# Cognitive Algorithms
## Lecture 3

# Linear Regression

Klaus-Robert Müller, Johannes Niediek,
Hannah Boldt, Augustin Krause, Jonas Müller, Joanina Oltersdorff, Ken Schreiber

Technische Universität Berlin
Machine Learning Group

# Organizational

**Tutorials next week (changes due to holiday)**

| | | | |
|---|---|---|---|
| Hannah | H2053 | Tue 21.05. | 12.00 |
| Augustin | FH 314 | Tue 21.05. | 14.00 |
| Joanina | Zoom | Wed 22.05. | 10.00 |
| Jonas | MAR 4.033 | Thu 23.05. | 10.00 |

**Bi-weekly tutorials from June onwards**

| | | | |
|---|---|---|---|
| Joanina | Zoom | Mondays | 10.00 |
| Hannah | H2053 | Tuesdays | 12.00 |
| Augustin | | no more tutorials | |
| Jonas | FH 303 | Mondays | 14.00 |

Please remember the Q&A session via Zoom by Ken, each Wednesday at 12.00.

# Summary of last lecture

- Correlations between features can affect classification accuracy
- Linear Discriminant Analysis (LDA) maximizes *between class variance* while minimizing *within class variance*
- If data has multivariate normal distribution with equal class covariances, then LDA is the optimal classifier
- We want our model to **generalize** well. We need to test this on data that was not used during training.

# Estimating covariance matrices

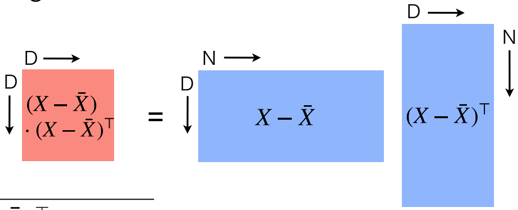
Given $n$ data points $\boldsymbol{x}_i \in \mathbb{R}^d$ in a data matrix $X \in \mathbb{R}^{d\times n}$ the empirical estimate of the **covariance matrix** is defined as

$$\hat{\Sigma} = \frac{1}{n}(X - \bar{X})(X - \bar{X})^\top ,$$

where the estimate of the expected value is given by the mean

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i , \quad \bar{X} = (\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \ldots, \bar{\boldsymbol{x}}) \in \mathbb{R}^{d\times n}$$

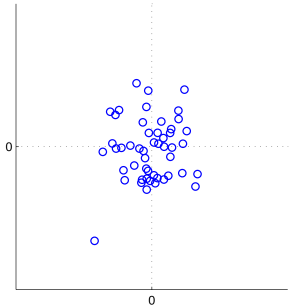
The diagonal entries of $\hat{\Sigma}$ are estimates of the variance.




We call $(X - \bar{X})(X - \bar{X})^\top$ the *empirical scatter matrix*

# Correlated data and linear mappings

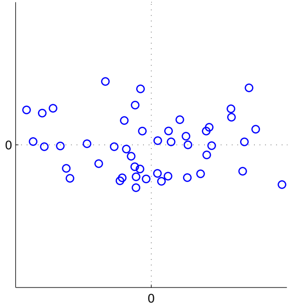
Uncorrelated




$$x \sim \mathcal{N}(0,1)$$

$$XX^\top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

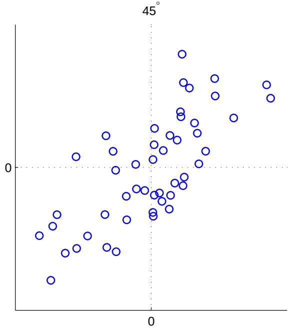
Uncorrelated, scaled




$$\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} X$$

$$XX^\top = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}$$

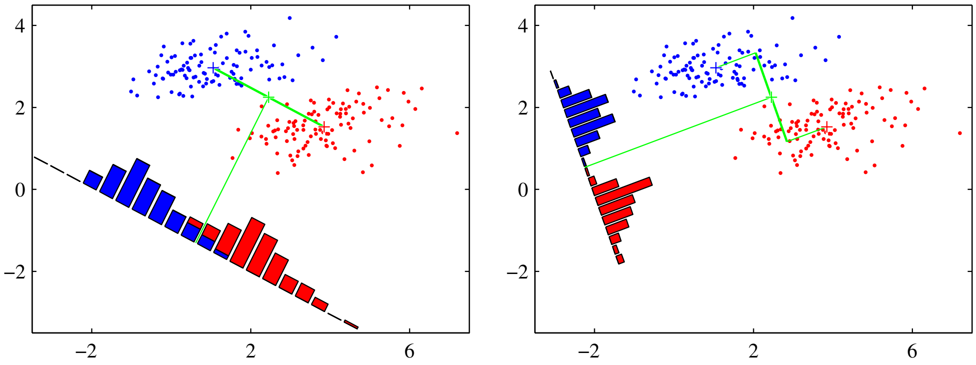
Scaled, rotated by 45°




$$\begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} X$$

$$XX^\top = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

# Linear Discriminant Analysis (LDA)




**Goal:** Find a (normal vector of a linear decision boundary) $\boldsymbol{w}$ that

- Maximizes mean class difference, and
- Minimizes variance in each class

# Linear Discriminant Analysis (LDA)

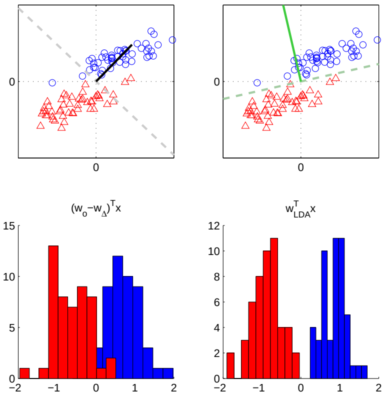
Optimization problem:

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\boldsymbol{w}^\top S_B \boldsymbol{w}}{\boldsymbol{w}^\top S_W \boldsymbol{w}}$$

