UNIT 2

Basics of Supervised Machine Learning

Johannes Kofler

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright statement:

Lecture Supervised Techniques: Planned Topics

- UNIT 1: Overview of Supervised Machine Learning
- UNIT 2: Basics of Supervised Machine Learning
- UNIT 3: Support Vector Machines
- UNIT 4: Random Forests and Gradient Boosting
- UNIT 5: Logistic Regression
- UNIT 6: Artificial Neural Networks
- UNIT 7: Special Network Architectures

Planned topics for UNIT 2

- The probabilistic framework
- Three introductory examples:
  1. A Gaussian classification task
  2. k-nearest neighbors
  3. Linear and polynomial regression
- The bias-variance trade-off
- Evaluation of classifiers for unbalanced data sets:
  Confusion matrix, ROC, AUC, PR curves

How does supervised learning work in a nutshell:

1. Acquire labeled dataset (input features + target values)
2. Divide dataset into training and test set
3. Select preprocessing pipeline, features, and model class based on training set
4. Optimize the model parameters on the training set
5. Optionally use validation set or CV to determine best model (hyperparameters)
6. Go back to step 3 if evaluation on validation/training set gave new insights
7. Use test set to calculate estimate for generalization error/risk

We also introduced the formal framework, whose basic building blocks we recall next:

Recap of formal framework: Data set

- One object is represented as feature vector of length d:

$$\mathbf{x} = (x^{(1)}, \dots, x^{(d)})^T$$

- Dataset consits of l objects with feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_l$
- We are given a target value $y_i \in \mathbb{R}$ for each sample $\mathbf{x}_i$
- All target values: target/label vector:

$$\mathbf{y} = (y_1, \dots, y_l)^T$$

- Often dataset is summarized as data matrix:

$$\mathbf{Z} = \begin{pmatrix} \mathbf{X} \\ \mathbf{y}^T \end{pmatrix} = \begin{pmatrix} x_1^{(1)} & \dots & x_l^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(d)} & \dots & x_l^{(d)} \\ y_1 & \dots & y_l \end{pmatrix}$$

Recap of formal framework: Model and loss function

- How do we get the "best" model?
  1. How does our model perform on our data? – Loss function
  2. How will it perform on (unseen) future data? (i.e. how will it generalize?) – Generalization error/risk
- Assume we have a model $g(\mathbf{x}; \mathbf{w})$, parameterized by $\mathbf{w}$
- Its output should be as close as possible to the true target value y
- We use a loss function

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

  to measure how close our prediction is to the true target.

Recap of formal framework: Generalization error/risk and Empirical Risk Minimization

- The generalization error or risk is the expected loss on future data:

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

- In practice, we hardly have any knowledge about $p(\mathbf{x}, y)$. Precise definition: next slide.
- Minimize the empirical risk R_{emp} on our dataset (Empirical Risk Minimization):

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

The probabilistic framework: Part 1

- Previous slide: assume that future data are generated according to joint distribution of inputs and outputs.
- The joint density (finitely many possibilities: joint probability distribution) is denoted as $p(\mathbf{z}) = p(\mathbf{x}, y)$.
- Further important probabilistic objects in this context:
  1. Marginal distributions:
     - $p(\mathbf{x})$: density/probability of observing input vector $\mathbf{x}$ (regardless of target value)
     - $p(y)$: density/probability of observing target value y
  2. Conditional distributions:
     - $p(\mathbf{x}\,|\,y)$: density/probability of input value $\mathbf{x}$ for a given y
     - $p(y\,|\,\mathbf{x})$: density/probability to observe y for a given input $\mathbf{x}$

The probabilistic framework: Part 2

- By definition of conditional probability:

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

- Bayes' Theorem:

$$p(y \,|\, \mathbf{x}) = \tfrac{p(\mathbf{x}|y)\, p(y)}{p(\mathbf{x})}, \quad p(\mathbf{x} \,|\, y) = \tfrac{p(y|\mathbf{x})\, p(\mathbf{x})}{p(y)}$$

- Marginal densities are obtained by integrating out:

$$p(\mathbf{x}) = \int\limits_{\mathbb{R}} p(\mathbf{x}, y)\, \mathrm{d}y = \int\limits_{\mathbb{R}} p(\mathbf{x} \,|\, y)\, p(y)\, \mathrm{d}y$$
$$p(y) = \int\limits_{X} p(\mathbf{x}, y)\, \mathrm{d}\mathbf{x} = \int\limits_{X} p(y \,|\, \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

- Next slides: use these concepts to provide an example where g can be calculated explicitly

Binary classification with 0-1 loss: Part 1

- Recall 0-1 loss: $L_{\mathbf{zo}}(y, g(\mathbf{x}; \mathbf{w})) = \begin{cases} 0 & y = g(\mathbf{x}; \mathbf{w}) \\ 1 & y \neq g(\mathbf{x}; \mathbf{w}) \end{cases}$
- Inserting this into the general formula of the risk, we obtain:

  $$R(g(.; \mathbf{w})) = \int\limits_X \int\limits_{\mathbb{R}} p(\mathbf{x}, y \neq g(\mathbf{x}; \mathbf{w}))\, \mathrm{d}y\, \mathrm{d}\mathbf{x},$$

  i.e. the misclassification probability.
- Now we use binary classification with only two possible labels $y = \pm 1$. Then $\int \mathrm{d}y \rightarrow \sum_{y=\pm 1}$ and

  $$R(g(.; \mathbf{w})) = \int\limits_X \sum_{y=\pm 1} p(\mathbf{x}, y \neq g(\mathbf{x}; \mathbf{w}))\, \mathrm{d}\mathbf{x}$$

- In case also the features are discrete:

  $$R(g(.; \mathbf{w})) = \sum_{x \in X} \sum_{y=\pm 1} p(\mathbf{x}, y \neq g(\mathbf{x}; \mathbf{w}))$$

Binary classification with 0-1 loss: Part 2

- Together with definition of conditional probability:

$$R(g(.;\mathbf{w})) = \int\limits_X \left\{ \begin{array}{ll} p(\mathbf{x}, y=-1) & \text{if } g(\mathbf{x};\mathbf{w}) = +1 \\ p(\mathbf{x}, y=+1) & \text{if } g(\mathbf{x};\mathbf{w}) = -1 \end{array} \right\} \mathrm{d}\mathbf{x}$$

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

- Optimal classifier: so-called Bayes-optimal classifier:

$$g_{\mathbf{opt}}(\mathbf{x}) = \left\{ \begin{array}{ll} +1 & \text{if } p(y=+1 \mid \mathbf{x}) > p(y=-1 \mid \mathbf{x}) \\ 0 & \text{if } p(y=+1 \mid \mathbf{x}) = p(y=-1 \mid \mathbf{x}) \\ -1 & \text{if } p(y=-1 \mid \mathbf{x}) > p(y=+1 \mid \mathbf{x}) \end{array} \right\}$$

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

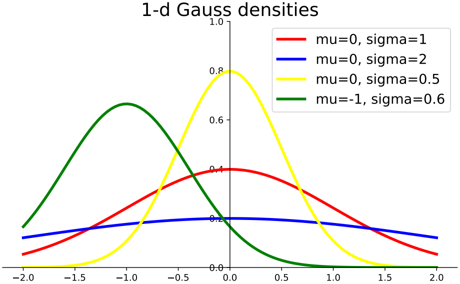
- Resulting minimal risk:

$$R_{\min} = \int\limits_X \min[p(y=-1 \mid \mathbf{x}), p(y=+1 \mid \mathbf{x})]\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

- Next: formulas for Gauss distributed $p(\mathbf{x} \mid y = \pm 1)$

Interlude: intuition for multivariate Gauss distribution: Part 1

- Recall: 1-dimensional Gaussian distribution $N(\mu, \sigma)$; probability density:
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$
μ and σ^2 denote the mean and the variance, respectively.



