Machine learning foundations

Mitko Veta

Eindhoven University of Technology
Department of Biomedical Engineering

2022

Learning goals

At the end of this lecture you will:

- Have an understanding of the goal of machine learning (ML) models.
- *Have a good understanding of basic mathematical concepts used in ML and be able to apply them in the design and implementation of ML methods.
- Have a good understanding of the basic principles of machine learning (ML) and be able to apply them in the analysis of ML methods.
- Be able to design good experimental setups for developing ML models.
- *Have a good understanding of the different evaluation measures for ML models.

* Covered in video lectures

Overview

Topics covered in this lecture:

1. Gradient-based optimization

2. Two simple machine learning models
   - Linear model
   - Nearest-neighbours model

3. Model capacity, underfitting and overfitting

4. Model selection

5. Bias and variance trade-off

Gradient-based optimization

Materials:

- Chapters I.4 and I.5 from Goodfellow et al., *Deep Learning*
- Kolter et al., "Linear Algebra Review and Reference"

Gradient

- Let $f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$ be a function that takes $m \times n$ matrix $\boldsymbol{A}$ as input and returns a real number (scalar).
- A **gradient** of f with respect to A is the matrix

$$\nabla_{\boldsymbol{A}} f(\boldsymbol{A}) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} & \cdots & \frac{\partial f}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \frac{\partial f}{\partial A_{m2}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

- i.e. an $m \times n$ matrix with

$$(\nabla_{\boldsymbol{A}} f(\boldsymbol{A}))_{ij} = \frac{\partial f}{\partial A_{ij}}$$

- The size of the gradient of $\boldsymbol{A}$ is the same as the size of A.

Gradient

- In the special case when A is a vector we obtain the (possibly more familiar) gradient

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}$$

- In general to define a gradient we require that the function returns a **real** value.

Jacobian

- The Jacobian $\boldsymbol{J_f}$ is a generalization of the gradient for vector valued functions.
- Let $\boldsymbol{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a function that takes n-dimensional vector $\boldsymbol{x}$ as input and returns a m-dimensional vector as an output.
- The Jacobian $\boldsymbol{J_f}$ is defined as

$$\boldsymbol{J_f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

- Note that for the special case of a scalar-valued function, the Jacobian is the transpose of the gradient.

Optimization

- Most machine learning methods involve some kind of optimization.
  - One exception is the k-Nearest neighbour classifier introduced later.
- Optimization means minimizing or maximizing some function $f(\boldsymbol{x})$, i.e. finding the values of $\boldsymbol{x}$ for which $f(\boldsymbol{x})$ has a minimum or a maximum.
- Notation: $\boldsymbol{x}^* = \operatorname{argmin} f(\boldsymbol{x})$

Gradient-based optimization

- The derivative tells us how to change x in order to make a small improvement of $f(x)$.
- Therefore, derivatives can be useful in optimization.

Two simple machine learning models

Materials:

- Chapter 2.3 from Friedman et al., *The Elements of Statistical Learning*

The learning task

- Given a value of the input vector $\boldsymbol{x}$ make a good prediction of the output y, denoted as $\hat{y}$.
- Both y and $\hat{y}$ should take values from the same numerical set.
- Similarly, g and $\hat{g}$ should both take values from the same set $\mathcal{G}$.
- We suppose that we have available a set of measurements $(\boldsymbol{x}_i, y_i)$ or $(\boldsymbol{x}_i, g_i)$ $(1 \leq i \leq N)$ called **training data** (in matrix form: $(\boldsymbol{X}, \boldsymbol{y})$ and/or $(\boldsymbol{X}, \boldsymbol{g})$).
- Our task is to construct a prediction rule based on the training data.

The learning task

Example:

- **Variable values**: Let g (and therefore also $\hat{g}$) be two valued (categorical), e.g. $\mathcal{G} = \{\text{BLUE}, \text{ORANGE}\}$.
- **Encoding of gs with ys**: Then each class can be encoded binary, i.e., with $y \in \{0, 1\}$, e.g., BLUE and ORANGE, would correspond to 0 and 1, respectively.
- **Predicted output values**: $\hat{y}$ ranges over the interval $[-\infty, +\infty]$ (of which $\{0, 1\}$ is a subset).
- **Prediction rule**: $\hat{g}$ is assigned a (class label) BLUE if $\hat{y} < 0.5$ and ORANGE, otherwise.

Two simple approaches to prediction

- Linear model fit
  - strong assumptions about the structure of the decision boundary
- k-nearest neighbours
  - weak assumptions about the structure of the decision boundary

Linear model fit by least squares

- Despite relative simplicity one of the most important statistical tools
- Input vector $\boldsymbol{x}^T = (x_1, x_2, \ldots, x_p)$
- Output y predicted using the model

$$\hat{y} = \hat{w}_0 + \sum_{j=1}^{p} x_j \hat{w}_j$$

- $\hat{w}_i$ $(0 \leq i \leq p)$ are the parameters of the linear model
- In vector form

$$\hat{y} = \hat{\boldsymbol{w}}^T \boldsymbol{x} = \boldsymbol{x}^T \hat{\boldsymbol{w}}$$

using the fact that the scalar (inner) product of two vectors is a commutative operation.

Linear model fit by least squares

- We assume that w_0 is in $\boldsymbol{w}$ and 1 is included in $\boldsymbol{x}$.
- $\hat{y}$ is a scalar, but in general can be a k-vector $\hat{\boldsymbol{y}}$, in which case $\boldsymbol{w}$ becomes a $p \times k$ matrix of coefficients.

Linear model fit by least squares

- There are many ways to fit a linear model to a training dataset.
- **Least squares** method
  - We need to find coefficients $\hat{w}_i$ which minimize the error estimated with the **residual sum of squares**

$$\text{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2$$

    assuming N input-output pairs.
- $\text{RSS}(\boldsymbol{w})$ is a quadratic function.
- A minimum always exists though not necessarily a unique one.

Linear model fit by least squares

- We look for the solution $\hat{\boldsymbol{w}}$ using the matrix notation:
- $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T$ is the vector formed from the N output vectors and $\boldsymbol{X}$ is an $N \times p$ matrix

$$\text{RSS}(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- To find the minimum we differentiate with respect to $\boldsymbol{w}$ which gives

$$-2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

For details about the derivation check equations 4 and 5 in this document.

Linear model fit by least squares

- To find the minimum our derivative must be $\mathbf{0}$, hence:

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

$$\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \mathbf{0}$$

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$$

- If $\boldsymbol{X}^T\boldsymbol{X}$ is non-singular there exists a unique solution given by

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Linear model fit by least squares

- For each input $\boldsymbol{x}_i$ there corresponds the fitted output

$$\hat{y}_i = \hat{y}_i(\boldsymbol{x}_i) = \hat{\boldsymbol{w}}^T \boldsymbol{x}_i$$

.