Setting the gradient to zero we obtained:

$$\boldsymbol{w} \propto S_W^{-1}(\boldsymbol{w}_o - \boldsymbol{w}_\Delta)$$

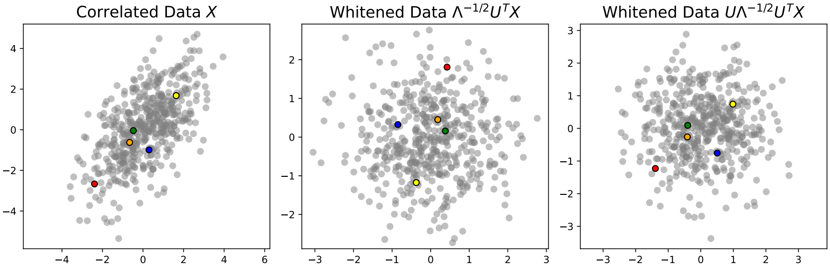
# NCC vs. LDA



# Whitening

Transforms data to data with covariance matrix that is the identity.

$\rightarrow$ Data are decorrelated after whitening



# Generalization and model evaluation

The goal of classification is **generalization**: Correct categorization/prediction of new data

How can we estimate generalization performance?

$\rightarrow$ **Test set**

- Train model on part of data (training set)
- Test model on other part of data (test set)

# From classification to regression

What if our labels are not in $\{-1, +1\}$ but in $\mathbb{R}$?

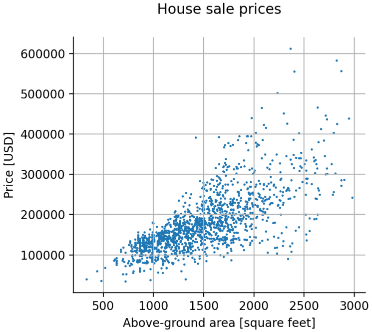
| $y \in \{-1, +1\}$ | $y \in \mathbb{R}$ |
|---|---|
| Classification | **Regression** |

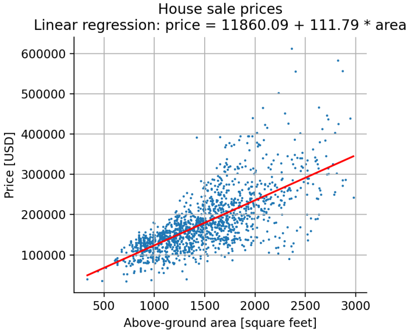
The most basic and best understood type of regression is **linear regression** (or ordinary least squares (OLS)) using a *least-squares cost function*.

# Linear regression - application examples

- Estimate price of a house
- Describe processes in physics/engineering
- Control a hand prosthesis based on electric activity measured on the arm
- Predict sales as a function of advertisement budgets for TV, radio and newspaper.
- Predict stock prices
- . . . many, many more. . .

# Simple linear regression




How to find the regression line?

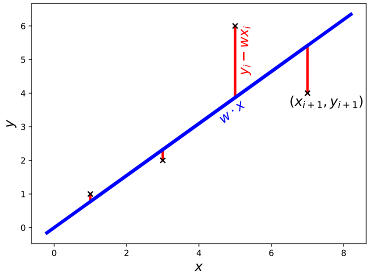
(data from kaggle, a great data science platform
`https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/`)

# Simple linear regression (one-dimensional data)

Given data $x_1, \dots, x_n \in \mathbb{R}$ and labels $y_1, \dots, y_n \in \mathbb{R}$, the goal is to predict new $y$ using an (affine) linear function

$$f(x) = w \cdot x + b$$

We will first focus on a simpler version without intercept $f(x) = w \cdot x$.
Approach: Minimize the **squared error** to find the $w$




$$\mathcal{E}(w) = \sum_{i=1}^{n} (y_i - w \cdot x_i)^2$$

- differentiable
- analytically solvable
- (optimal under normality assumptions)

# Least-squares error: general case

Given data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ with $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ the goal is to find a weight vector $\boldsymbol{w} \in \mathbb{R}^d$ to predict $y$ for a new $\boldsymbol{x}$ via

$$y = \boldsymbol{w}^\top \boldsymbol{x}.$$

Approach: find $\boldsymbol{w}$ by minimizing the **least-squares error** [Gauß, 1809; Legendre, 1805], defined as

$$\mathcal{E}_{LSQ}(\boldsymbol{w}) = \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 \tag{1}$$

C.F. Gauß (1777–1855)

A.M. Legendre (1752–1833)

# Supplementary: optimality of LSE when assuming Gaussian noise

$$y = w \cdot x + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

$$\begin{aligned} p(y|w) &= \mathcal{N}(y|w \cdot x, \sigma_\epsilon^2) \\ &= \frac{1}{\sqrt{2\pi}\sigma_\epsilon} \exp\left\{ -\frac{1}{2} \left( \frac{y - w \cdot x}{\sigma_\epsilon} \right)^2 \right\} \end{aligned}$$

Maximizing $p(y|w)$ as a function of $w$ is equivalent to maximizing the logarithm of $p(y|w)$ (because it is monotonically increasing).

$$\begin{aligned} \operatorname*{argmax}_w p(y|w) &= \operatorname*{argmax}_w \log p(y|w) \\ &= \operatorname*{argmax}_w \left( - \underbrace{(y - w \cdot x)^2}_{\text{Least-squares error}} \right) \end{aligned}$$

For more details, see Chapter 1.2.5 in Bishop [2007].