Interlude: intuition for multivariate Gauss distribution: Part 2

- Density of d-variate $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$-distributed random variable:
  $$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \cdot \sqrt{\det \boldsymbol{\Sigma}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right]$$
  $\boldsymbol{\mu}$: d-dimensional vector, $\boldsymbol{\Sigma}$: invertible symmetric $d \times d$ matrix
- Let's consider two independent 1-dimensional Gaussians with means μ_1, μ_2 and variances σ_1^2, σ_2^2
- Joint density: $p(x_1, x_2) = p_1(x_1)\, p_2(x_2) \rightarrow$ exponent in $p(x_1, x_2)$ is just sum of x_1-exponent and x_2-exponent:
  $$-\frac{1}{2} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix}^T \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}^{-1} \begin{pmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{pmatrix} =$$
  $$-\frac{(x_1-\mu_1)^2}{2\sigma_1^2} - \frac{(x_2-\mu_2)^2}{2\sigma_2^2}$$

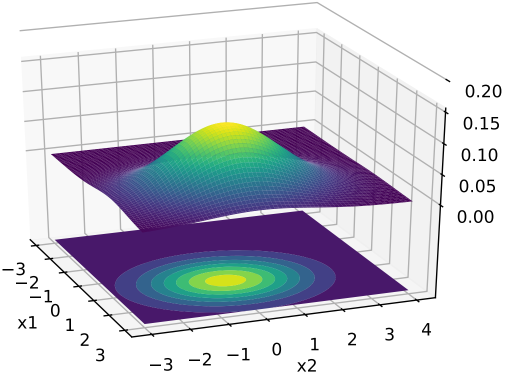
★ ★

Interlude: intuition for multivariate Gauss distribution: Part 3

- This structure also makes sense for non-diagonal matrices $\mathbf{\Sigma}$ which account for dependencies between the d variables.
- Constant $\frac{1}{(2\pi)^{d/2}\sqrt{\det \mathbf{\Sigma}}}$ is included to make the distribution normalized.
- Next: visualization: What will be observed?
  - The mean vector localizes peaks of densities.
  - The level curves are ellipses. Their axis lengths in each direction are proportional to the diagonal entries of $\mathbf{\Sigma}$, i.e. the variances in the different directions.
  - The smaller the variance in some direction, the more tightly peaked the Gaussian in this direction.
  - Covariances account for rotation around coordinate axes.

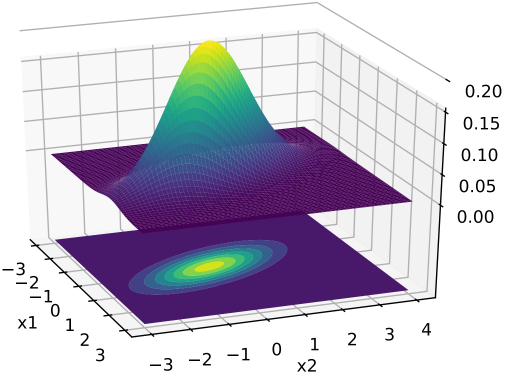
Interlude: intuition for multivariate Gauss distribution: Part 4

2-d Gaussian density with $\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$



Interlude: intuition for multivariate Gauss distribution: Part 5

2-d Gaussian density with $\boldsymbol{\mu} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$, $\boldsymbol{\Sigma} = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} \end{pmatrix}$



Explicit example: Gaussian classifier: Part 1

- We assume that samples are drawn from Gaussians:

$$
\begin{aligned}
p(\mathbf{x} \mid y = -1) \sim N(\boldsymbol{\mu}_{-1}, \boldsymbol{\Sigma}_{-1}) \\
p(\mathbf{x} \mid y = +1) \sim N(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1})
\end{aligned}
$$

- Find useful expression for classification function:

$$
\begin{aligned}
\bar{g}(\mathbf{x}) &= p(y = +1 \mid \mathbf{x}) - p(y = -1 \mid \mathbf{x}) \\
&= \frac{1}{p(\mathbf{x})} \left[p(\mathbf{x} \mid y = +1)\, p(y = +1) \right. \\
&\qquad \left. - \, p(\mathbf{x} \mid y = -1)\, p(y = -1) \right] \\
\hat{g}(\mathbf{x}) &= \ln p(\mathbf{x} \mid y = +1) + \ln p(y = +1) \\
&\qquad - \ln p(\mathbf{x} \mid y = -1) - \ln p(y = -1)
\end{aligned}
$$

Explicit example: Gaussian classifier: Part 2

- Using the formula for $g_{\mathbf{opt}}$, we can infer

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

- Plugging in the definitions: optimal classification border $\hat{g}(\mathbf{x}) = 0$ is d-dimensional hyper-quadric (without proof):

$$-\tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c = 0.$$

- Where:
  1. $\mathbf{A} = \boldsymbol{\Sigma}_{+1}^{-1} - \boldsymbol{\Sigma}_{-1}^{-1}$
  2. $\mathbf{b} = \boldsymbol{\Sigma}_{+1}^{-1}\boldsymbol{\mu}_{+1} - \boldsymbol{\Sigma}_{-1}^{-1}\boldsymbol{\mu}_{-1}$
  3. $c = -\frac{1}{2}\boldsymbol{\mu}_{+1}^T\boldsymbol{\Sigma}_{+1}^{-1}\boldsymbol{\mu}_{+1} + \frac{1}{2}\boldsymbol{\mu}_{-1}^T\boldsymbol{\Sigma}_{-1}^{-1}\boldsymbol{\mu}_{-1} - \frac{1}{2}\ln\det\boldsymbol{\Sigma}_{+1} + \frac{1}{2}\ln\det\boldsymbol{\Sigma}_{-1} + \ln p(y=+1) - \ln p(y=-1)$

- Next slides: provide visualizations

Explicit example: Gaussian classifier: Part 3: Concrete Example Assumptions

- $p(\boldsymbol{x} \mid y=+1) \sim N(\boldsymbol{\mu}_{+1}, \boldsymbol{\Sigma}_{+1})$ with:

$$\boldsymbol{\mu}_{+1} = (0.4, 0.8) \qquad \boldsymbol{\Sigma}_{+1} \approx \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.005 \end{pmatrix}$$

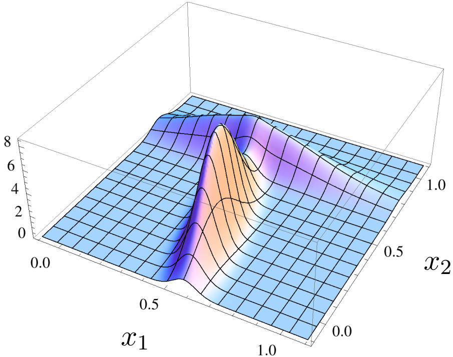
- $p(\mathbf{x} \mid y=-1) \sim N(\mu_{-1}, \boldsymbol{\Sigma}_{-1})$ with:

$$\boldsymbol{\mu}_{-1} = (0.5, 0.3) \qquad \boldsymbol{\Sigma}_{-1} \approx \begin{pmatrix} 0.004 & -0.007 \\ -0.007 & 0.04 \end{pmatrix}$$

- $p(y=+1) = \frac{55}{120} \approx 0.46,\ p(y=-1) = \frac{65}{120} \approx 0.54$

