Foundations of Machine Learning
Applications for Genomic Data

Prof. Dr. Dominik Grimm

University of Applied Sciences Weihenstephan-Triesdorf

Technical University of Munich – Campus Straubing for Biotechnology and Sustainability

Get your Smartphone/Tablets or Laptops!

http://pingo.upb.de

ID: 781448

What is Machine Learning?

Lecture Material & Code

GitHub https://github.com/grimmlab/lecture_ml4genomics

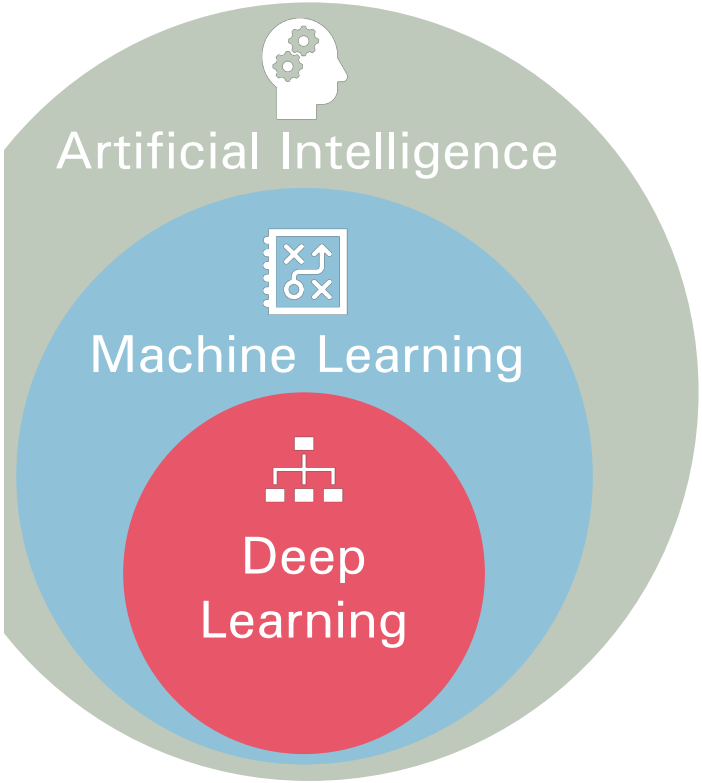
Overview

- **Part 1:** Foundations of Machine Learning
- **Part 2:** Two ML Methods in Detail – Regression vs. Classification
- **Part 3:** Real-World Applications of ML in Genomics and Genetics
- **Part 4:** Applied Machine Learning – How to use Python with SciKit-Learn

Part 1

Foundations of Machine Learning

The AI World



Artificial Intelligence

Any intelligent machine, software or technique that perceives its environment and **makes decisions like humans** would do