# Simple linear regression: analytical solution

$$\mathcal{E}(w) = \sum_{i=1}^{n} (y_i - w \cdot x_i)^2$$

Compute the derivative w.r.t. $w$

$$\frac{\partial \mathcal{E}(w)}{\partial w} = \sum_{i=1}^{n} 2(y_i - w \cdot x_i) \cdot (-x_i).$$

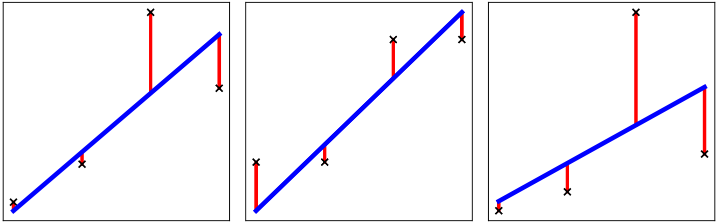
Set to zero and solve for $w$:

$$\sum_{i=1}^{n} 2(y_i - w \cdot x_i) \cdot (-x_i) = 0 \implies -\sum_{i=1}^{n} y_i x_i + w \sum_{i=1}^{n} x_i^2 = 0$$

$$\implies w = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i x_i}$$

# How does OLS behave?

How does the predicted label behave for different samples (marked as $\times$)?



OLS is sensitive to outliers,
because we minimize the squared distance between $y$ and $w \cdot x$.
Therefore, large deviations have a large effect.

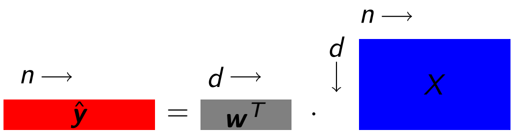
# Linear regression

Let's look at samples with more than one feature and write everything in matrix notation:

Let $n$ be the number of samples, so $\boldsymbol{y} \in \mathbb{R}^{1\times n}$ and $X \in \mathbb{R}^{d\times n}$.

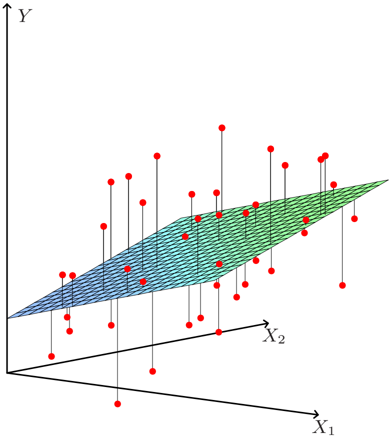
The prediction $\hat{y}$ of our Linear Regression model then becomes

$$\boldsymbol{y} \approx \hat{\boldsymbol{y}} = \boldsymbol{w}^\top X.$$




The goal is still to find $\boldsymbol{w}$ that minimizes the least-squares error.

# Linear regression




$\hat{y} = w_1 \cdot x_1 + w_2 \cdot x_2$

The target variable $\hat{y} \in \mathbb{R}$ is modeled as a **linear combination** $\boldsymbol{w} \in \mathbb{R}^d$ of $d$ features $\boldsymbol{x} \in \mathbb{R}^d$

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

# Linear regression with basis functions

The target variable $\hat{y} \in \mathbb{R}$ can be modeled as a **linear combination** $\boldsymbol{w} \in \mathbb{R}^{\tilde{d}}$ of $\tilde{d}$ features $\phi(\boldsymbol{x}) \in \mathbb{R}^{\tilde{d}}$

$$\hat{y} = \boldsymbol{w}^\top \phi(\boldsymbol{x})$$

where $\phi(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), ...\phi_{\tilde{d}}(\boldsymbol{x}))$ denotes a vector of (possibly non-linear) *basis functions*.

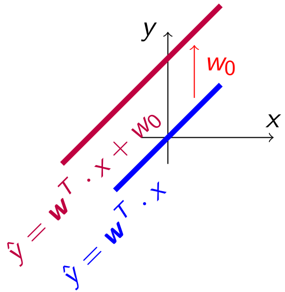
The basis function can also be $\phi(\boldsymbol{x}) = \boldsymbol{x}$.
Generally $\phi(\boldsymbol{x})$ allows us to model more complex functions.

# Intercept term

For non-centered data:
use an *intercept* (often called *bias*) term

$$\hat{y} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}^T \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + w_0$$

$$= \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$




This unifies the notation. Basis function: $\phi([x_1, x_2, \dots, x_d]^T) = [1, x_1, x_2, \dots, x_d]^T$

# Linear Regression: non-linear $\phi(\boldsymbol{x})$

The general model is $\hat{y} = \boldsymbol{w}^\top \phi(\boldsymbol{x})$ with
$\hat{y} \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^{\tilde{d}}$, $\boldsymbol{x} \in \mathbb{R}^d$, $\phi : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$.

Consider the following example:

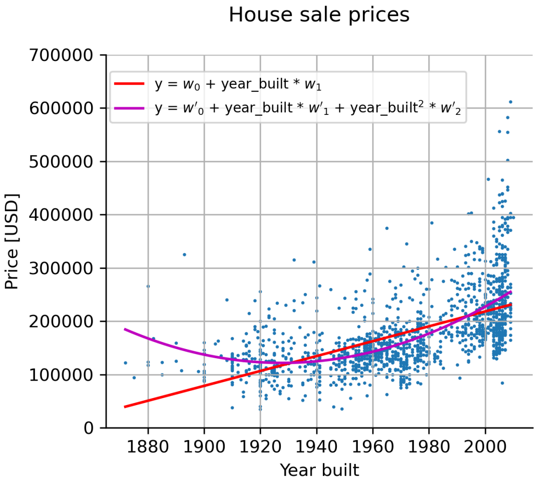
$$\hat{y} = 0.5x^3 - 3x.$$

Here $d = 1$, $\tilde{d} = 2$, $\phi(x) = \begin{bmatrix} x^3 \\ x \end{bmatrix}$ and $\boldsymbol{w} = \begin{bmatrix} 0.5 \\ -3 \end{bmatrix}$.

The feature map $\phi$ is a fixed part of the model. The vector $\boldsymbol{w}$ is found via linear regression (it depends on the data!).

We can use non-linear basis functions to apply linear regression in higher-dimensional space and predict a non-linear function!