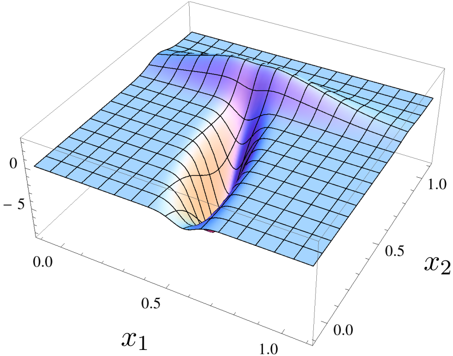
Explicit example: Gaussian classifier: Part 4: Density plot

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



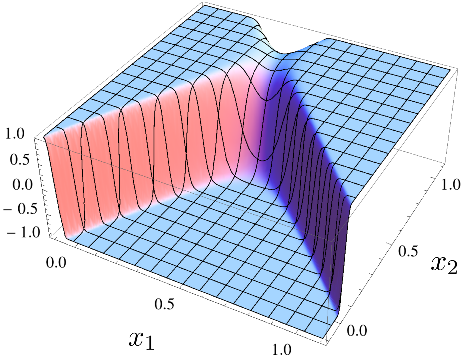
Explicit example: Gaussian classifier: Part 5: Plot of $\tilde{g}$

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

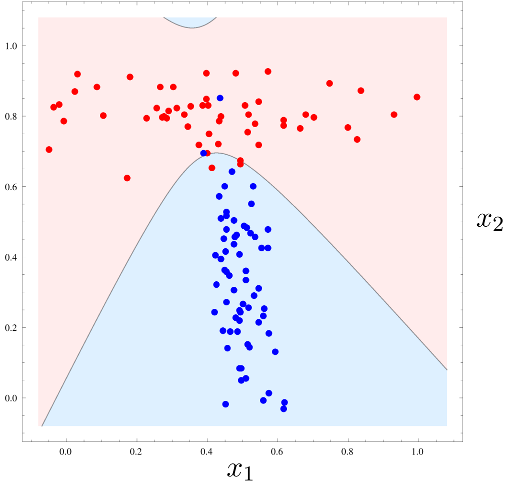


Explicit example: Gaussian classifier: Part 6: Plot of Discriminant function $\bar{g}$

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



Explicit example: Gaussian classifier: Part 7: Plot of Data and Decision Boundary



What about practice?

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

Therefore:

1. Estimate the prediction function with other methods
2. Estimate the generalization error

★★

A more practical example: k-nearest Neighbors Classifier: Part 1: Basics

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

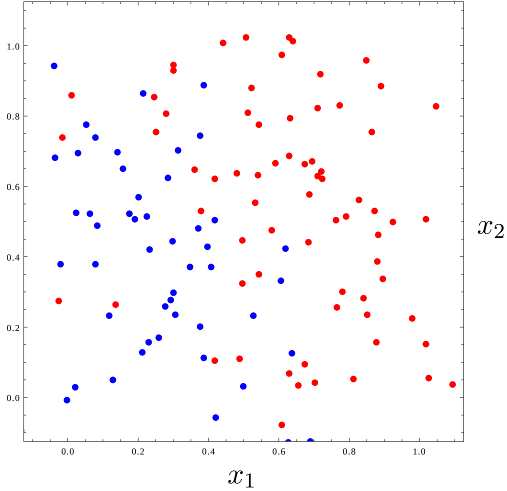
$$g_{k\text{-NN}}(\mathbf{x};\mathbf{Z}) = \text{class that occurs most often among the } k \text{ samples in } \mathbf{Z} \text{ closest to } \mathbf{x}$$

- For $k = 1$: nearest neighbor classifier:

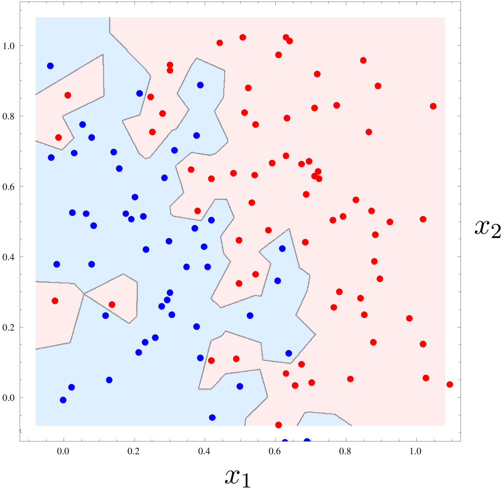
$$g_{\text{NN}}(\mathbf{x};\mathbf{Z}) = \text{class of the sample that is closest to } \mathbf{x}$$

- In case of ties: e.g. random class assignment or class with larger number of samples is assigned
- k-NN regression: output is the average value of the k nearest neighbors

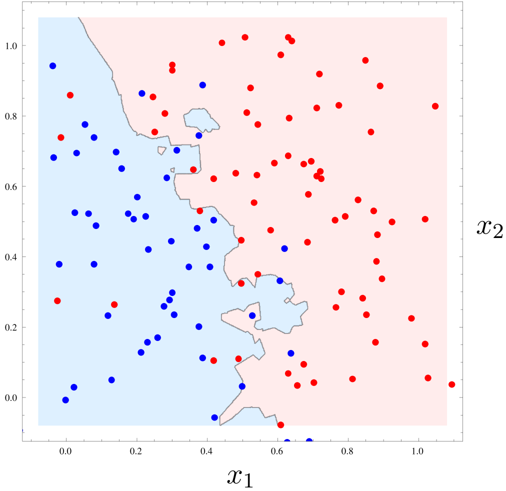
k-nearest Neighbors Classifier: Part 2: Plot of Data set



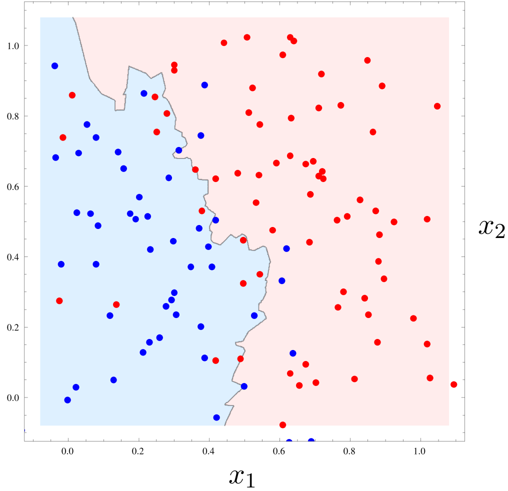
k-nearest Neighbors Classifier: Part 3: k-NN with $k = 1$