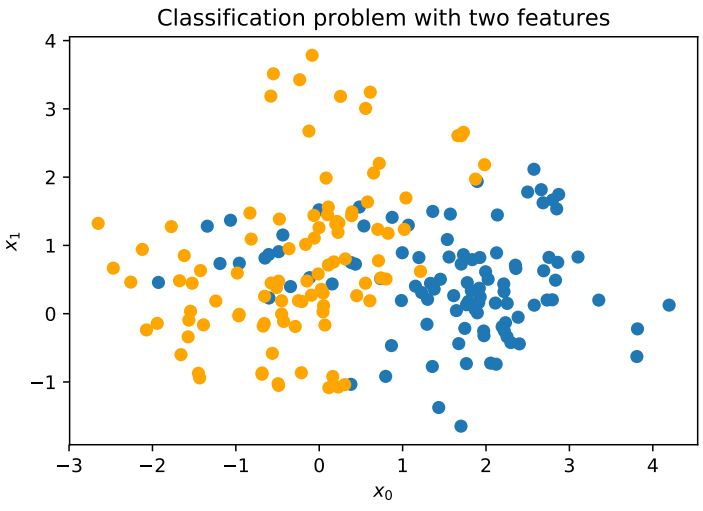
- This is called "making a prediction" for $\boldsymbol{x}_i$.
- The entire fitted surface (hyperplane) is fully characterized by the parameter vector $\hat{\boldsymbol{w}}$.
- After fitting the model, we can "discard" the training dataset.

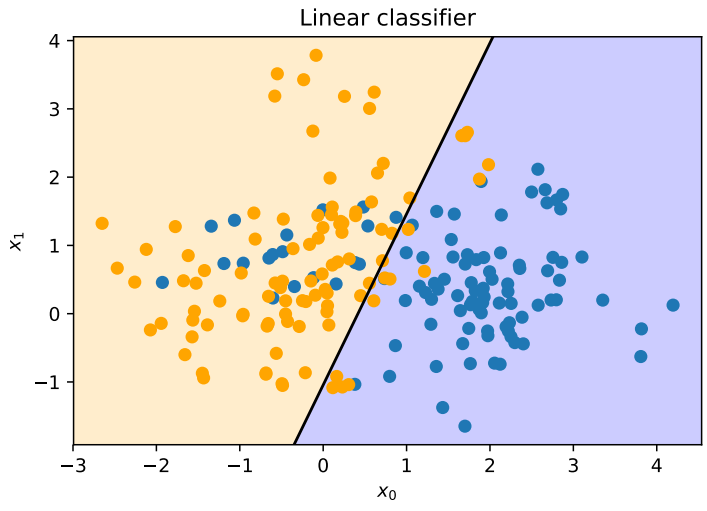
Example: Linear model fit by least squares

- Scatter plot (on next slide) of training data on a pair of inputs x_1 and x_2
- Output class variable g has two values BLUE and ORANGE.
- Linear regression model fitted with the response variable y coded as 0 for BLUE and 1 for ORANGE.
- Fitted values $\hat{y}$ converted to a fitted class variable $\hat{g}$ as

$$\hat{g} = \begin{cases} \text{BLUE} & \text{if } \hat{y} \leq 0.5 \\ \text{ORANGE} & \text{if } \hat{y} > 0.5 \end{cases}$$

Example: Linear model fit by least squares





Example: Linear model fit by least squares

- Wrong classifications on both sides of the boundary
- Are the errors caused by the model or are they unavoidable?
- Two possible scenarios
  - **Scenario 1**: data generated from bivariate Gaussian distribution
  - **Scenario 2**: data generated from 10 Gaussian distributions; the means of these distributions are also distributed as Gaussian
- In Scenario 1 the linear boundary is the best we can do since the overlap is inevitable.
- In Scenario 2 the linear boundary is unlikely to be optimal (in fact the boundary is non-linear and disjoint).

Nearest-neighbours model

- In nearest-neignbour methods $\hat{y}(\boldsymbol{x})$ is determined based on the inputs (points) in the training set $\mathcal{T}$ which are "closest" to the input $\boldsymbol{x}$.
- k-nearest neighbour fit is defined as

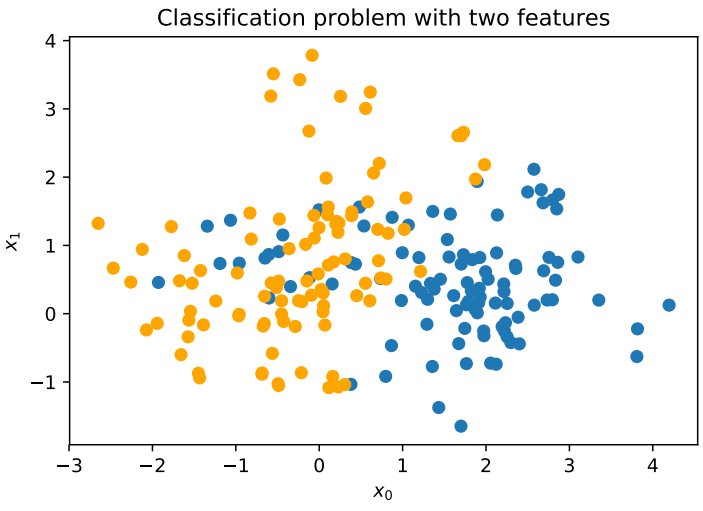
$$\hat{y}(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x}_i \in N_k(\boldsymbol{x})} y_i$$

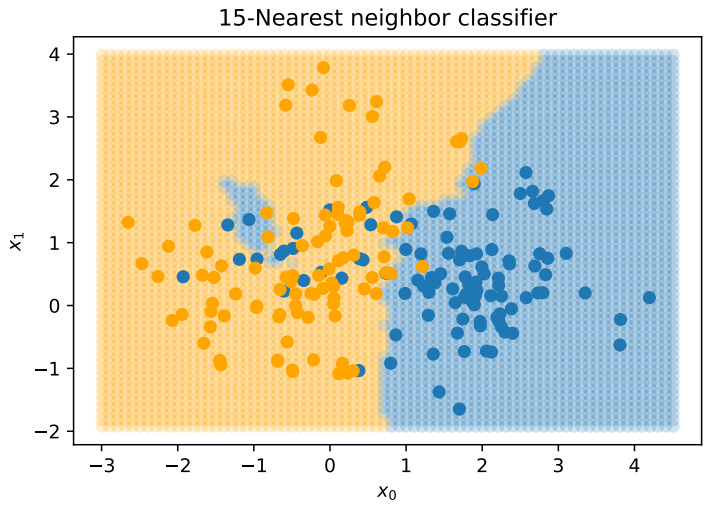
  where $N_k(\boldsymbol{x})$ is the neighbourhood of $\boldsymbol{x}$ consisting of the k "closest" points to $\boldsymbol{x}$.
- "Closeness" requires a definition of **metrics**.
- For the moment we assume Euclidian distance (each $\boldsymbol{x}$ is a point in the hyperspace).
- An average of the classes of the k closest points (but only for binary classification problem.