# Example with non-linear basis functions



# Linear regression: minimizing LSE

To minimize the least-squares loss function in eq. 1

$$
\begin{aligned}
\mathcal{E}_{LSQ}(\boldsymbol{w}) &= \sum_{i=1}^{n}(y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 \\
&= \|\boldsymbol{y} - \boldsymbol{w}^\top X\|^2 \\
&= \boldsymbol{y}\boldsymbol{y}^\top - 2\boldsymbol{w}^\top X\boldsymbol{y}^\top + \boldsymbol{w}^\top XX^\top \boldsymbol{w}
\end{aligned}
$$

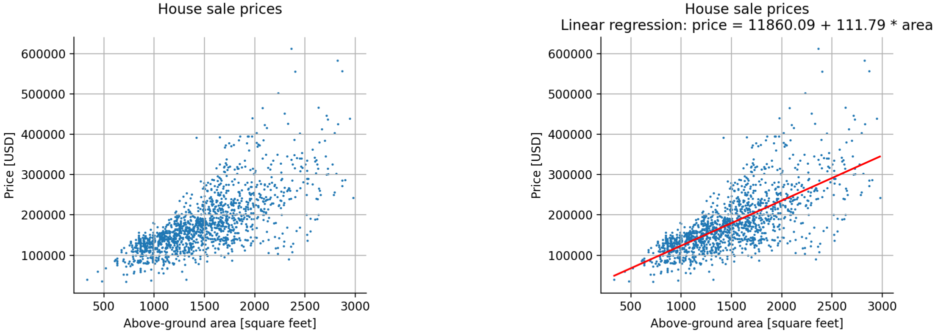
We compute the derivative with respect to $\boldsymbol{w}$

$$
\frac{\partial \mathcal{E}_{LSQ}(\boldsymbol{w})}{\partial \boldsymbol{w}} = -2X\boldsymbol{y}^\top + 2XX^\top \boldsymbol{w}
$$

set it to zero and solve for $\boldsymbol{w}$

$$
\begin{aligned}
-2X\boldsymbol{y}^\top + 2XX^\top \boldsymbol{w} &= 0 \\
XX^\top \boldsymbol{w} &= X\boldsymbol{y}^\top \\
\boldsymbol{w} &= (XX^\top)^{-1}Xy^\top \qquad (2)
\end{aligned}
$$

# Back to initial example




Now you know how to calculate the regression line.

Data from kaggle, a great data science platform
`https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/`

# Does correlation influence the performance of linear regression?

For a new data point $\boldsymbol{z} \in \mathbb{R}^d$ and centered data, we have

$$\begin{aligned} \mathbf{z} &\mapsto \mathbf{w}^\top \mathbf{z} \\ \boldsymbol{w} &= (XX^\top)^{-1} X y^\top \end{aligned}$$

We can decompose:

$$\mathbf{w}^\top \mathbf{z} = yX^\top (XX^\top)^{-1} \boldsymbol{z} = y \underbrace{X^\top U \Lambda^{-1/2}}_{\text{whitened } X^\top} \cdot \underbrace{\Lambda^{-1/2} U^\top \boldsymbol{z}}_{\text{whitened } \boldsymbol{z}}$$

where $XX^\top = U\Lambda U^\top$ is the eigenvalue decomposition of $XX^\top$

$\Rightarrow$ LR is not susceptible to correlation in the features (different from NCC)
(But note: numerical problems can occur, e.g. if $X$ has near-collinear features)

# Linear Regression for vector labels

We now want to predict vector-valued labels $y \in \mathbb{R}^m$

For a measurement $X \in \mathbb{R}^{d \times n}$, $Y \in \mathbb{R}^{m \times n}$ the model is

$$Y = W^\top X$$

where $W^\top \in \mathbb{R}^{m \times d}$ is a **linear mapping** from data to labels.

# Linear Regression for Vector Labels

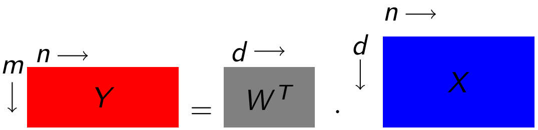
Given Data $X \in \mathbb{R}^{d \times n}$ and labels $Y \in \mathbb{R}^{m \times n}$, the error function for multiple linear regression is

$$\mathcal{E}_{MLR}(W) = ||Y - W^\top X||_F^2 \tag{3}$$

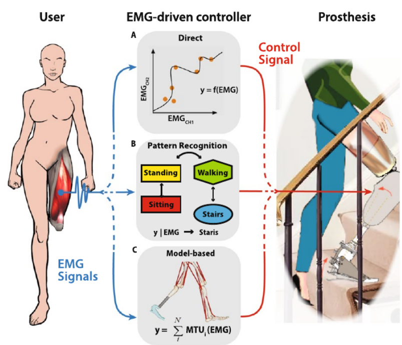
where $||A||_F = \sqrt{\sum_i^n \sum_j^d A_{ij}^2}$ denotes the Frobenius norm and $W^\top \in \mathbb{R}^{m \times d}$

Eq. 3 is minimized by (see also eq. 2)

$$W = (XX^\top)^{-1} XY^\top$$



# Application example: myoelectric control of prostheses

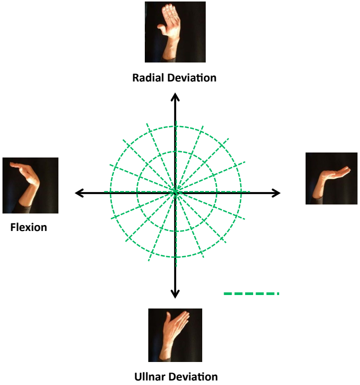



Cimolato et al. [2022]

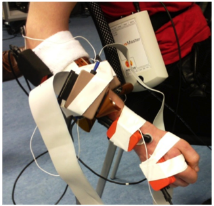
Neurons activate muscles via electric discharges
Electric activity can be measured non-invasively


hand prosthesis
Only 2 degrees of freedom are controlled (open/close, rotate)
Controlled by muscle activity

