

## Cognitive Algorithms Lecture 3

# Linear Regression

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

Technische Universität Berlin  
Machine Learning Group

## Cognitive Algorithms Lecture 3

# Linear Regression

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

Technische Universität Berlin  
Machine Learning Group

## Cognitive Algorithms Lecture 3

# Linear Regression

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

Technische Universität Berlin  
Machine Learning Group

## 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

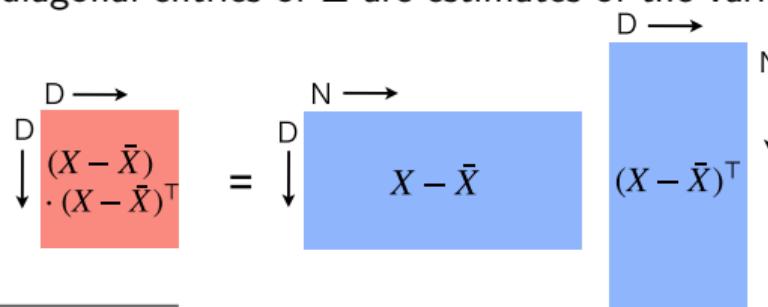
Given  $n$  data points  $\mathbf{x}_i \in \mathbb{R}^d$  in a data matrix  $X \in \mathbb{R}^{d \times n}$   
 the empirical estimate of the **covariance matrix** is defined as

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

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

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

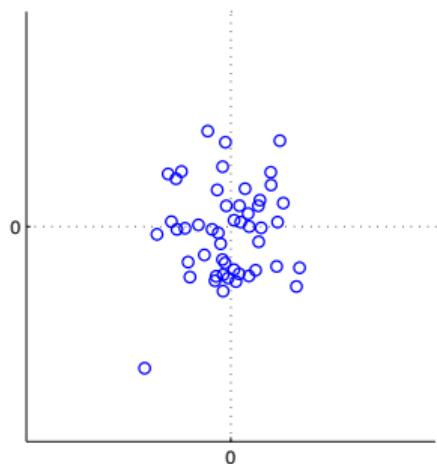
The diagonal entries of  $\hat{\Sigma}$  are estimates of the variance.



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

## Correlated data and linear mappings

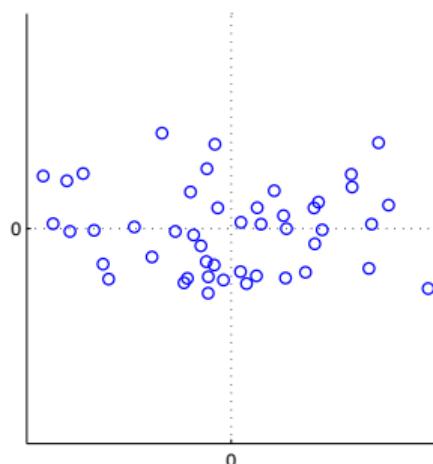
### Uncorrelated



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

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

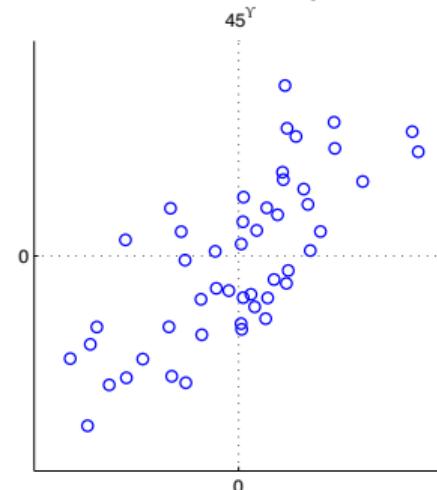
Uncorrelated, scaled



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

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

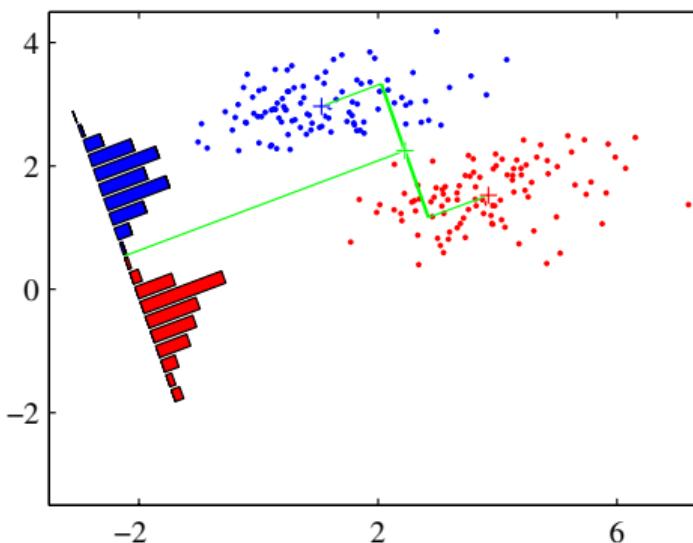
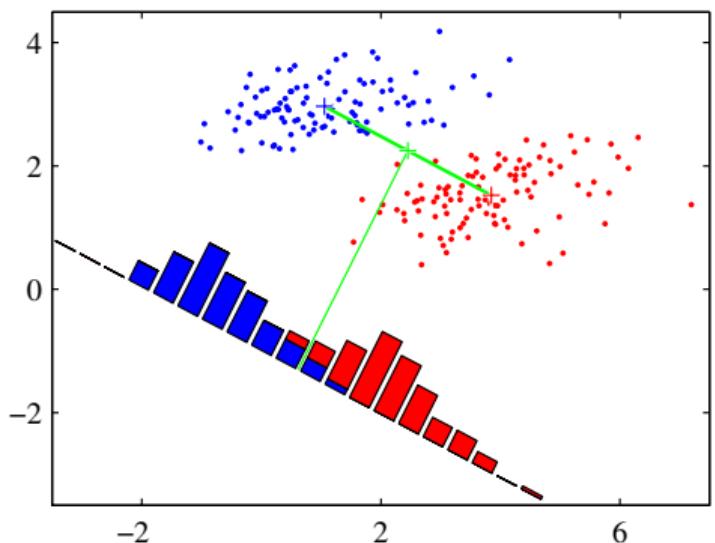
Scaled, rotated by 45°



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

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

## Linear Discriminant Analysis (LDA)



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

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

# Linear Discriminant Analysis (LDA)

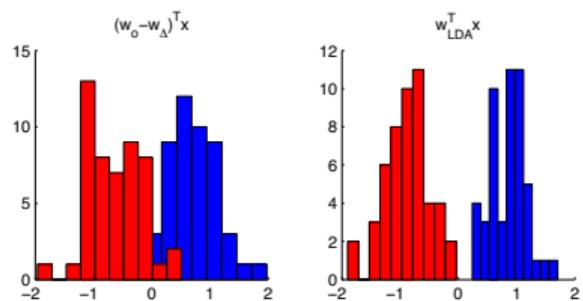
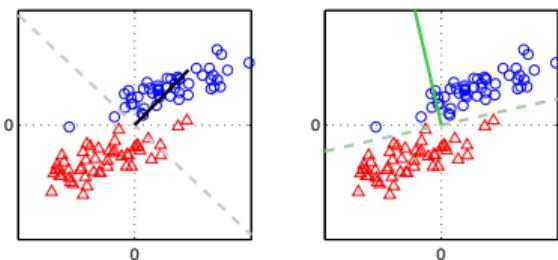
Optimization problem:

$$\operatorname{argmax}_{\mathbf{w}} \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}$$

Setting the gradient to zero we obtained:

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

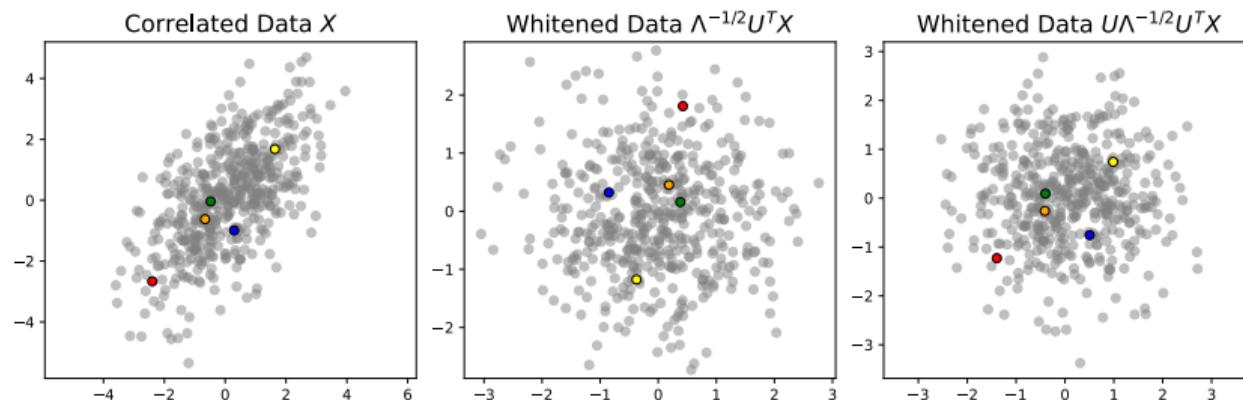
# NCC vs. LDA



# Whitening

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

→ 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?

→ **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  $\mathbb{R}$ ?

$$\begin{array}{c|c} y \in \{-1, +1\} & y \in \mathbb{R} \\ \text{Classification} & \text{Regression} \end{array}$$

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

最基本且最为理解的回归类型是线性回归（或普通最小二乘法 (OLS)），它使用最小二乘成本函数。

## 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/>)

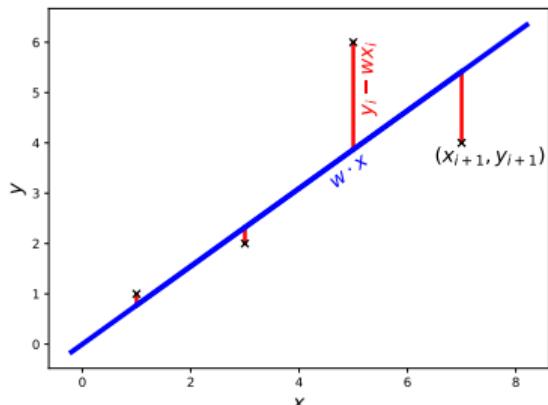
## Simple linear regression

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

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

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

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



C.F. Gauß (1777–1855)



A.M. Legendre (1752–1833)

## Supplementary: optimality of LSE when assuming Gaussian noise

附加信息：在假设高斯噪声时，最小二乘估计（LSE）的最优性

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

$$p(y|w) = \mathcal{N}(y|w \cdot x, \sigma_\epsilon)$$

$$= \frac{1}{\sqrt{2\pi}\sigma_\epsilon} \exp \left\{ -\frac{1}{2} \left( \frac{y - w \cdot x}{\sigma_\epsilon} \right)^2 \right\}$$

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

$$\operatorname{argmax}_w p(y|w) = \operatorname{argmax}_w \log p(y|w)$$

$$= \operatorname{argmax}_w \left( - \underbrace{(y - w \cdot x)^2}_{\text{Least-squares error}} \right)$$

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.  $w$

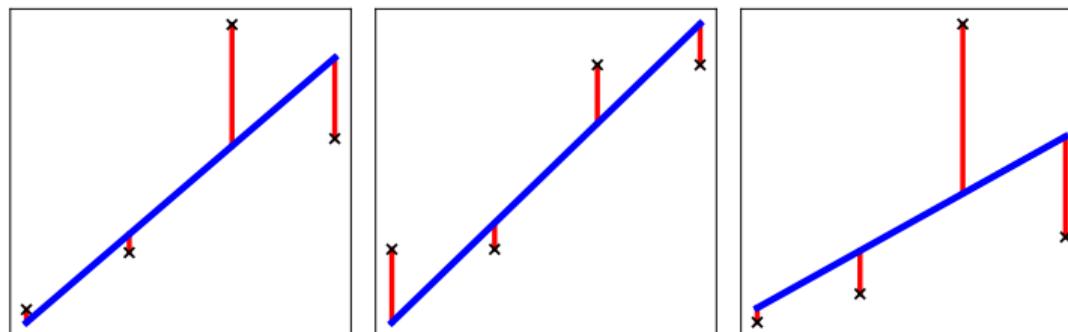
$$\frac{\partial \mathcal{E}(w)}{\partial w} = \sum_{i=1}^n 2(y_i - w \cdot x_i) \cdot (-x_i).$$

Set to zero and solve for  $w$ :

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

## 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.

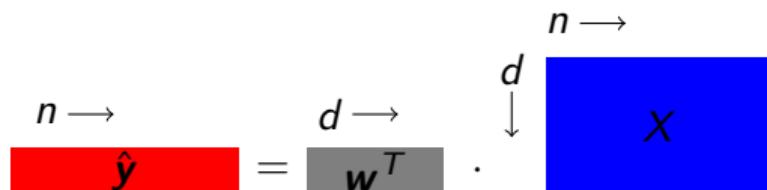
## Linear regression

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

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

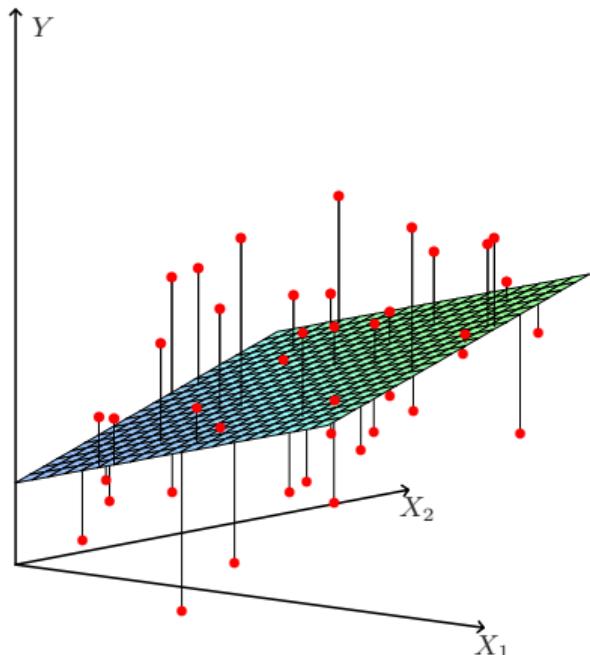
The prediction  $\hat{\mathbf{y}}$  of our Linear Regression model then becomes

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



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

## Linear regression

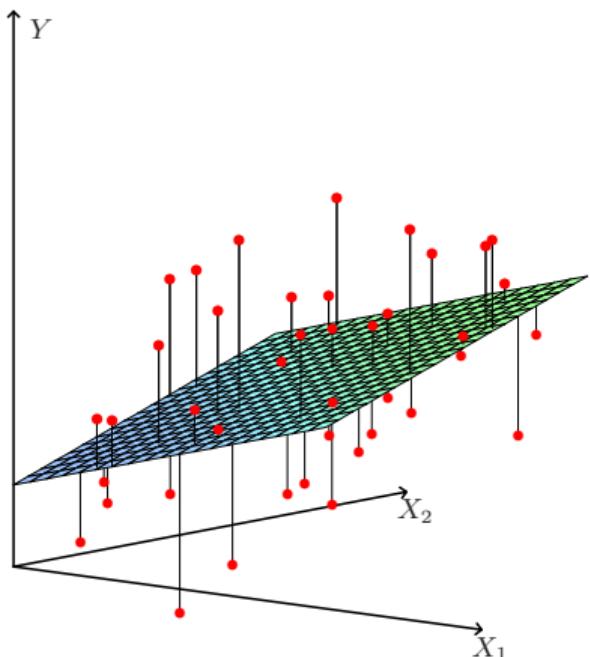


The target variable  $\hat{y} \in \mathbb{R}$  is modeled as a **linear combination**  $\mathbf{w} \in \mathbb{R}^d$  of  $d$  features  $\mathbf{x} \in \mathbb{R}^d$