Back to the BLUE and ORANGE example

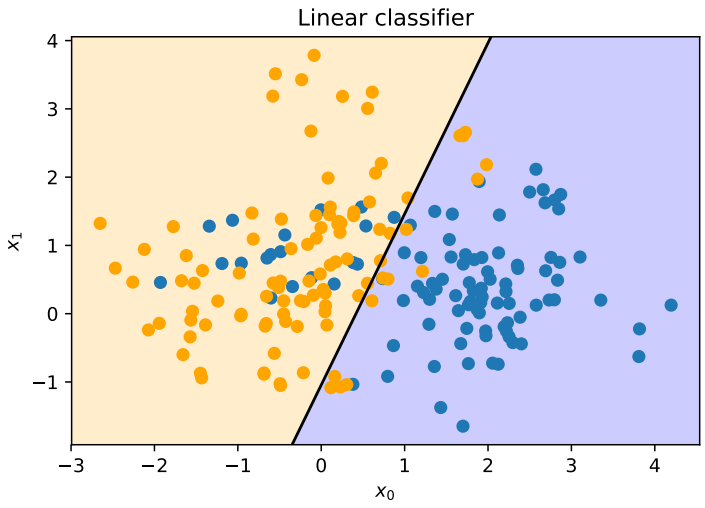
- We use the same training data as in the linear model example.
- New borderline between the classes generated with 15-nearest-neighbour model.
- Since ORANGE is encoded as 1 $\hat{y}$ is the proportion of ORANGE points in the 15-neighbourhood.
- Class ORANGE assigned to $\boldsymbol{x}$ if $\hat{y}(\boldsymbol{x}) > 0.5$ (majority is ORANGE).

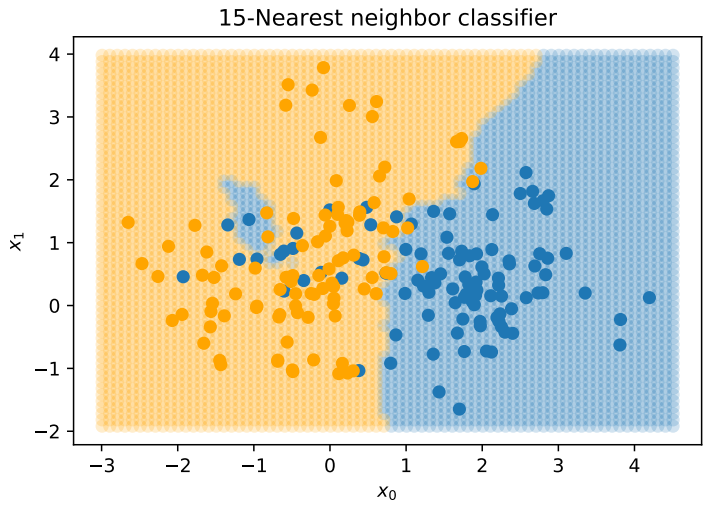
15-Nearest neighbour classifier



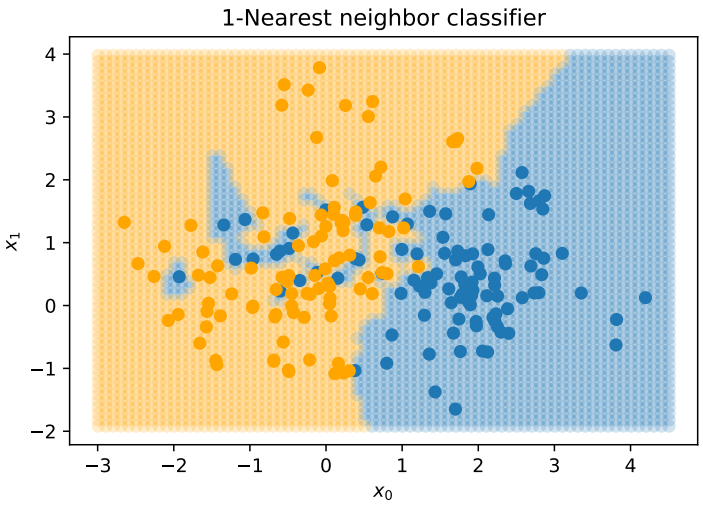


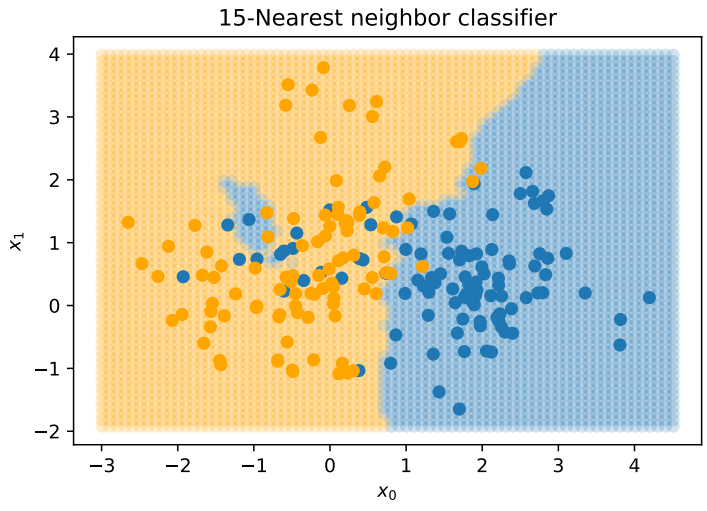
Linear classifier vs. 15-Nearest neighbour