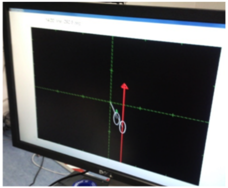
# Acquisition of training data




Experimental Paradigm

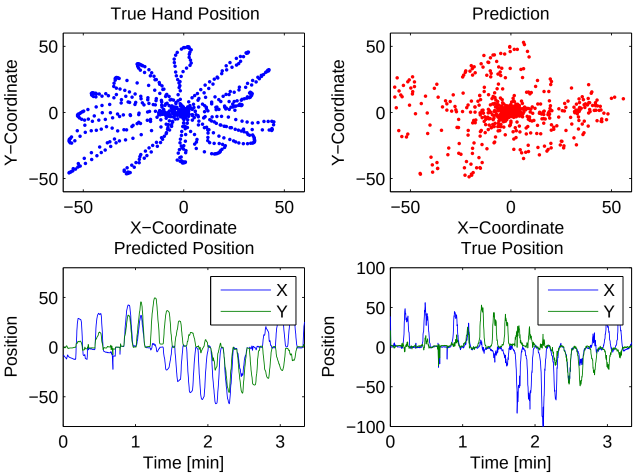


Motion Capture System

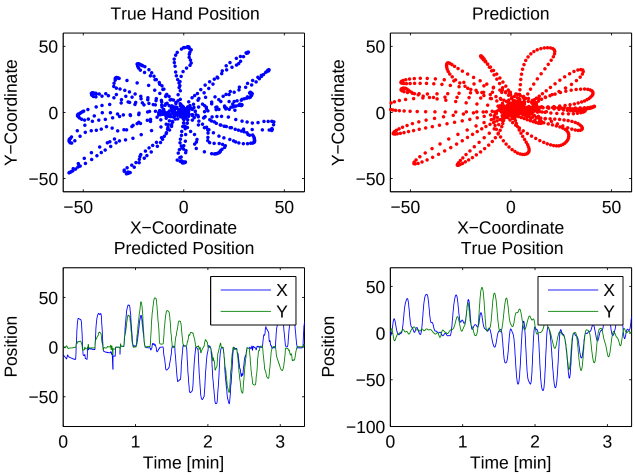


Visual Feedback

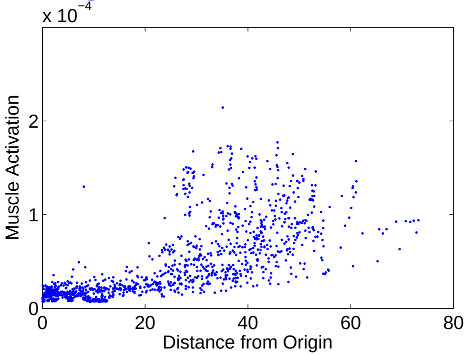
# Results from linear regression



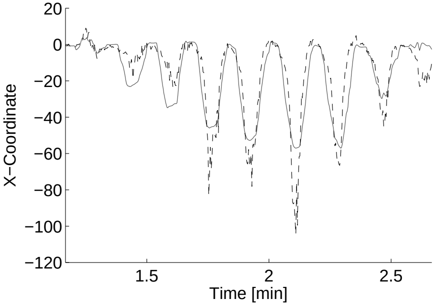
# Results linear regression - smoothed



# Linear regression



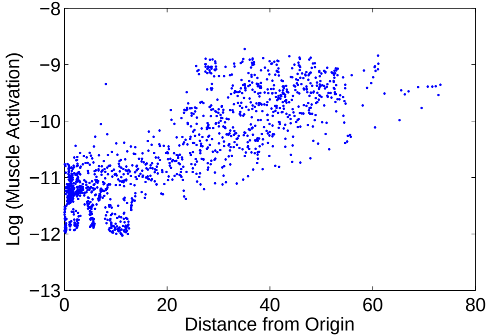

Hand position is a *non-linear* function of muscle activation



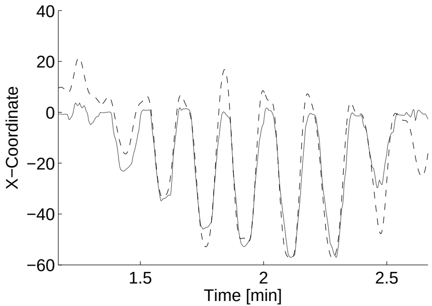

Weak muscle activation
$\rightarrow$ True hand position (gray) **under**estimated (dashed)

Strong muscle activation
$\rightarrow$ True hand position **over**estimated

# Linear(ized) Regression

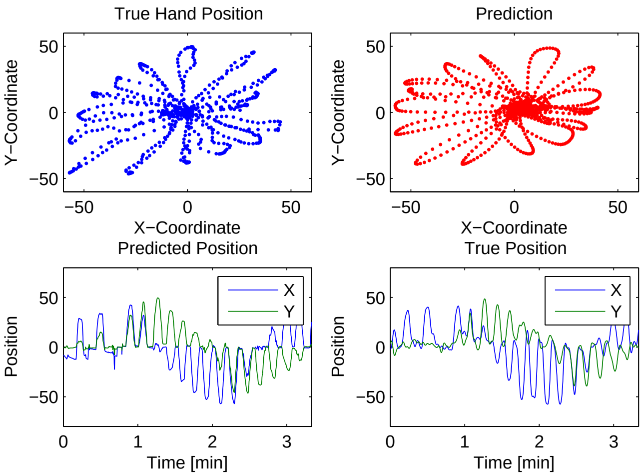



Hand position is *almost linearly* related to log of muscle activation




Strong muscle activation
→ hand position *less* **over**estimated

# Results linear regression - smoothed and log features



# The statistical model of linear regression

Linear Model:

$$y = \boldsymbol{w}^\top \cdot \boldsymbol{x} + \epsilon$$

Linear Regression: estimates

$$\hat{\boldsymbol{w}} = (XX^\top)^{-1}Xy$$

from given data $X, y$.