Machine Learning

Constructing and studying systems that can **learn** from **data**

Deep Learning

Branch of Machine Learning that uses **neural networks** for learning from data

Types of Machine Learning Systems

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Supervised Machine Learning

Regression Example: Predicting House Prices

Size: 150 m^2
Price: 500k $

Size: 250 m^2
Price: 1000k $

Size: 120 m^2
Price: 370k $

Size: 100 m^2
Price: 350k $

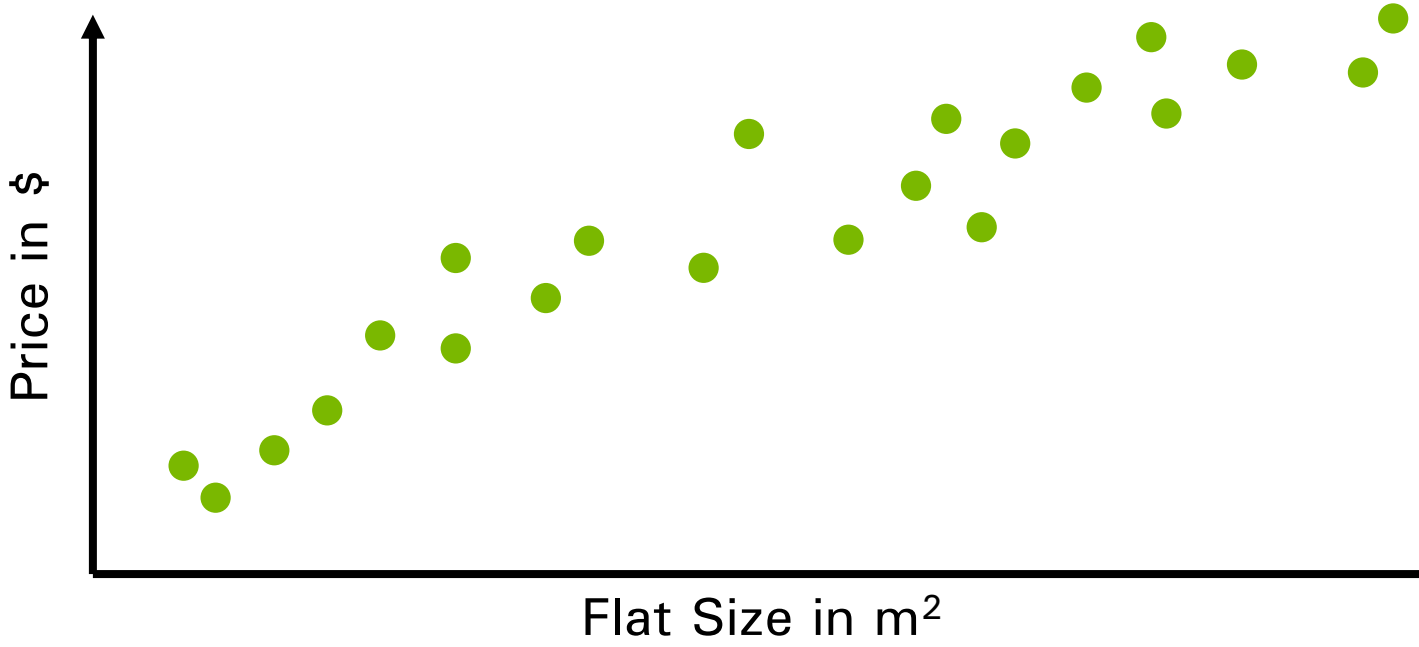
Size: 154 m^2
Price: 450k $

Size: 80 m^2
Price: 280k $

Size: 54 m^2
Price: 150k $

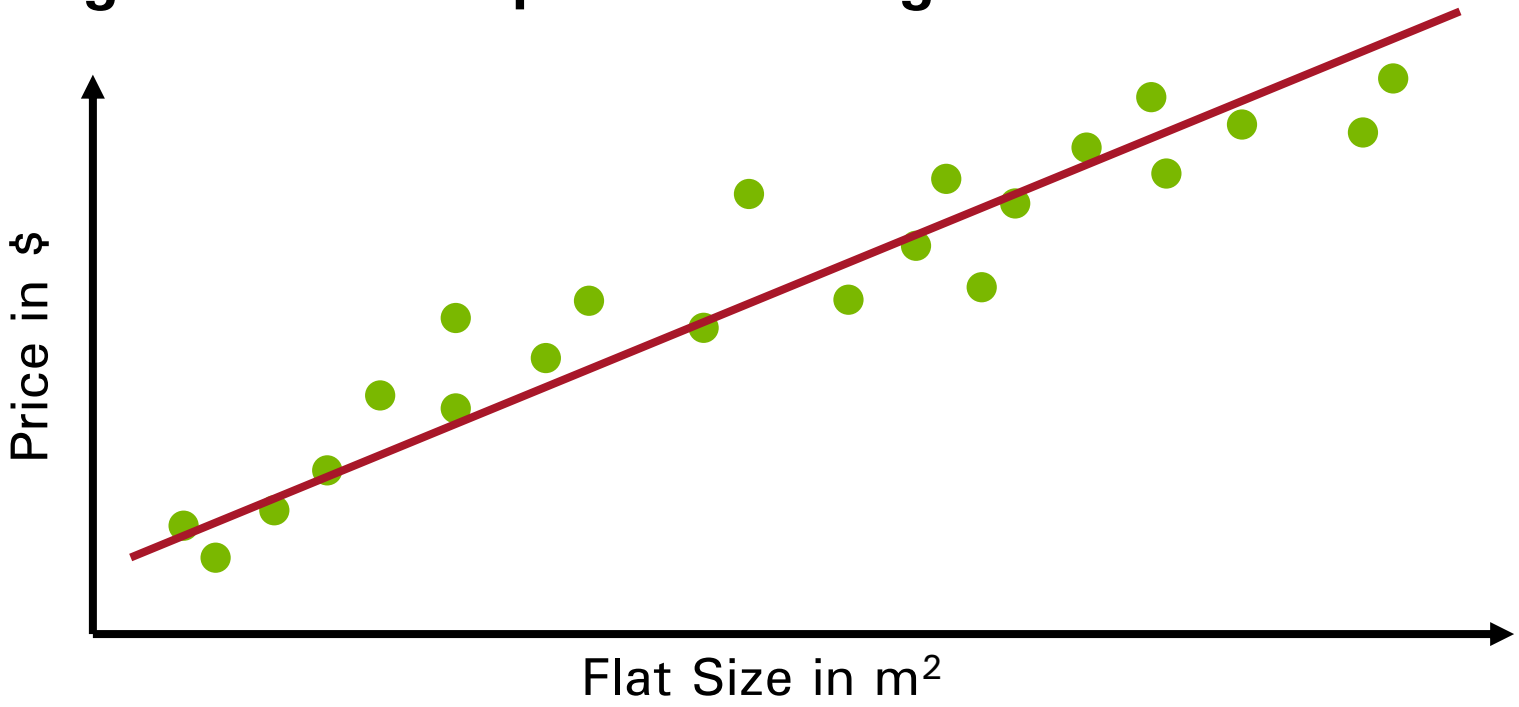
Supervised Machine Learning

Regression Example: Predicting House Prices



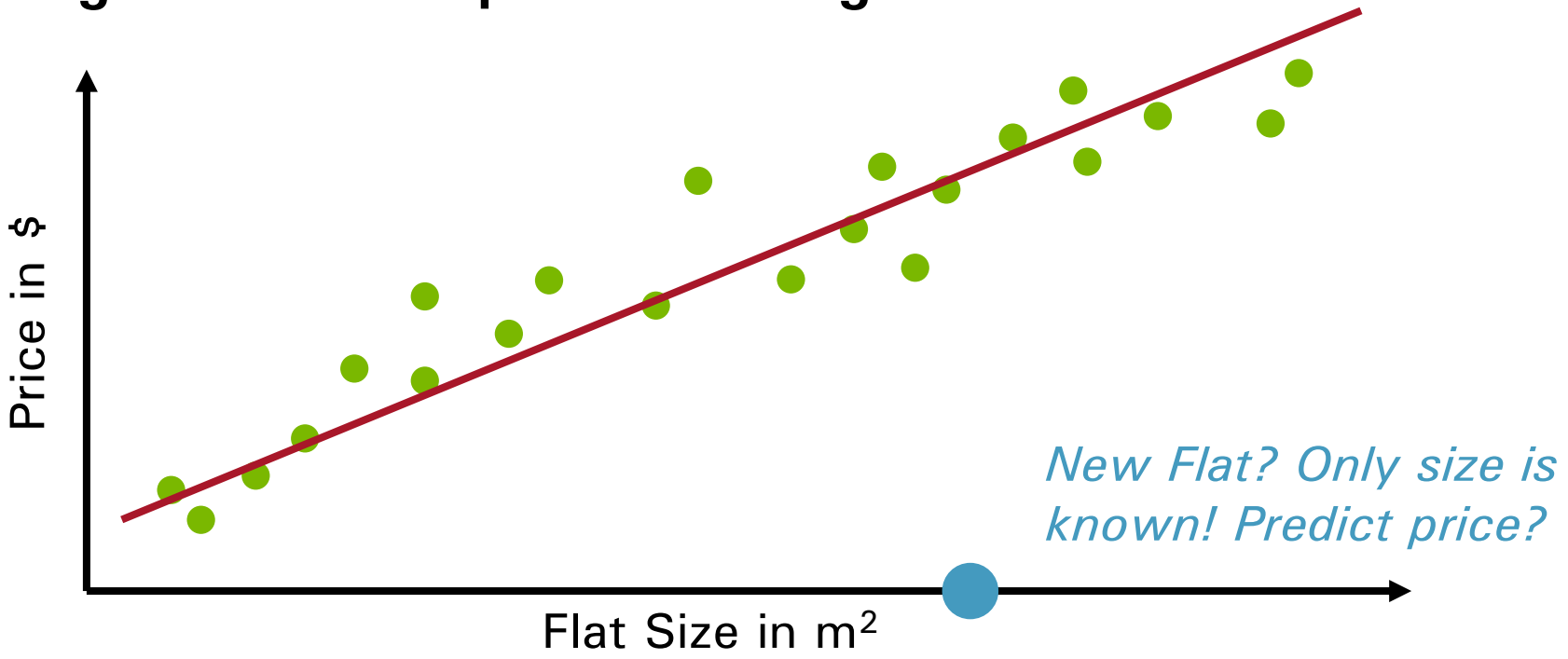
Supervised Machine Learning

Regression Example: Predicting House Prices



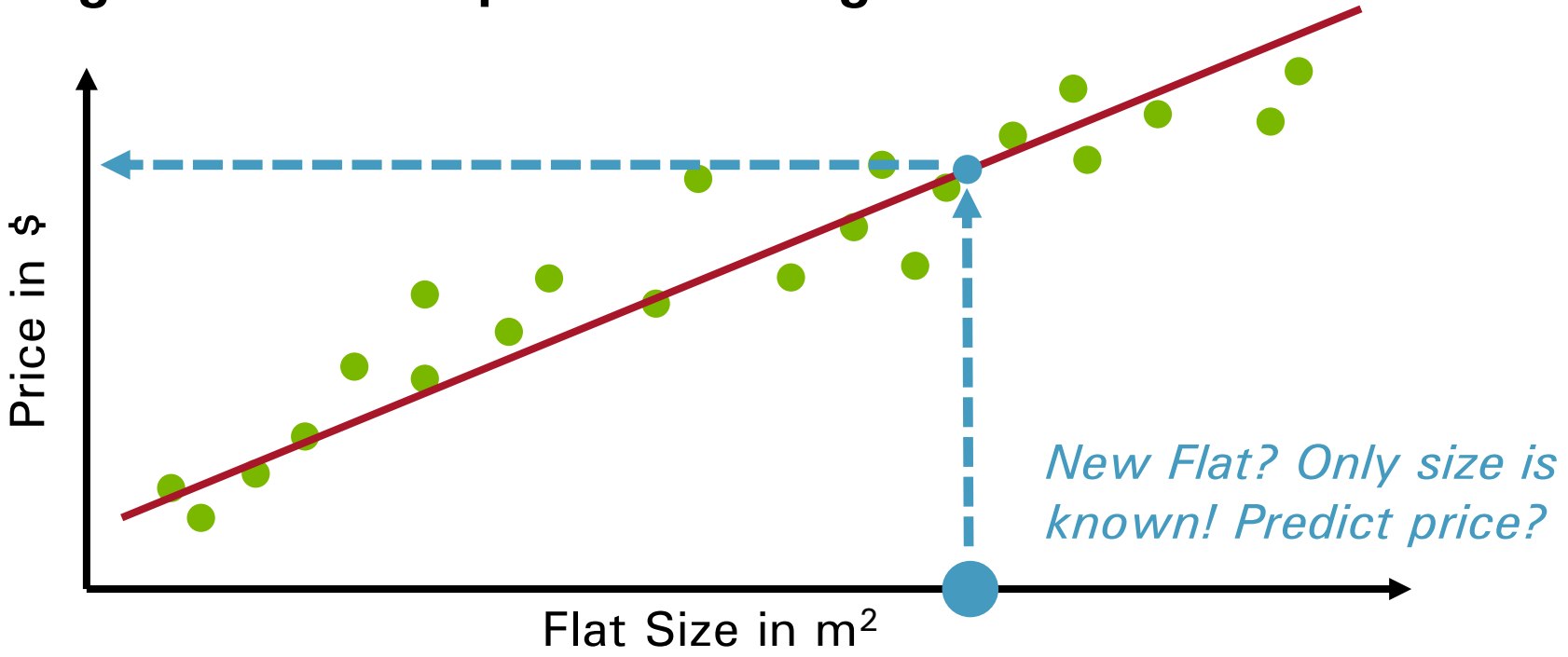
Supervised Machine Learning

Regression Example: Predicting House Prices



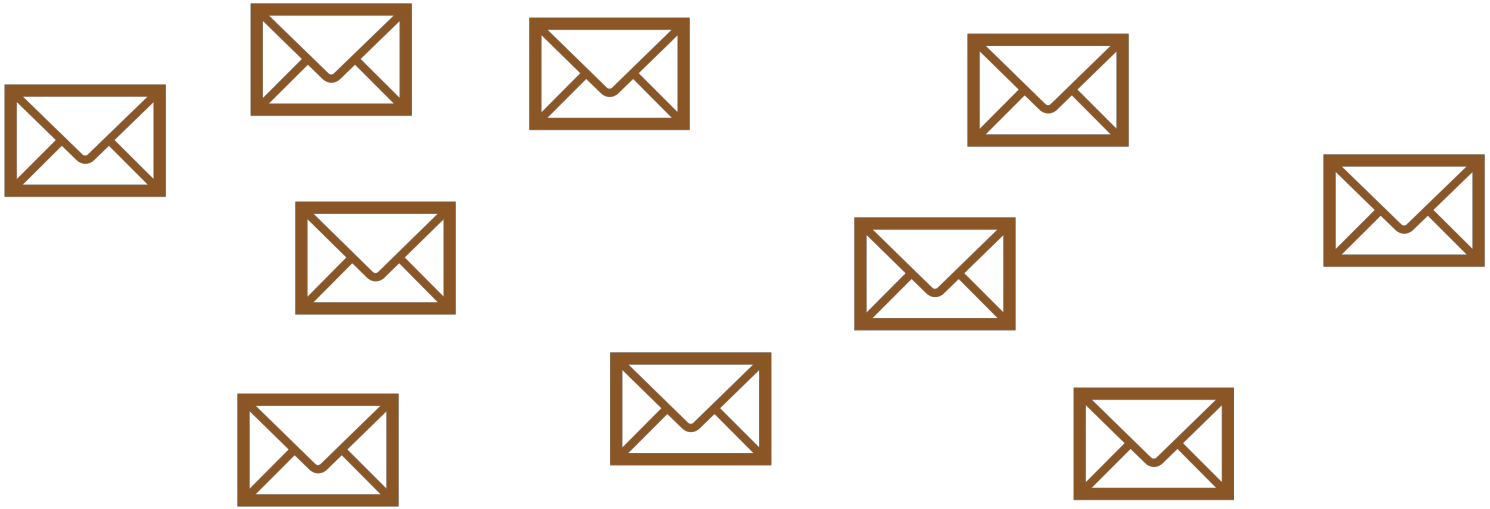
Supervised Machine Learning

Regression Example: Predicting House Prices



Supervised Machine Learning

Classification Example: Predicting Spam vs No Spam

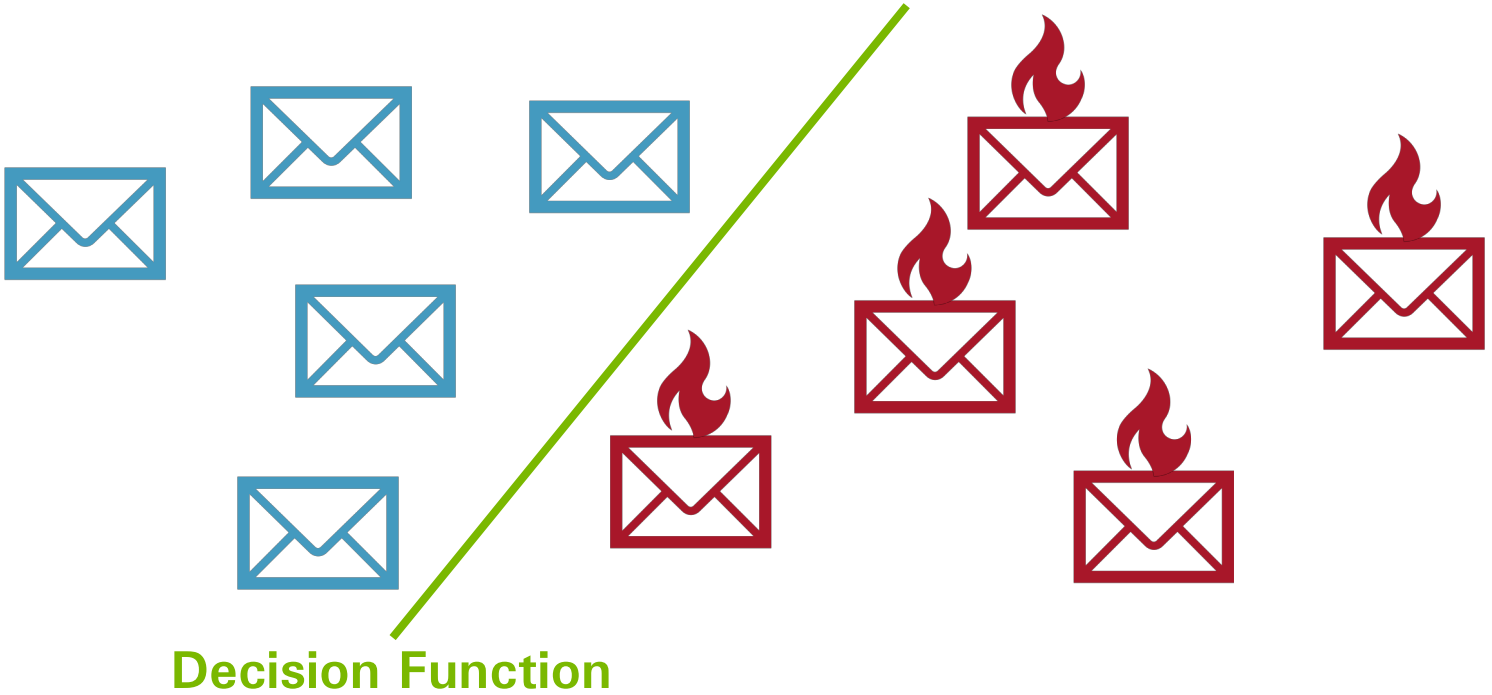


Supervised Machine Learning

Classification Example: Predicting Spam vs No Spam

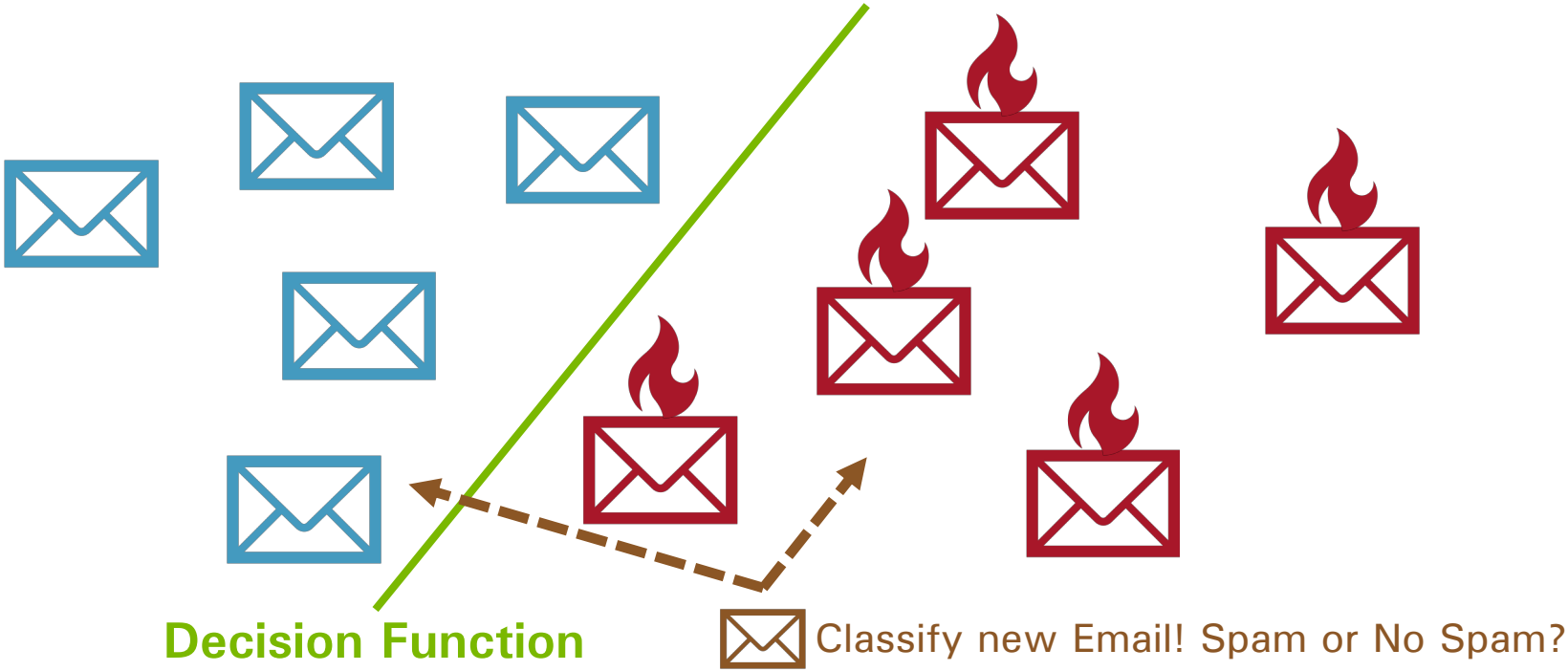
Supervised Machine Learning

Classification Example: Predicting Spam vs No Spam

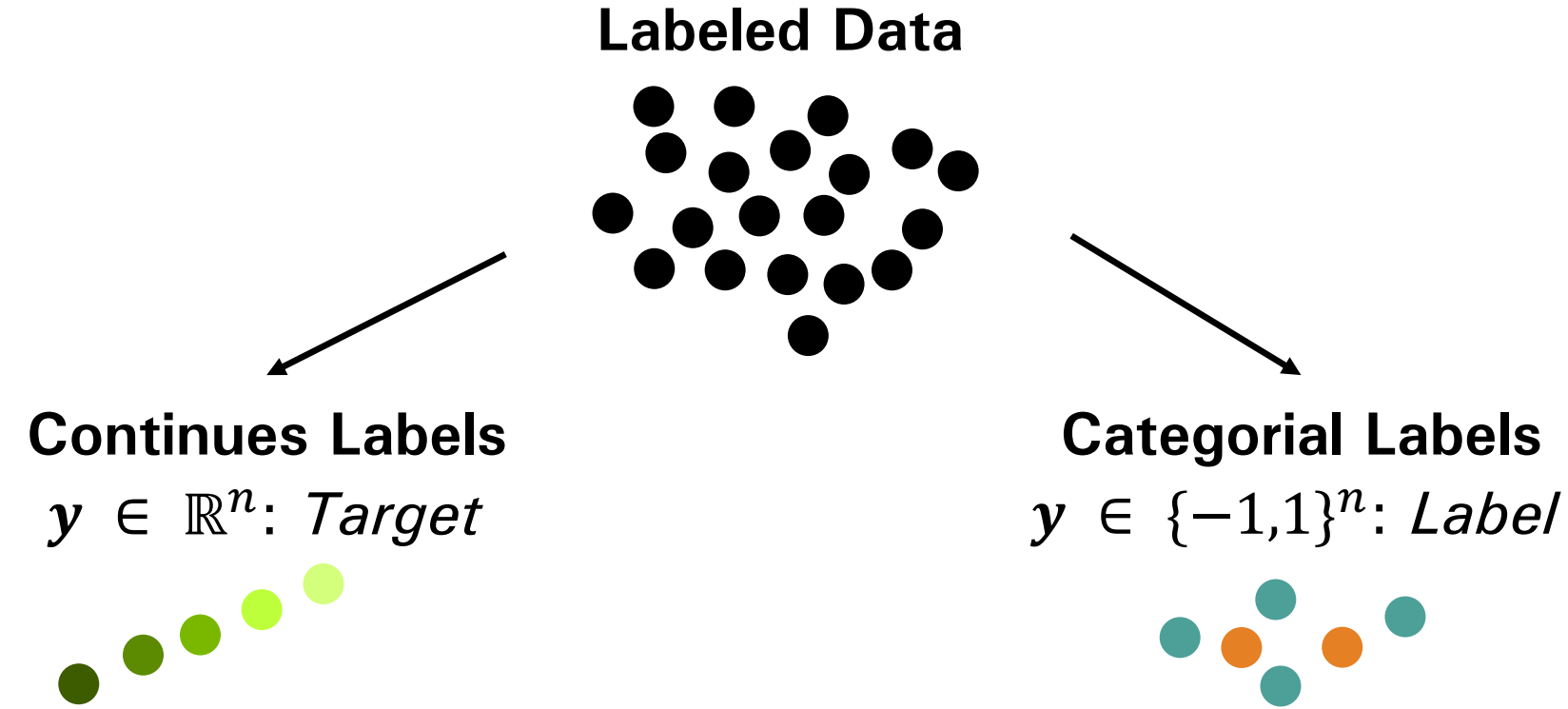


Supervised Machine Learning

Classification Example: Predicting Spam vs No Spam



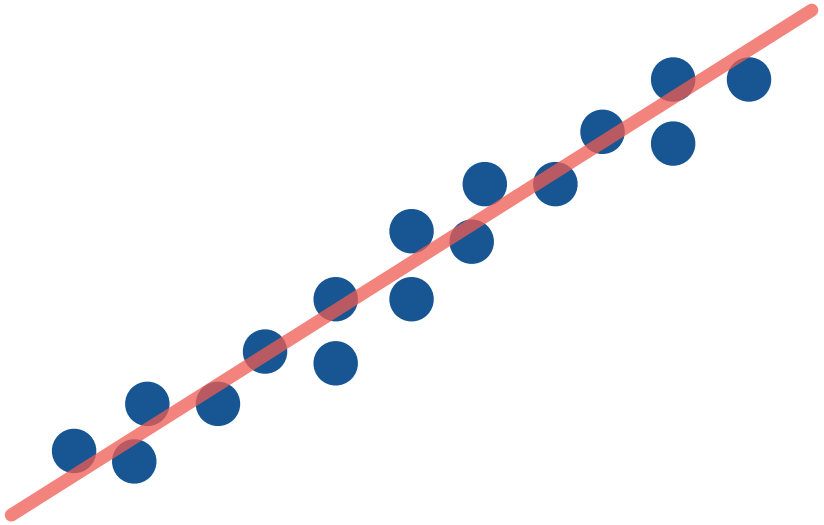
Supervised Machine Learning



Supervised Machine Learning

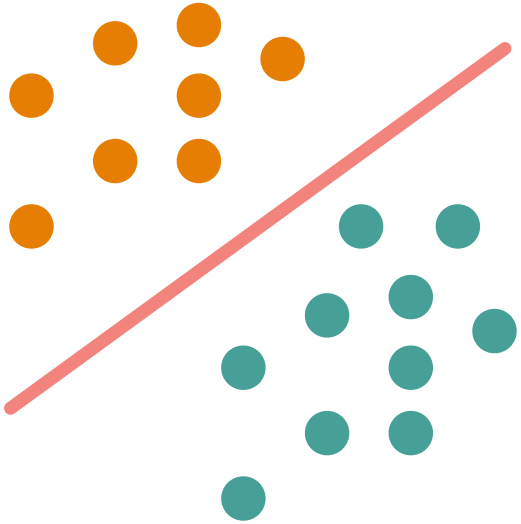
Regression

$\boldsymbol{y} \in \mathbb{R}^n$: *Target*



Classification

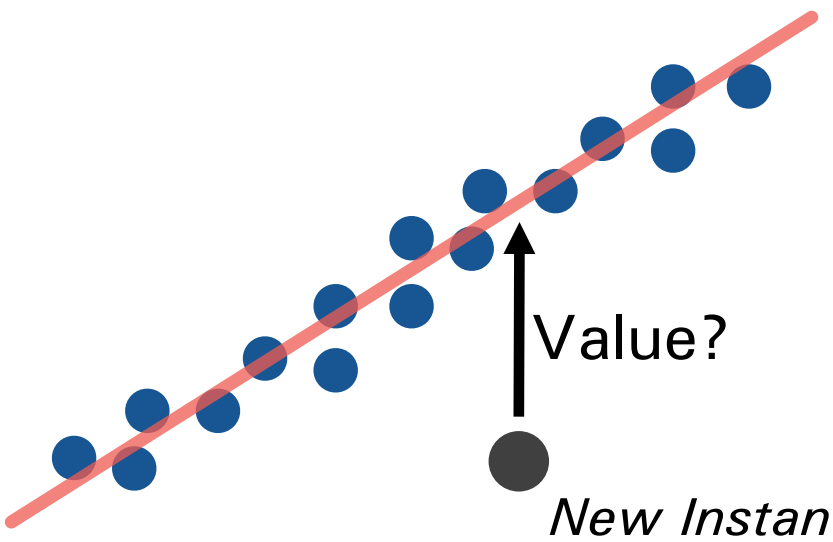
$\boldsymbol{y} \in \{-1,1\}^n$: *Label*