1-Nearest neighbour vs. 15-Nearest neighbour





Comparison of techniques

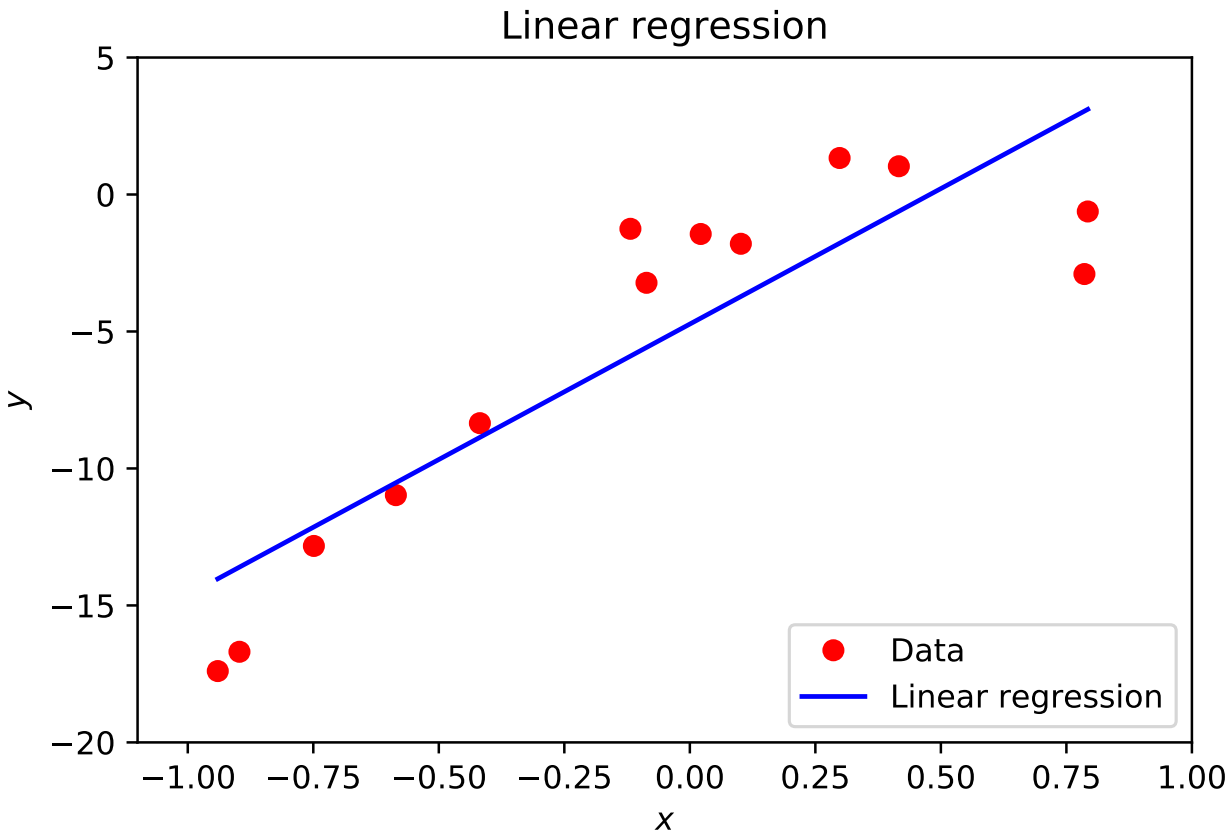
- At first sight it looks like k-NN has only one parameter, k versus p parameters (number of weights w_i) of the linear model.
- The **effective** number of parameters of k-NN is N/k which is in general bigger than p (N is the size of the training set).
- For instance, assume non-overlapping neighbourhoods
  - There will be N/k neighbourhoods.
  - To each neighbourhood there correspond one parameter (the mean of the elements of the neighbourhood).

Model capacity, underfitting and overfitting

Materials:

- Chapter I.5.2 from Goodfellow et al., *Deep Learning*

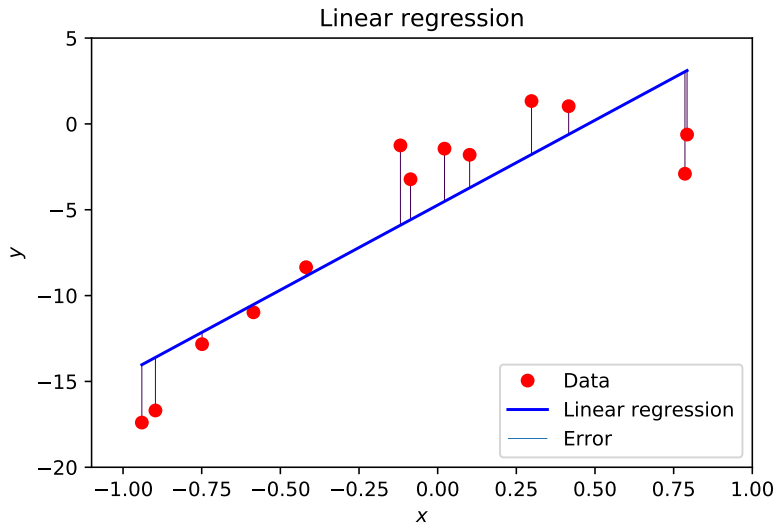
Linear regression

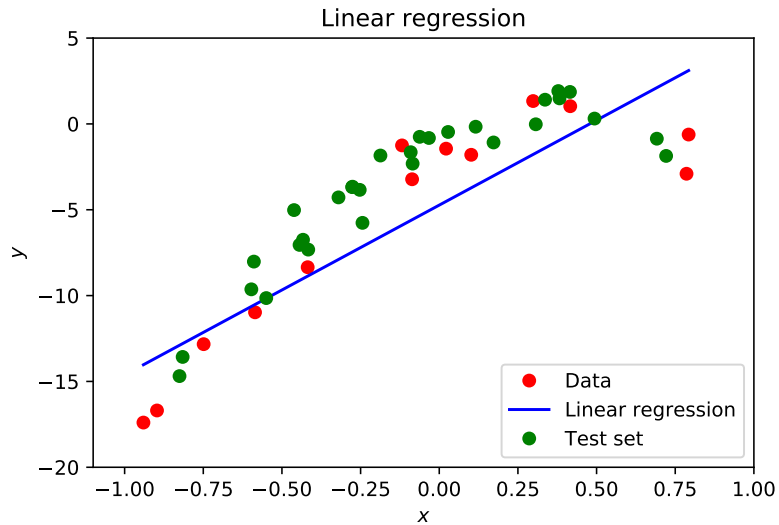


$$\hat{y} = \hat{w}_0 + \sum_{i=1}^{n} x_i \hat{w}_i$$
$$\hat{y} = \boldsymbol{x}^T \hat{\boldsymbol{w}}$$

Generalization

- The central challenge in machine learning is to design an algorithm which will perform well on new data (different from the training set data).
- This ability is called **generalization**.
- **Training error** is the error computed on the training set.





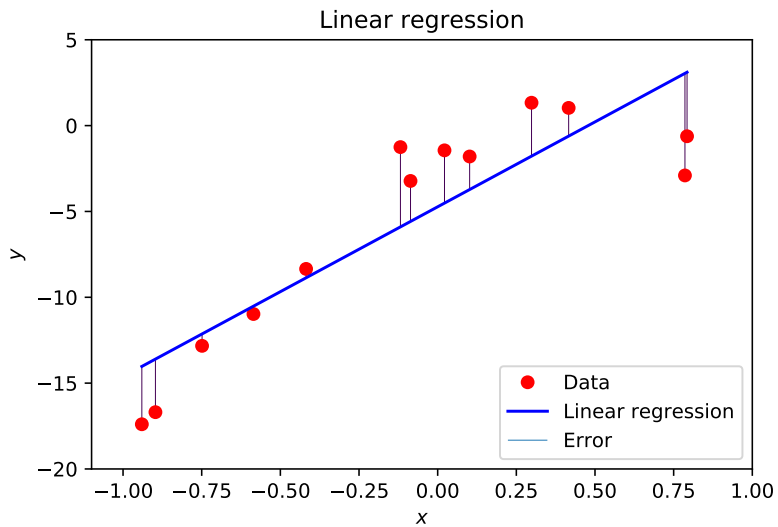
Example: Linear regression

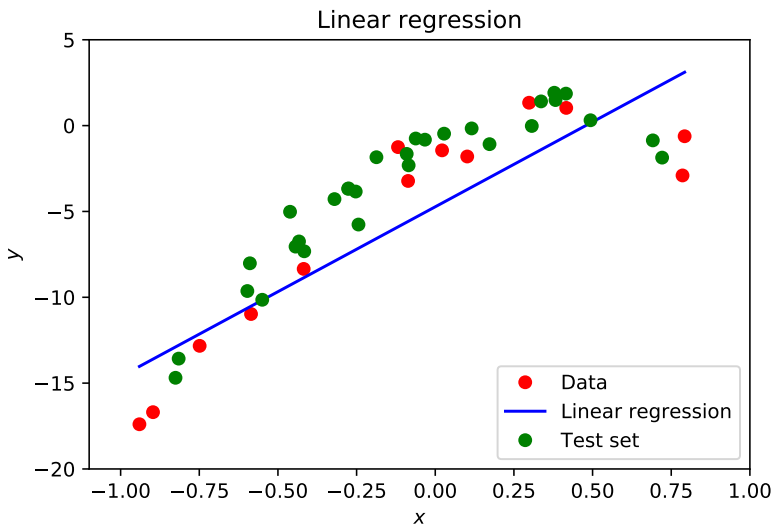
- Previously, we trained the model by minimizing the training error