# Random variable (recap)

A mapping $X : \Omega \to \mathbb{R}$ which assigns a real value to every elementary event, is called a real-valued random variable.

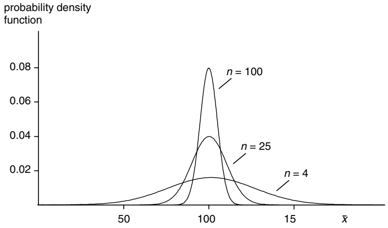
$\Omega$ is the sample space, the set of all possible outcomes.

Example: tossing a coin

$$X(\omega) = \begin{cases} 0, \text{if } & \omega = \text{tail} \\ 1, \text{if } & \omega = \text{head} \end{cases} \quad \text{for } \omega \in \Omega$$

# The sampling distribution of an estimator

**Example: Mean of a random variable**




Consider random variables $x_i \sim \mathcal{N}(\mu, \sigma^2)$ independent, identically distributed (i.i.d.).

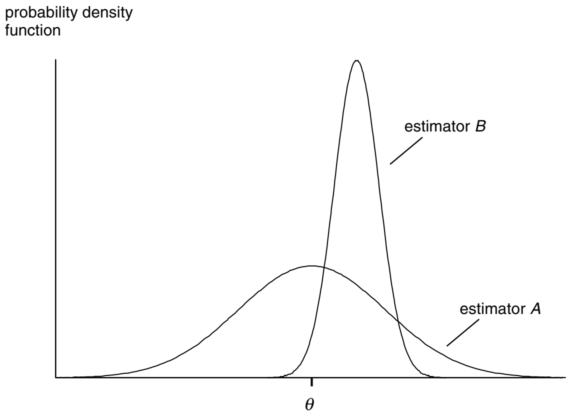
Draw $n$ data points and *estimate* mean on $n$ data points:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$\hat{\mu}$ is a function of the data and thus itself a random variable.

The sampling distribution is the distribution of values that $\hat{\mu}$ takes.

# Desirable properties of estimators

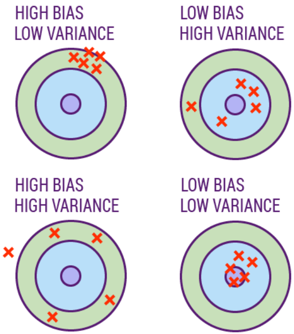



**Unbiased** The estimator's expected value is the true value of the parameter being estimated ($A$ in the Fig.)

**Small estimator variance** ($B$ has a smaller variance than $A$)

**Robust** not unduly affected by outliers or other small deviations from the model assumptions

# Bias and variance




Source: Ikompass [2019]

# Gauss-Markov Theorem

Under the model assumption $y = \boldsymbol{w}^\top \cdot \boldsymbol{x} + \epsilon$ with uncorrelated noise $\epsilon$, our ordinary least squares estimator $\hat{\boldsymbol{w}} = (XX^\top)^{-1}Xy$ is the Best Linear Unbiased Estimator (BLUE), i.e. the minimum variance unbiased estimator that is linear in y.

But: in some cases biased estimators with lower variance might be more suitable.

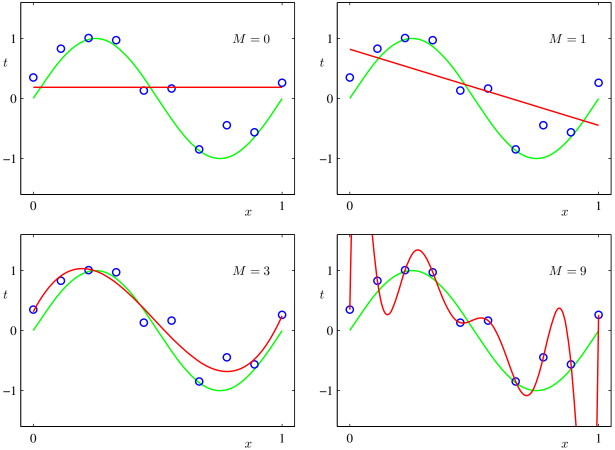
# Example: polynomial regression

Higher degrees of polynomials are not necessarily better.

$$\phi_M(x) = [x^0, x^1, \ldots, x^M]^T$$
$$\hat{y} = \boldsymbol{w}^T \phi_M(x)$$

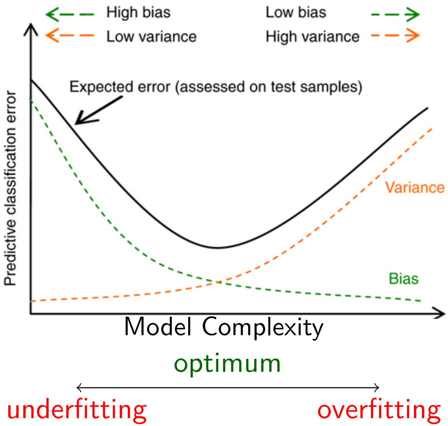
Weights:

| | $M=0$ | $M=1$ | $M=6$ | $M=9$ |
|---|---|---|---|---|
| $w_0^\star$ | 0.19 | 0.82 | 0.31 | 0.35 |
| $w_1^\star$ | | -1.27 | 7.99 | 232.37 |
| $w_2^\star$ | | | -25.43 | -5321.83 |
| $w_3^\star$ | | | 17.37 | 48568.31 |
| $w_4^\star$ | | | | -231639.30 |
| $w_5^\star$ | | | | 640042.26 |
| $w_6^\star$ | | | | -1061800.52 |
| $w_7^\star$ | | | | 1042400.18 |
| $w_8^\star$ | | | | -557682.99 |
| $w_9^\star$ | | | | 125201.43 |




[Bishop, 2007]

# The bias-variance trade-off




## Careful. . .

*Bias* and *variance* are terms with multiple (related) usages

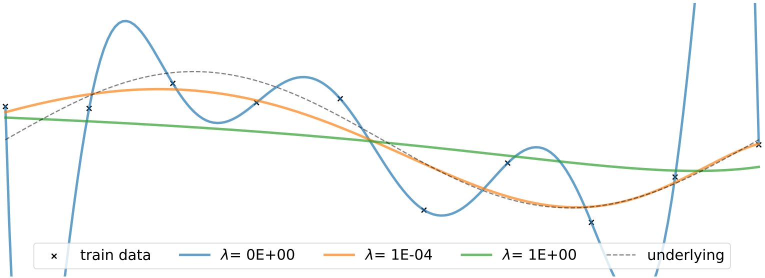
- $\boldsymbol{w}^\top \boldsymbol{x} + b$, $w \cdot x + b$
- Bias/variance of an estimator
- Bias/variance of a general ML model

# Ridge regression

Often it is important to **control the complexity** of the solution $\boldsymbol{w}$.