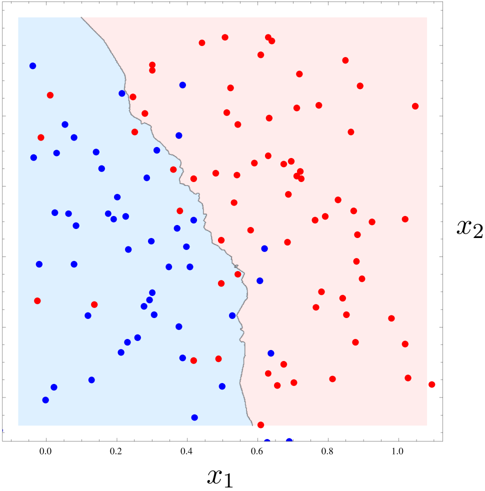
k-nearest Neighbors Classifier: Part 4: k-NN with $k = 5$



k-nearest Neighbors Classifier: Part 5: k-NN with $k = 13$



k-nearest Neighbors Classifier: Part 6: k-NN with $k = 25$



Another explicit example: Linear regression in $d = 1$: Part 1: Basics

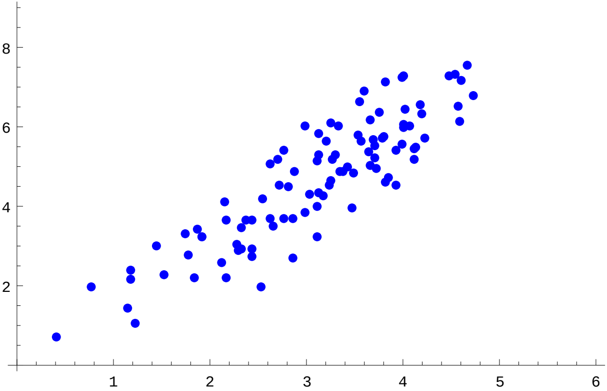
- Main ingredients:
  1. Dataset $\mathbf{Z} = \{(x_i, y_i) \mid i = 1, \ldots, l\}$ with $x_i, y_i \in \mathbb{R}$
  2. Linear classifier: $g(x; w_0, w_1) = w_0 + w_1\, x$
  3. Averaged quadratic loss:

$$\begin{aligned} Q(\mathbf{Z}; w_0, w_1) &= \frac{1}{l} \sum_{i=1}^{l} L_{\mathbf{q}}(y_i, g(x_i; w_0, w_1)) \\ &= \frac{1}{l} \sum_{i=1}^{l} \big(w_0 + w_1\, x_i - y_i\big)^2 \end{aligned}$$

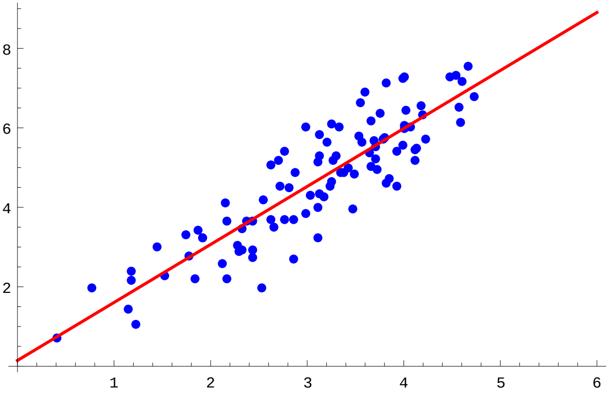
- Aim: Find solution (w_0, w_1) that minimizes $Q(\mathbf{Z}; w_0, w_1)$. Calculus and linear algebra lead to explicit formula
- More details and intuitions: Unit 5 or consider e.g. the course “Basic Methods of Data Analysis”

Linear regression in $d = 1$: Part 2: Plot of Data

All subsequent plots are y versus x.



Linear regression in $d = 1$: Part 3: Plot of Data + Regression Line



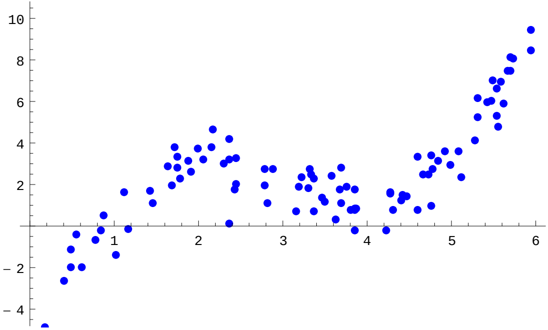
Polynomial regression in $d = 1$: Part 1: Basics

- For more complex data: more complex models; try polynomials
- Main ingredients:
  1. Dataset $\mathbf{Z} = \{(x_i, y_i) \mid i = 1, \ldots, l\}$ with $x_i, y_i \in \mathbb{R}$
  2. Polynomial classifier of degree m:
     $g(x; w_0, w_1, \ldots, w_m) = w_0 + w_1\, x + w_2\, x^2 + \cdots + w_m\, x^m$
  3. Averaged quadratic loss
- Again, there exists a unique global solution with an explicit formula for the parameter vector $\mathbf{w} = (w_0, w_1, \ldots, w_m)^T$:

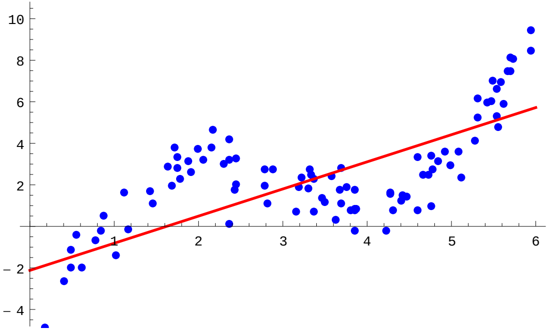
$$\mathbf{w} = \left(\tilde{\mathbf{X}}^T\, \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^T\, \mathbf{y} \quad \text{with} \quad \tilde{\mathbf{X}} = (\mathbf{1}, \mathbf{x}, \mathbf{x}^{[2]}, \ldots, \mathbf{x}^{[m])})$$

  The design matrix $\tilde{\mathbf{X}}$ is a Vandermonde matrix.

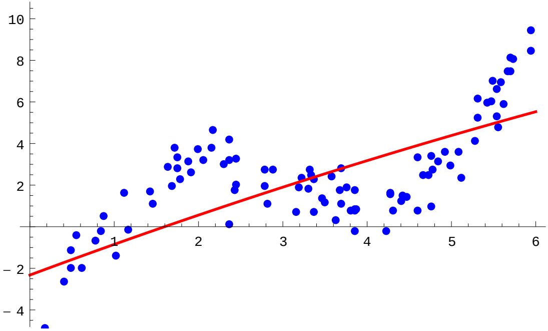
Polynomial regression in $d = 1$: Part 2: Plot of data



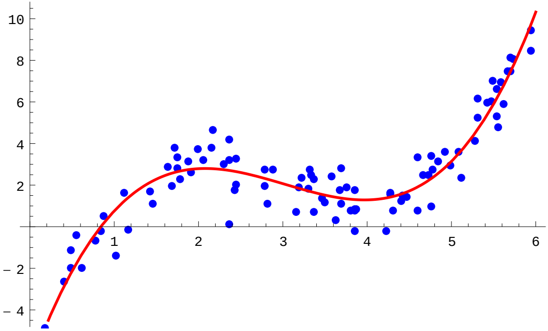
Polynomial regression in $d = 1$: Part 3: Regression with degree $m = 1$



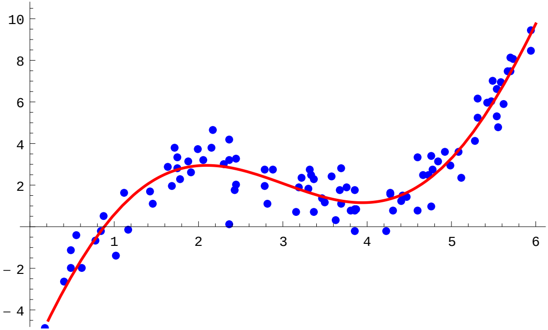
Polynomial regression in $d = 1$: Part 4: Regression with degree $m = 2$



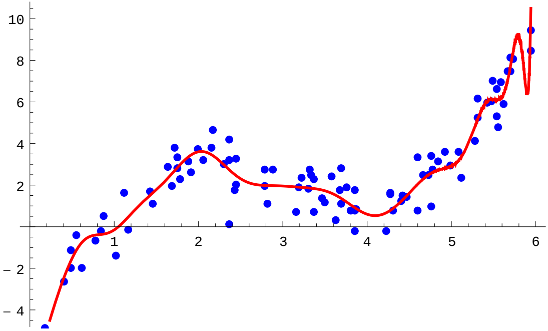
Polynomial regression in $d = 1$: Part 5: Regression with degree $m = 3$