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

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

## Linear regression with basis functions



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

2.8

$$\phi(\mathbf{x}) = (\mathbf{x}, \mathbf{x}^2, \mathbf{x}^3)$$

$$\mathbf{w}^\top \phi(\mathbf{x}) = w_0 \mathbf{x} + w_1 \mathbf{x}^2 + w_2 \mathbf{x}^3$$

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

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

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

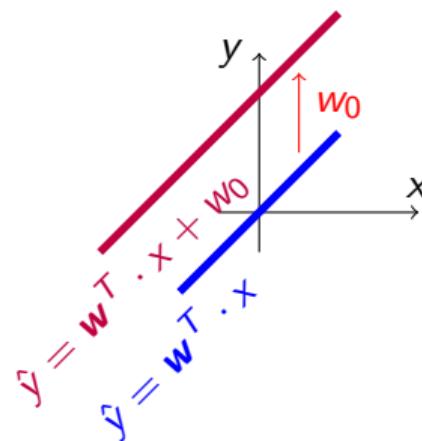
The basis function can also be  $\phi(\mathbf{x}) = \mathbf{x}$ . Generally  $\phi(\mathbf{x})$  allows us to model more complex functions.

## Intercept term

For non-centered data:  
use *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:  $\phi([x_1, x_2, \dots, x_d]^T) = [1, x_1, x_2, \dots, x_d]^T$

## 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/>

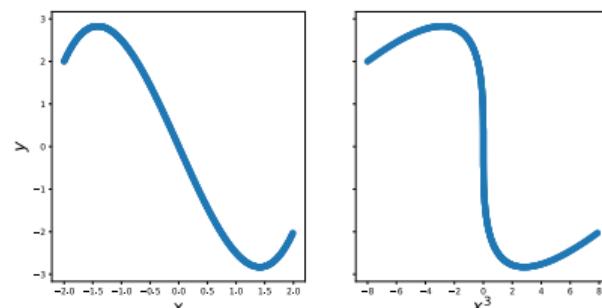
## Linear Regression: non-linear $\phi(x)$

Polynomials as an example for  $\phi(x)$ :

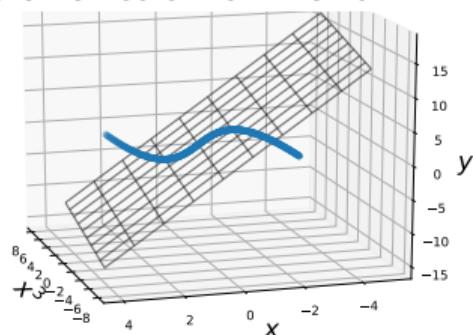
$$\hat{y} = 0.5x^3 - 3x$$

Here  $\phi(x) = \begin{bmatrix} x^3 \\ x \end{bmatrix}$  and  $w = \begin{bmatrix} 0.5 \\ -3 \end{bmatrix}$

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

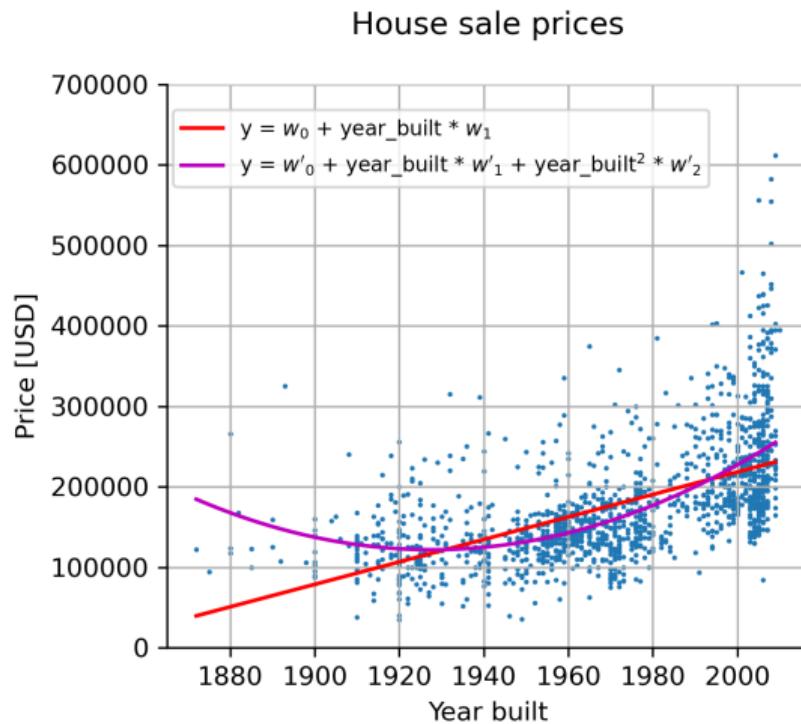


$\hat{y}$  as a function of  $x$  and  $x^3$



$\hat{y}$  lies on a plane in  $x, x^3$  space

## Example with non-linear basis functions



## Linear regression: minimizing LSE

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

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

We compute derivative w.r.t.  $\mathbf{w}$

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

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

$$-2\mathbf{X}\mathbf{y}^\top + 2\mathbf{X}\mathbf{X}^\top \mathbf{w} = 0$$

$$\mathbf{X}\mathbf{X}^\top \mathbf{w} = \mathbf{X}\mathbf{y}^\top$$

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1} \mathbf{X}\mathbf{y}^\top$$

(2)

## Linear Regression

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

$$\begin{aligned}\mathbf{z} &\mapsto \mathbf{w}^T \cdot \mathbf{z} \\ \mathbf{w} &= \left( \frac{1}{n} \mathbf{X} \mathbf{X}^T \right)^{-1} \frac{1}{n} \mathbf{X} \mathbf{y}^T = \frac{1}{n} \mathbf{X} \mathbf{y}^T\end{aligned}$$

Suppose

$\mathbf{X}$  and  $\mathbf{y}$  centered  $\rightarrow$  mean in every dimension is 0

$$\frac{1}{n} \mathbf{X} \mathbf{X}^T = I$$

Then

$$\begin{aligned}\mathbf{w}^T \mathbf{z} &= \frac{1}{n} \mathbf{y} \mathbf{X}^T \mathbf{z} \\ &= [\text{Cov}(\mathbf{x}^1, \mathbf{y}), \dots, \text{Cov}(\mathbf{x}^d, \mathbf{y})] \cdot \mathbf{z} \\ &= \text{Cov}(\mathbf{x}^1, \mathbf{y}) \cdot z^1 + \dots + \text{Cov}(\mathbf{x}^d, \mathbf{y}) \cdot z^d\end{aligned}$$

Here the features do not influence each others coefficients

## Does correlation influence the performance of linear regression?

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

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

We can decompose:

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

where  $\mathbf{X}\mathbf{X}^\top = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$  is the eigenvalue decomposition of  $\mathbf{X}\mathbf{X}^\top$

⇒ LR is not susceptible to correlation in the features (different from NCC!)