This is done by constraining the norm of $\boldsymbol{w}$ (regularization)

$$\mathcal{E}_{RR}(\boldsymbol{w}) = ||\boldsymbol{y} - \boldsymbol{w}^\top X||^2 + \lambda||\boldsymbol{w}||^2$$



# Ridge regression

Computing the derivative with respect to $\boldsymbol{w}$ yields

$$\frac{\partial \mathcal{E}_{RR}(\boldsymbol{w})}{\partial \boldsymbol{w}} = -2X\boldsymbol{y}^\top + 2XX^\top \boldsymbol{w} + 2\lambda \boldsymbol{w}.$$

Setting the gradient to zero and rearranging terms, the optimal $\boldsymbol{w}$ is

$$\begin{aligned} 2XX^\top \boldsymbol{w} + 2\lambda \boldsymbol{w} &= 2X\boldsymbol{y}^\top \\ (XX^\top + \lambda I)\boldsymbol{w} &= X\boldsymbol{y}^\top \\ \boldsymbol{w} &= (XX^\top + \lambda I)^{-1} X\boldsymbol{y}^\top \end{aligned}$$

One can show (calculate) that for $\lambda \neq 0$, this estimator is biased and has a smaller variance than the OLS estimator

[Hoerl and Kennar, 1970; Tychonoff, 1943]

# (Multi-)linear (ridge) regression algorithm

**Computes:** Weight matrix $W$ for linear mapping of $\mathbb{R}^{d+1} \to \mathbb{R}^m$

**Input:** Data $\{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}^m$, ridge $\lambda$

Include offset parameters (row vector of $n$ ones)

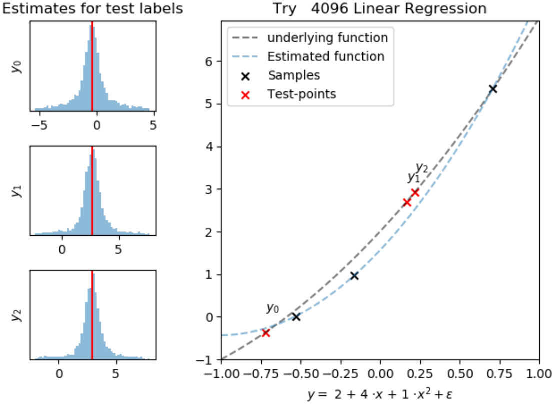
$$X = \begin{bmatrix} \mathbb{1} \\ X \end{bmatrix}$$

$$W = (XX^\top + \lambda I)^{-1} XY^\top$$

**Output:** $W$

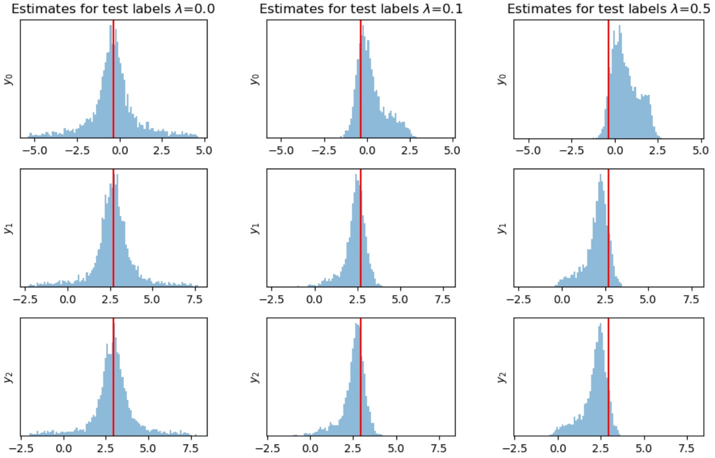
# Bias and variance of $y_{test}$

How do bias and variance of predicted labels behave
→ Let's simulate
For LR it looks unbiased (on average correct), but high variance
Let's look at the effect of regularization



# Effect of $\lambda$ on bias and variance

- For RR we see that estimates are biased, but less variance
- How can we choose optimal $\lambda$?

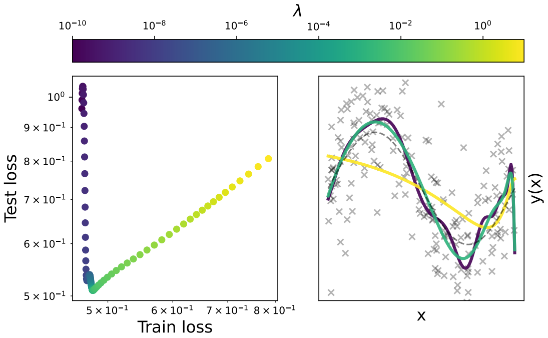


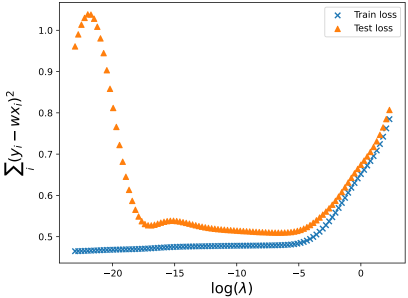
# Model selection

How can we find the best $\lambda$?