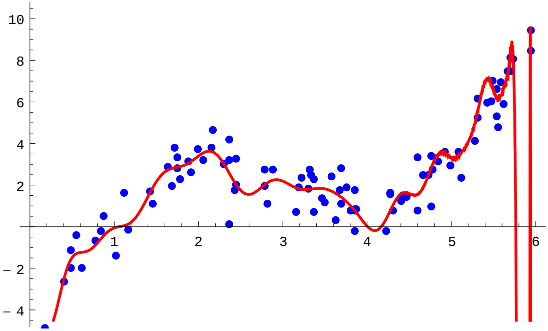
Polynomial regression in $d = 1$: Part 6: Regression with degree $m = 5$



Polynomial regression in $d = 1$: Part 7: Regression with degree $m = 25$



Polynomial regression in $d = 1$: Part 8: Regression with degree $m = 75$



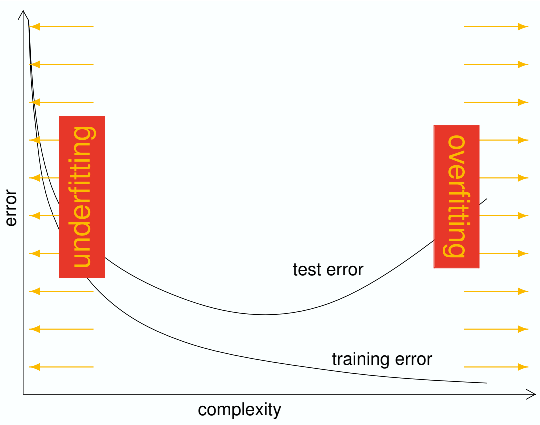
★ ★

Bias-Variance Tradeoff: Part 1: Intuition

- Previous examples: instance of one of the basic problems of supervised machine learning: bias-variance tradeoff.
- Recall from Unit 1:
  1. Underfitting: model is too coarse to fit training or test data (too low model class complexity): e.g. $k = l$ or $m = 1$
  2. Overfitting: model fits well to training data but not to future/test data (too high model class complexity): e.g. $k = 1$ or $m = 75$
- This rather general situation (which often occurs in practice) is illustrated in the next slides.
- We will also discuss these issues in more detail and on a more formal level.

Bias-Variance Tradeoff: Part 2: Notorious situation in practice

★ ★



- Next slides: Explicit example of quadratic loss, where a nice decomposition with proper interpretation is possible

Bias-Variance Decomposition for Quadratic Loss: Part 1

- $\mathbf{Z}_l$: sample set of l elements
- Object of interest: expected prediction error (EPE) for $\mathbf{x}_0 \in X$:

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

- By assumption: $y \,|\, \mathbf{x}_0$ and the selection of training samples are independent, thus:

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

- A short calculation yields (see exercises):

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

Bias-Variance Decomposition for Quadratic Loss: Part 2

1. The first term

$$\mathrm{Var}[y \,|\, \mathbf{x}_0]$$

measures the label variance, i.e. the amount to which the label y varies at $\mathbf{x}_0$: unavoidable error.

2. The second term

$$\text{bias}^2 = \Big(\mathrm{E}[y \,|\, \mathbf{x}_0] - \mathrm{E}_{\mathbf{Z}_l}\big[g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l))\big]\Big)^2$$

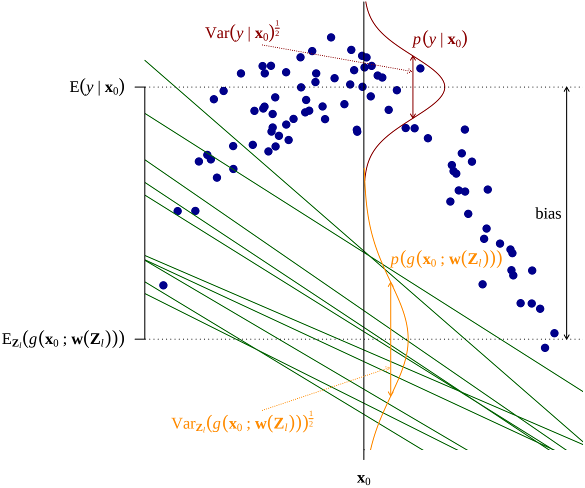
measures how close the model in average approximates the average target y at $\mathbf{x}_0$: squared bias.

3. The third term,

$$\text{variance} = \mathrm{E}_{\mathbf{Z}_l}\Big[\big(g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l)) - \mathrm{E}_{\mathbf{Z}_l}[g(\mathbf{x}_0; \mathbf{w}(\mathbf{Z}_l))]\big)^2\Big]$$

is the variance of the model 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: Part 3


$\mathrm{Var}(y \mid \mathbf{x}_0)^{\frac{1}{2}}$
$p(y \mid \mathbf{x}_0)$
$\mathrm{E}(y \mid \mathbf{x}_0)$
bias
$p(g(\mathbf{x}_0\,;\,\mathbf{w}(\mathbf{Z}_l)))$
$\mathrm{E}_{\mathbf{Z}_l}(g(\mathbf{x}_0\,;\,\mathbf{w}(\mathbf{Z}_l)))$
$\mathrm{Var}_{\mathbf{Z}_l}(g(\mathbf{x}_0\,;\,\mathbf{w}(\mathbf{Z}_l)))^{\frac{1}{2}}$
$\mathbf{x}_0$

Bias-Variance Decomposition for Quadratic Loss: Part 4: Possible Simplifications

- Assume $y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon$, where f is deterministic and ε is a random variable with mean zero and variance σ_ε^2 and which is independent of $\mathbf{x}$. Then:

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

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

Bias-Variance Decomposition for Binary Loss: Part 1

- Assume that we are given a binary classification task, i.e. $y \in \{-1, +1\}$ and $g(\mathbf{x}; \mathbf{w}) \in \{-1, +1\}$.
- As $L_{\mathrm{zo}} = \frac{1}{4} L_{\mathrm{q}}$:

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

- Important: if g isn't the binary classification function, the above representation is not valid

Bias-Variance Decomposition for Binary Loss: Part 2: Further Simplifications

Using $p_R = p(y = +1 \mid \mathbf{x_0})$ and $p_O = p_{\mathbf{Z}_l}(g(\mathbf{x_0}; \mathbf{w}(\mathbf{Z}_l)) = +1)$, we obtain (blackboard):

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

and hence

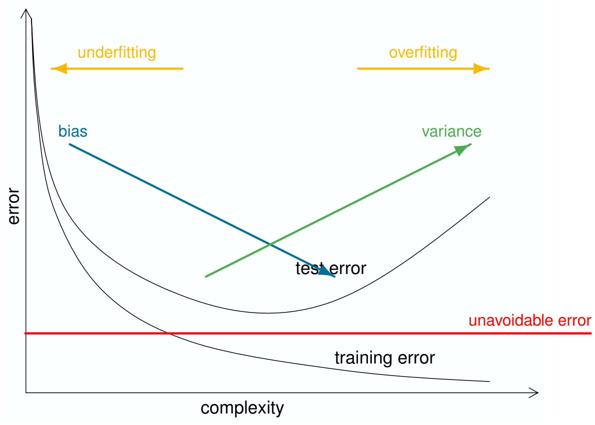
$$\text{EPE}(\mathbf{x}_0) = \underbrace{p_R\,(1 - p_R)}_{\text{unavoidable error}} + \underbrace{(p_R - p_O)^2}_{\text{squared bias}} + \underbrace{p_O\,(1 - p_O)}_{\text{model variance}}.$$

The Bias-Variance Trade-off: Part 3

- It seems intuitively reasonable that the bias decreases with increasing complexity of the model class.
  Take-away: 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 increasing complexity of the model class.
  Take-away: 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: Part 4



The Bias-Variance Trade-off: Part 5: Summary

- 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
- Empirical risk minimization 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 high chance to produce over-fitted models.