## 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

$$\mathcal{E}_{MLR}(W) = \|Y - W^\top X\|_F^2 = \sum_j^m (\vec{y}_j - \vec{w}_j^\top x)^2 \quad (3)$$

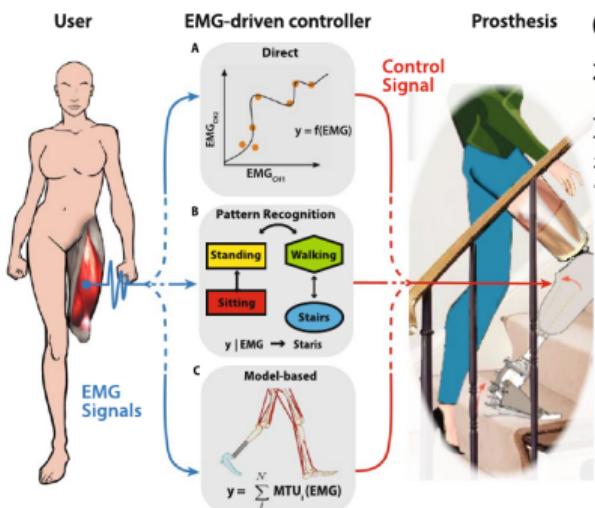
where  $\|A\|_F = \sqrt{\sum_i^n \sum_j^d A_{ij}^2}$  denotes the Frobenius norm and  $W^\top \in \mathbb{R}^{m \times d}$

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

$$W = (X X^\top)^{-1} X Y^\top \quad \begin{matrix} \text{all different dimensions of labels} \\ \text{(not the data)} \end{matrix}$$

$$m \begin{matrix} n \rightarrow \\ \downarrow \end{matrix} Y = \begin{matrix} d \rightarrow \\ \downarrow \end{matrix} W^\top \cdot \begin{matrix} d \rightarrow \\ \downarrow \end{matrix} X$$

# Application example: myoelectric control of prostheses



Cimolato et al. [2022]

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

应用示例：肌电控制假肢

Cimolato等人 [2022]

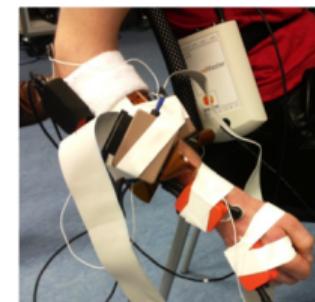
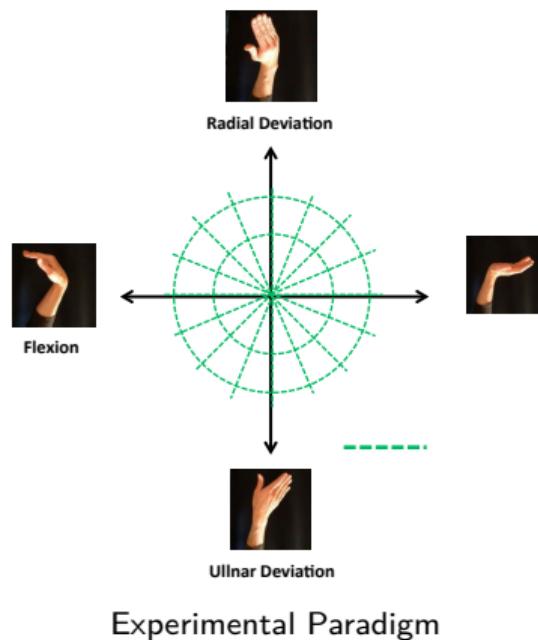
神经通过电放电激活肌肉，电活动可以进行非侵入性测量，手部假肢只控制2个自由度（开合和旋转），由肌肉活动来进行控制。



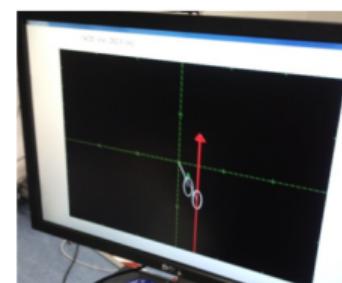
hand prosthesis

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

## Acquisition of training data

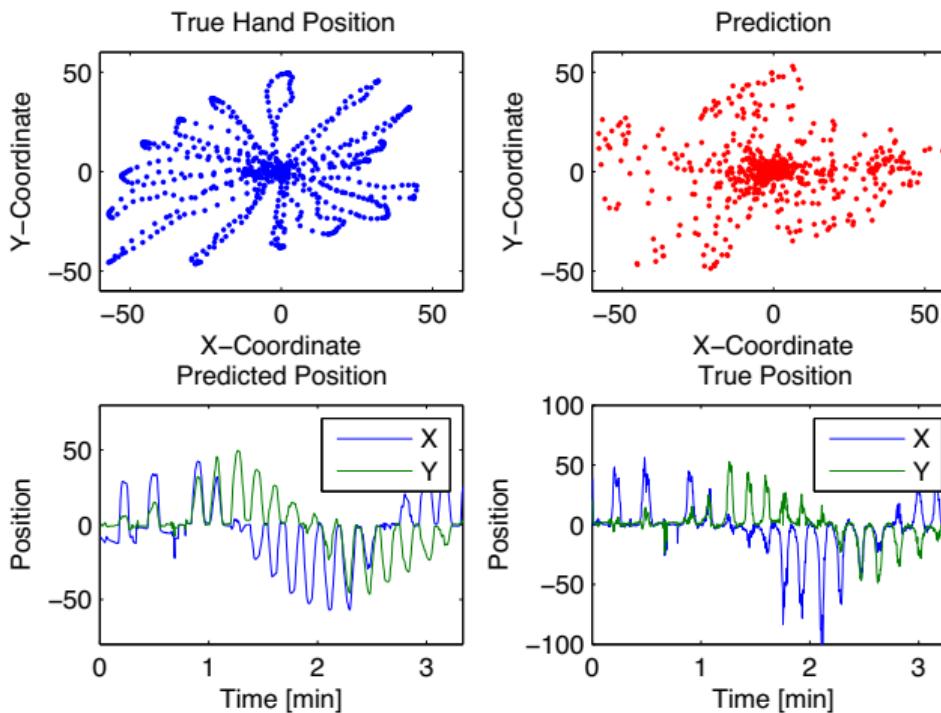


Motion Capture System



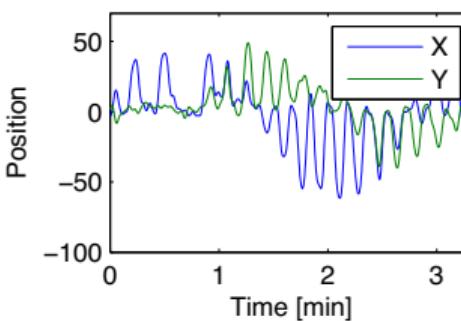
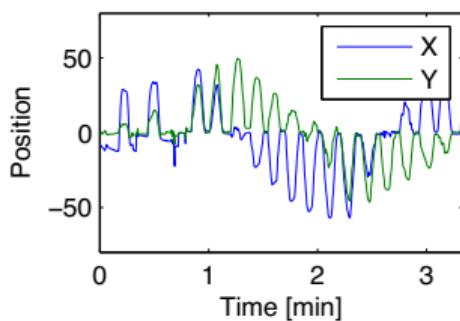
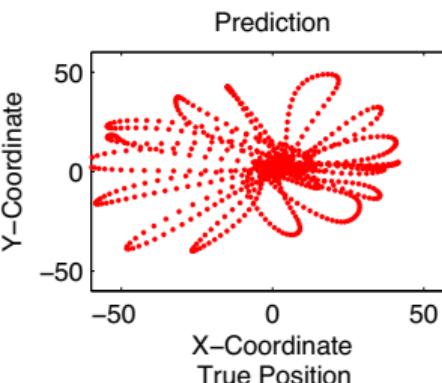
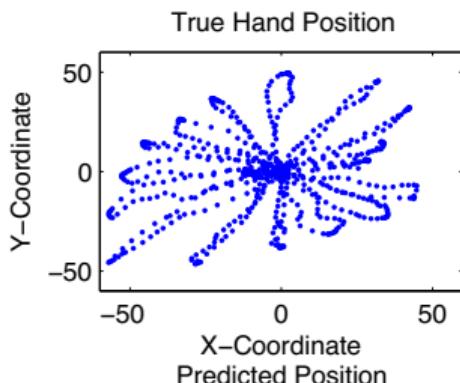
Visual Feedback