Supervised Machine Learning

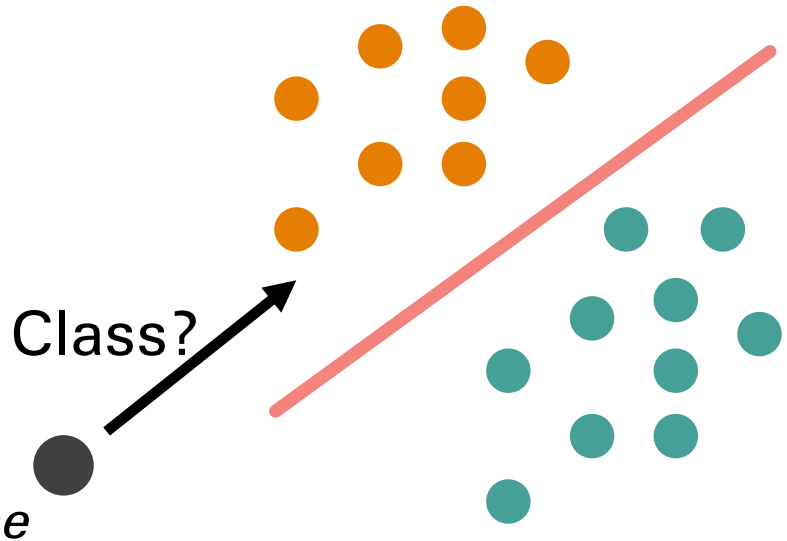
Regression

$\boldsymbol{y} \in \mathbb{R}^n$: *Target*



Classification

$\boldsymbol{y} \in \{-1,1\}^n$: *Label*



Types of Machine Learning Systems

Supervised Learning

Ground Truth

Direct Feedback

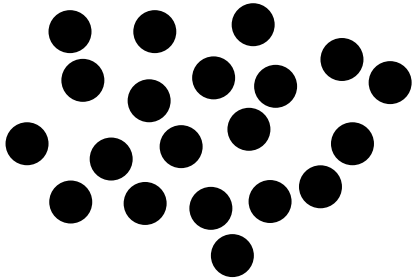
Prediction

Unsupervised Learning

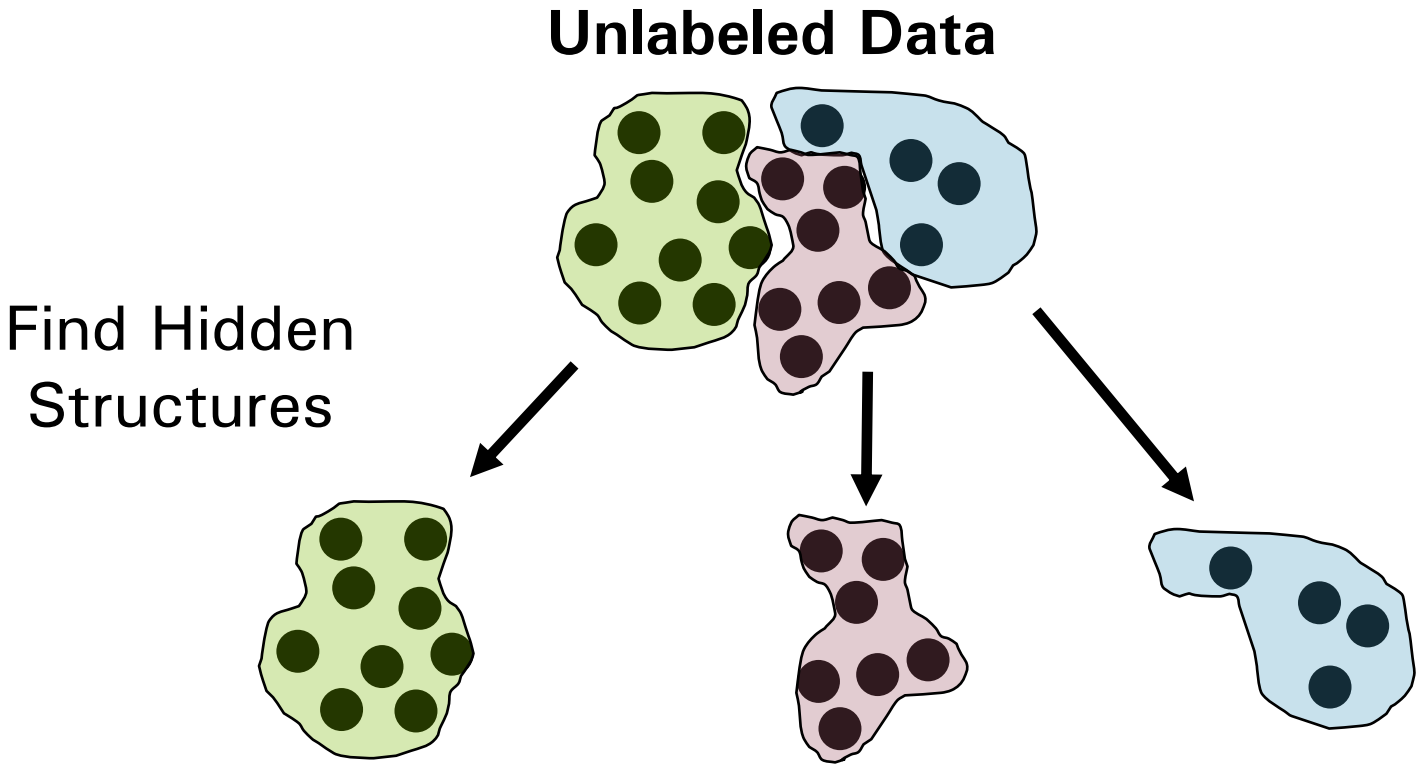
Reinforcement Learning

Unsupervised Machine Learning

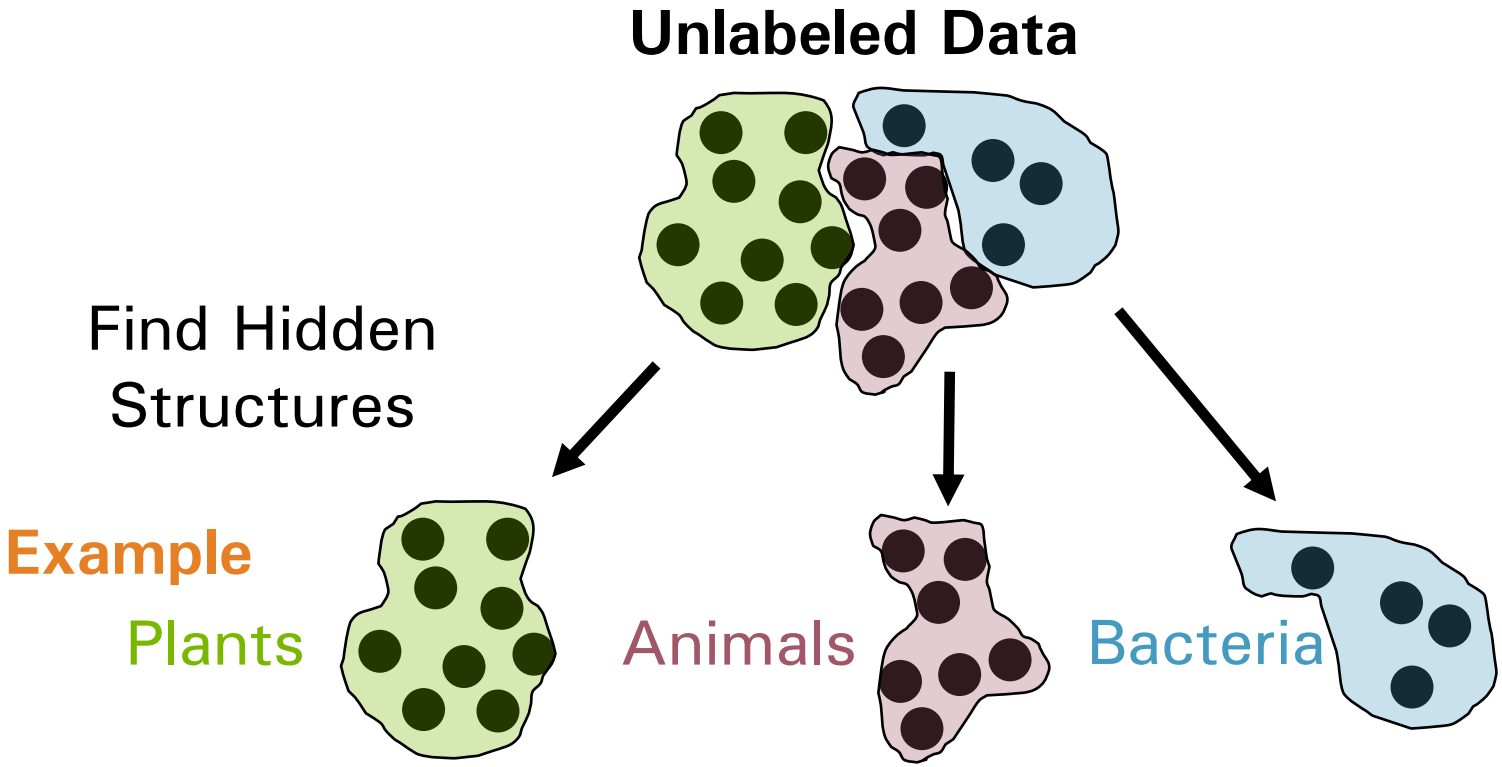
Unlabeled Data



Unsupervised Machine Learning



Unsupervised Machine Learning



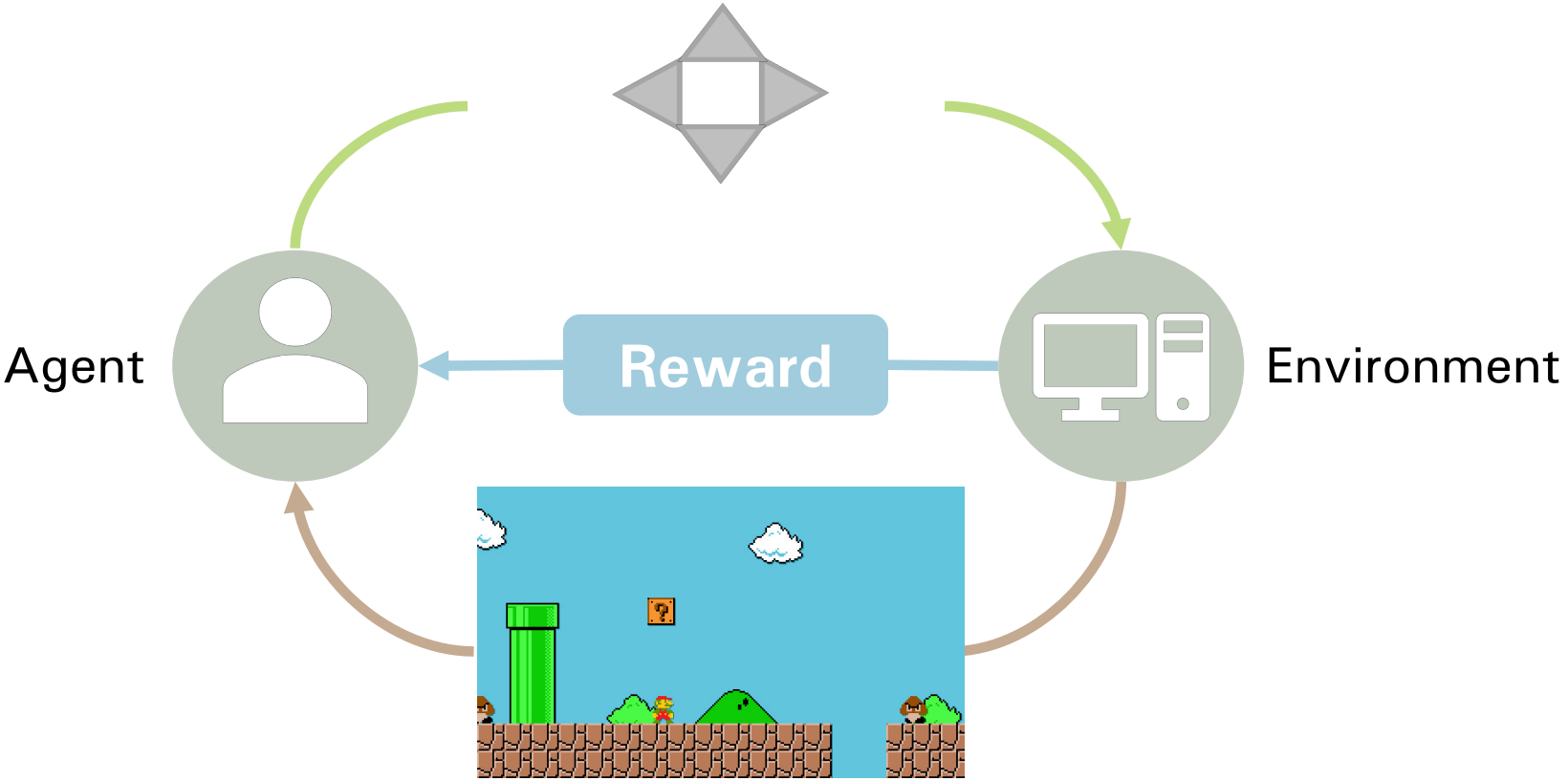
Types of Machine Learning Systems

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Ground Truth	No Ground Truth	
Direct Feedback	No Feedback	
Prediction	Find Hidden Structures	

Reinforcement Learning

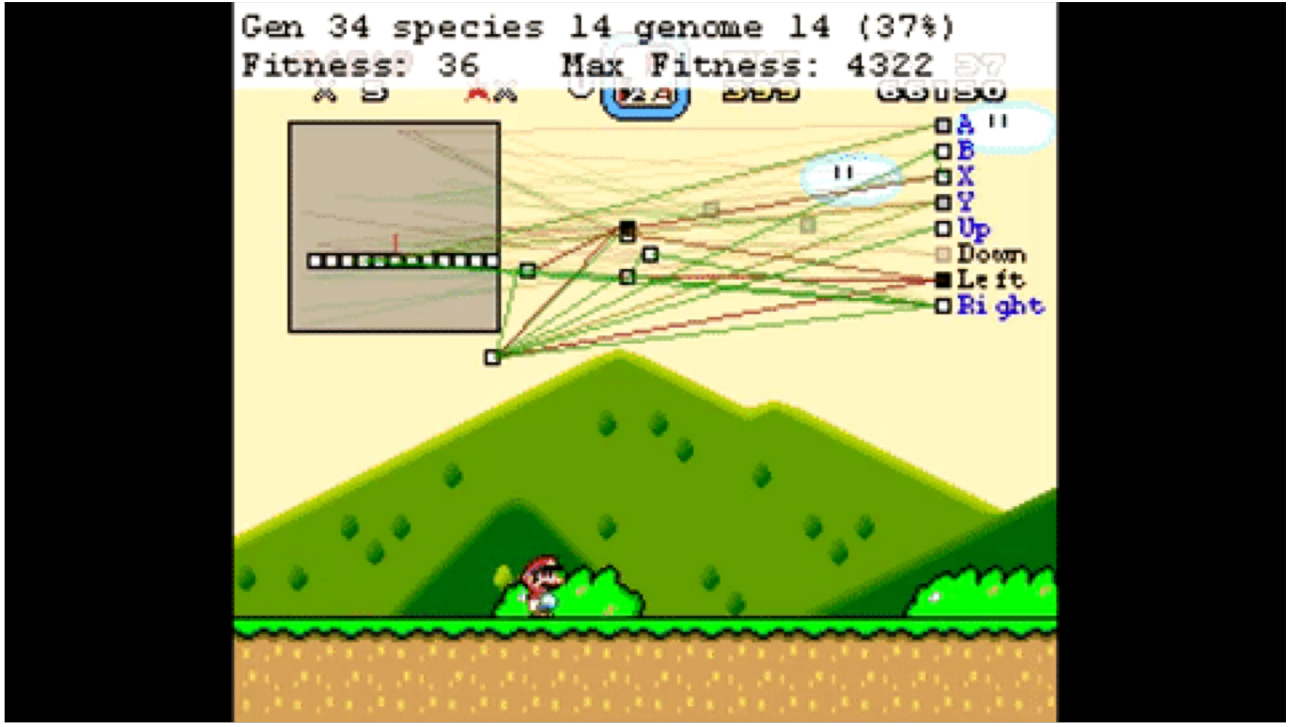
Reinforcement Learning: Maximize long-term reward by “trail & error”

Reinforcement Learning



Reinforcement Learning

Mari/o (Generation 34)



https://origins.asu.edu/blog/great-debate-ai

Reinforcement Learning

Smart Home Automation: Learn to anticipate human needs

https://en.wikipedia.org/wiki/Nest_Learning_Thermostat

Types of Machine Learning Systems

Supervised Learning	Unsupervised Learning	Reinforcement Learning
Ground Truth	No Ground Truth	Decision Process
Direct Feedback	No Feedback	Reward System
Prediction	Find Hidden Structures	Learn Series of Actions

How could ML be used for bio-(medical) applications?