Evaluation of classifiers: possible pitfalls

- So far: only performance measure was generalization error based on $L_{\mathbf{zo}}$
- Another frequent problem mentioned already in Unit 1: unbalanced data sets
- What if misclassification cost depends on the sample's class? (A miss may be much more expensive than a false alarm.)
- Can we define a general performance measure independent of class distributions and misclassification costs?
- To answer these questions: introduce confusion matrices

Confusion matrix for binary classification: Part 1

For a given sample $(\mathbf{x}, y)$ with $y = \pm 1$ and a classifier $g(.) = \pm 1$ there are four possible cases:

- True Positive (TP) if $y = +1$ and $g(\mathbf{x}) = +1$ (hit)
- True Negative (TN) if $y = -1$ and $g(\mathbf{x}) = -1$ (correct rejection)
- False Positive (FP) if $y = -1$ and $g(\mathbf{x}) = +1$ (false alarm)
- False Negative (FN) if $y = +1$ and $g(\mathbf{x}) = -1$ (miss)

Confusion matrix for binary classification: Part 2

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

The entries #TP, #FP, #TN, and #FN denote the numbers of true positives, false positives, true negatives, and false negatives, respectively, for the given data set.

Evaluation measures

- Positives: #P $=$ #TP$+$#FN
- Negatives: #N $=$ #TN$+$#FP
- True Positive Rate (aka Recall, Sensitivity, Hit Rate): proportion of correctly identified positives: TPR $= \frac{\#\text{TP}}{\#\text{P}} = \frac{\#\text{TP}}{\#\text{TP}+\#\text{FN}}$.
- True Negative Rate (aka Specificity, Selectivity): proportion of correctly identified negatives: TNR $= \frac{\#\text{TN}}{\#\text{N}} = \frac{\#\text{TN}}{\#\text{TN}+\#\text{FP}}$.
- False Positive Rate (aka Fall-Out): proportion of negative examples that were incorrectly classified as positives: FPR $= \frac{\#\text{FP}}{\#\text{N}} = \frac{\#\text{FP}}{\#\text{TN}+\#\text{FP}} = 1 -$ TNR.
- False Negative Rate (aka Miss Rate): proportion of positive examples that were incorrectly classified as negatives: FNR $= \frac{\#\text{FN}}{\#\text{P}} = \frac{\#\text{FN}}{\#\text{TP}+\#\text{FN}} = 1 -$ TPR.
- Accuracy: proportion of correctly classified items: ACC $= \frac{\#\text{TP}+\#\text{TN}}{\#\text{TP}+\#\text{TN}+\#\text{FP}+\#\text{FN}}$.
- Precision (aka Positive Predicted Value): proportion of predicted positive examples that were correct: PREC $= \frac{\#\text{TP}}{\#\text{TP}+\#\text{FP}}$.

Evaluation measures 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.

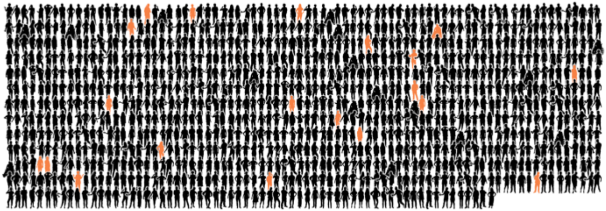
$$\text{MCC} = \frac{\#\text{TP}\cdot\#\text{TN}-\#\text{FP}\cdot\#\text{FN}}{\sqrt{(\#\text{TP}+\#\text{FP})(\#\text{TP}+\#\text{FN})(\#\text{TN}+\#\text{FP})(\#\text{TN}+\#\text{FN})}}$$

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

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

- Next: case study; then generalize previously introduced concepts to multi-class classification

Case study



Blackboard

Confusion matrix for multi-class classification

Now: 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

- The accuracy of a classifier $g(.)$ is defined as

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

  i.e. as proportion of correctly classified samples.

- Other evaluation measures cannot be generalized to the multi-class case in a straightforward way.

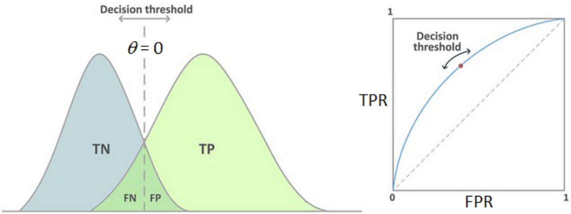
Other performance measures for multi-class classification

- However: We can 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 define all previously introduced evaluation measures (for class j).

Choice of Decision Threshold



Picture adapted from https://towardsdatascience.com/

- Decision threshold $\theta \to -\infty$: everything becomes labeled with $+1$, hence #TN $\to 0$, #FN $\to 0$, TPR $\to 1$, FPR $\to 1$
- Decision threshold $\theta \to +\infty$: everything becomes labeled with -1, hence #TP $\to 0$, #FP $\to 0$, TPR $\to 0$, FPR $\to 0$

★ ★

General performance of discriminant function

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

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

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

ROC Curves

- ROC stands for Receiver Operator Characteristic. Comes from signal detection theory.
- ROC curves are simple means for evaluating the performance of a binary classifier independent of class distributions and misclassification costs.
- Basic idea: plot true positive rate (TPR) vs. false positive rate (FPR) while varying the classification threshold.

ROC Curves: Practical Realizations

- 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 curve (AUC)

- $\text{AUC} = \int_0^1 \text{TPR}\,\text{dFPR} = \int_{+\infty}^{-\infty} \text{TPR}(\theta)\,\text{FPR}'(\theta)\,\text{d}\theta$
- 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 $\frac{1}{2}$ in average, hence:
  1. AUC $< \frac{1}{2}$: worse than random
  2. AUC $> \frac{1}{2}$: better than random.
- For which (binary) classifier would you pay more? (1) AUC = 0.75 or (2) AUC = 0.1?

★ ★

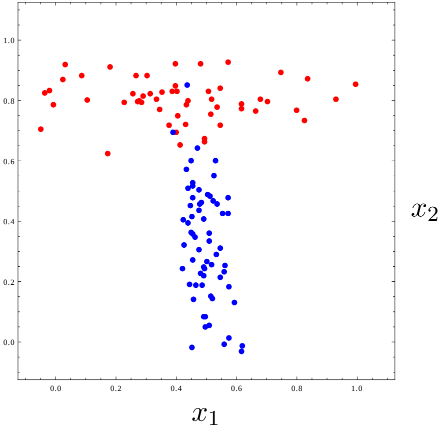
Example Problem

Consider the following classification results (already ordered):

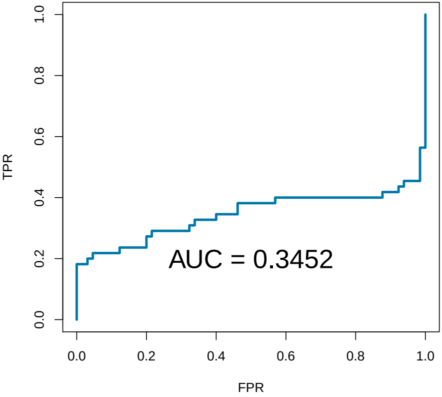
$\bar{g}(x)$	y
1.04	1
0.39	1
0.16	1
0.15	-1
0.08	1
-0.18	1
-0.27	-1
-0.39	-1
-0.52	-1
-0.99	-1