## Results from linear regression

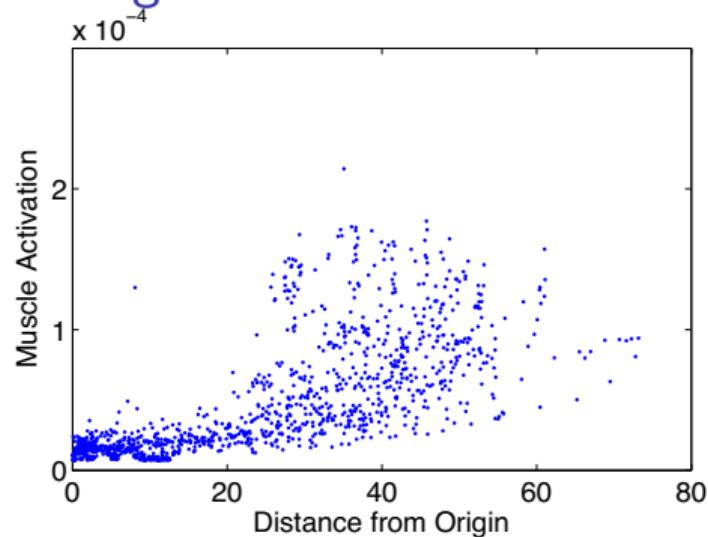


## Results linear regression - smoothed

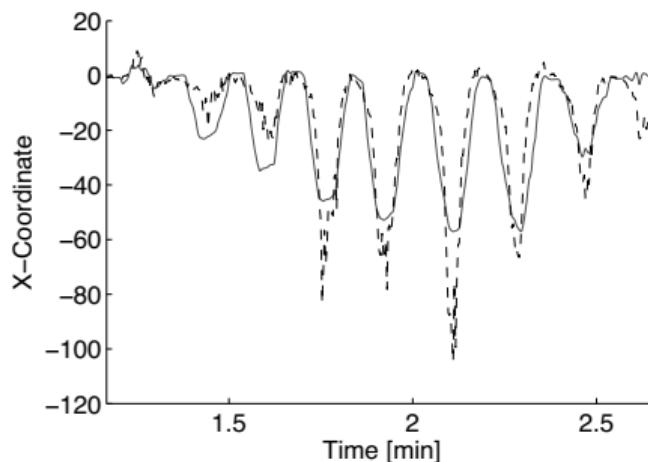
We smooth the predicted labels



## Linear regression



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



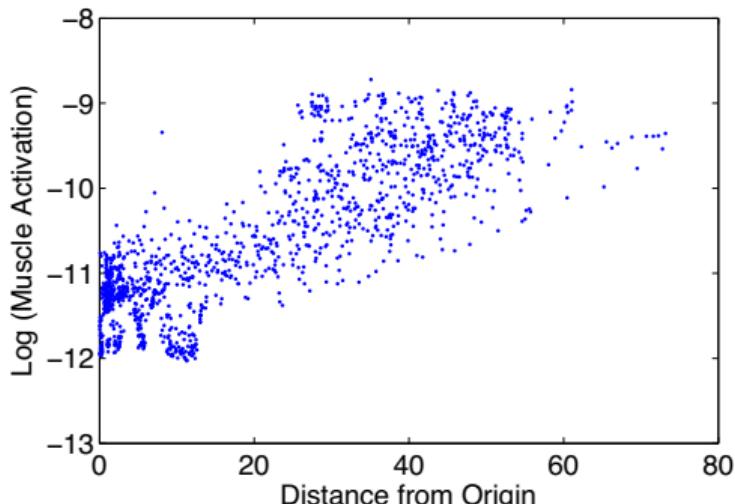
Weak muscle activation

→ True hand position (gray) **underestimated** (dashed)

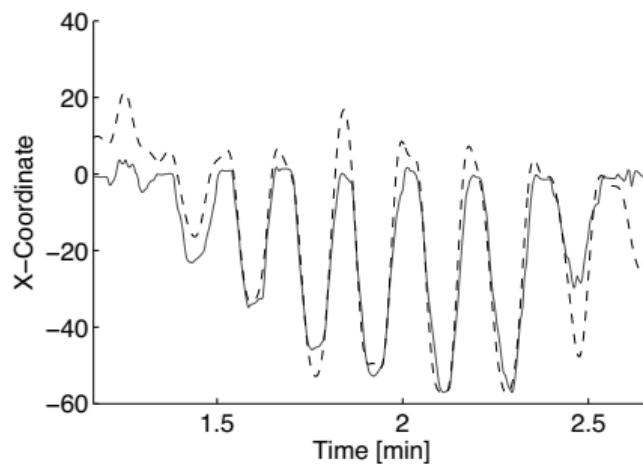
Strong muscle activation

→ True hand position **overestimated**

## Linear(ized) Regression

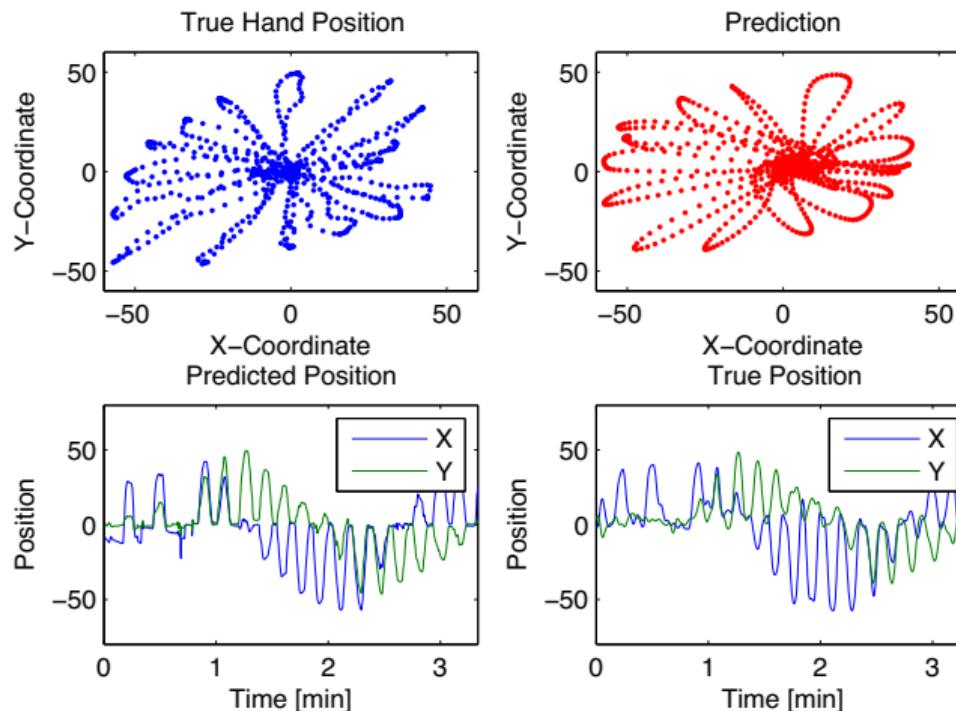


Hand position is *almost linearly* related to log of muscle activation



Strong muscle activation  
→ hand position *less overestimated*

## Results linear regression - smoothed and log features



# The statistical model of linear regression

Linear Model:

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

Linear Regression: estimates

$$\hat{\mathbf{w}} = (X X^\top)^{-1} X y$$

from given data  $X, y$ .