Machine Learning Workflow

Machine Learning Workflow

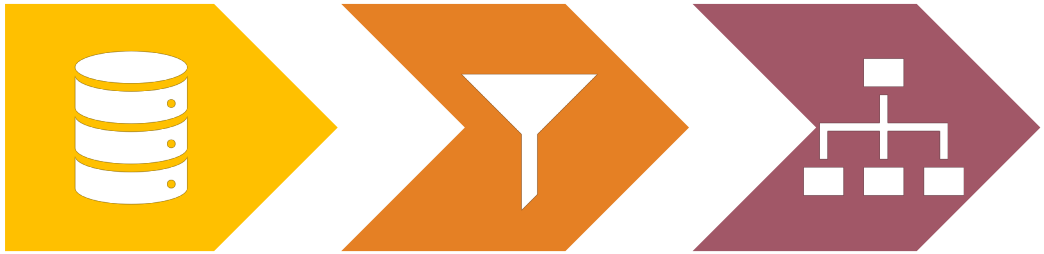
STEP 1
DATA
COLLECTION

Machine Learning Workflow

STEP 1
DATA
COLLECTION

STEP 2
DATA
PREPROCESSING
& CLEANING

Machine Learning Workflow

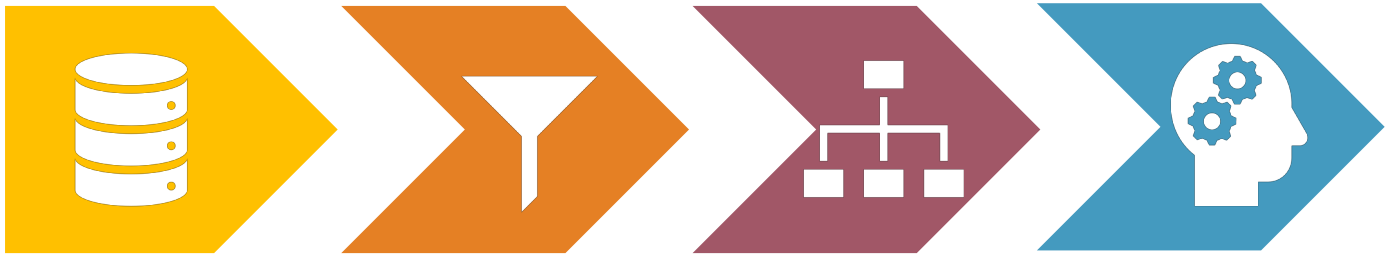


STEP 1
DATA
COLLECTION

STEP 2
DATA
PREPROCESSING
& CLEANING

STEP 3
MODEL
SELECTION AND
TRAINING

Machine Learning Workflow



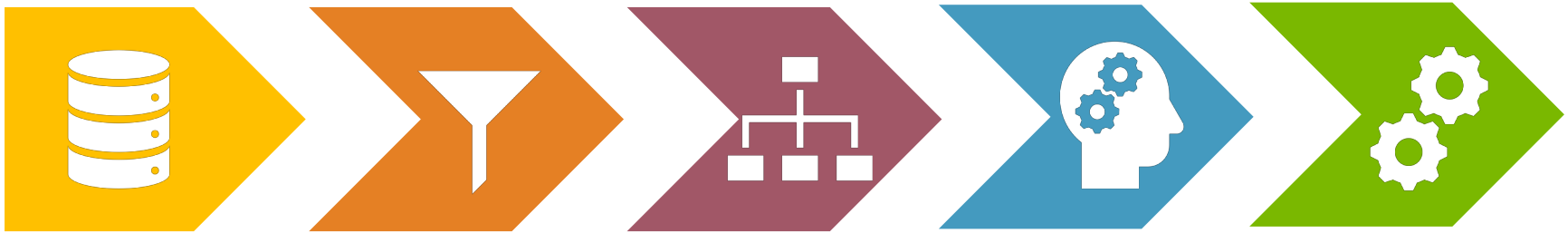
STEP 1
DATA
COLLECTION

STEP 2
DATA
PREPROCESSING
& CLEANING

STEP 3
MODEL
SELECTION AND
TRAINING

STEP 4
INFERENCE
GAIN INSIGHTS

Machine Learning Workflow



STEP 1
DATA COLLECTION

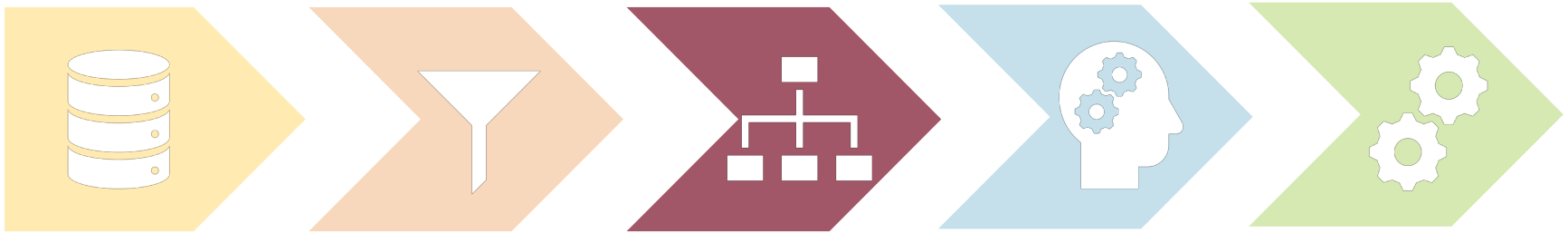
STEP 2
DATA PREPROCESSING & CLEANING

STEP 3
MODEL SELECTION AND TRAINING

STEP 4
INFERENCE GAIN INSIGHTS

STEP 5
DEPLOY MODEL

Machine Learning Workflow



STEP 1
DATA COLLECTION

STEP 2
DATA PREPROCESSING & CLEANING

STEP 3
MODEL SELECTION AND TRAINING

STEP 4
INFERENCE GAIN INSIGHTS

STEP 5
DEPLOY MODEL

Machine Learning Models

Part 2

Linear Regression, Ridge Regression and Support Vector Machines

Regression – Univariate Linear Regression

Goal:
Model the relationship between a singe **feature x** (explanatory variable) and a **continuous target variable y** (response variable)

$$y = b + w_1 x$$

b: bias or error term (y-axis intercept)

w_1: weight coefficient of feature x

Regression – Univariate Linear Regression

Goal:
Model the relationship between a singe **feature x** (explanatory variable) and a **continuous target variable y** (response variable)

$$y = b + w_1 x$$

b: bias or error term (y-axis intercept)

w_1: weight coefficient of feature x

Parameters b, w_1 are unknown.

We have to learn these parameters!

Regression – Univariate Linear Regression

Example

b and w_1 are initialized with random values → The resulting y value is called a prediction (Predictions or estimated values are indicated with the "head" symbol, e.g. $\hat{y}$)

$$\hat{\boldsymbol{y}} = b + w_1 \boldsymbol{x}$$

Regression – Univariate Linear Regression

Example

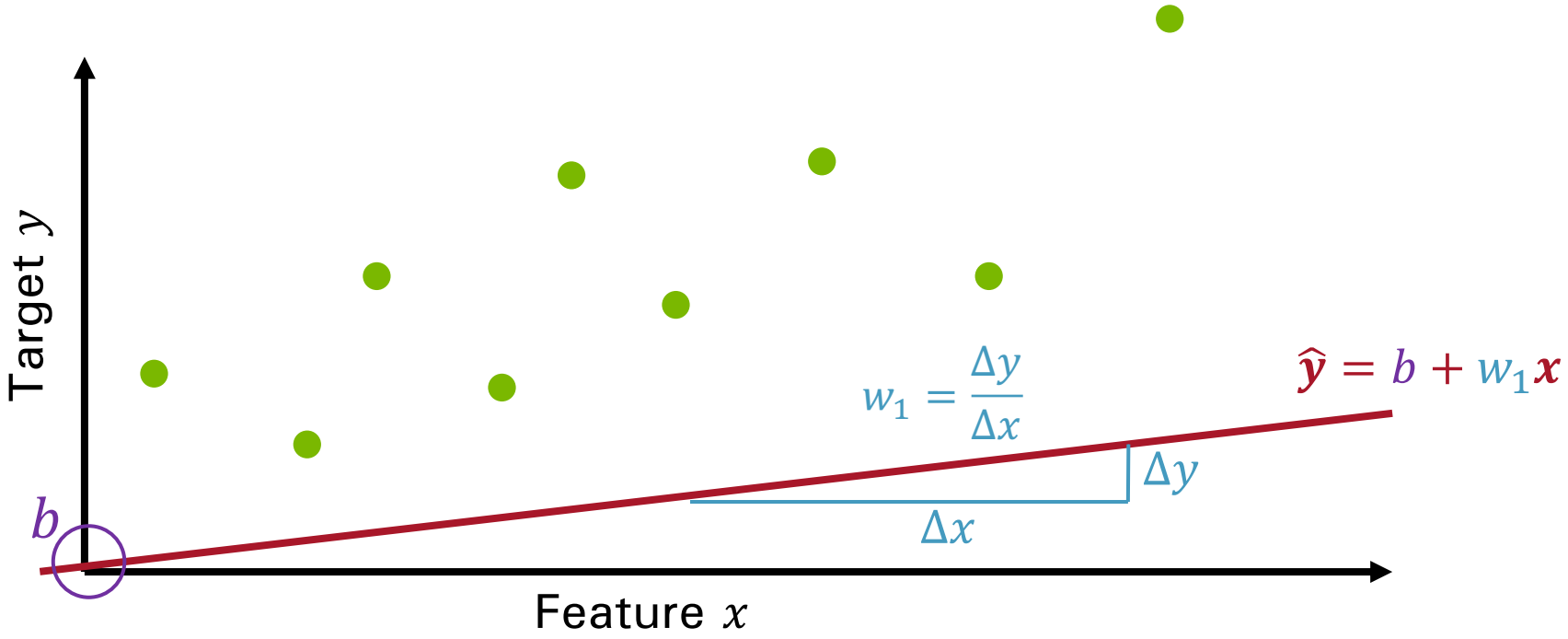
b and w_1 are initialized with random values → The resulting y value is called a prediction (Predictions or estimated values are indicated with the "head" symbol, e.g. $\hat{y}$)

$$\hat{y} = b + w_1 x$$

How good is our prediction, based on random values?