One option: grid search

$\rightarrow$ try out e.g. $\lambda \in \{0, 0.1, ..., 0.9, 1.0\}$ and choose the one with the lowest error on test set



# What if we have a small data-set?

## Standard approach

Split the data into train and test

$$[\underbrace{x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}}_{\text{train}}, \underbrace{x_{i_5}, x_{i_6}}_{\text{test}}]$$

Then

- **Train** your model on the training data
- **Test** your model on the test data

We are not using the full data-set

- Test set could be sampled badly

## Solution

$k$-fold Cross-Validation:

- fold 1 $[\underbrace{x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}}_{\mathcal{F}_1^{\text{train}}}, \underbrace{x_{i_5}, x_{i_6}}_{\mathcal{F}_1^{\text{test}}}]$
- fold 2 $[\underbrace{x_{i_1}, x_{i_2}}_{\mathcal{F}_2^{\text{test}}}, \underbrace{x_{i_3}, x_{i_4}, x_{i_5}, x_{i_6}}_{\mathcal{F}_2^{\text{train}}}]$
- fold 3 ...

For each fold:

- **Train** your model on the training data
- **Test** your model on the test data

# Cross-validation

Split data set in $k$ different random **training** and **test** data

fold 1 $[\underbrace{x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}}_{\mathcal{F}_1^{\text{train}}}, \underbrace{x_{i_5}, x_{i_6}}_{\mathcal{F}_1^{\text{test}}}]$

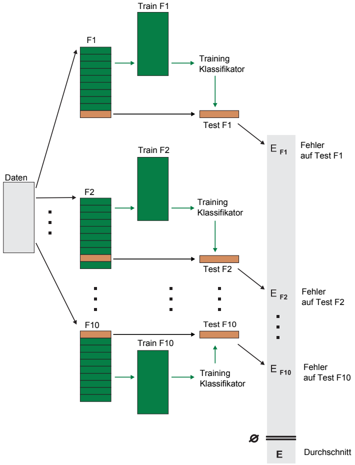
fold 2 $[\underbrace{x_{i_1}, x_{i_2}}_{\mathcal{F}_2^{\text{test}}}, \underbrace{x_{i_3}, x_{i_4}, x_{i_5}, x_{i_6}}_{\mathcal{F}_2^{\text{train}}}]$

fold 3 ...

For each fold:

**Train** your model on the training data

**Test** your model on the test data

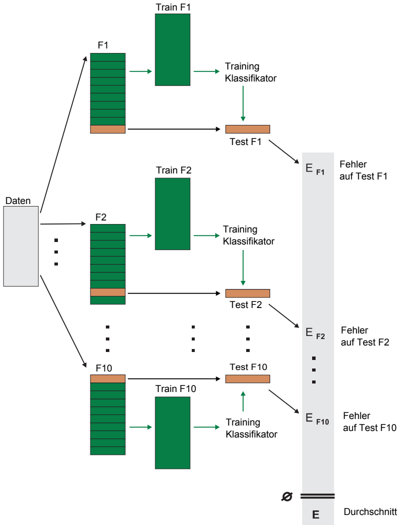


# Cross-validation

**Algorithm 1:** Cross-Validation

**Require:** Data $(x_1, y_1) \dots, (x_N, y_N)$, Number of CV folds $F$

1: # Split data in $F$ **disjunct** folds
2: **for** folds $f = 1, \dots, F$ **do**
3:     # Train model on folds $\{1, \dots, F\} \setminus f$
4:     # Compute prediction error on fold $f$
5: **end for**
6: # Average prediction error



# Cross-validation: Can be used differently

**Model Evaluation**
*"How well does my model perform?"*
Report **mean evaluation score** – e.g. accuracy – across folds

**Model Selection**
*"What hyperparameter should I use?"*
Do grid search on every fold.
Take parameter with the highest mean test score across folds

- You can't do both at the same time with simple cross-validation!
  - If we did both on the same test fold:
    - We would be too optimistic because we use same test set for optimizing and evaluating
- After CV you still need to train your model on the whole data-set

# Comparison of Supervised Algorithms

| Algorithm | Solution | Assumption |
|---|---|---|
| NCC<br>LDA | $w = Xy^T$<br>$w = S^{-1}Xy^T$ | $y_t \in \left\{ \frac{1}{n_{+1}}, -\frac{1}{n_{-1}} \right\}$<br>NCC: Isotropic Normal distribution<br>LDA: Equal within-class covariances,<br>Multivariate Normal distribution |
| Linear Regression | $w = (XX^\top)^{-1}Xy^\top$ | $y_i \in \mathbb{R}$<br>Gaussian Noise |

# Summary

Linear Regression

- is a generic framework for prediction
- straightforwardly extends to vector labels
- can model nonlinear dependencies between data and labels
- can be made more robust (Ridge Regression)

Cross-Validation

- Data-efficient method for model selection & model evaluation
- Only use if your bottleneck is data

# References


C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2007.

A. Cimolato, J. J. M. Driessen, L. S. Mattos, E. De Momi, M. Laffranchi, and L. De Michieli. EMG-driven control in lower limb prostheses: a topic-based systematic review. *Journal of NeuroEngineering and Rehabilitation*, 19 (1), 2022. ISSN 1743-0003. doi: 10.1186/s12984-022-01019-1. URL https://doi.org/10.1186/s12984-022-01019-1.

C. F. Gauß. Theoria motus corporum coelestium in sectionibus conicis solem ambientium. *Göttingen*, 1809.

A. E. Hoerl and R. W. Kennar. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

Ikompass. Bias variance tradeoff, 2019. URL http://www.ikompass.edu.sg/trainings/data_science_ccc-big-data-foundation-2/darts/.

A.-M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*, chapter Sur la methode des moindres quarres. Firmin Didot, http://imgbase-scd-ulp.u-strasbg.fr/displayimage.php?pos=-141297, 1805.

A. N. Tychonoff. On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39(5):195–198, 1943.