## Random variable (recap)

A mapping  $X : \Omega \rightarrow \mathbb{R}$  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

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

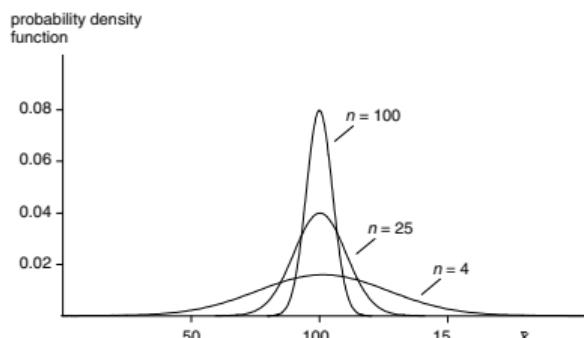
# The sampling distribution of an estimator

## 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:

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

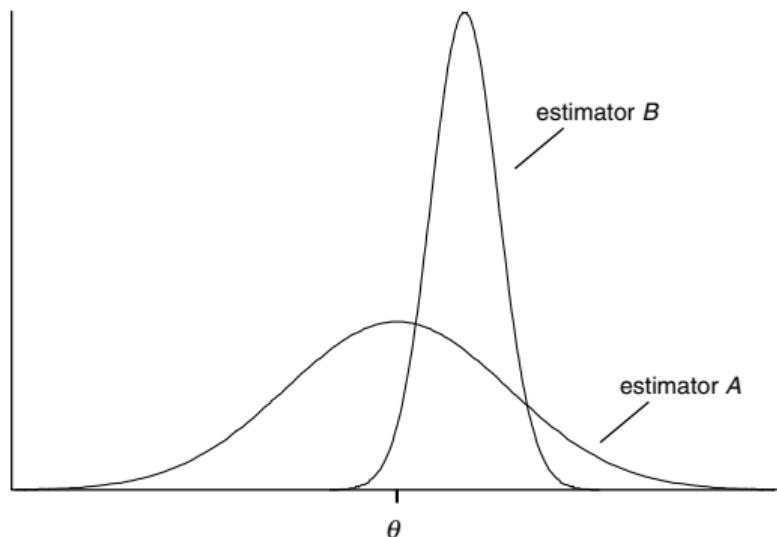


$\hat{\mu}$  is a function of the data and thus itself a random variable.

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

# Desirable properties of estimators

probability density  
function



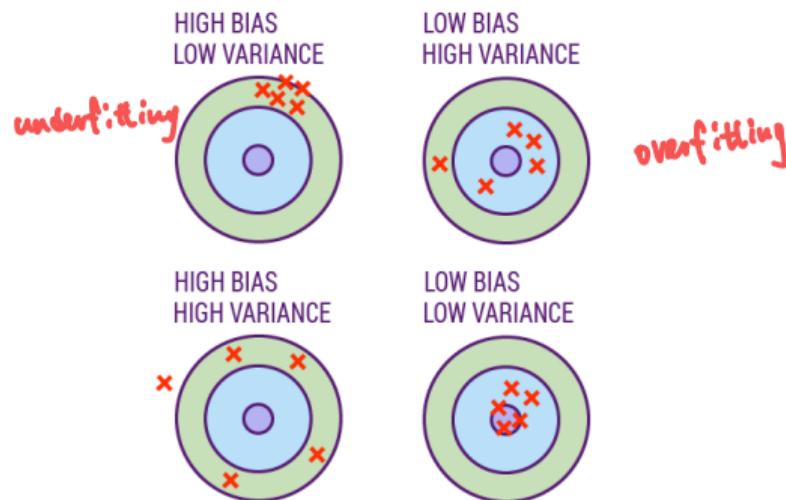
无偏 估计器的期望值是被估计参数的真实值 (图中的 A)。  
小估计器方差  
(B 的方差小于 A)  
稳健，不受异常值或其他与模型假设的微小偏差的不当影响

**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  $y = \mathbf{w}^\top \cdot \mathbf{x} + \epsilon$  with uncorrelated noise  $\epsilon$ , our ordinary least squares estimator  $\hat{\mathbf{w}} = (X X^\top)^{-1} X y$  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.

高斯-马尔可夫定理

在模型假设  $y = \mathbf{w}^\top \cdot \mathbf{x} + \epsilon$  下，其中  $\epsilon$  是不相关的噪声，我们的普通最小二乘估计器  $\hat{\mathbf{w}} = (X X^\top)^{-1} X y$  是最佳线性无偏估计 (BLUE)，即在  $y$  中是线性的、方差最小的无偏估计器。

但是：在某些情况下，具有较低方差的有偏估计器可能更合适。

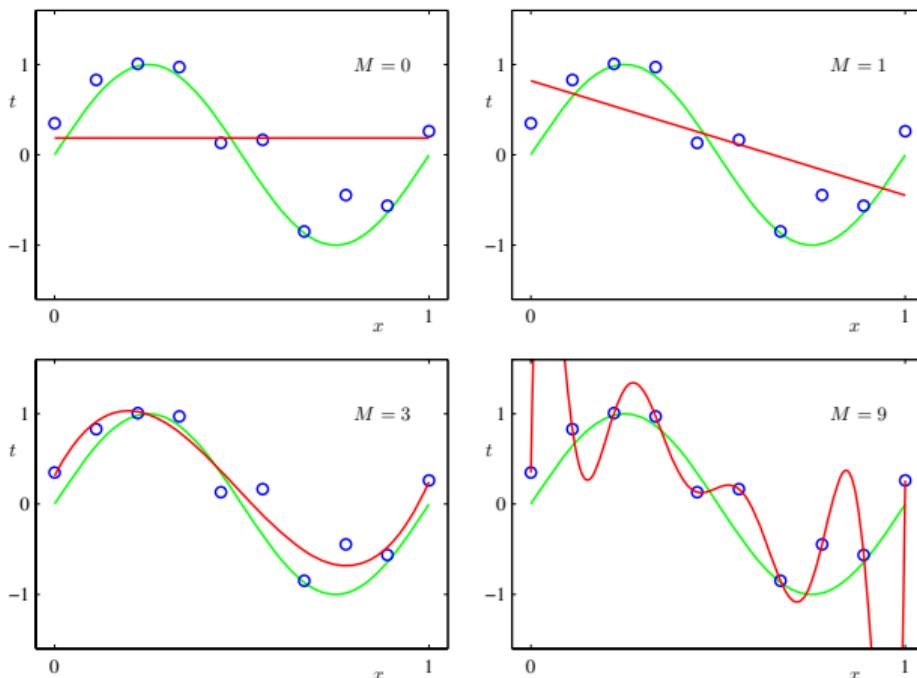
## Example: polynomial regression

$$\phi_M(x) = [x^0, x^1, \dots, x^M]^T$$

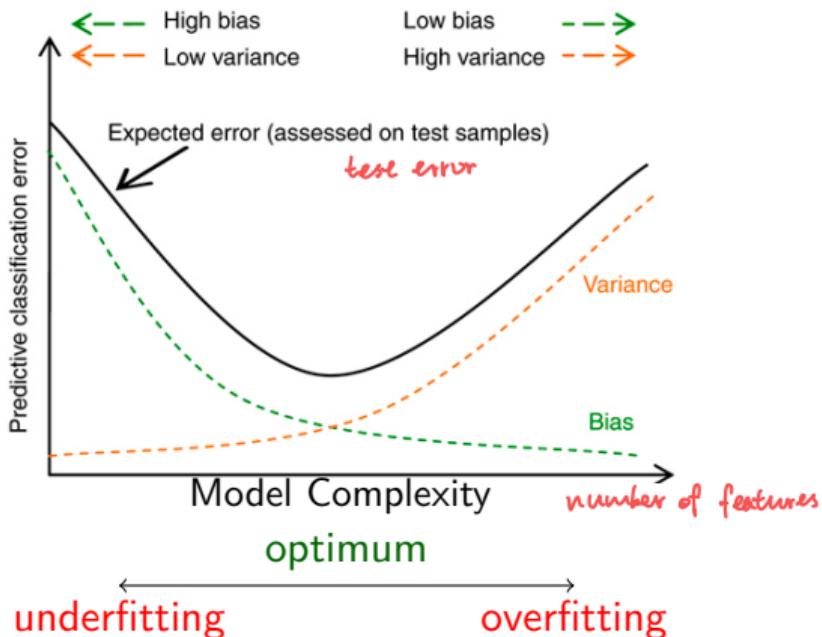
$$\hat{y} = \mathbf{w}^T \phi_M(x)$$

Weights:

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



# The bias-variance trade-off



Careful...