$$\frac{1}{m^{(\text{train})}} \left\| \boldsymbol{X}^{(\text{train})} \hat{\boldsymbol{w}} - \boldsymbol{y}^{(\text{train})} \right\|_2^2$$

- We would like actually to minimize the test error

$$\frac{1}{m^{(\text{test})}} \left\| \boldsymbol{X}^{(\text{test})} \hat{\boldsymbol{w}} - \boldsymbol{y}^{(\text{test})} \right\|_2^2$$





Statistical learning theory

- **Statistical learning theory** provides methods to mathematically reason about the performance on the test set although we can observe only the training set.
- This is possible under some assumptions about the data sets
  - The training and test data are generated by drawing from a probability distribution over data sets. We refer to that as **data-generating process**.
  - **i.i.d. assumptions**
    - Examples in each data sets are **independent** from each other.
    - The training data set and the test data set are **identically distributed**, i.e., drawn from the same probability distribution.

Model capacity

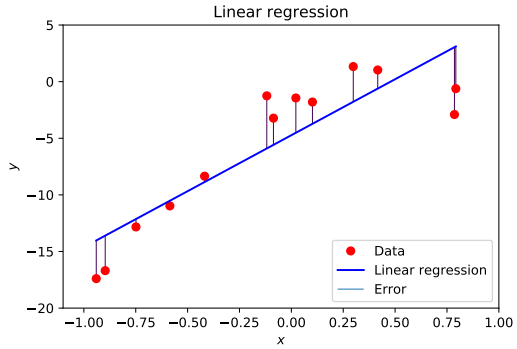
- A **capacity of the model** is its ability to fit a wide variety of functions.
- The capacity can be controlled by choosing its **hypothesis space**, i.e. the set of functions from which the learning algorithm is allowed to select the solution.
- Example: The linear regression algorithm has the set of all linear functions as its hypothesis space.

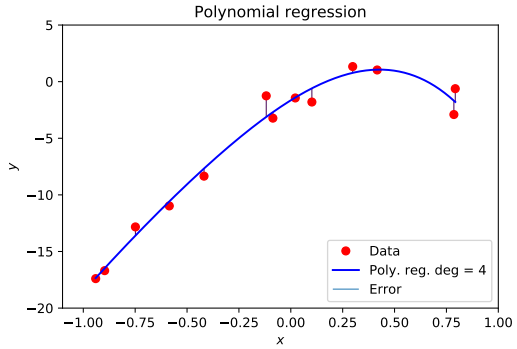
Polynomial regression

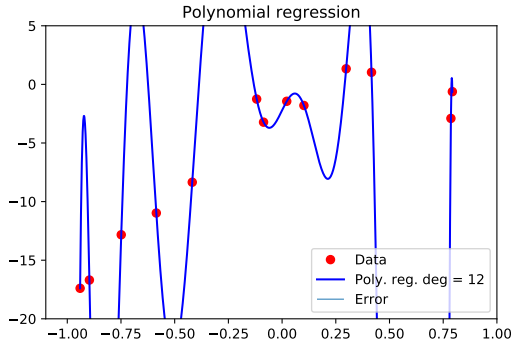
- The linear regression algorithm can be generalized to include all polynomial functions instead of just the linear ones.
- Moving to degree two to we obtain: $\hat{y} = b + w_1 x + w_2 x^2$.
  - This can be seen as adding a new feature x^2.
  - In fact, we can generalize this approach to create all sorts of hypothesis spaces, e.g.: $\hat{y} = b + w_1 x + w_2 \sin(x) + w_3 \sqrt{x}$.
- The **outuput** is still a **linear** function of the parameters, so in principle it can be trained in the same way as the linear regression.

Polynomial regression

A comparison of a linear, degree-4, and degree-12 polynomials as predictors

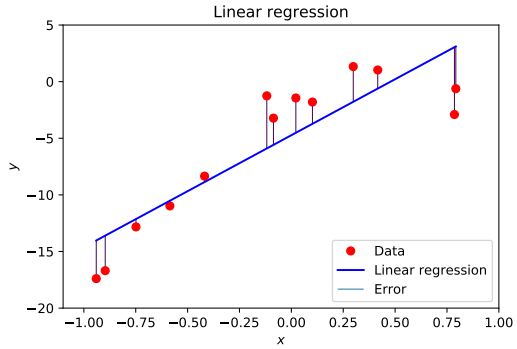


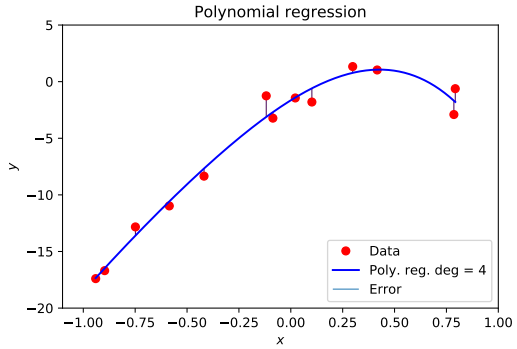


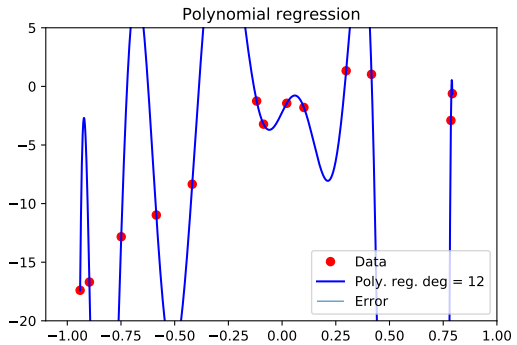


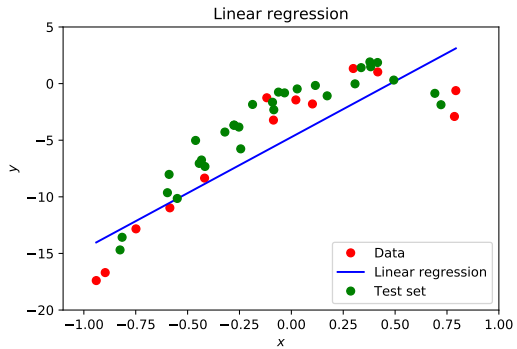
Polynomial regression

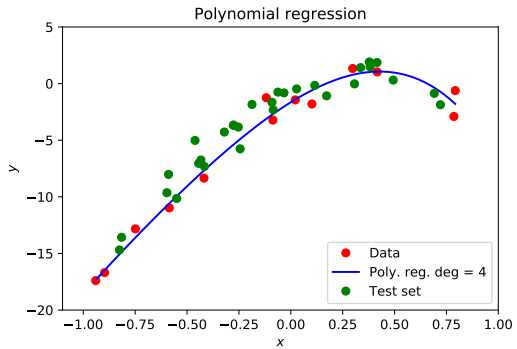
A comparison of a linear, degree-4, and degree-12 polynomials as predictors