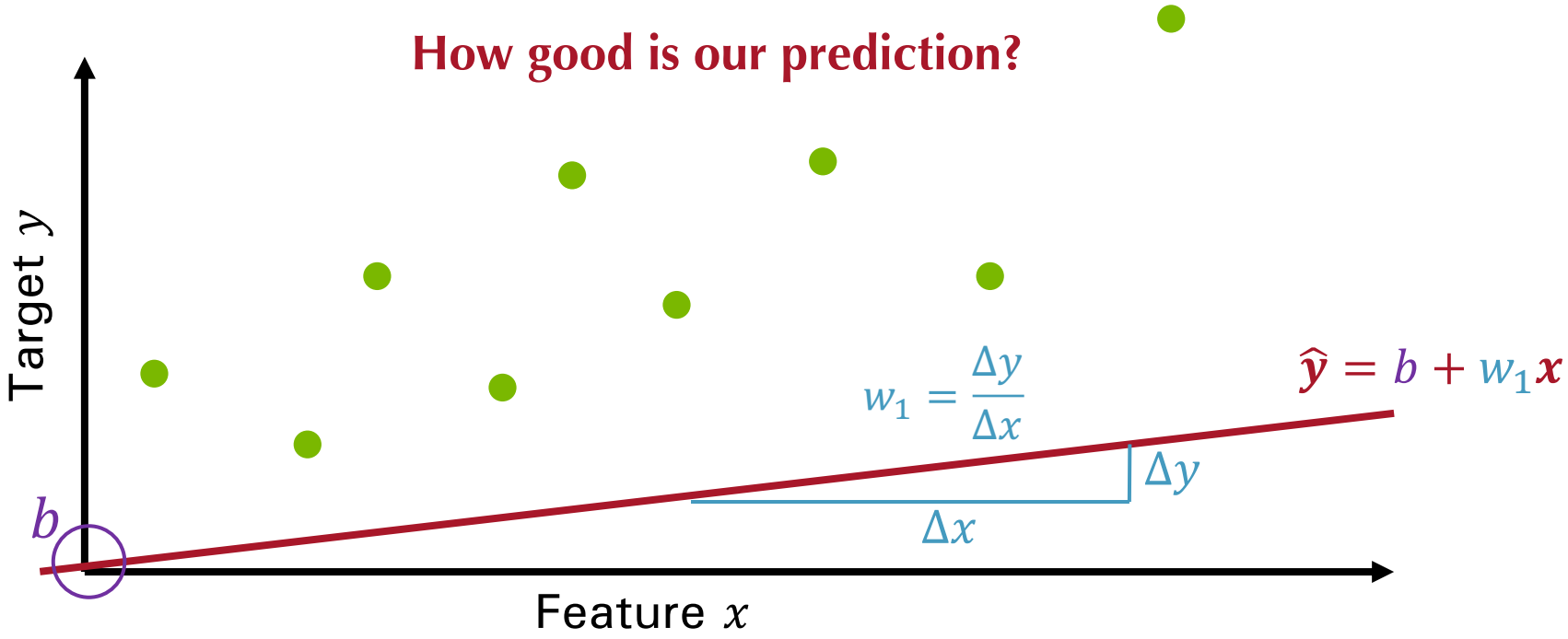
Univariate Linear Regression

b and w_1 are initialized with random values



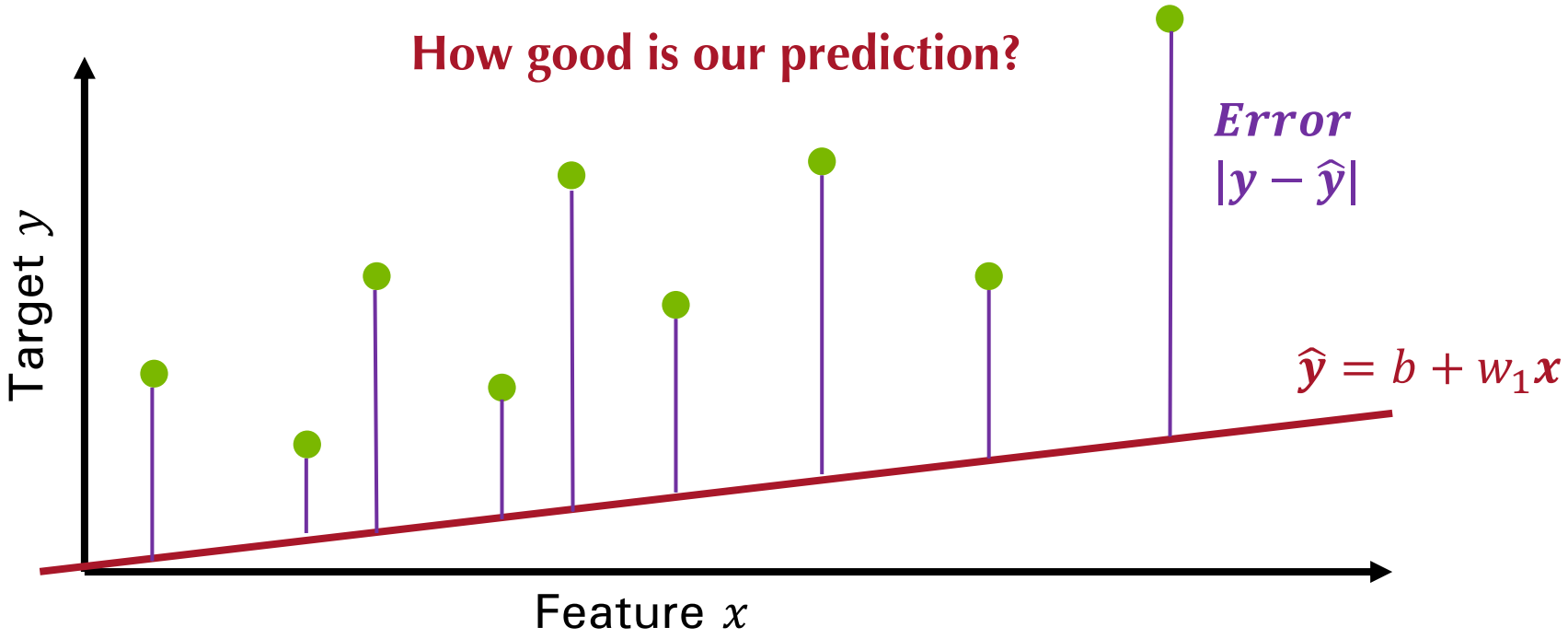
Univariate Linear Regression

b and w_1 are initialized with random values



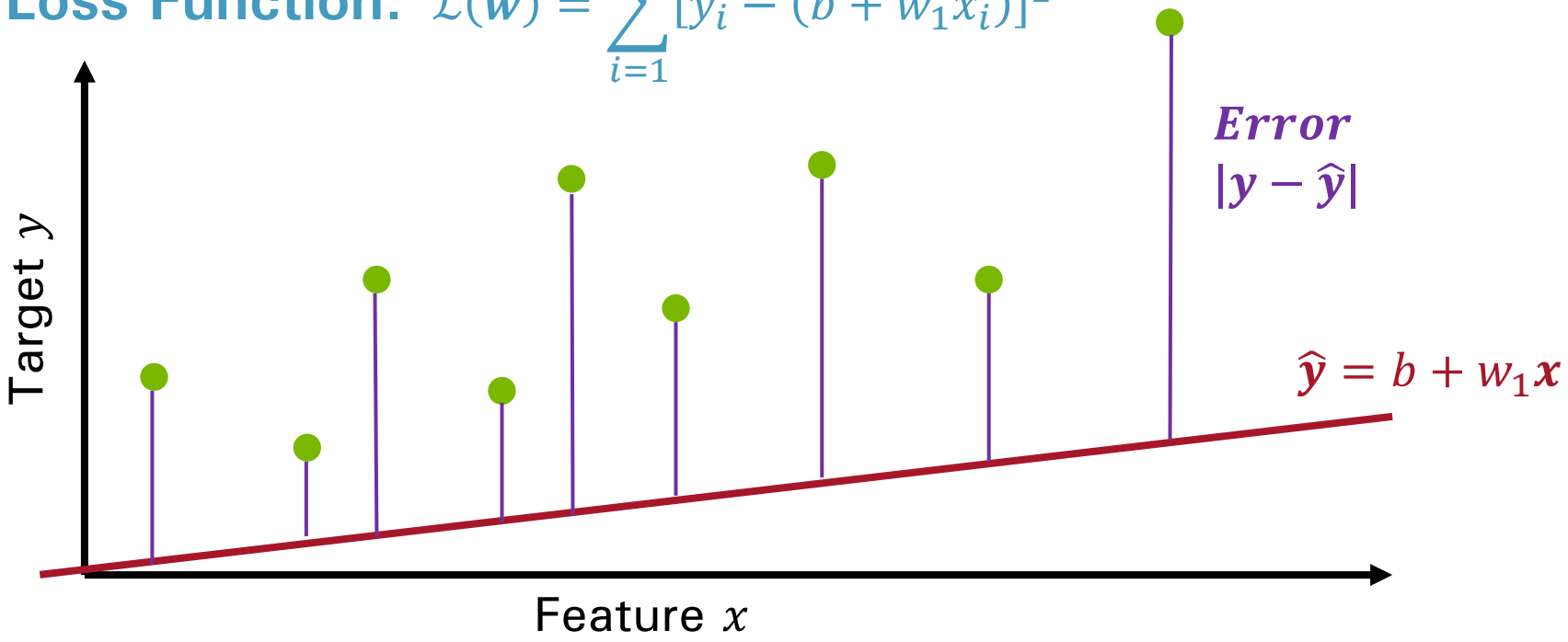
Univariate Linear Regression

b and w_1 are initialized with random values



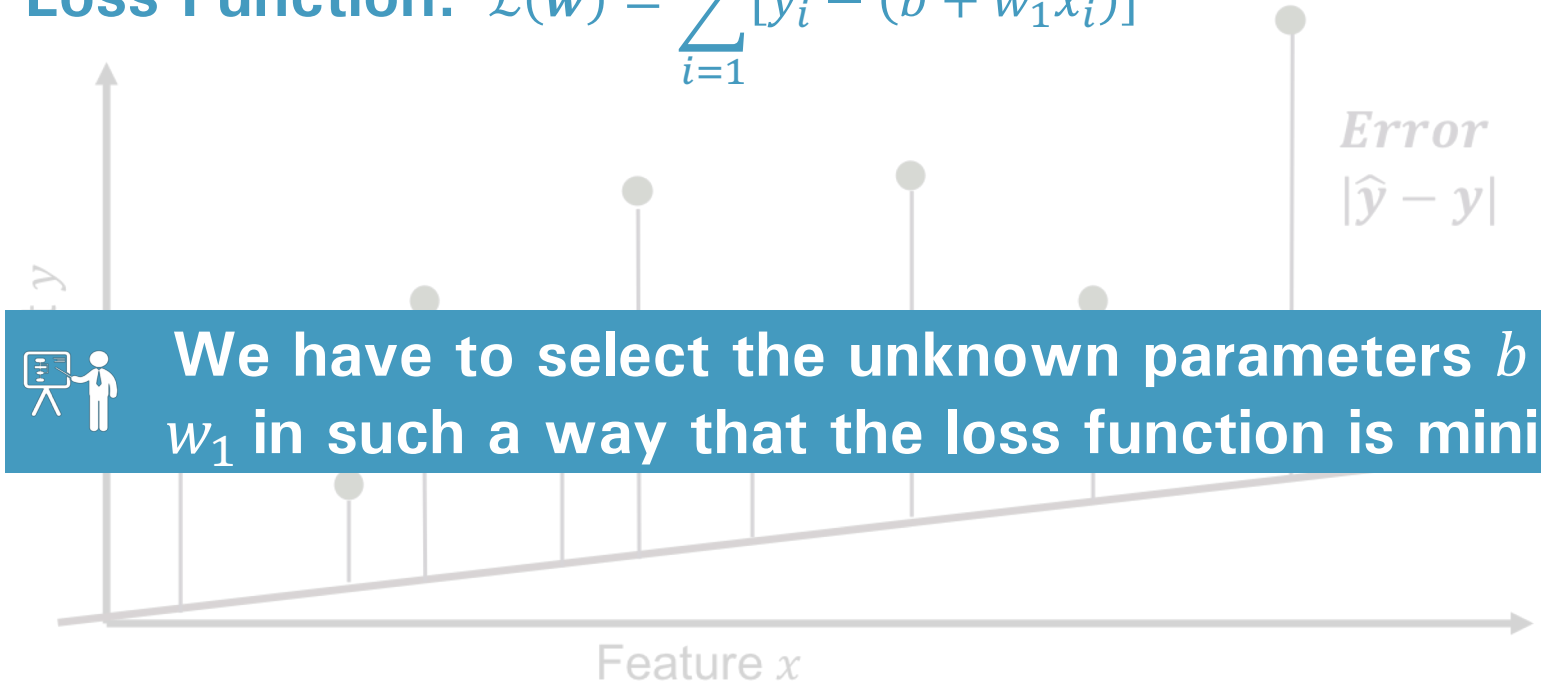
Univariate Linear Regression

Loss Function: $\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} [y_i - (b + w_1 x_i)]^2$



Univariate Linear Regression

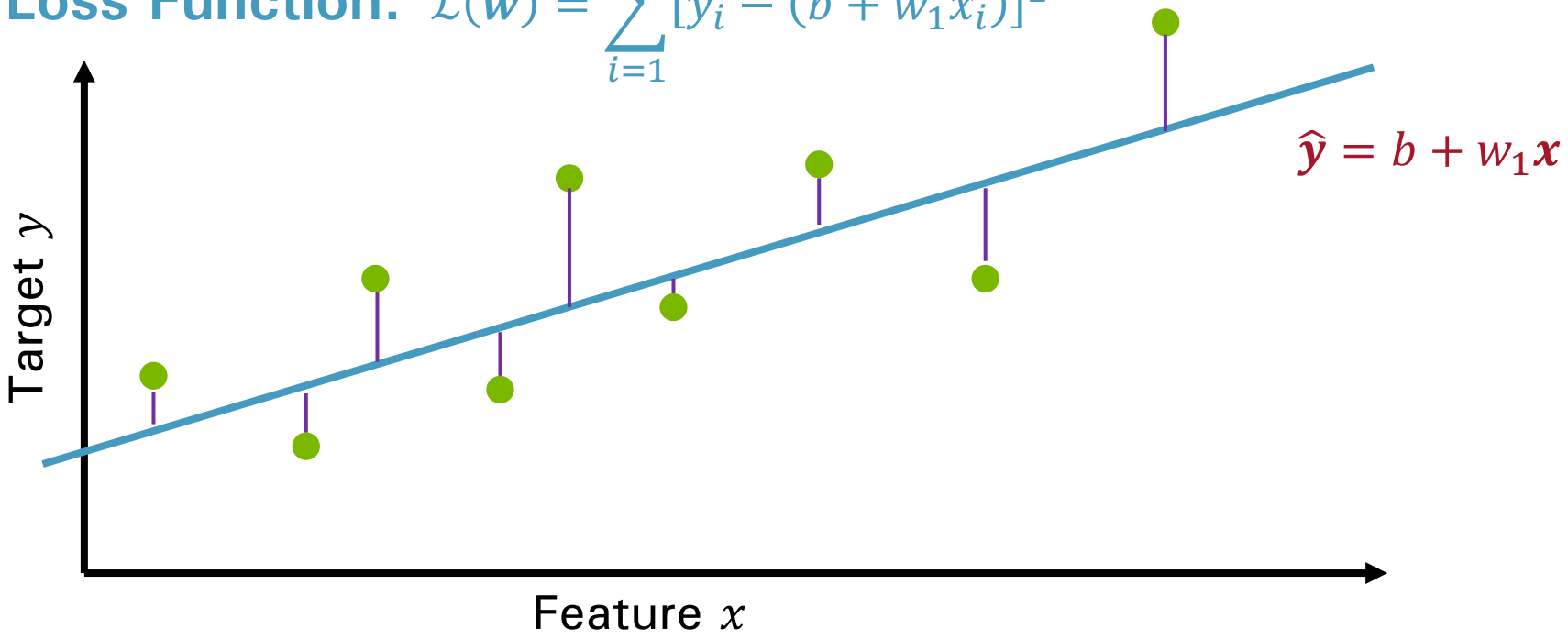
Loss Function: $\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} [y_i - (b + w_1 x_i)]^2$



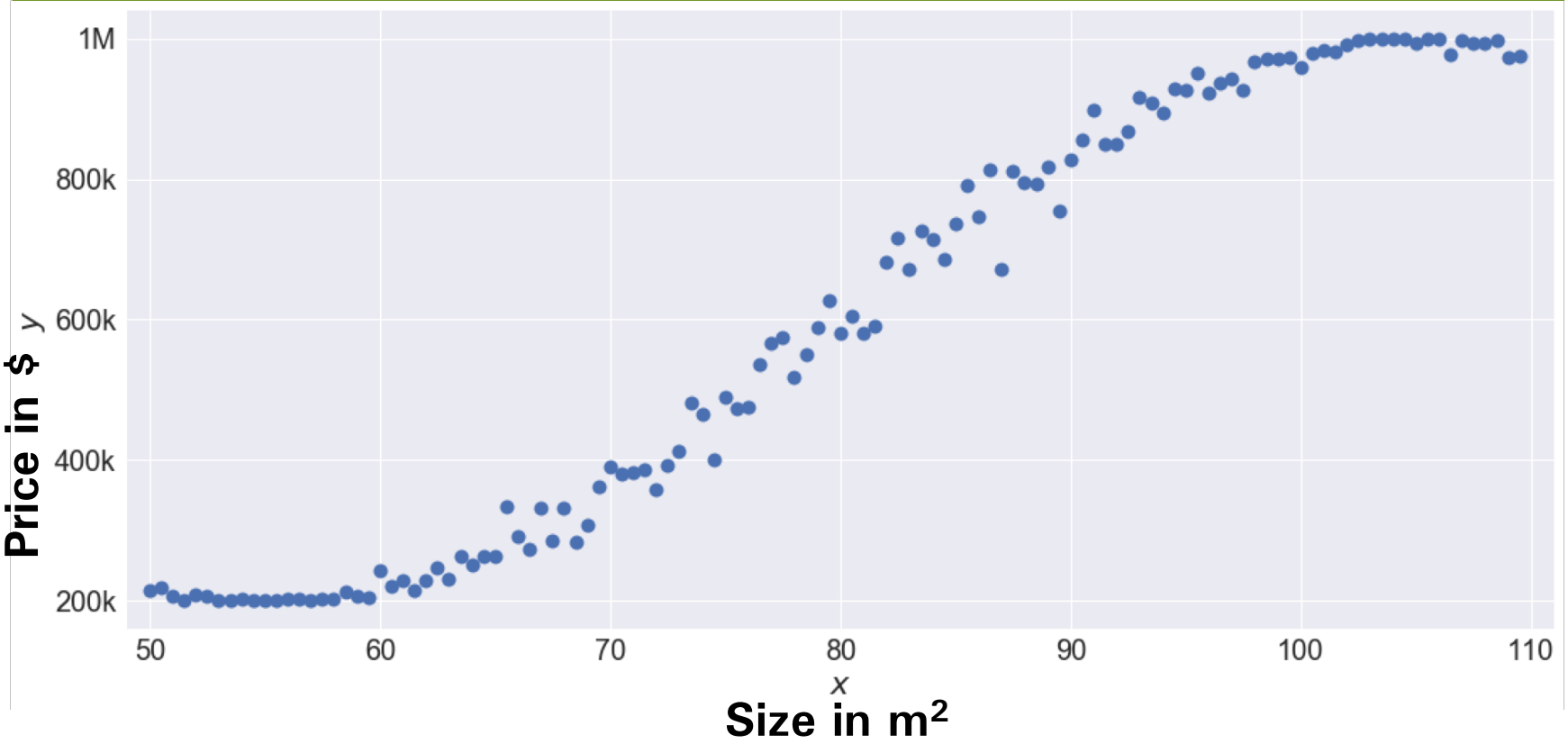
We have to select the unknown parameters b and w_1 in such a way that the loss function is minimal!

Univariate Linear Regression

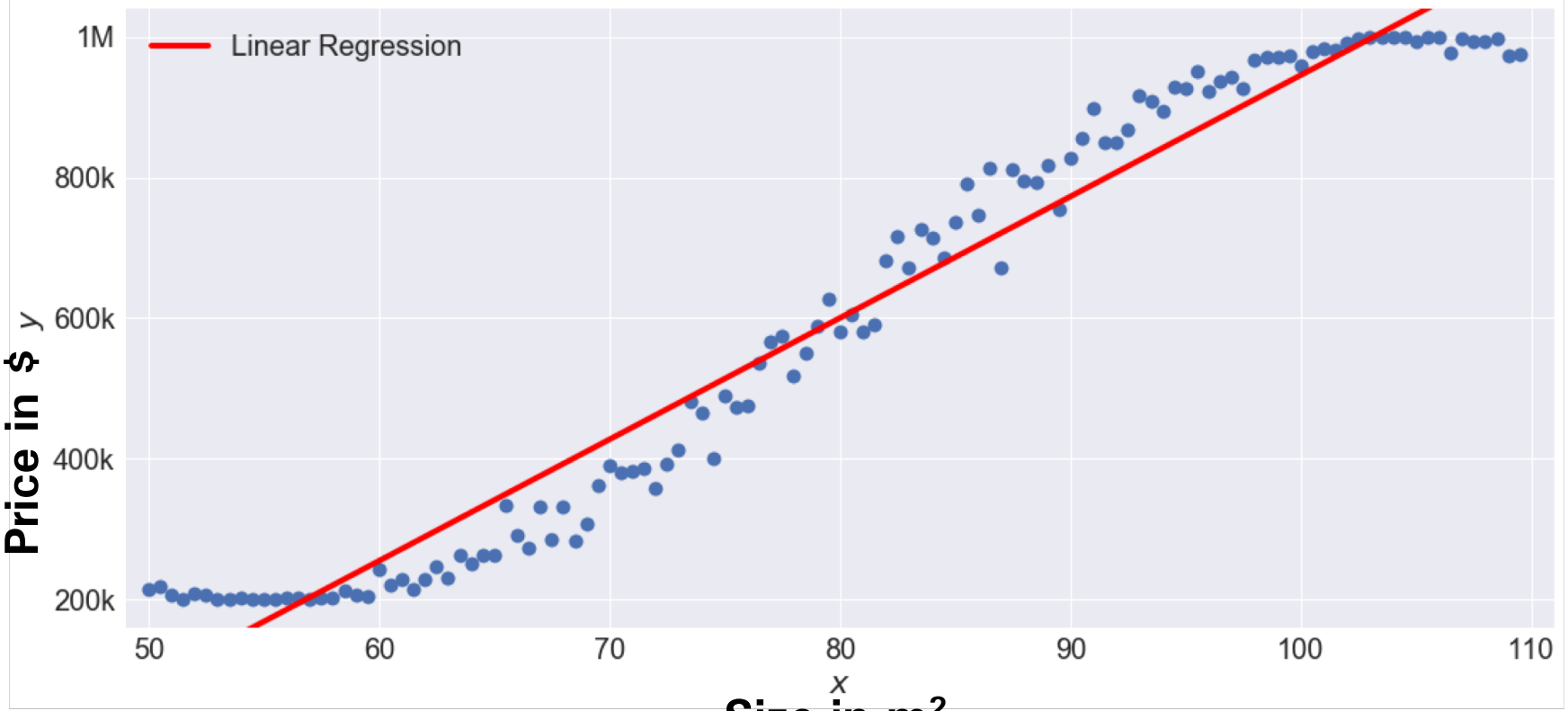
Loss Function: $\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} [y_i - (b + w_1 x_i)]^2$



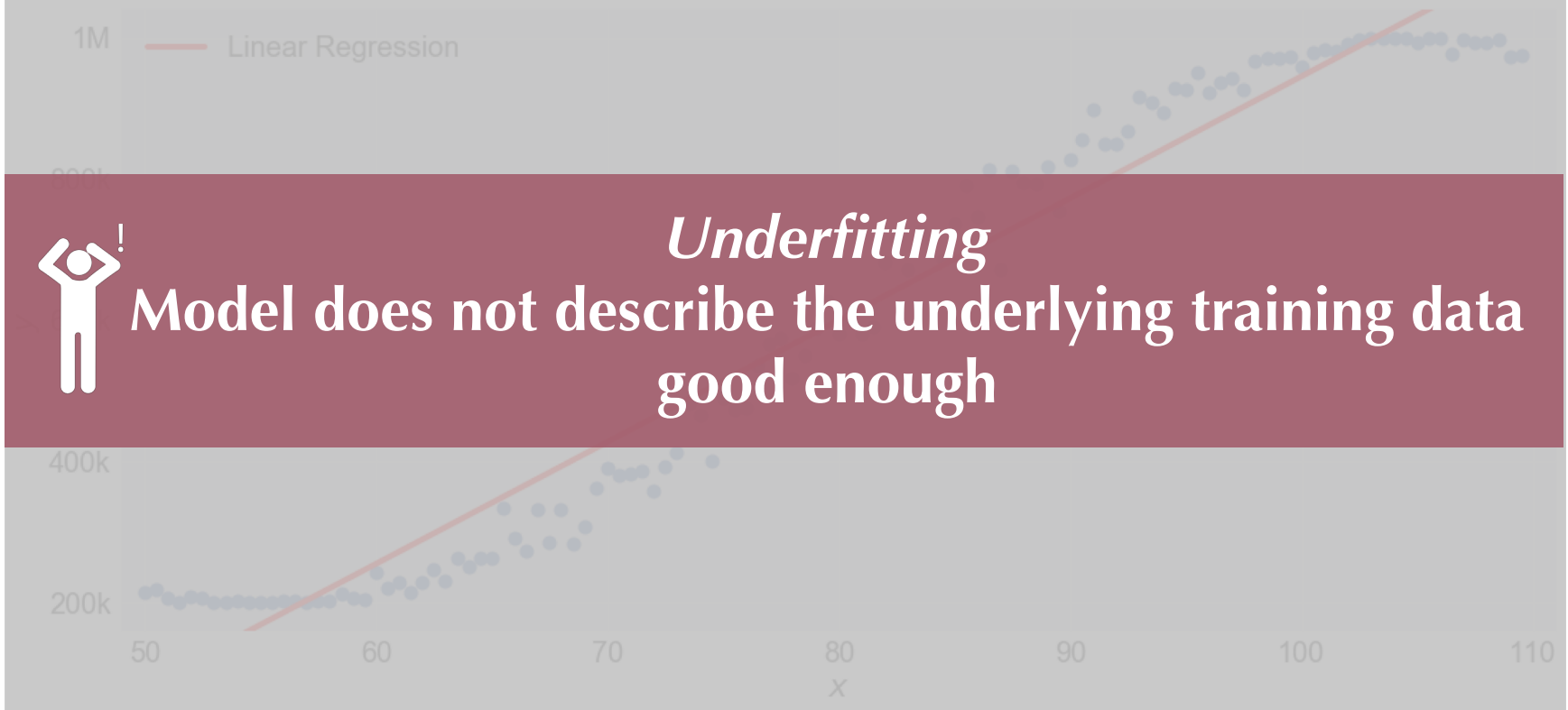
Example


1M
800k
600k
400k
200k
y
Price in $
50
60
70
80
90
100
110
x
Size in m²

Example


Linear Regression
1M
800k
600k
400k
200k
Price in $ y
50
60
70
80
90
100
110
x
Size in m²