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

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

岭回归

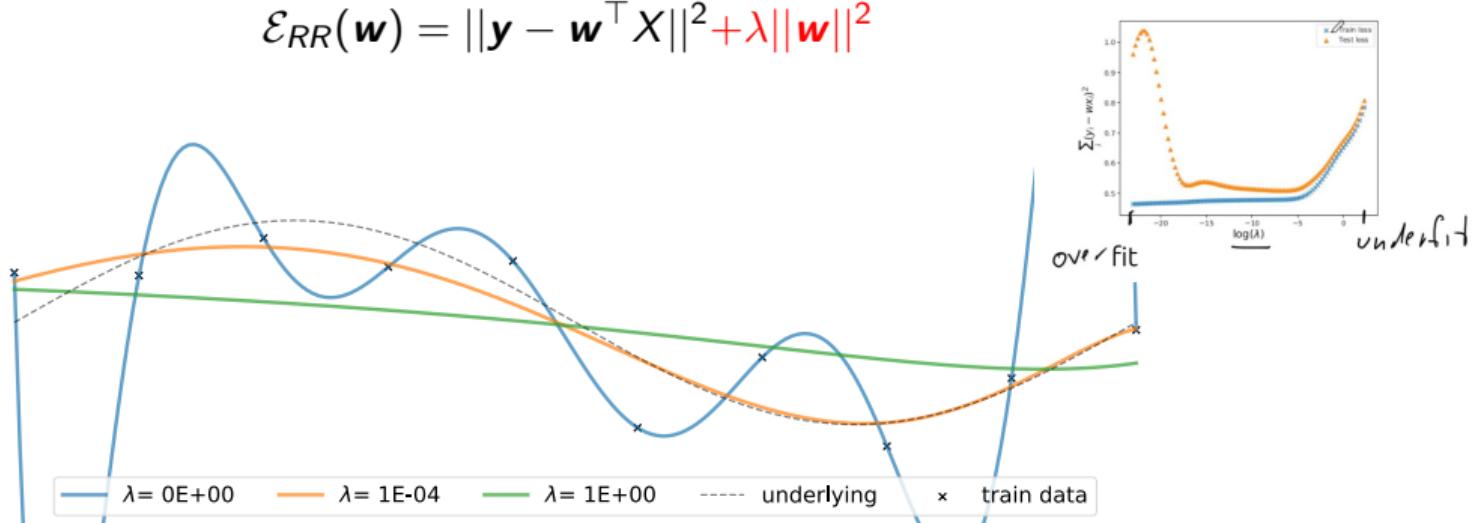
## Ridge regression

通常，控制解 $w$ 的复杂性非常重要。这可以通过约束 $w$ 的范数（正则化）来实现。

Often it is important to **control the complexity** of the solution  $w$ .

This is done by constraining the norm of  $w$  (regularization)

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



## Ridge regression

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

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

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

$$2XX^\top \mathbf{w} + 2\lambda\mathbf{w} = 2X\mathbf{y}^\top$$

$$(XX^\top + \lambda I)\mathbf{w} = X\mathbf{y}^\top$$

$$\mathbf{w} = (XX^\top + \lambda I)^{-1}X\mathbf{y}^\top$$

*Biased estimator, but smaller  
variance*

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  $\mathbb{R}^{d+1} \rightarrow \mathbb{R}^m$

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

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

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

$$W = (X X^\top + \lambda I)^{-1} X Y^\top$$

**Output:**  $W$

## Bias and variance of $y_{test}$

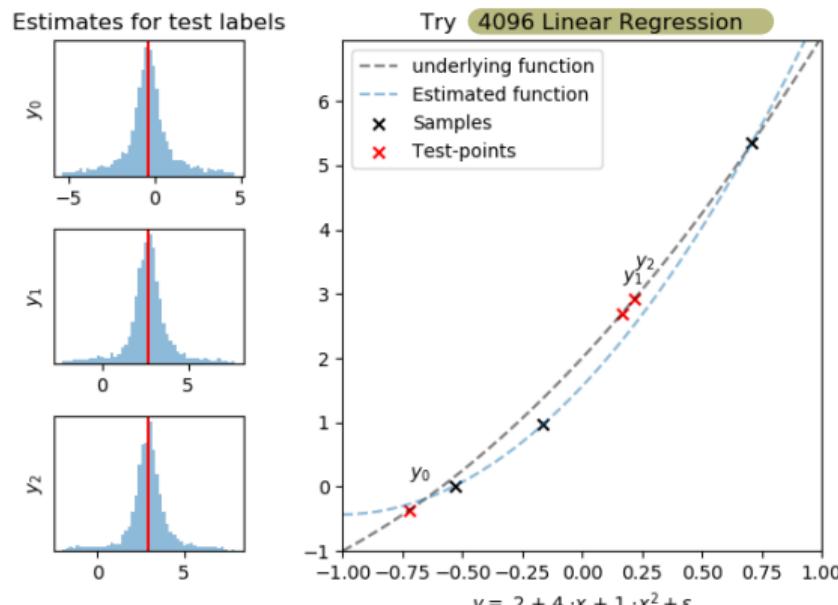
How do bias and variance of predicted labels behave  
 → Let's simulate  
 For LR it looks unbiased (on average correct), but high variance  
 Let's look at the effect of regularization