Compute and visualize the ROC curve by going through the (ordered) values $\bar{g}(x)$ and considering the correct or false classifications. Compute the AUC. (solution: blackboard)

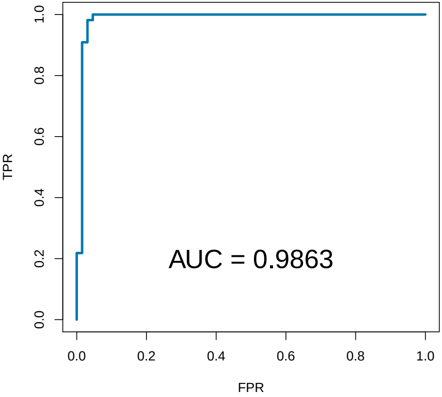
ROC-Example: Gaussian classifier revisited



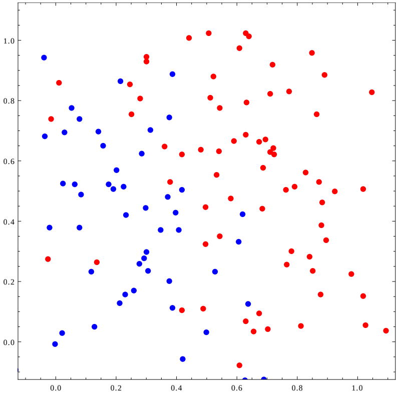
ROC curve for $\bar{g}((x_1, x_2)) = x_1$



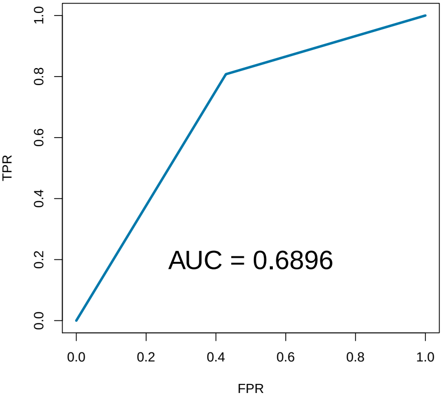
ROC curve for $\bar{g}((x_1, x_2)) = x_2$



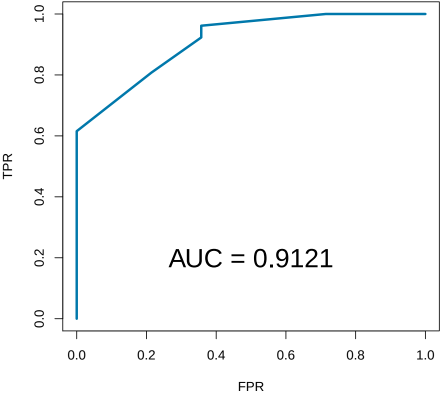
ROC example: k-NN example revisited (note: discriminant function by k-NN Regression)



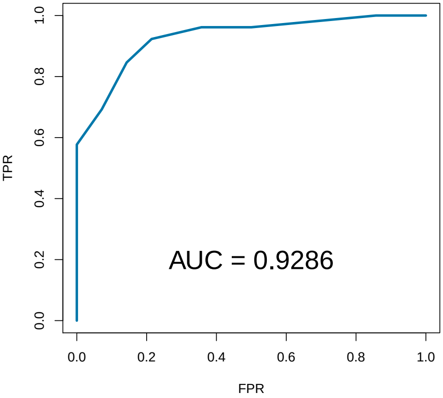
ROC curves for k-NN example (75% training, 25% test samples): $k = 1$



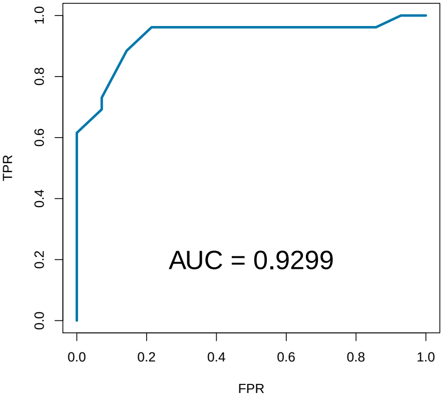
ROC curves for k-NN example (75% training, 25% test samples): $k = 5$



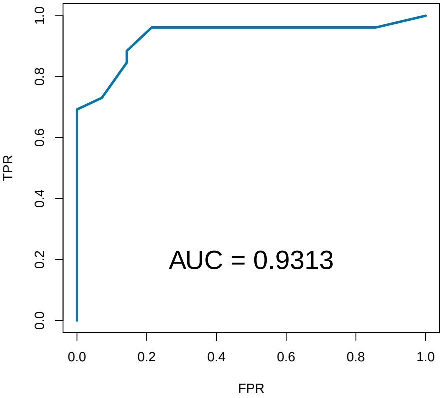
ROC curves for k-NN example (75% training, 25% test samples): $k = 9$



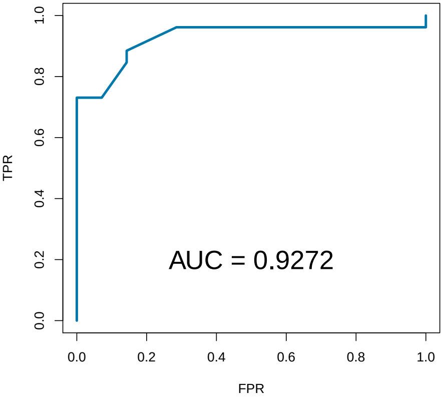
ROC curves for k-NN example (75% training, 25% test samples): $k = 13$



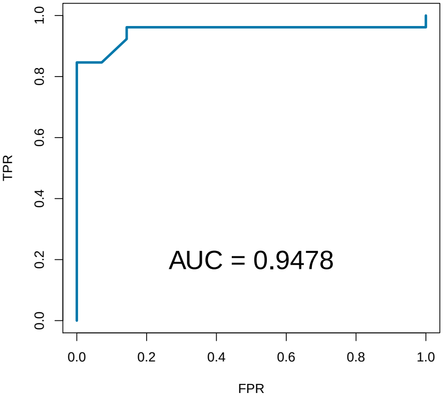
ROC curves for k-NN example (75% training, 25% test samples): $k = 17$



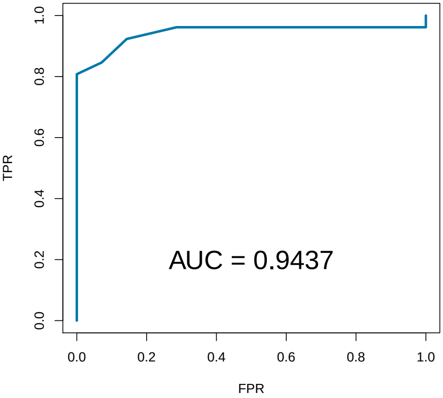
ROC curves for k-NN example (75% training, 25% test samples): $k = 21$