Example



Underfitting

Model does not describe the underlying training data good enough

Multiple Linear Regression

Goal:
Model the relationship between **multiple features x** and a **single continuous target variable y** (response variable)

$$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$

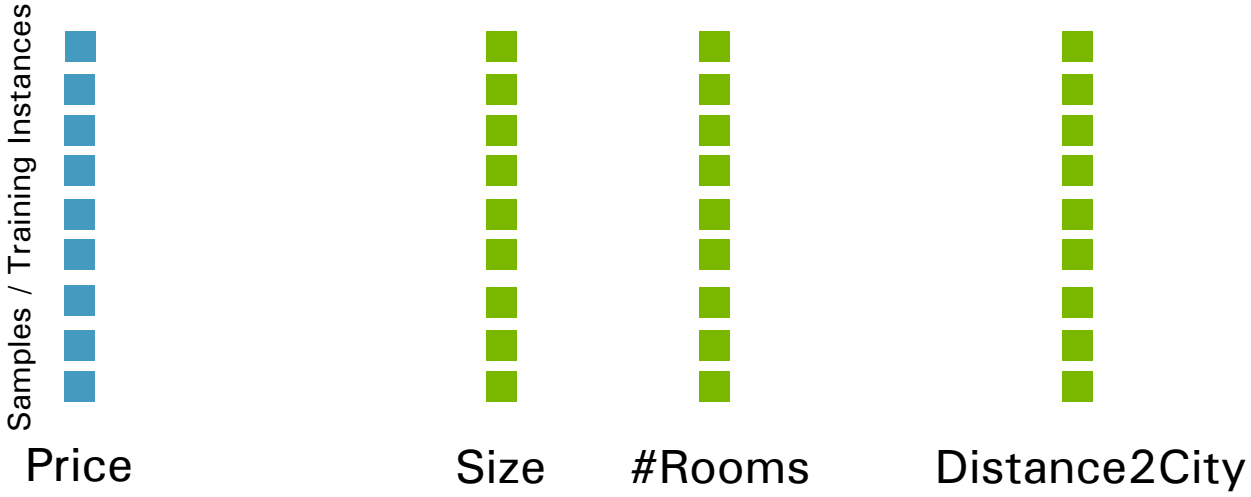
w_0 is the bias b
$x_0 = 1$

Multiple Linear Regression

Goal:

Model the relationship between **multiple features x** and a **single continuous target variable y** (response variable)

$$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$$



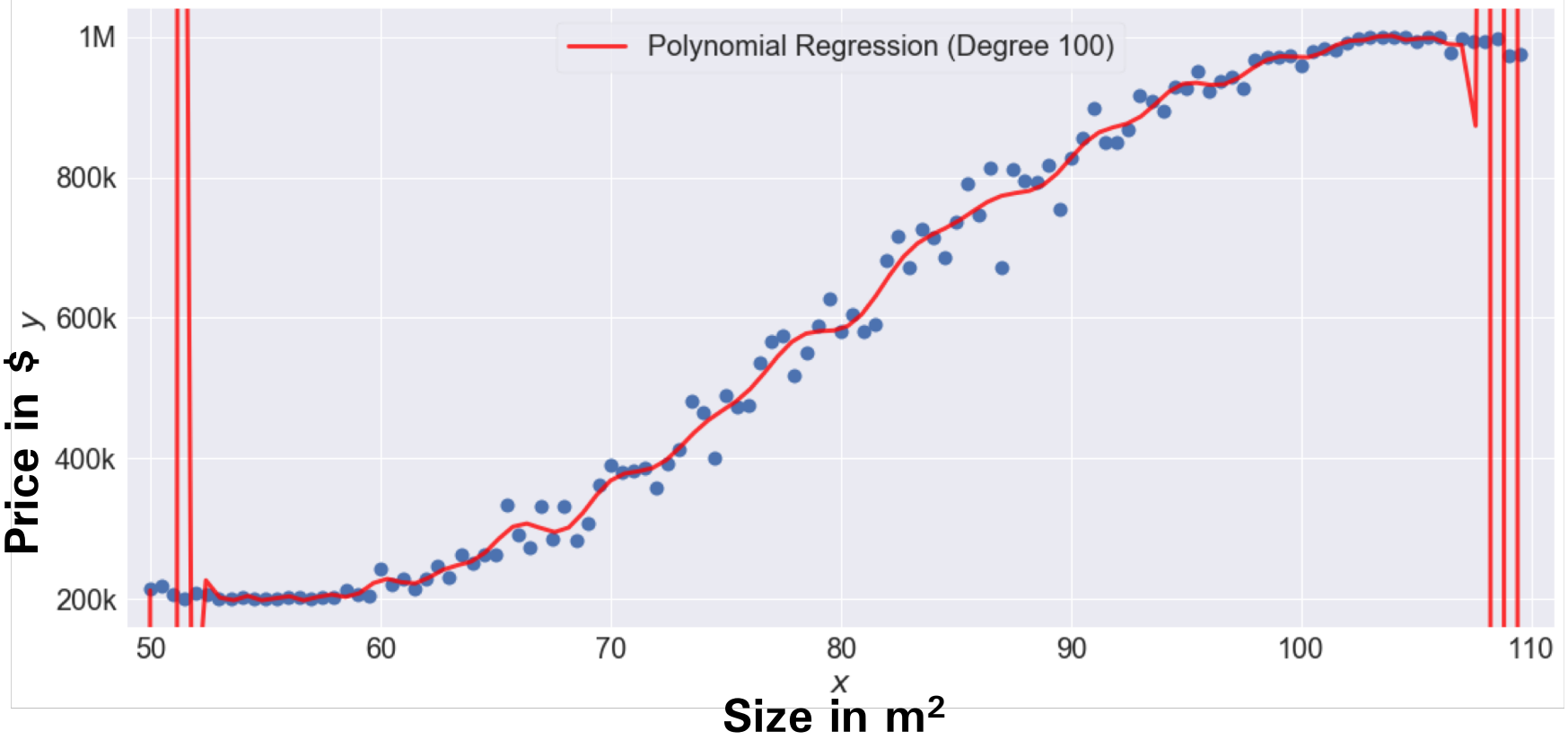
Polynomial Regression

Goal:
Compute higher order combinations of features and add them to the model to model non-linear relationships

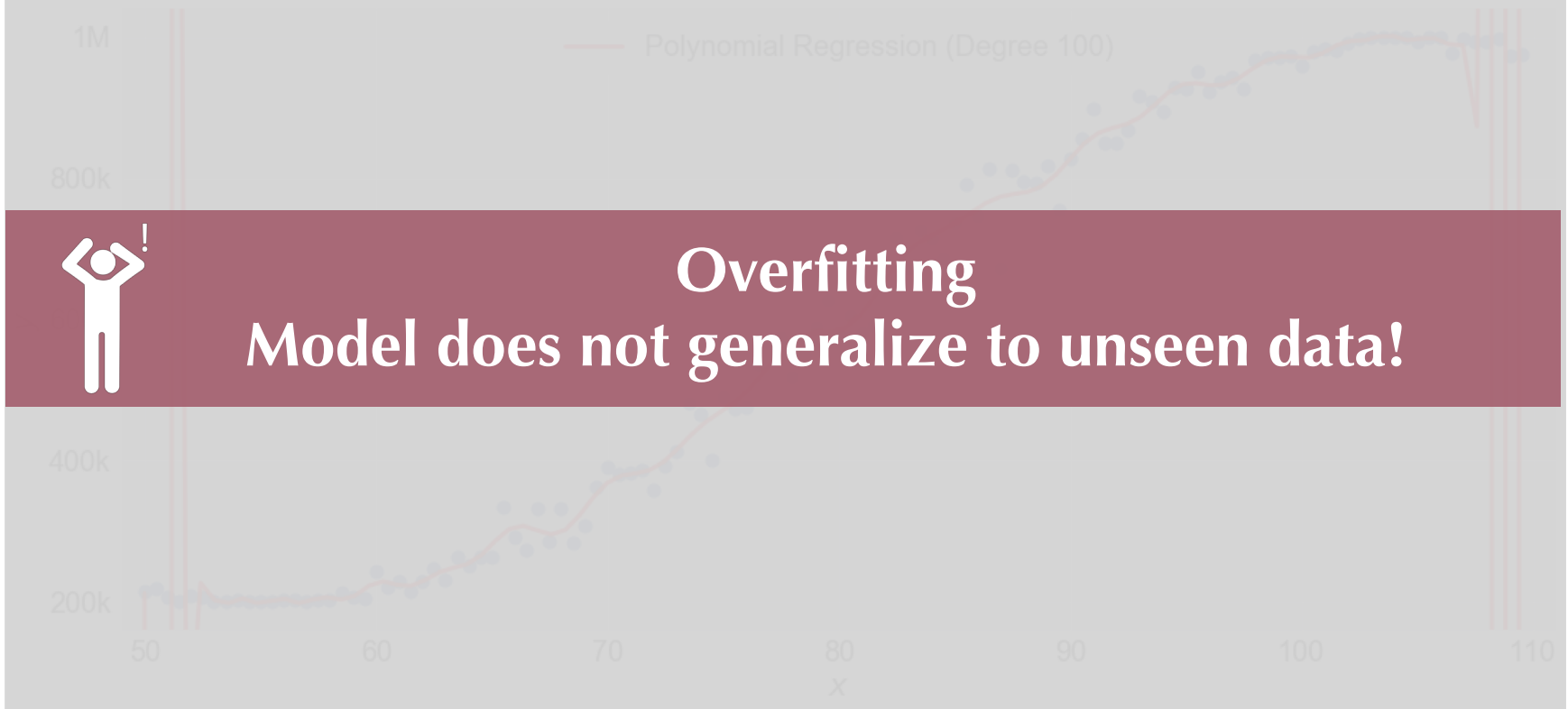
Example
Only a single feature is available (e.g. the size of a house) which shows a non-linear behavior → add higher order combinations of that feature to the model

$$y = w_0 x_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1^3 + \ldots + w_m x_1^m$$

Polynomial Regression with degree 100



Polynomial Regression with degree 100



Overfitting
Model does not generalize to unseen data!

Ridge Regression

Regression Loss:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - (b + \boldsymbol{w}^T \boldsymbol{x}_i)]^2$$

Ridge Regression

Ridge Regression Loss:

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - (b + \boldsymbol{w}^T \boldsymbol{x}_i)]^2 + \alpha\|\boldsymbol{w}\|_2^2$$

Regularization Term

α is an unknown hyperparameter

Ridge Regression

Ridge Regression Loss:

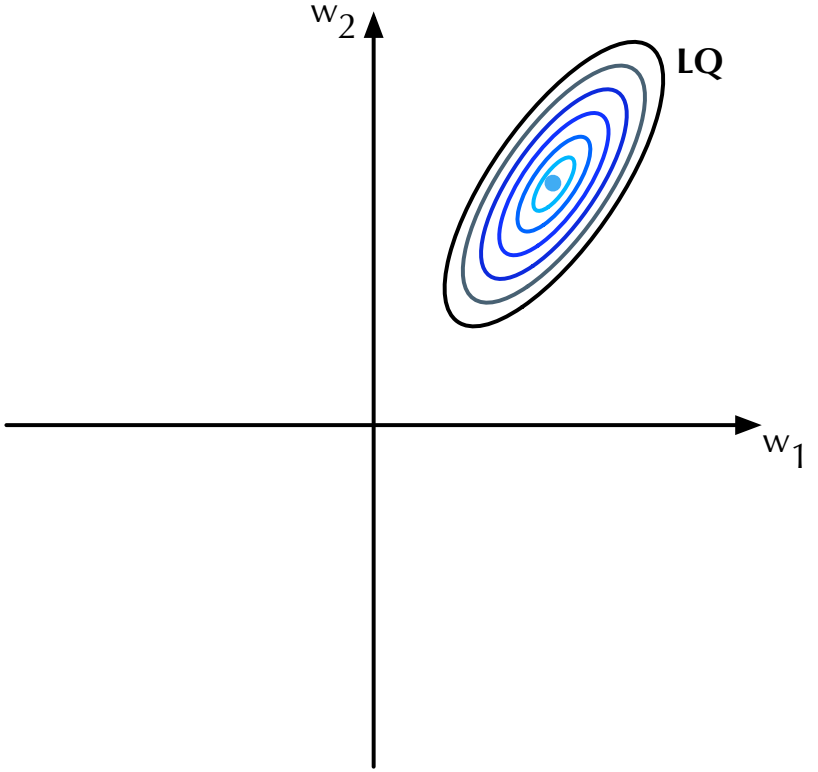
$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - (b + \boldsymbol{w}^T \boldsymbol{x}_i)]^2 + \alpha\|\boldsymbol{w}\|_2^2$$

Regularization Term

$$\|\boldsymbol{w}\|_2^2 = \sum_j w_j^2 = \boldsymbol{w}^T\boldsymbol{w}$$

Squared L2-Norm

What does the regularization term do?

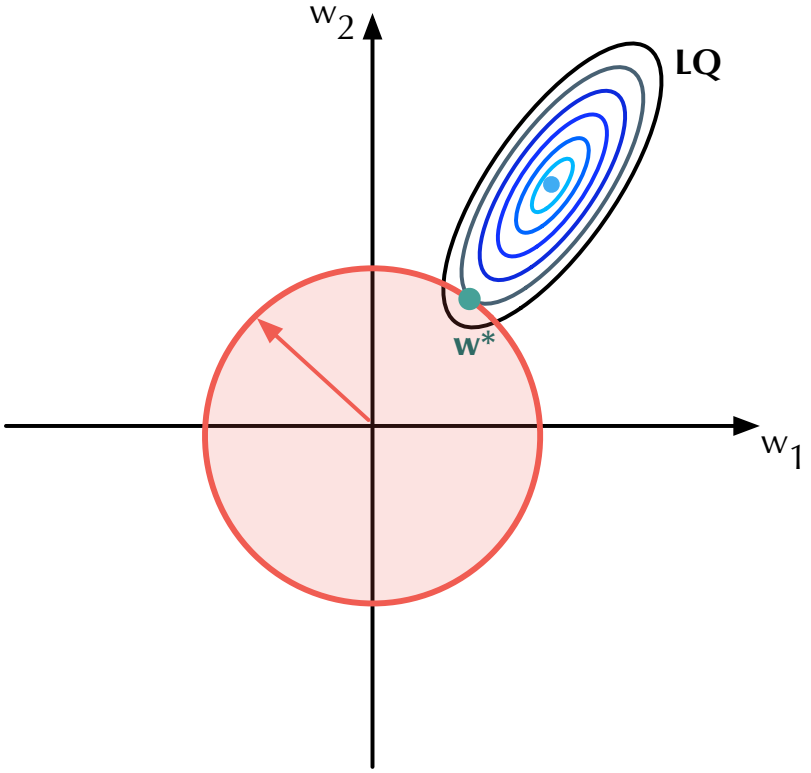


$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - (b + \boldsymbol{w}^T \boldsymbol{x}_i)]^2$$

$$\mathcal{L}(\boldsymbol{w}) = \mathrm{MSE}(\boldsymbol{w})$$

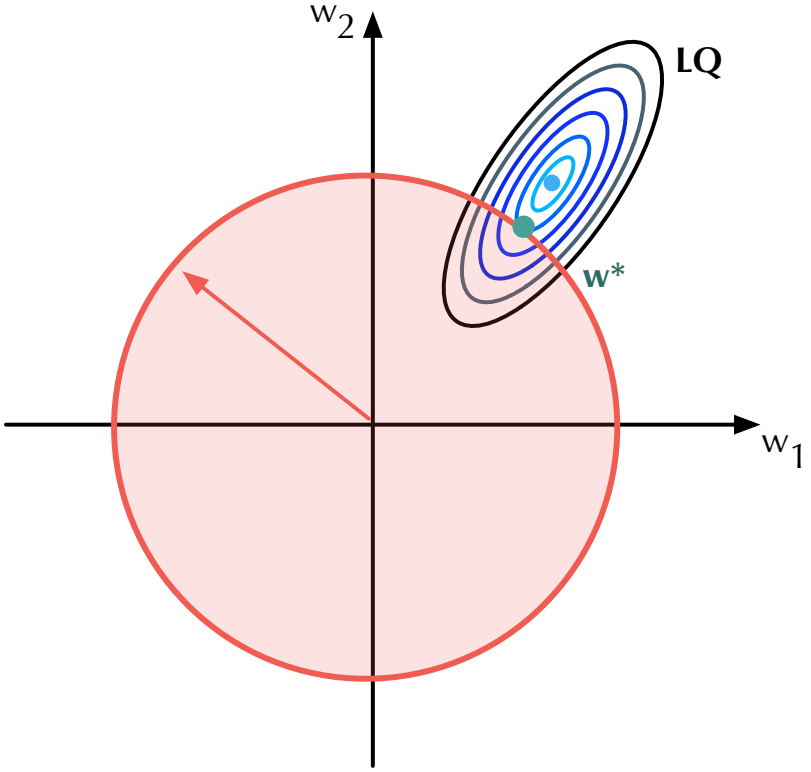
MSE := Mean Squared Error

What does the regularization term do?



$$\mathcal{L}(\boldsymbol{w}) = \mathrm{MSE}(\boldsymbol{w}) + \alpha \|\boldsymbol{w}\|_2^2$$

What does the regularization term do?