$y_{test}$ 的偏差和方差

预测标签的偏差和方差如何表现

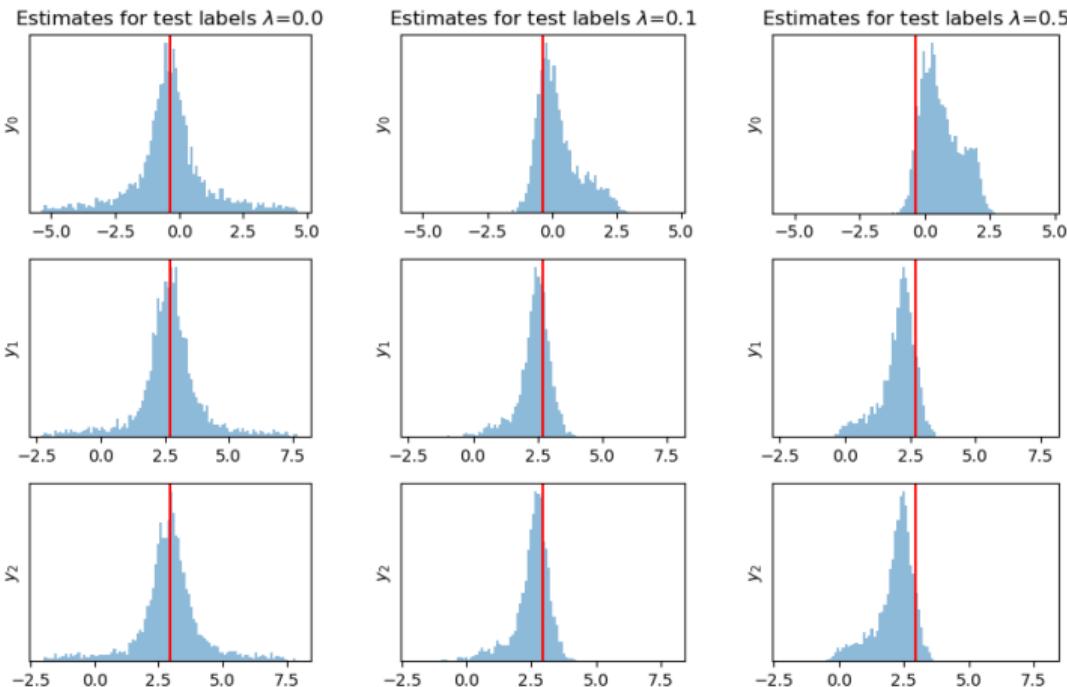
→ 让我们进行模拟

对于线性回归 (LR) , 它看起来是无偏的 (平均正确) , 但方差较高。让我们看看正则化的影响。



## Effect of $\lambda$ on bias and variance

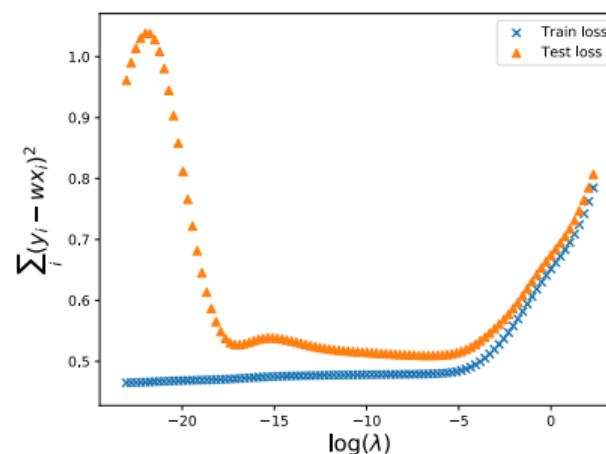
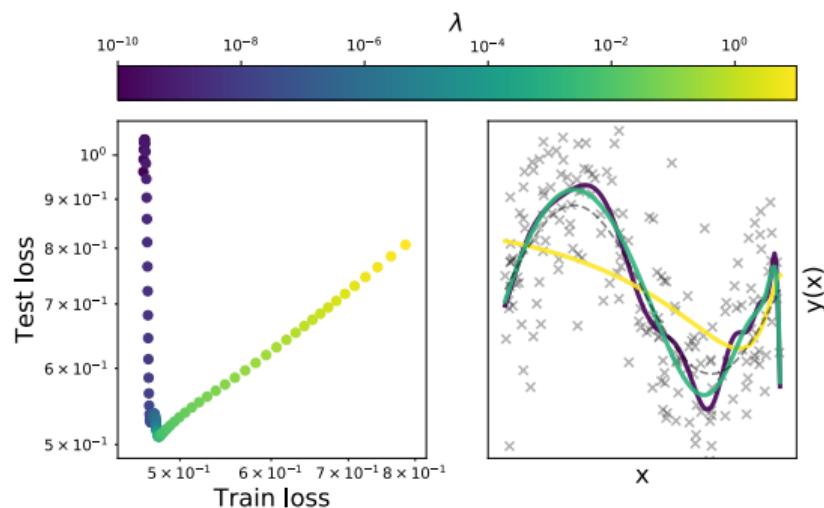
- For RR we see that estimates are biased, but less variance
- How can we choose optimal  $\lambda$ ?



# Model selection

How can we find the best  $\lambda$ ?  
 One option: grid search

→ try out e.g.  $\lambda \in \{0, 0.1, \dots, 0.9, 1.0\}$  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

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

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:

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

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

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

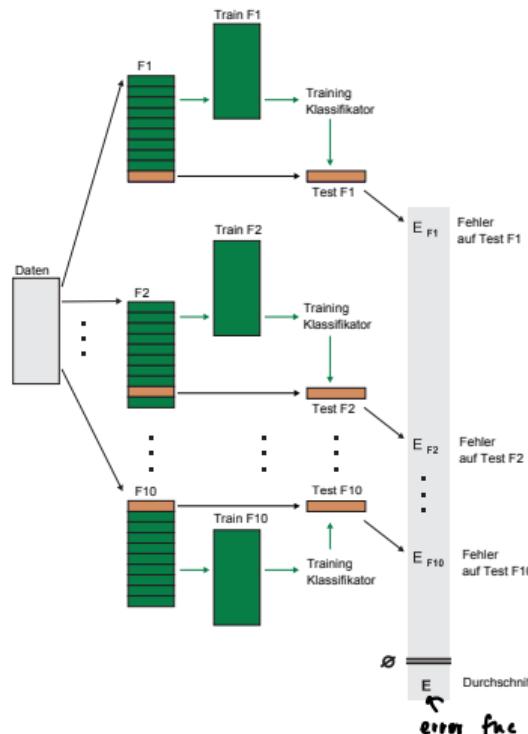
fold 2  $[x_{i_1}, \underbrace{x_{i_2}, x_{i_3}, x_{i_4}}_{\mathcal{F}_2^{\text{test}}}, \underbrace{x_{i_5}, x_{i_6}}_{\mathcal{F}_2^{\text{train}}}]$

fold 3 ...

For each fold:

**Train** your model on the training data

**Test** your model on the test data



# Cross-validation

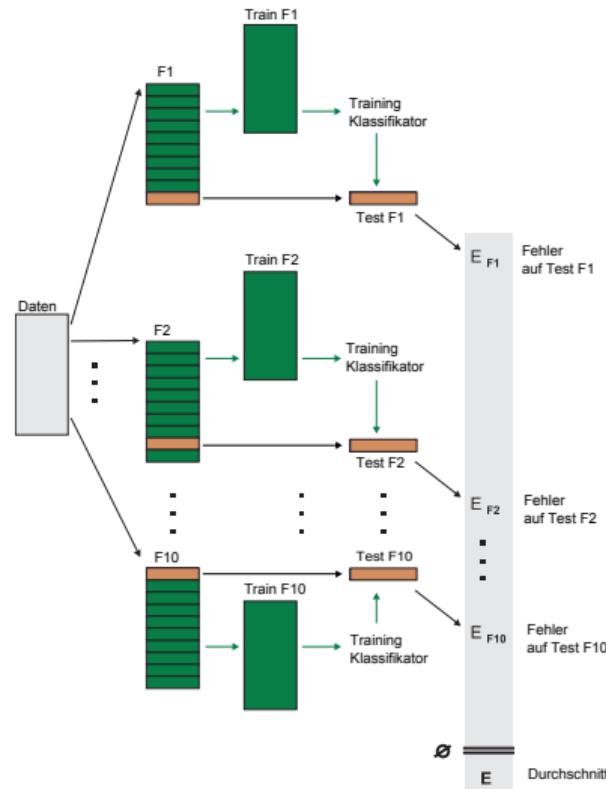
---

## 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	$w = Xy^T$	$y_t \in \left\{ \frac{1}{n+1}, -\frac{1}{n-1} \right\}$ NCC: Isotropic Normal distribution
LDA	$w = S^{-1}Xy^T$	LDA: Equal within-class covariances, Multivariate Normal distribution
Linear Regression	$w = (XX^\top)^{-1}Xy^\top$	$y_i \in \mathbb{R}$ 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. Driessens, L. S. Mattos, E. De Momi, M. Laffranchi, and L. De Micheli. 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/](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.