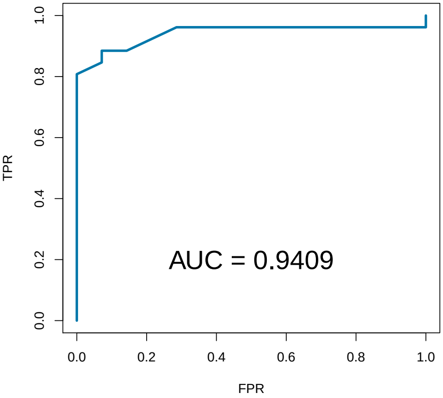
ROC curves for k-NN example (75% training, 25% test samples): $k = 25$



ROC curves for k-NN example (75% training, 25% test samples): $k = 29$



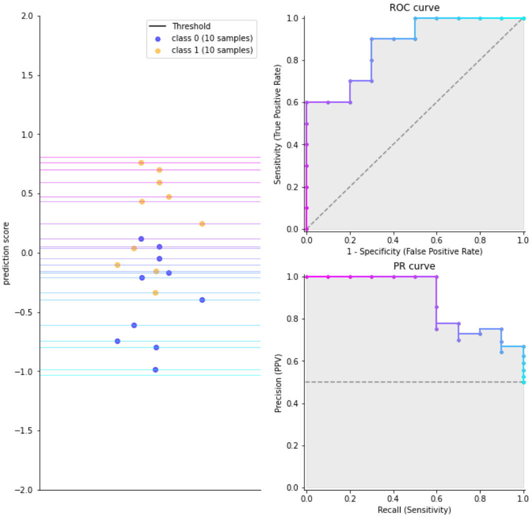
ROC curves for k-NN example (75% training, 25% test samples): $k = 33$



Precision-Recall (PR) curves

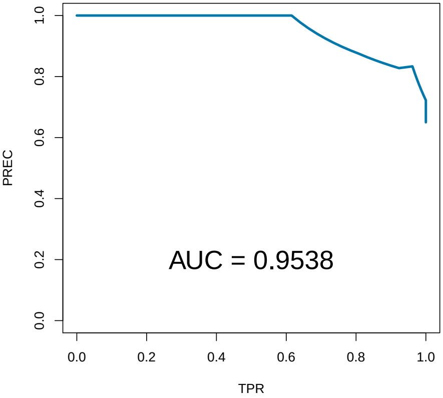
- For highly unbalanced data sets, in particular, if there are many true negatives, the ROC curves may not necessarily provide a very informative picture.
- For computing a precision-recall curve, similarly to ROC curves, sweep through all possible thresholds, but plot precision (vertical axis) versus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

ROC and PR

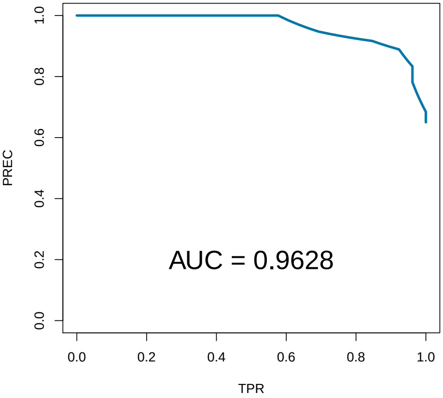


Source: link

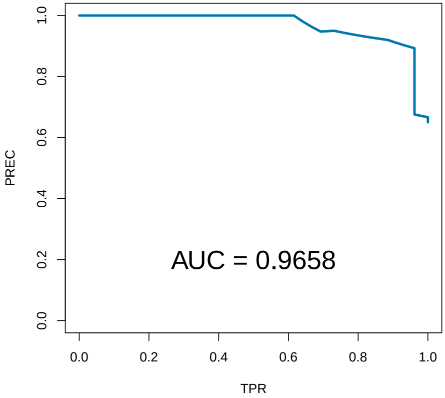
PR curves for k-NN example (75% training, 25% test samples): $k = 5$



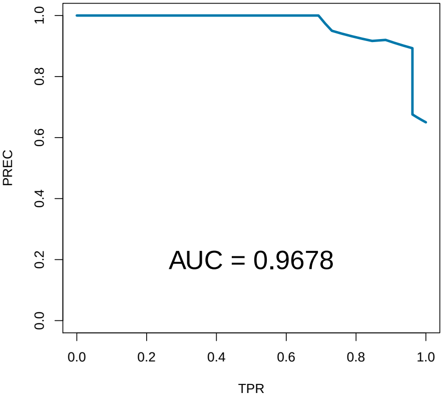
PR curves for k-NN example (75% training, 25% test samples): $k = 9$



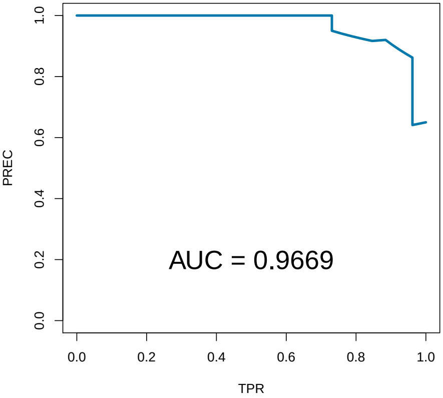
PR curves for k-NN example (75% training, 25% test samples): $k = 13$



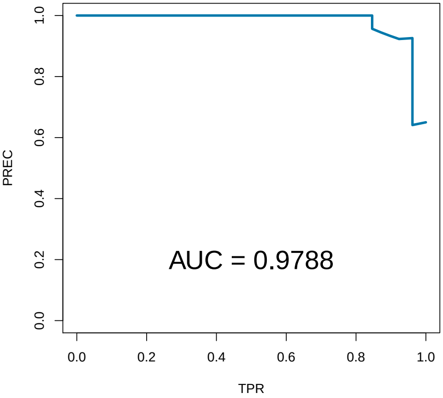
PR curves for k-NN example (75% training, 25% test samples): $k = 17$



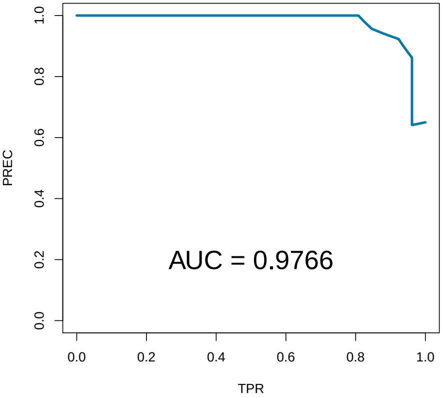
PR curves for k-NN example (75% training, 25% test samples): $k = 21$



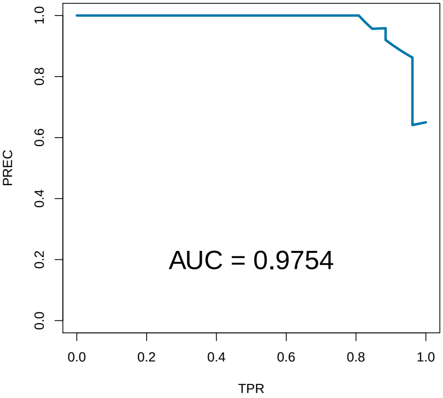
PR curves for k-NN example (75% training, 25% test samples): $k = 25$



PR curves for k-NN example (75% training, 25% test samples): $k = 29$



PR curves for k-NN example (75% training, 25% test samples): $k = 33$



Summary of Unit 2

- Recapitulation: basic concepts and notation from Unit 1
- The probabilistic framework and Bayes-optimal classifier
- Presentation of 3 introductory examples:
    1. A Gaussian classification task
    2. k-nearest neighbors
    3. Linear and polynomial regression
- The bias-variance trade-off
- Evaluation of classifiers for unbalanced data sets: Confusion matrix, ROC, AUC, PR curves

Next units: State-of-the-art methods for classification and regression