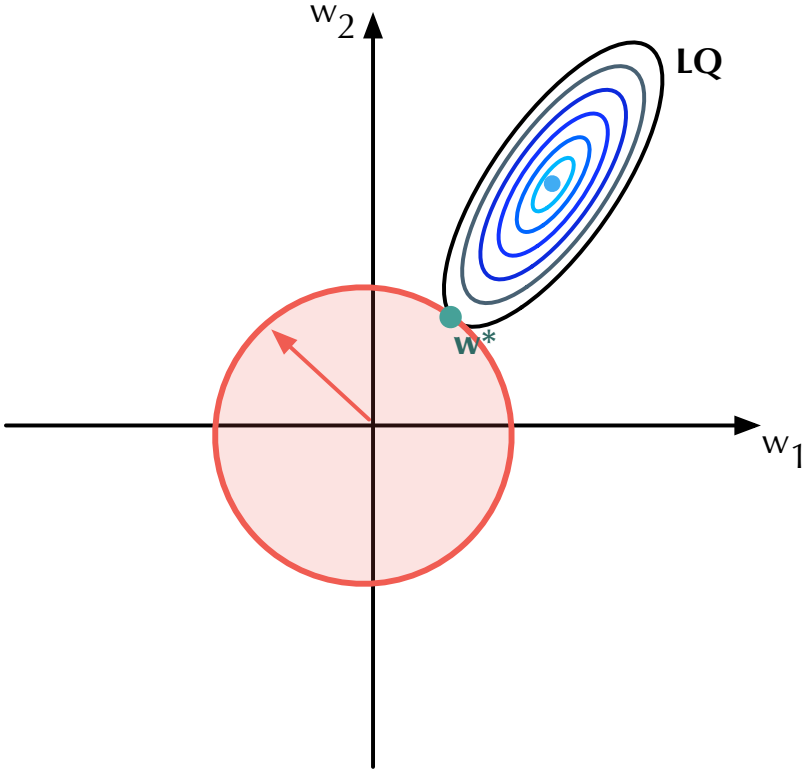


$$\mathcal{L}(\boldsymbol{w}) = \mathrm{MSE}(\boldsymbol{w}) + \alpha \|\boldsymbol{w}\|_2^2$$

Effect of small α

Regularization has a weak effect on parameters (danger of *overfitting*)

What does the regularization term do?

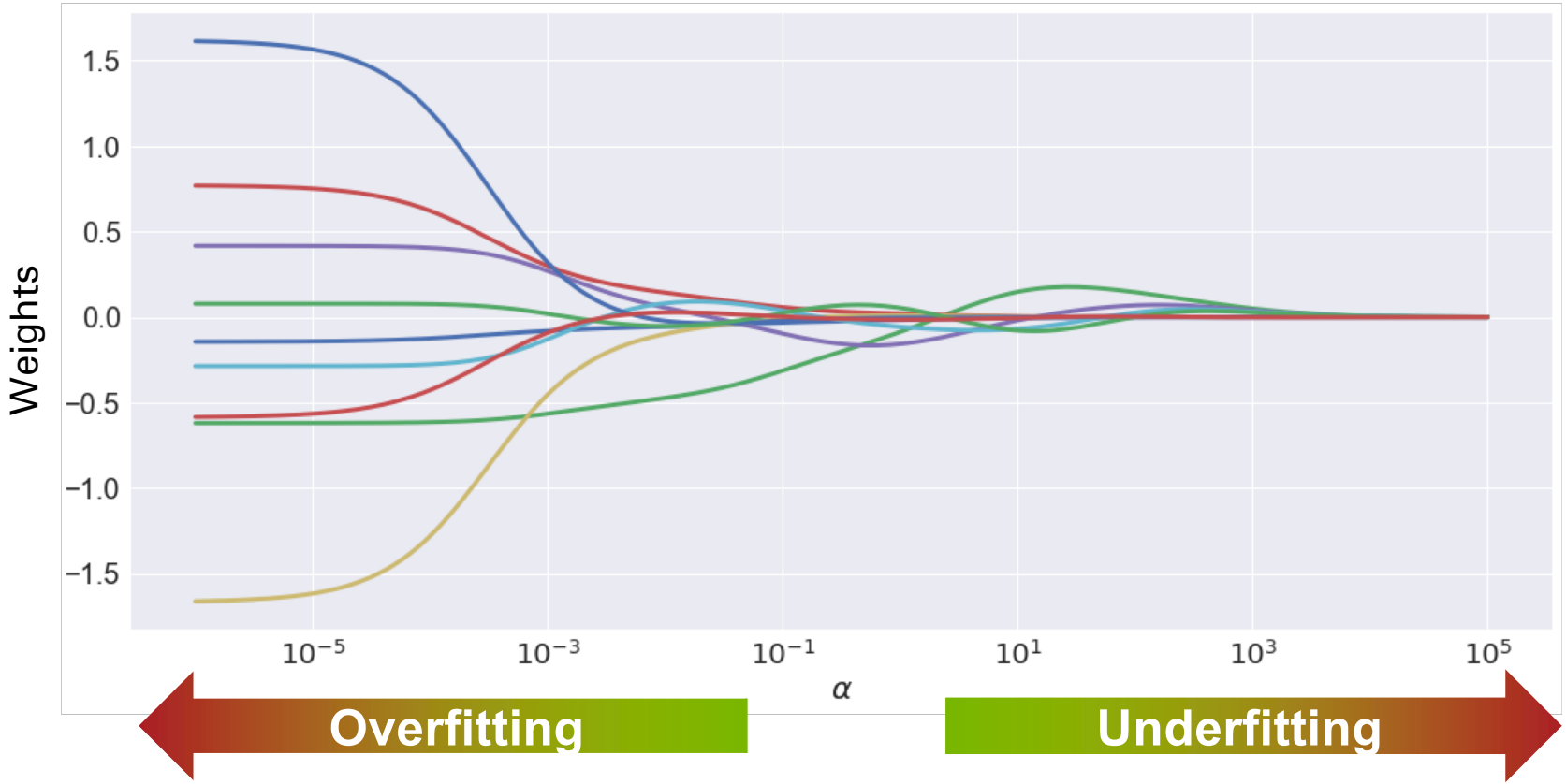


$$\mathcal{L}(\boldsymbol{w}) = \mathrm{MSE}(\boldsymbol{w}) + \alpha \|\boldsymbol{w}\|_2^2$$

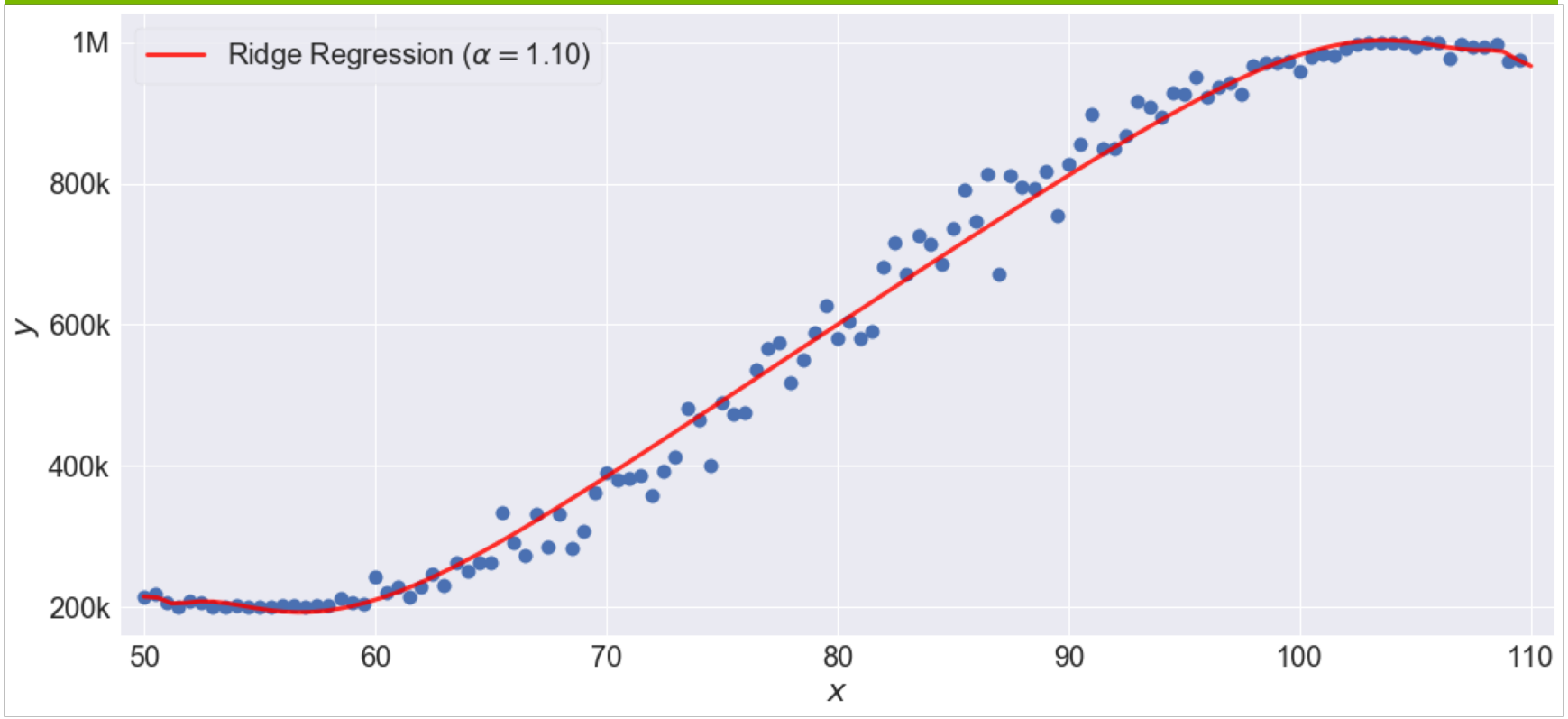
Effect of large α

Regularization has strong effect on parameters (danger of *underfitting*)

Effect of L2-Regularization Parameter on Weights

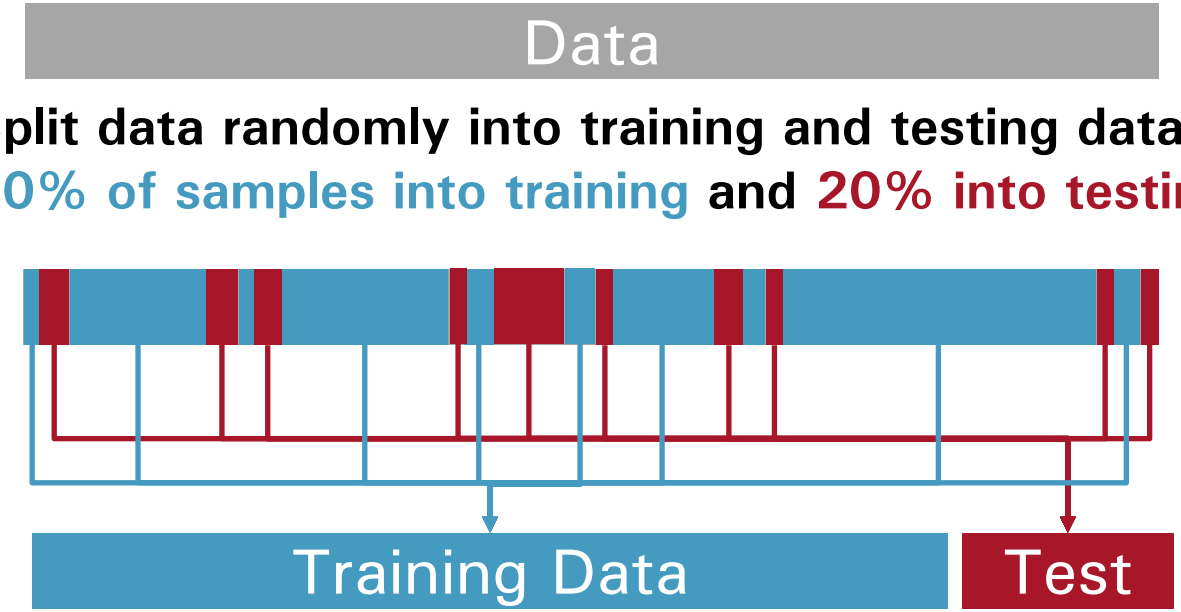


Example: Ridge Regression



How to train a model and how to find the optimal hyperparameter?

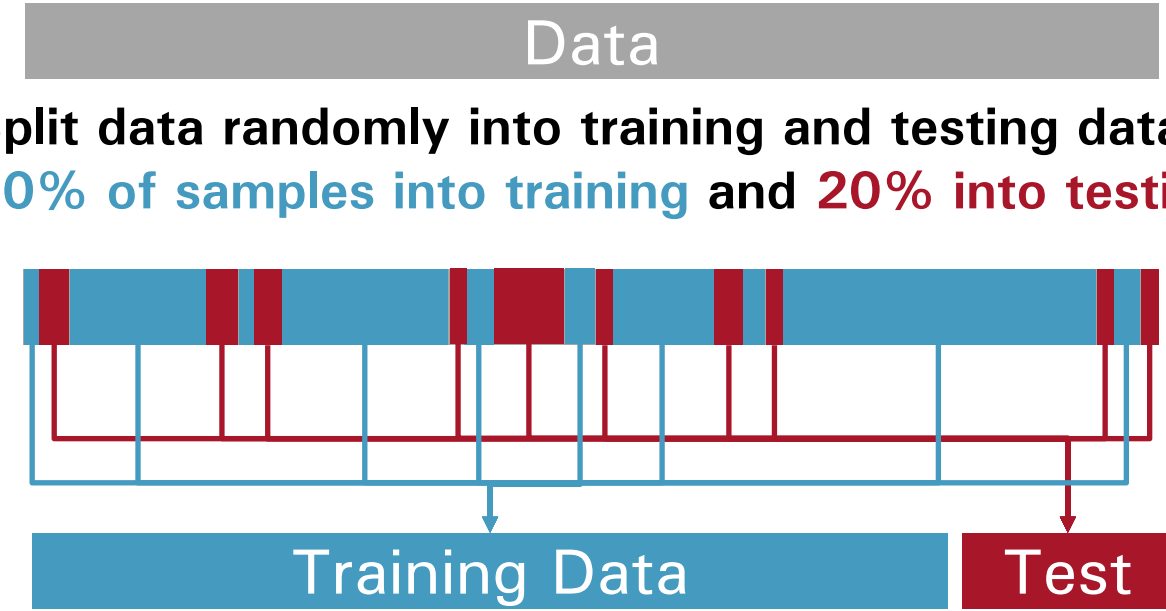
Train-Test Split



Split data randomly into training and testing data (e.g. 80% of samples into training and 20% into testing)

How to train a model and how to find the optimal hyperparameter?

Train-Test Split

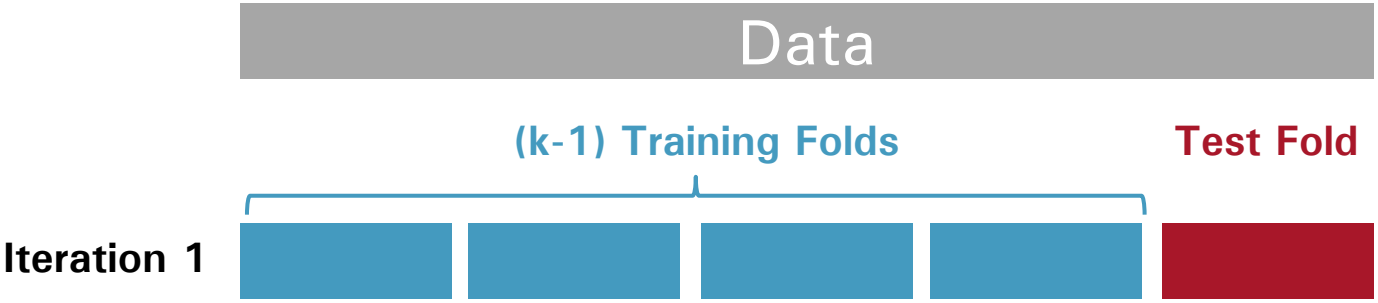


Split data randomly into training and testing data (e.g. 80% of samples into training and 20% into testing)

Not good for hyperparameter optimization (additional steps necessary)
Highly depending on the initial training and test split

How to train a model and how to find the optimal hyperparameter?