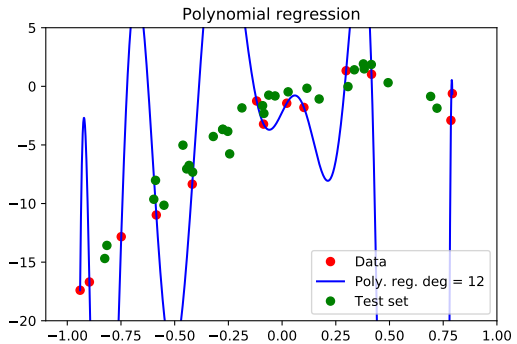








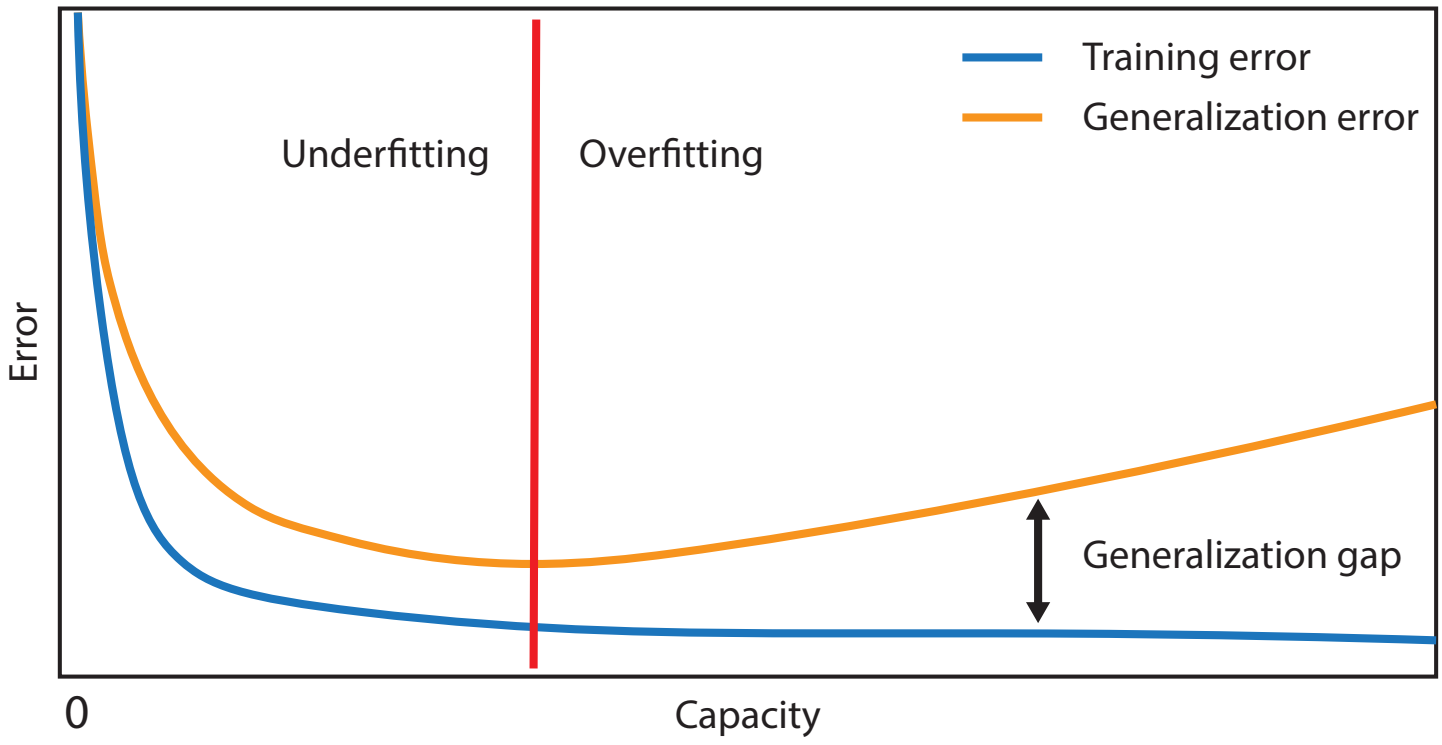




Generalization and capacity

- Simpler functions generalize more easily, but we still need to choose a sufficiently complex hypothesis (function) to obtain small training error.
- Typically training error decreases with the increase of the model capacity until an (asymptotic) value is reached.
- The generalization error is U-shaped with the capacity range split in an underfitting and an overfitting zone.

Generalization and capacity



Regularization

- In addition to increasing and decreasing of the hypothesis space, i.e., the capacity, we can influence the learning algorithm by **giving preference to one solution over another in the hypothesis space**.
- In case both functions are eligible we can define a condition to express preference about one of the functions.
- The unpreferred solution is chosen only if it gives significantly better performance with the training data.
- *More on regularization in the next lecture.*

Model selection

Materials:

- Chapter I.5.3 from Goodfellow et al., *Deep Learning*

Hyperparameters and validation sets

- **Hyperparameters** are settings that can be used to control the behaviour of the algorithm.
- In general, the hyperparameters are not modified by the learning algorithm itself.
- **Example**: In **polynomial regression** the degree of the polynomial is a **capacity** hyperparameter.
- A setting can be chosen to be hyperparameter when it is **difficult to optimize** or - more often - when its derivation from the training set **can lead to overfitting**.
  - Example: in polynomial regression we can always fit the data better with a higher degree polynomial.

Choice of training, validation, and test sets

- The **validation set** is used during training to predict the behaviour (generalization error) of the algorithm on new data, i.e., on the test set and to chose the hyperparameters.
- Ideally these two sets are disjoint.
- The validation set is chosen from the training data.
- The training data is split in two disjoint subsets.

Choice of training, validation, and test sets

Training	Validation	Test
Used to find the optimal **parameters** of the model.	Used to find the optimal **model** (hyper-parameters).	Used to estimate the **performance** of the optimal model.
w	$f(\cdot)$	$\|\|\hat{y} - y\|\|$

Cross-validation

- Dividing the data set into disjoint training and test sets can result in a result in a too small validation and/or test set.
- In such cases all data is used to estimate the generalization error.
- We use procedures that repeat the training and testing on different randomly chosen subsets or splits of the original data set.
- The most common such procedure is the **k-fold cross-validation**.

