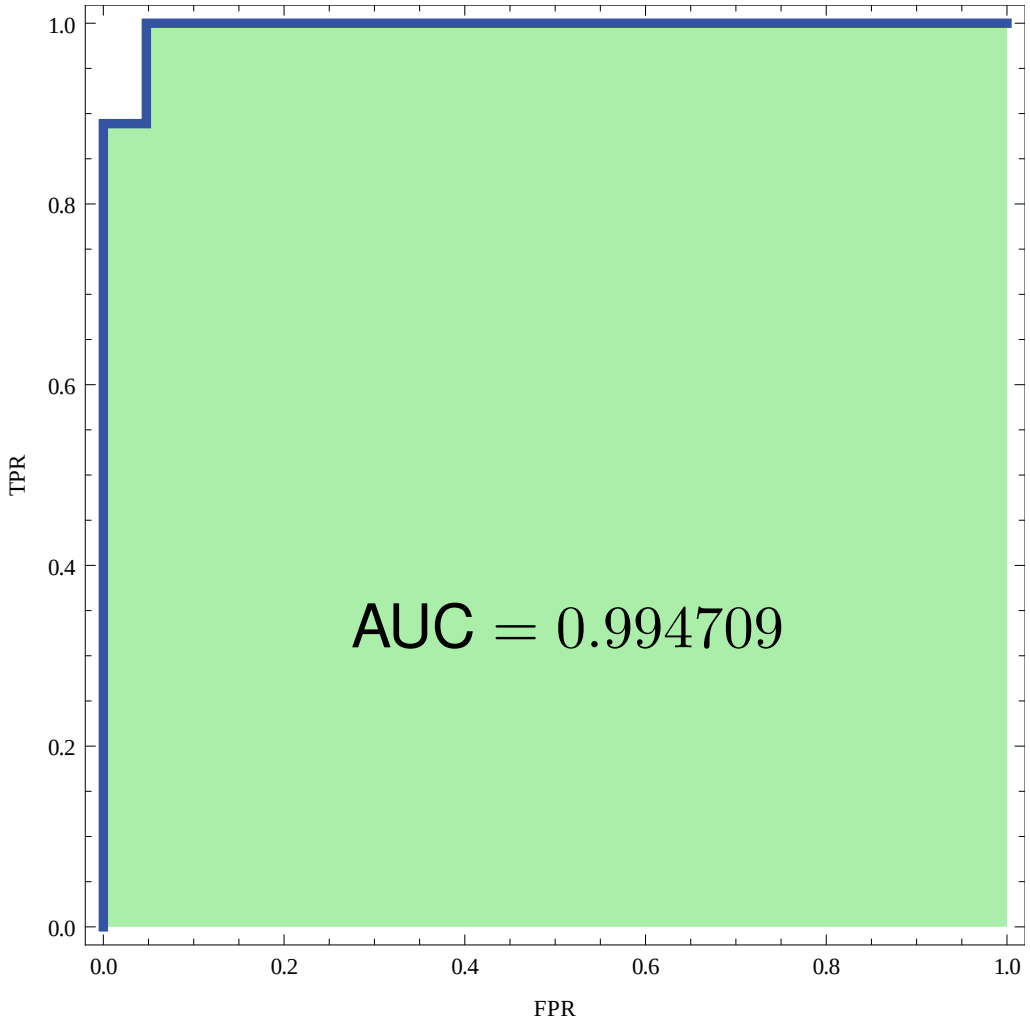
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 7$:



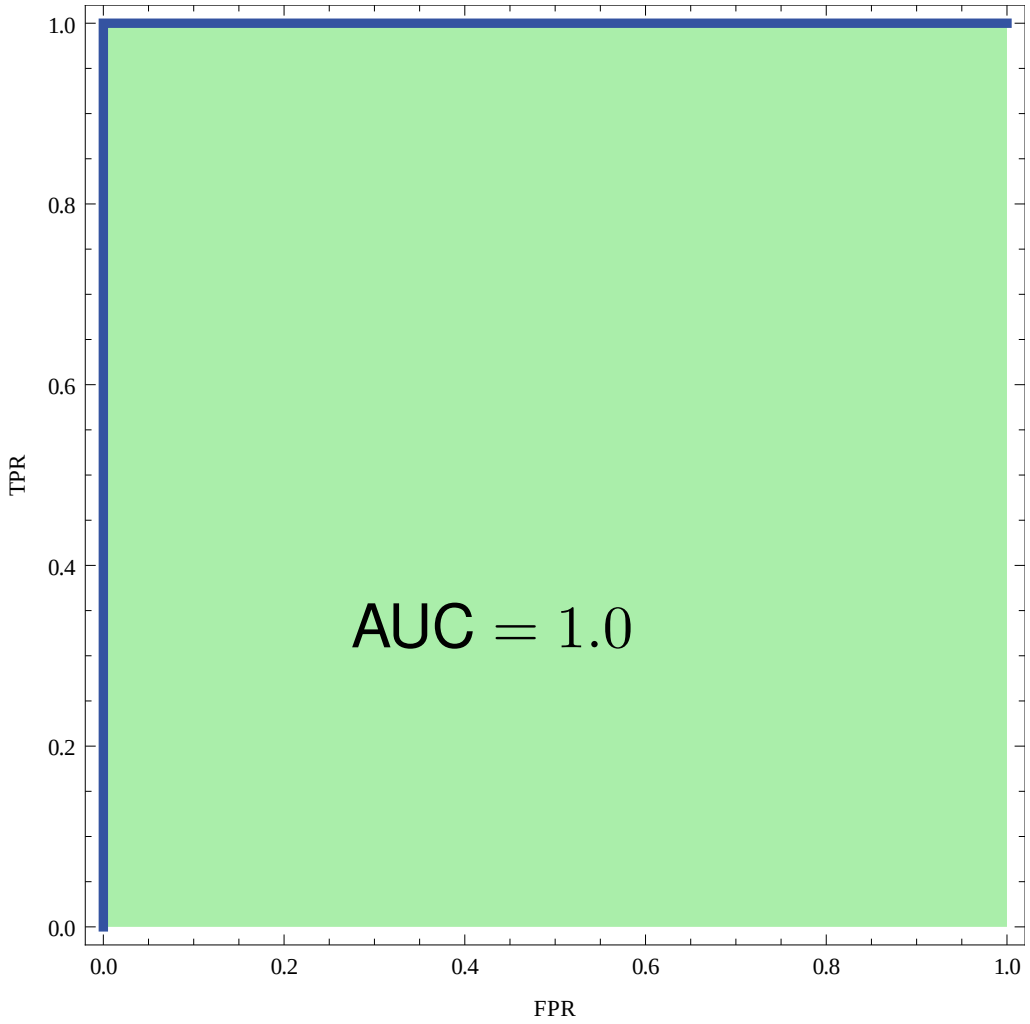
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 9$:



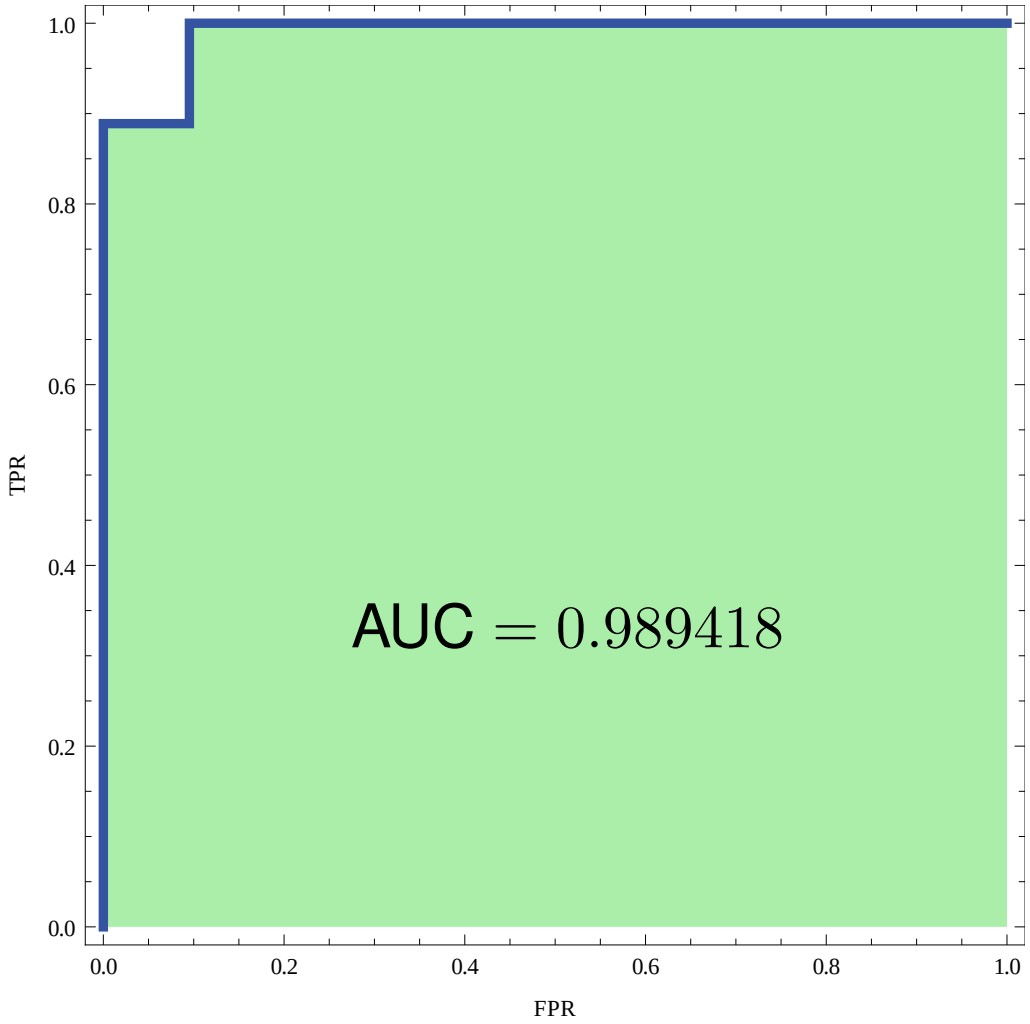
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 11$:



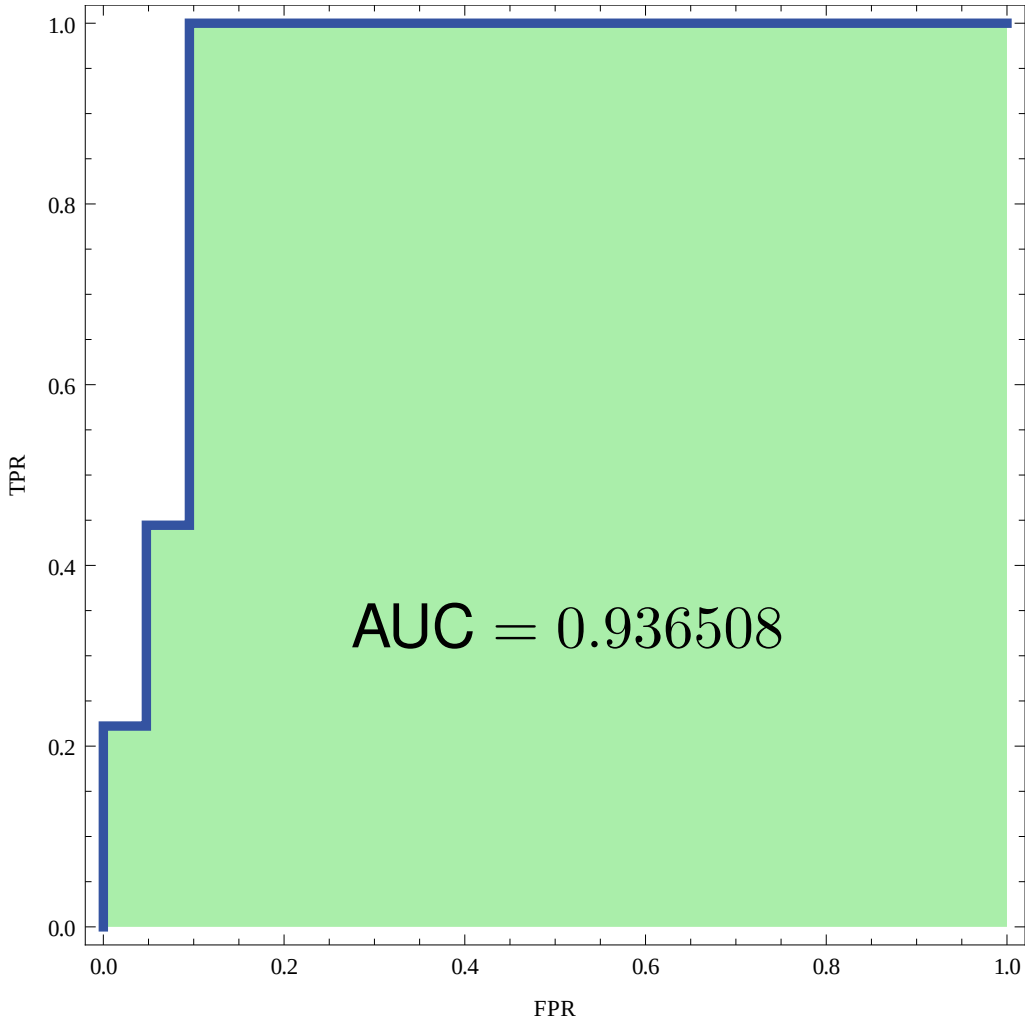
# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 13$:



# ROC Example: ROC Curves for KNN (75% training, 25% test samples)

$k = 39$:



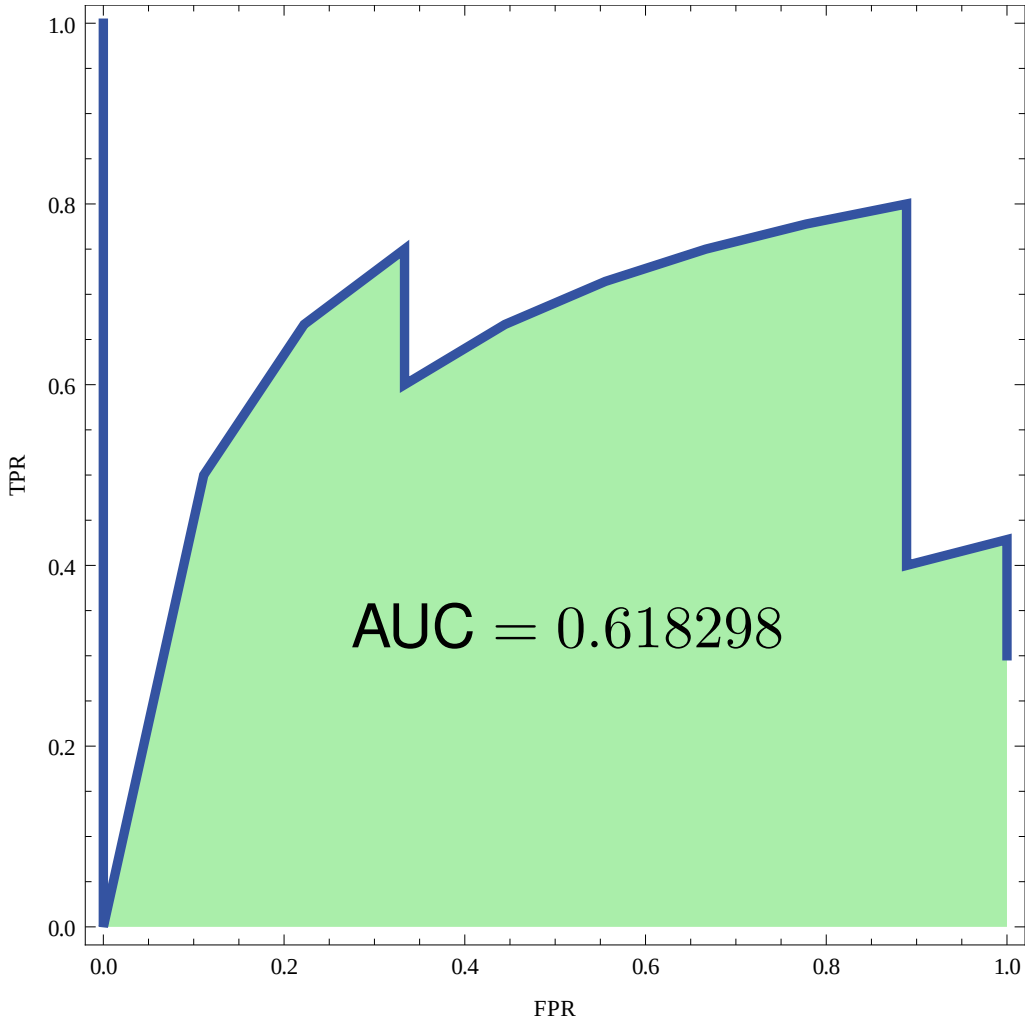
# Precision-Recall Curves

- 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

# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

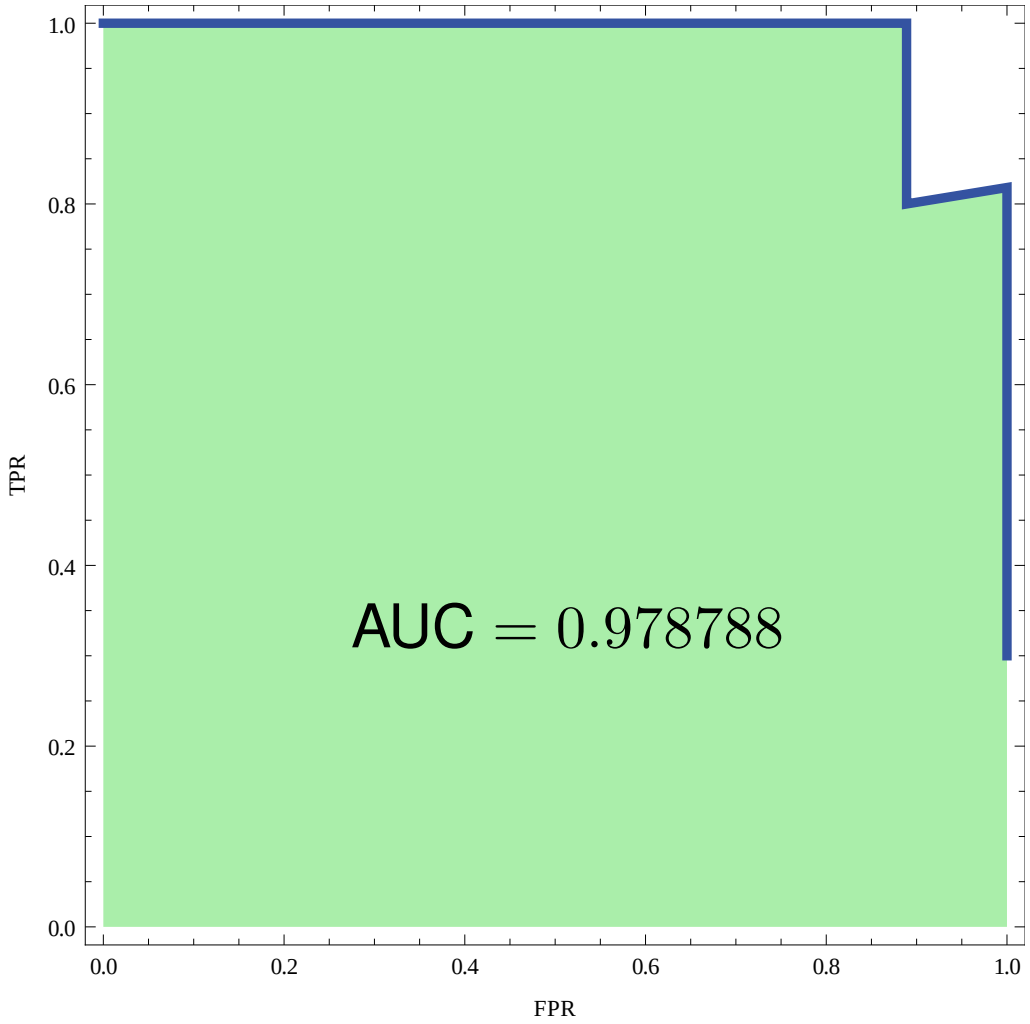
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 1$:



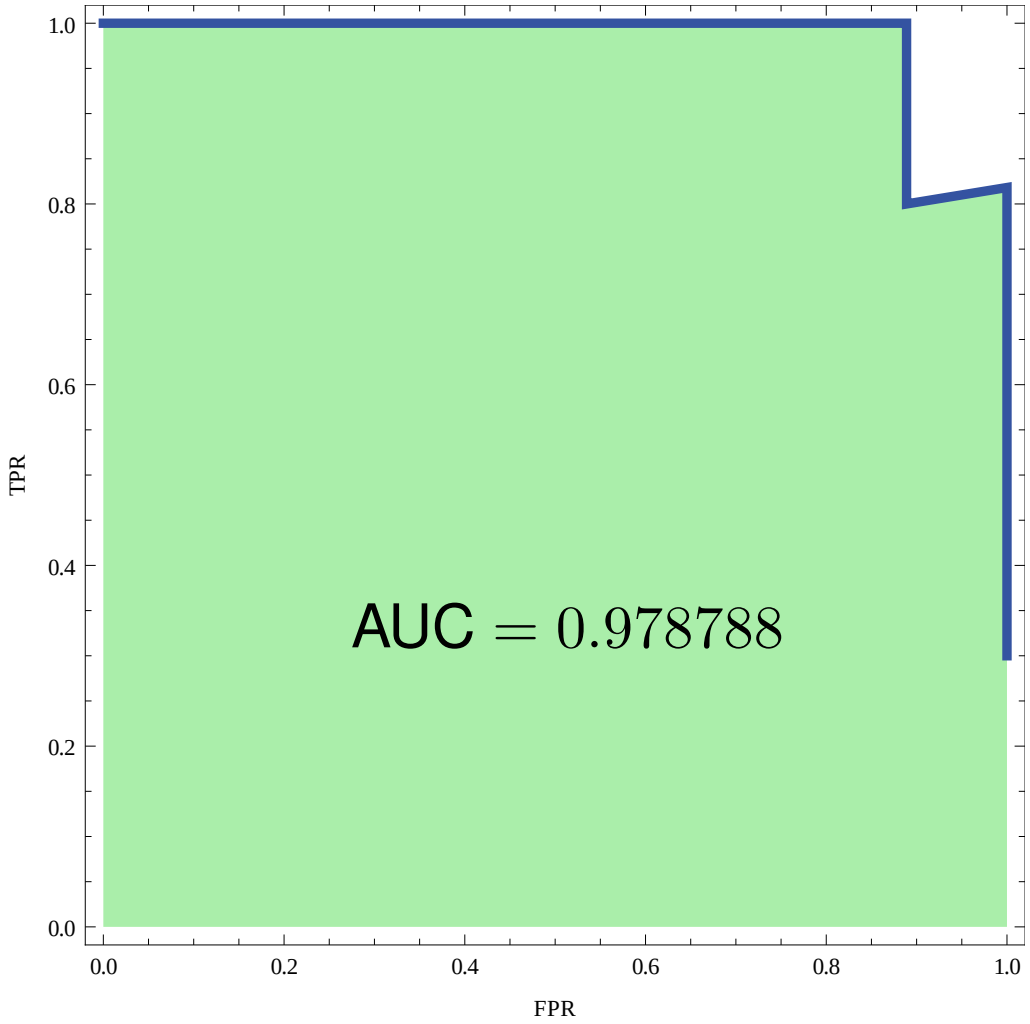
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 3$:



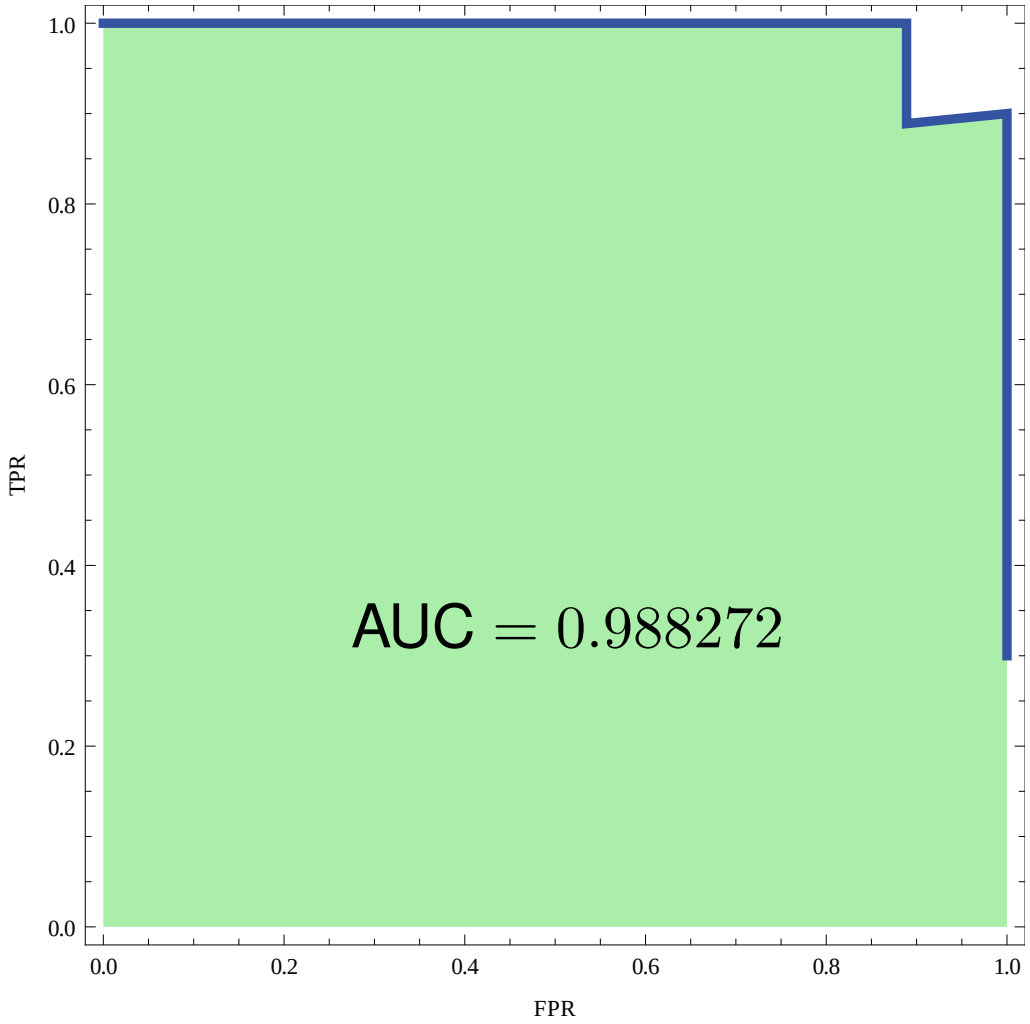
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 5$:



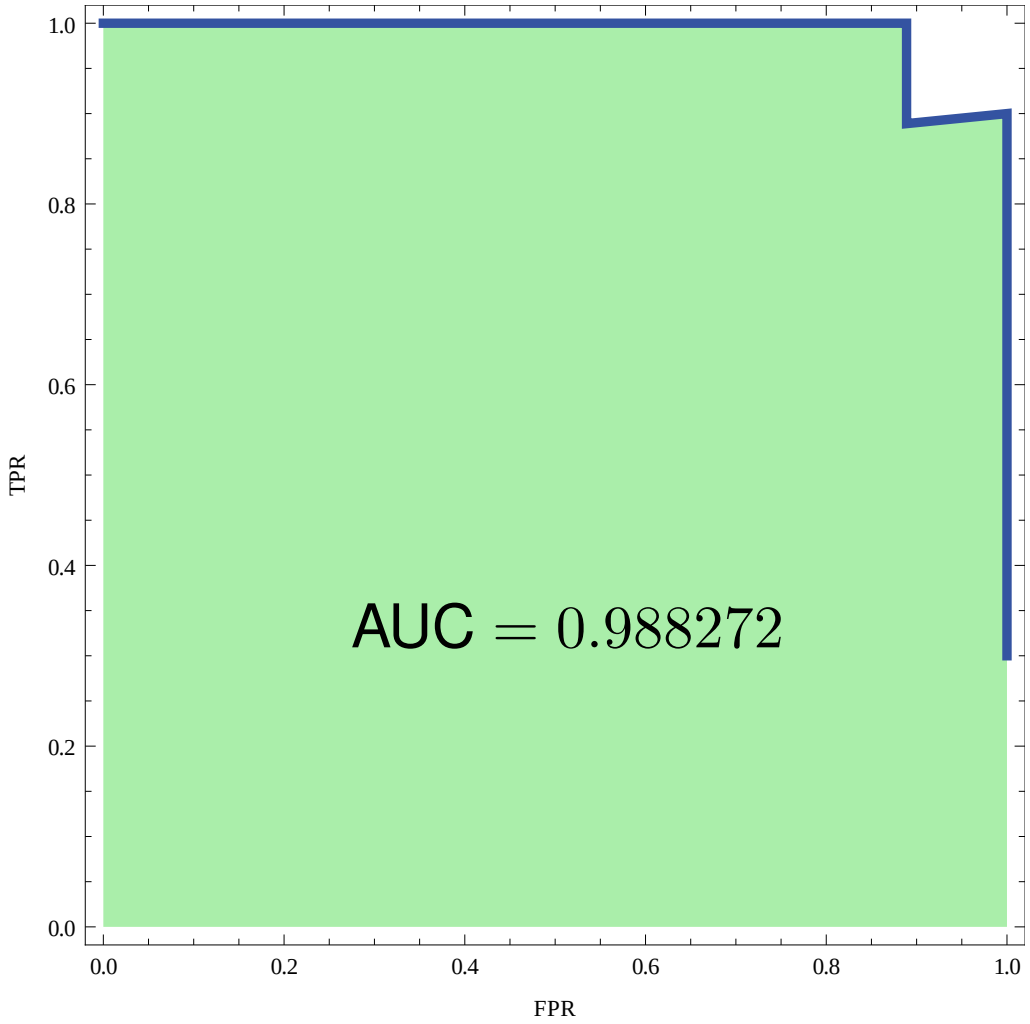
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 7$:



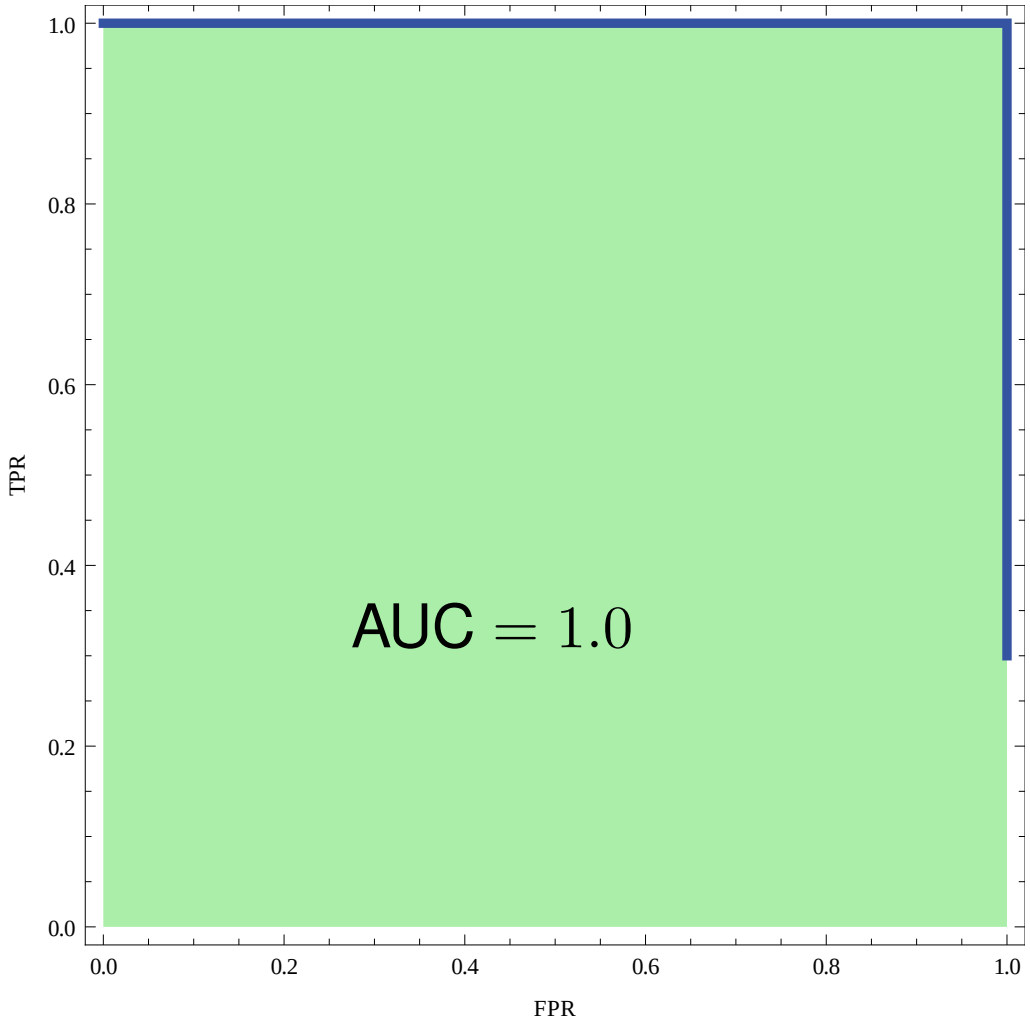
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 9$:



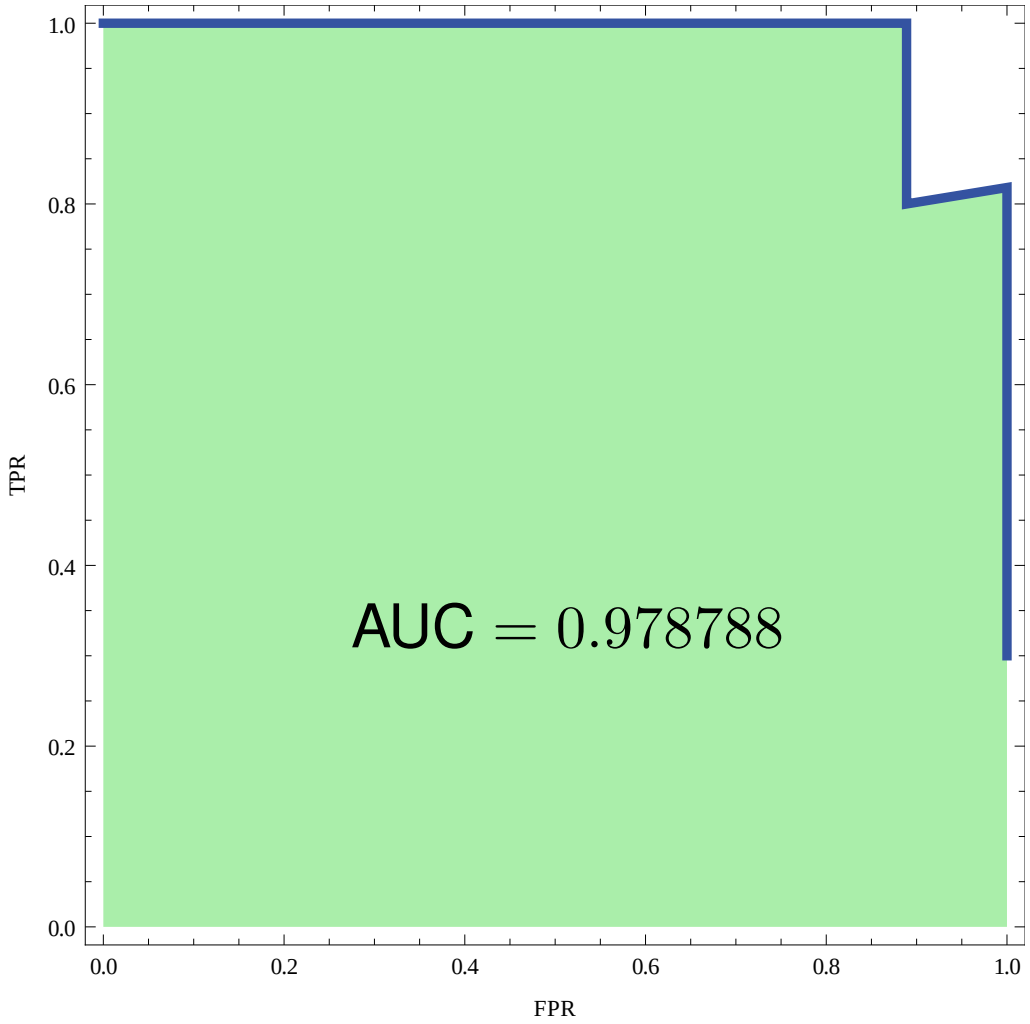
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 11$:



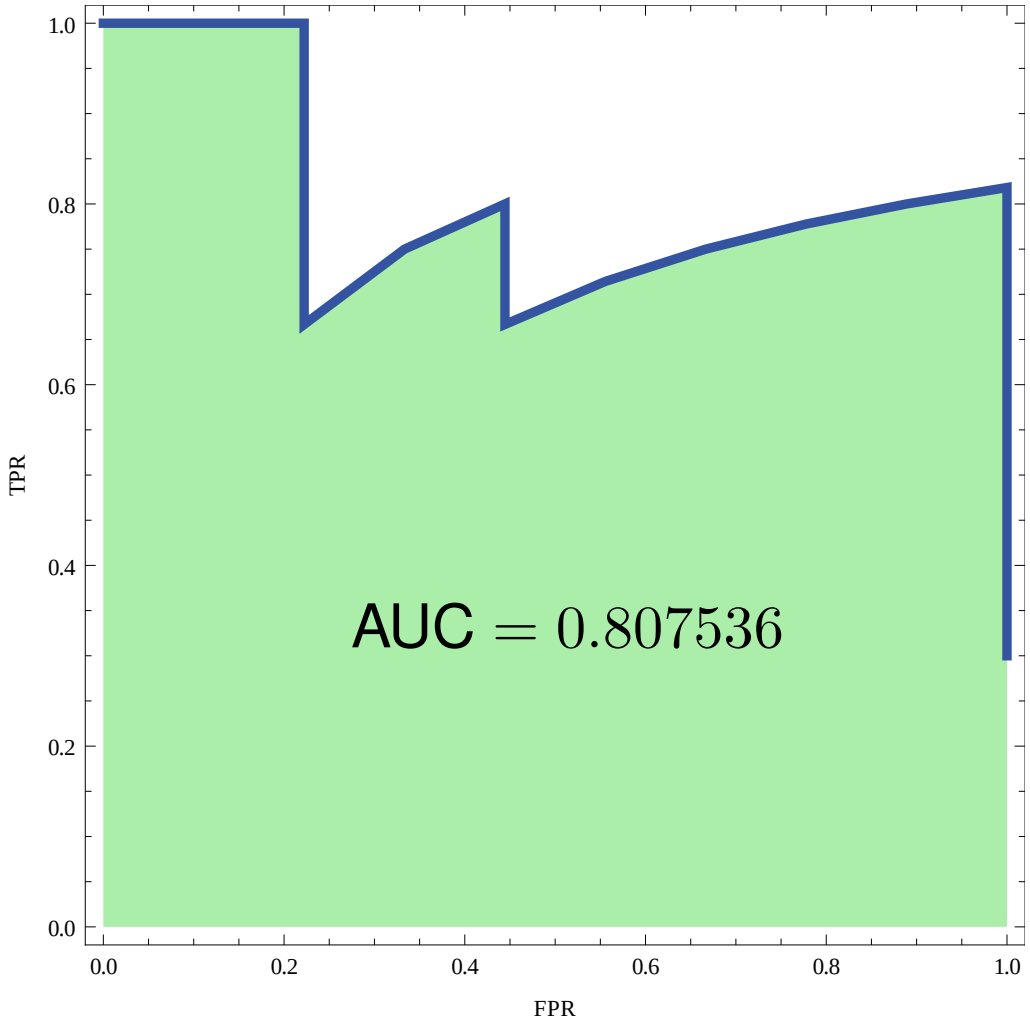
# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 13$:



# PRC Example: PRC Curves for KNN (75% training, 25% test samples)

$k = 39$:



# How Do We Actually Judge Results?

- We know that the lower the generalization error is, the better.
- *But how do we know whether an MSE of 0.1562 is just great or mega-bad?*
- We know that the higher the AUC is, the better. We further know that an AUC of around or below 0.5 is bad.
- *But how do we know whether an AUC of 0.65342 is just great or mega-bad?*

# Some Basic Facts

- The absolute values of MSE, AUC, and other evaluation measures are relatively meaningless if we want to know whether they could be improved or not.
- The reason is simply that there are usually influences that are beyond our control—e.g. unavoidable errors caused by noise.
- A common approach to assess absolute values of evaluation measures in machine learning is *significance testing*, which is concerned with finding out whether the results could have been observed by chance.

# Significance Testing According to Fisher

- With the hypothesis that an observation is the result of pure randomness, the $p$-*value* is the probability of obtaining a result at least as high/low as a given observation.
- So the $p$-value is a means of inductive evidence against the hypothesis that the observation has been observed by chance. Clearly, the smaller the $p$-value, the greater the evidence against the hypothesis.

# Significance Testing in Supervised Machine Learning (1/4)

- In supervised machine learning, we can use this idea to find out whether there is structure in the data.
- The hypothesis is that

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

holds, i.e. inputs and labels are independent from each other.

# Significance Testing in Supervised Machine Learning (2/4)

- Then, if inputs and outputs were actually independent, the results we have obtained for our data should be similar to the results that we obtain if we apply our learning algorithm to randomly sampled data according to the assumption above.
- The $p$-value is then the probability that a result at least as good (equally good or higher ACC, equally good or lower MSE, equally good or higher AUC, etc.) is obtained.

# Significance Testing in Supervised Machine Learning (3/4)

- It is clear that we can almost never compute the exact $p$-value.
- What we can do is the following:
  1. Repeat training a large number of times, each time using a different random label vector $(y^1, \ldots, y^l)$ (either by sampling $y^i$'s according to an estimated $p(y)$ or by randomly shuffling the label vector we have already)
  2. Compute a value $\tilde{p}$ as the proportion (relative frequency) of outcomes in which a result at least as good (equally good or higher ACC, equally good or lower MSE, equally good or higher AUC, etc. on the test set or *using cross-validation*) has been obtained as an estimate for the real $p$-value.

# Significance Testing in Supervised Machine Learning (4/4)

- A low $\tilde{p}$ indicates that the learning algorithm is only able to find better results in a small number of situations—most likely, when there is (at least as much) structure in the data.
- A high $\tilde{p}$ indicates that our results are not (much) better than if we had used a random label vector. The reasons for this may be twofold:
  1. There is not enough (not more than random) structure in our data (no specific relationship, missing features, too high noise, etc.) $\rightarrow$ this is a situation we cannot overcome.
  2. Our model/learning algorithm is so weak (at least for the given data) that it does not fit our data better than random data.

# Summary and Outlook

- In this unit, we have studied how to evaluate a given model:
  - Generalization error/risk
  - Estimates via test set method and cross validation
  - ROC analysis
  - Significance testing
- In some sense, we now have *a posteriori tools* for assessing the results we have obtained, but we do not know yet *how to create good models*.