Expectation (recap)

- The **expectation** or **expected** value of a function $f(x)$ with respect to a probability distribution $P(x)$ is the average value of f over all values x assuming they are drawn from P

$$\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x)$$

$$\mathbb{E}_{x\sim P}[f(x)] = \int p(x)f(x)dx$$

Variance (recap)

- The **variance** gives a measure of variation of the values of a random variable x

$$\mathrm{Var}(f(x)) = \mathbb{E}[(f(x) - E[f(x)])^2]$$

Bias and variance trade-off

Materials:

- Chapter I.5.4 from Goodfellow et al., *Deep Learning*

Point estimation

- For efficient design of learning algorithms it is useful to have formal characterizations of notions like generalization, overfitting and underfitting.
- To this end we introduce some definitions.
- **Point estimation** is the attempt to provide the single "best" prediction of some quantity of interest.
- The quantity of interest can be a single parameter, parameter vector of some model, e.g., the weights $\boldsymbol{w}$ in the linear regression model.

Point estimation

- Given a parameter $\boldsymbol{\theta}$ we denote its point estimate with $\hat{\boldsymbol{\theta}}$.
- As usual, let $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ be m independent and identically distributed (i.i.d.) data points.
- A **point estimator** or **statistic** is any function of the data

$$\hat{\boldsymbol{\theta}}_m = g(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}))$$

- This definition is very general. For instance, that the value returned by g need not be close to the true value $\boldsymbol{\theta}$. Also g might return a value which is outside the values that $\boldsymbol{\theta}$ is allowed to have.

Point estimation

- Of course, a good estimator is still a function that returns values close to θ.
- Since the data is drawn from a random process, point estimate $\hat{\theta}$ is considered to be a random variable and θ is fixed, but unknown parameter.

Bias

- A bias of an estimator $\hat{\boldsymbol{\theta}}_m$ is defined as

$$\text{bias}(\hat{\boldsymbol{\theta}}_m) = \mathbb{E}(\hat{\boldsymbol{\theta}}_m) - \boldsymbol{\theta}$$

  where the expectation is over the data and $\boldsymbol{\theta}$ is the true underlying value.
- An estimator $\hat{\boldsymbol{\theta}}_m$ is **unbiased** if $\text{bias}(\hat{\boldsymbol{\theta}}_m) = 0$. Note that this implies $\mathbb{E}(\hat{\boldsymbol{\theta}}_m) = \boldsymbol{\theta}$.
- $\hat{\boldsymbol{\theta}}_m$ is **asymptotically unbiased** if $\lim_{m\to\infty} \text{bias}(\hat{\boldsymbol{\theta}}_m) = 0$ (implying $\lim_{m\to\infty} \mathbb{E}(\hat{\boldsymbol{\theta}}_m) = \boldsymbol{\theta}$).

Bias: example

- **Example**: Consider samples $\{x^{(1)}, \ldots, x^{(m)}\}$ i.i.d distributed according to the Gaussian distribution

$$p(x^{(i)}; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{x^{(i)} - \mu}{\sigma^2}\right)$$

- The **sample mean** is a common estimator of the Gaussian mean parameter

$$\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} x(i)$$

Bias: example

We compute the bias as expectation by substituting the Gaussian distribution in the formula

$$\begin{aligned}
\text{bias}(\mu_m) &= \mathbb{E}[\mu_m] - \mu \\
&= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right] - \mu \\
&= \left(\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}[x^{(i)}]\right) - \mu \\
&= \left(\frac{1}{m}\sum_{i=1}^{m} \mu\right) - \mu \\
&= \mu - \mu = 0
\end{aligned}$$

The sample mean is an unbiased estimator of Gaussian mean parameter.

Bias: example

- **Example**: Estimators of the variance of a Gaussian distribution
- We compare two different estimators of the variance σ^2 parameter
- **Sample variance**

$$\hat{\sigma}^2 = \frac{1}{m} \sum_{1}^{m} \left(x^{(i)} - \hat{\mu}_m \right)^2$$

where $\hat{\mu}$ is the sample mean.

- We are interested in computing

$$\text{bias}(\hat{\sigma}_m^2) = \mathbb{E}[\hat{\sigma}_m^2] - \sigma^2$$

Bias: example

- First we evaluate $\mathbb{E}[\hat{\sigma}_m^2]$:

$$\mathbb{E}[\hat{\sigma}_m^2] = \mathbb{E}\left[\frac{1}{m}\sum_{1}^{m}\left(x^{(i)} - \hat{\mu}_m\right)^2\right] = \frac{m-1}{m}\sigma^2$$

- Back to the bias

$$\text{bias}(\hat{\sigma}_m^2) = \mathbb{E}[\hat{\sigma}_m^2] - \sigma^2 = \frac{m-1}{m}\sigma^2 - \sigma^2 = -\frac{\sigma^2}{m}$$

- Therefore the sample variance is a **biased** estimator.

Bias: example

- The **unbiased variance estimator** is defined as

$$\tilde{\sigma}^2 = \frac{1}{m-1} \sum_{1}^{m} \left(x^{(i)} - \hat{\mu}_m \right)^2$$

- Indeed

$$\mathbb{E}[\tilde{\sigma}_m^2] = \mathbb{E}\left[\frac{1}{m-1} \sum_{1}^{m} \left(x^{(i)} - \hat{\mu}_m \right)^2 \right] = \frac{m-1}{m-1} \sigma^2 = \sigma^2$$

and the bias is 0.

Variance and standard error

- Another important feature of an estimator is its variance.
- The **variance** of an estimator is simple its statistical variance $\text{Var}(\hat{\theta})$ over the training set as a random variable.
- Alternatively we can compute the **standard error** (the square root of the variance) $\text{SE}(\hat{\theta})$.
- The variance or the standard error provide a measure how much the estimate would vary as we resample the data independently from the underlying data generating process.
- We would prefer a relatively low variance of the estimator.

Variance and standard error

- The standard error of the mean estimator is given as

$$\mathrm{SE}(\hat{\mu}) = \sqrt{\mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}}$$

  where σ is the true variance of the distribution, i.e., the samples $x^{(i)}$.
- Neither the square root of the sample variance nor the square root of the unbiased estimator of the variance give an unbiased estimate of the standard deviation.
- Both approaches underestimate the true standard deviation.
- However, for large m the approximation works quite well.

Variance and standard error

- In machine learning the generalization error is estimated based on the sample mean of the error on the test set.
- The accuracy of the estimate depends on the number of the examples.
- From the statistical theory (central limit theorem) we know that the mean is distributed with normal distribution for which we can establish confidence intervals.
- For instance, the 95% confidence interval is given by

$$[\hat{\mu_m} - 1.96\text{SE}(\hat{\mu}_m), \hat{\mu_m} + 1.96\text{SE}(\hat{\mu}_m]$$

- Then we can say that algorithm A is better than algorithm B of the confidence upper bound for the error of A is less than the corresponding lower bound of B.

Trading off bias and variance to minimize mean squared error

- Bias and variance measure two different sources of error in an estimator.
- Bias measures the expected deviation with the true value of the estimator.
- Variance provides a measure of the deviation from the expected value of the estimator depending on the particular data sampling.

Trading off bias and variance to minimize mean squared error

- Often we need to make a trade-off between these two.
- The most common way to do this is via cross-validation.
- An alternative is to compare the **mean squared error** (MSE) of the estimates.

$$\text{MSE} = \mathbb{E}[(\hat{\theta}_m - \theta)^2] = \text{bias}(\hat{\theta}_m)^2 + \text{Var}(\hat{\theta}_m)$$

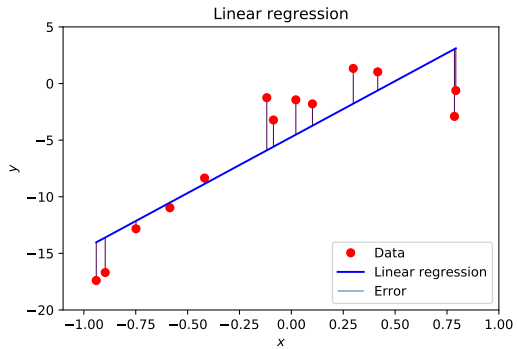
- The smaller MSE the better - so minimizing both the bias and variance is always preferable.

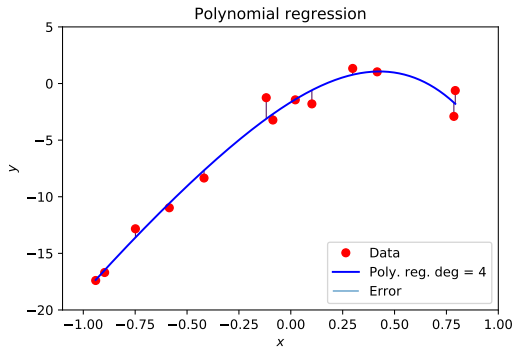
Bias and variance

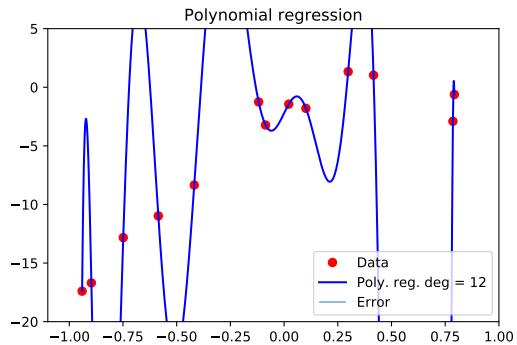
- Our original goal was to provide a mathematical support for the notions of capacity, underfitting, and overfitting.
- Indeed there is a close relationship between these three concepts and bias and variance.
- When generalization error is measured by MSE (and hence indirectly via bias and variance) increasing capacity tends to increase variance and decrease bias.
- Again the generalization as a function of capacity is given by an U-shaped curve.

Discussion

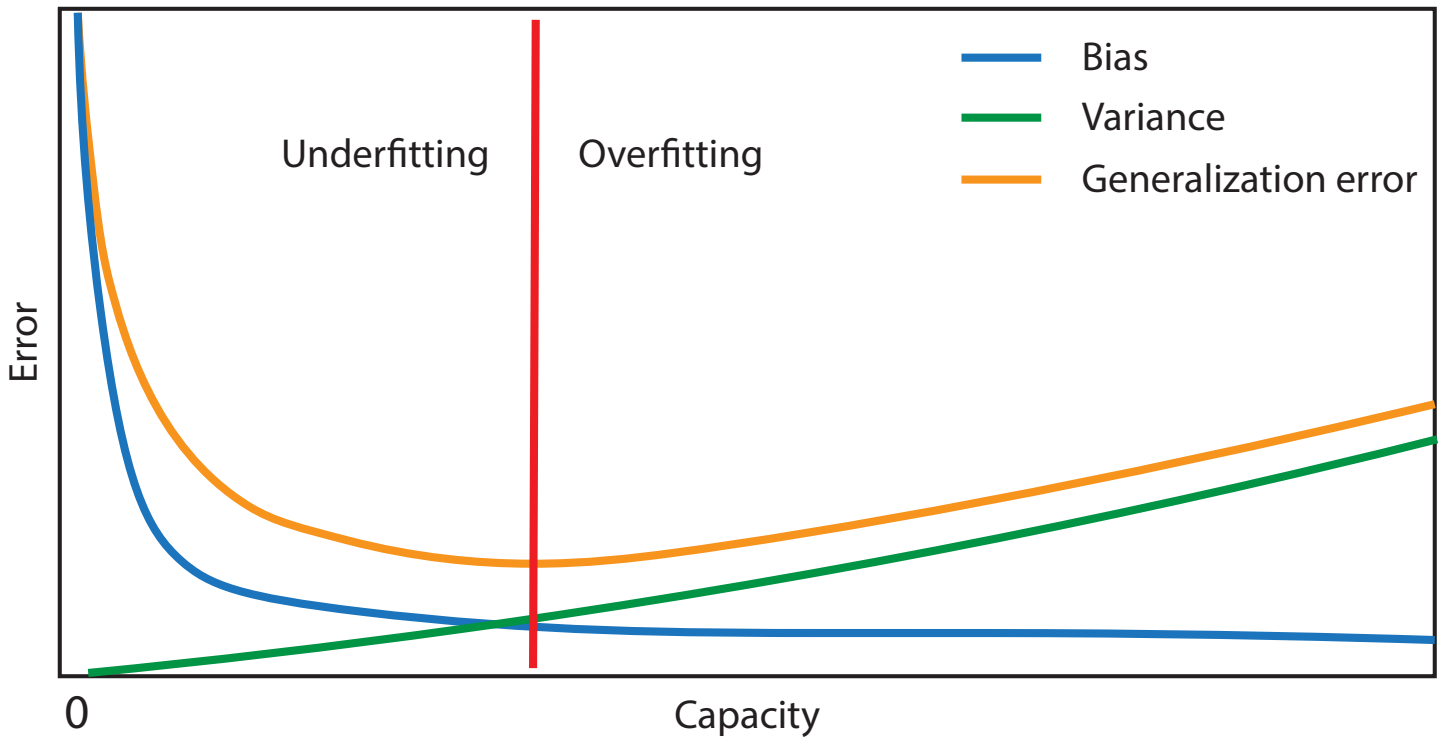
How will the estimated regression model change when one training data point is replaced with another one?







Bias and variance



Important: introductory topics covered in video lecture

These topics are covered in the extended slide stack and video lecture (links in Cavas) and are exam material:

- Linear algebra
- Probability theory
- Maximum-likelihood estimation
- Supervised and unsupervised algorithms
- Ensambling

Acknowledgements

The slides for this lecture were prepared by Mitko Veta and Dragan Bošnacki.
Some of the slides are based on the accompanying lectures of Goodfellow et al., *Deep Learning*.

References

- Friedman, J., T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics New York, 2001.
- Goodfellow, I., Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- Kolter, Z. and C. Do. "Linear Algebra Review and Reference". In: (2015).