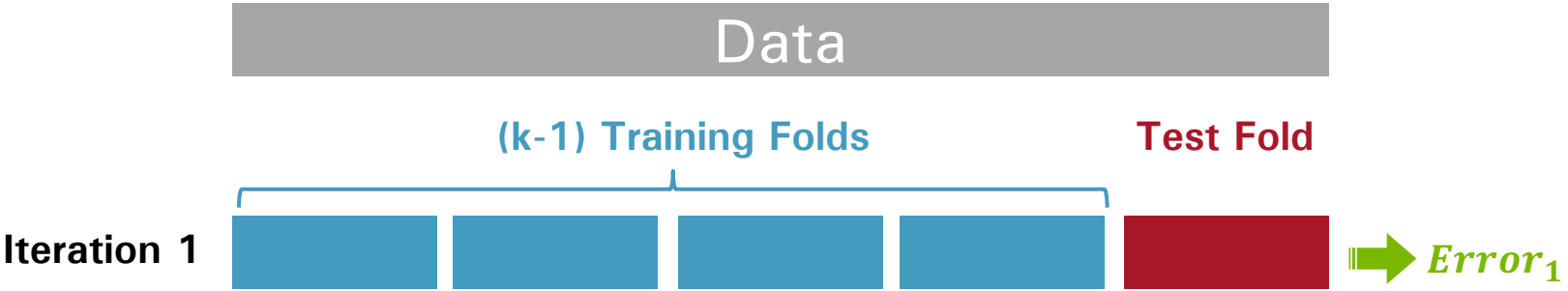
k-Fold Cross-Validation (e.g. 5-fold)



Split data randomly into k equal folds (if possible)

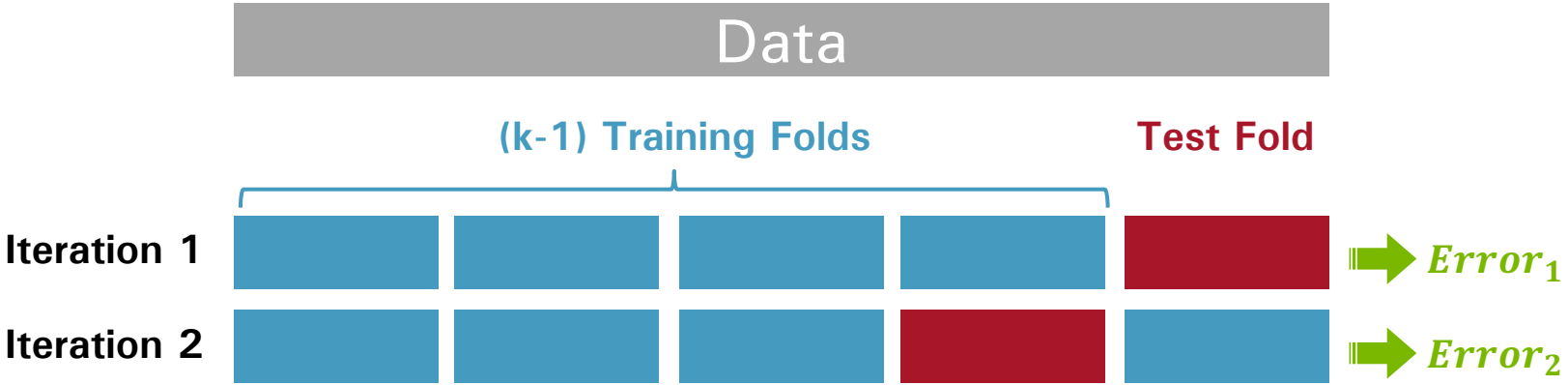
How to train a model and how to find the optimal hyperparameter?

k-Fold Cross-Validation (e.g. 5-fold)



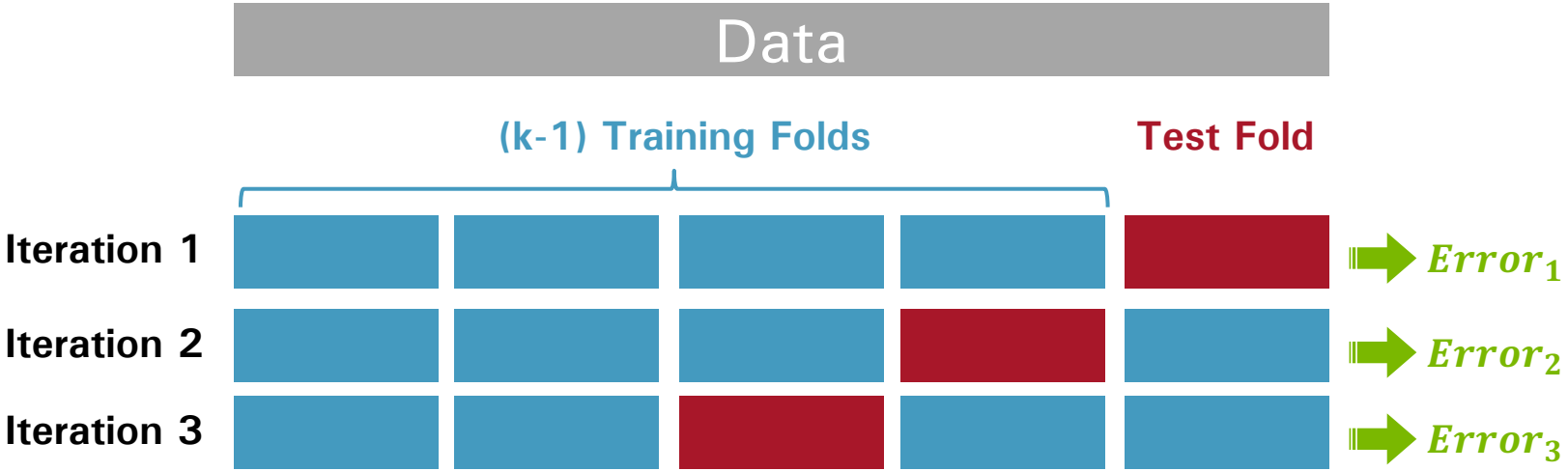
How to train a model and how to find the optimal hyperparameter?

k-Fold Cross-Validation (e.g. 5-fold)



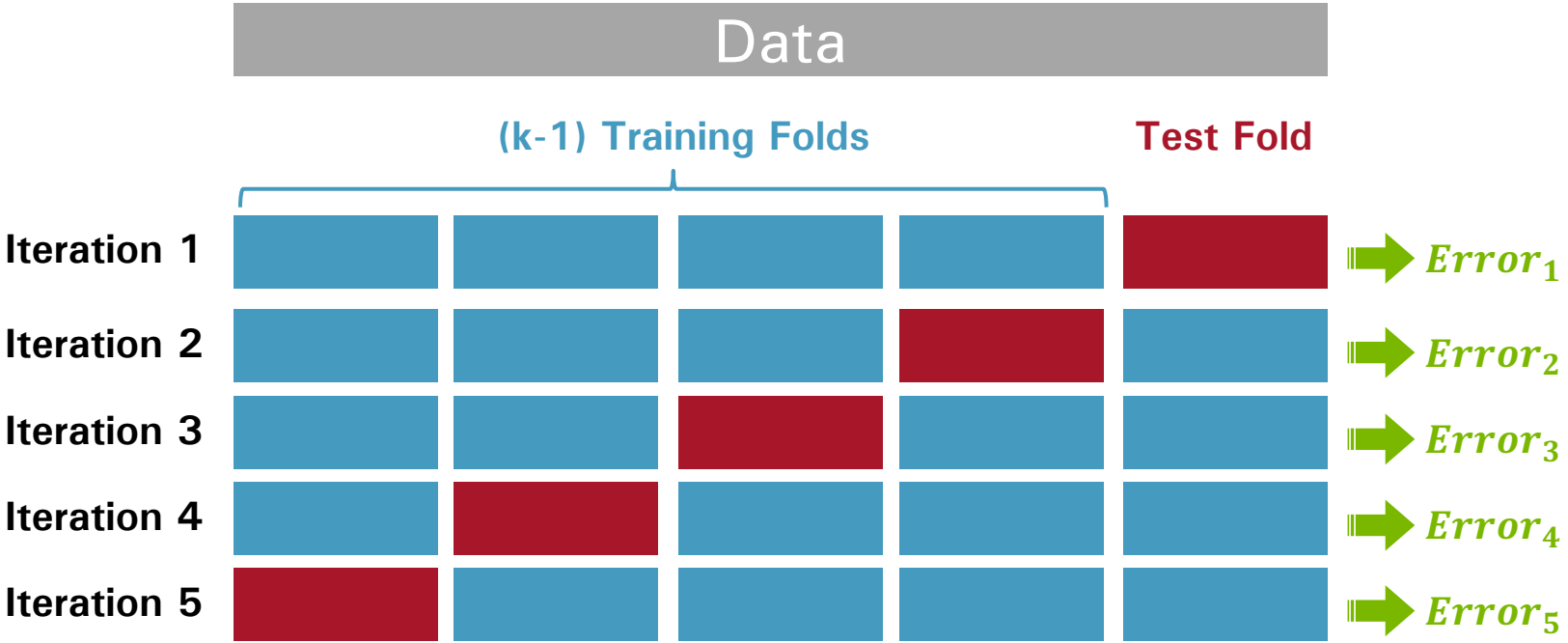
How to train a model and how to find the optimal hyperparameter?

k-Fold Cross-Validation (e.g. 5-fold)



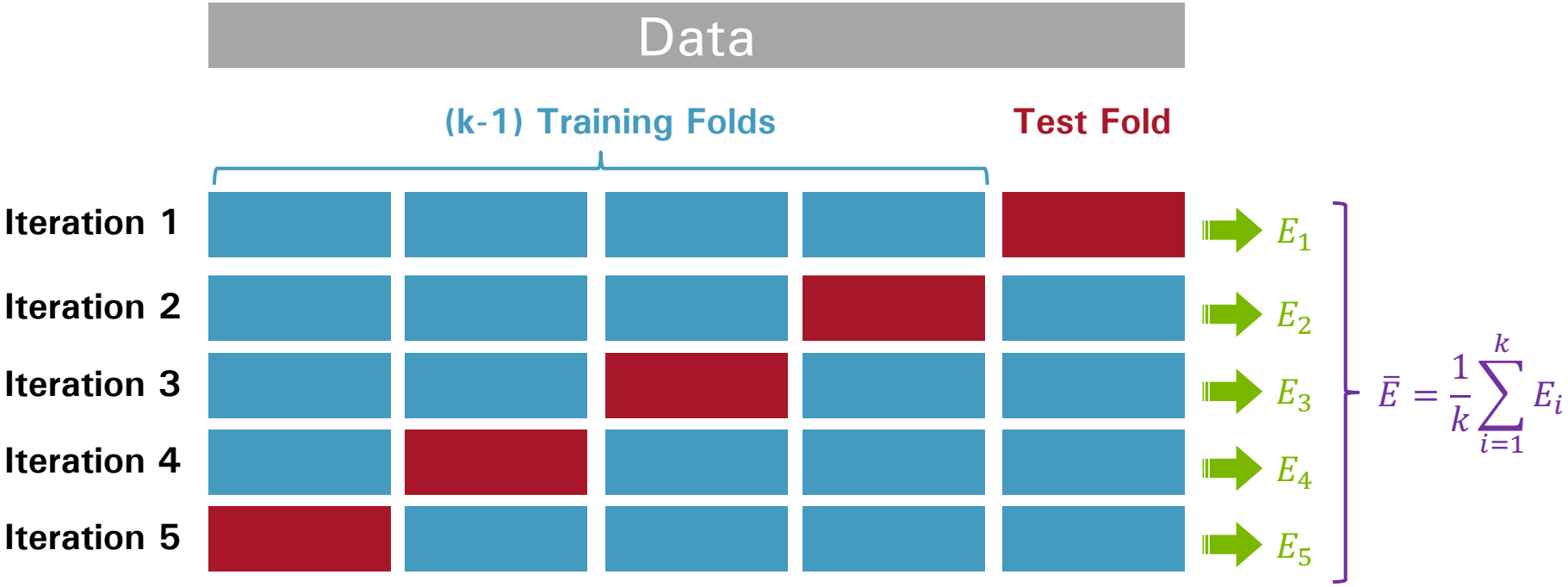
How to train a model and how to find the optimal hyperparameter?

k-Fold Cross-Validation (e.g. 5-fold)



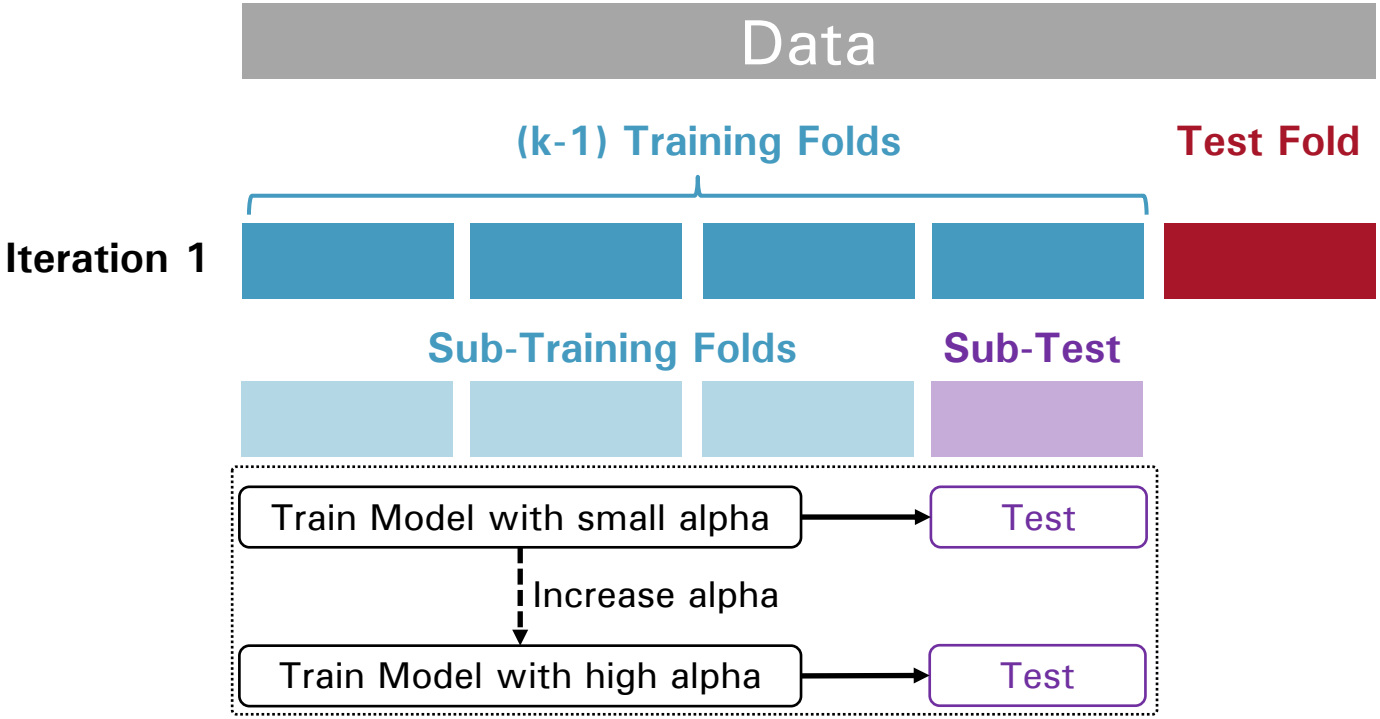
How to train a model and how to find the optimal hyperparameter?

k-Fold Cross-Validation (e.g. 5-fold)



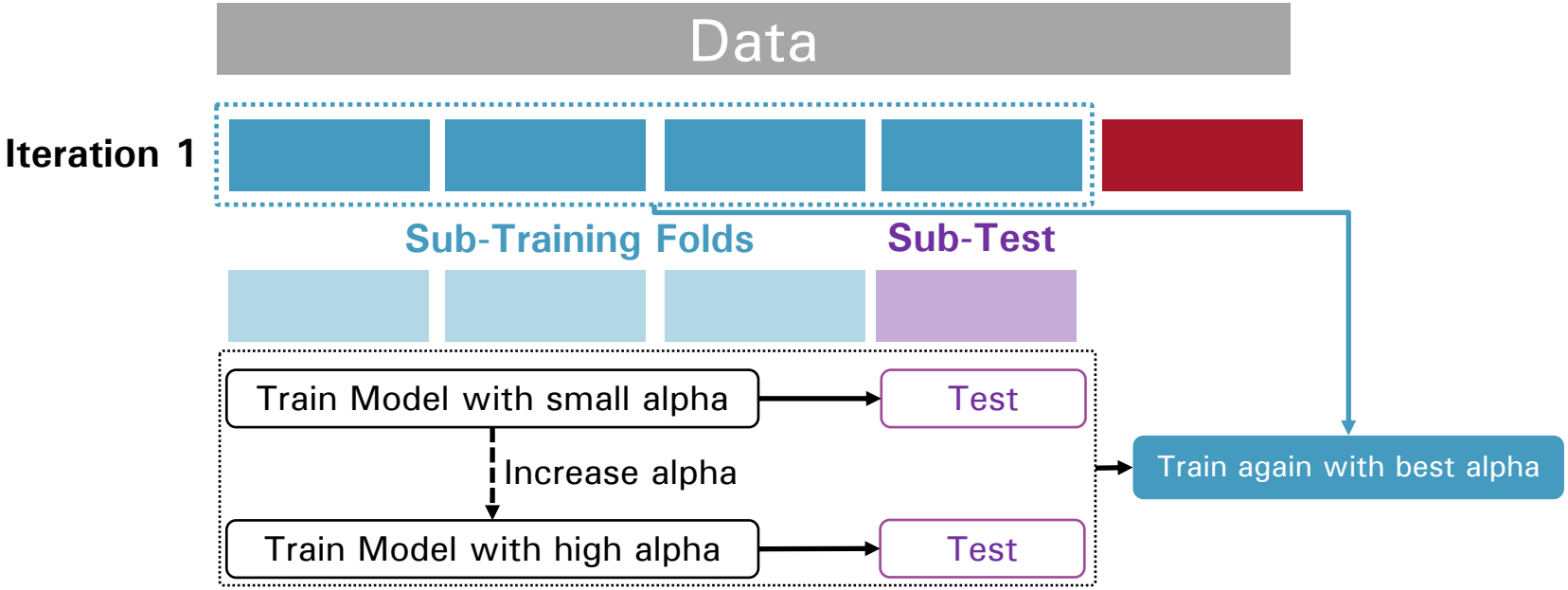
How to train a model and how to find the optimal hyperparameter?

Nested K-Fold Cross-Validation for Hyperparameter Optimization



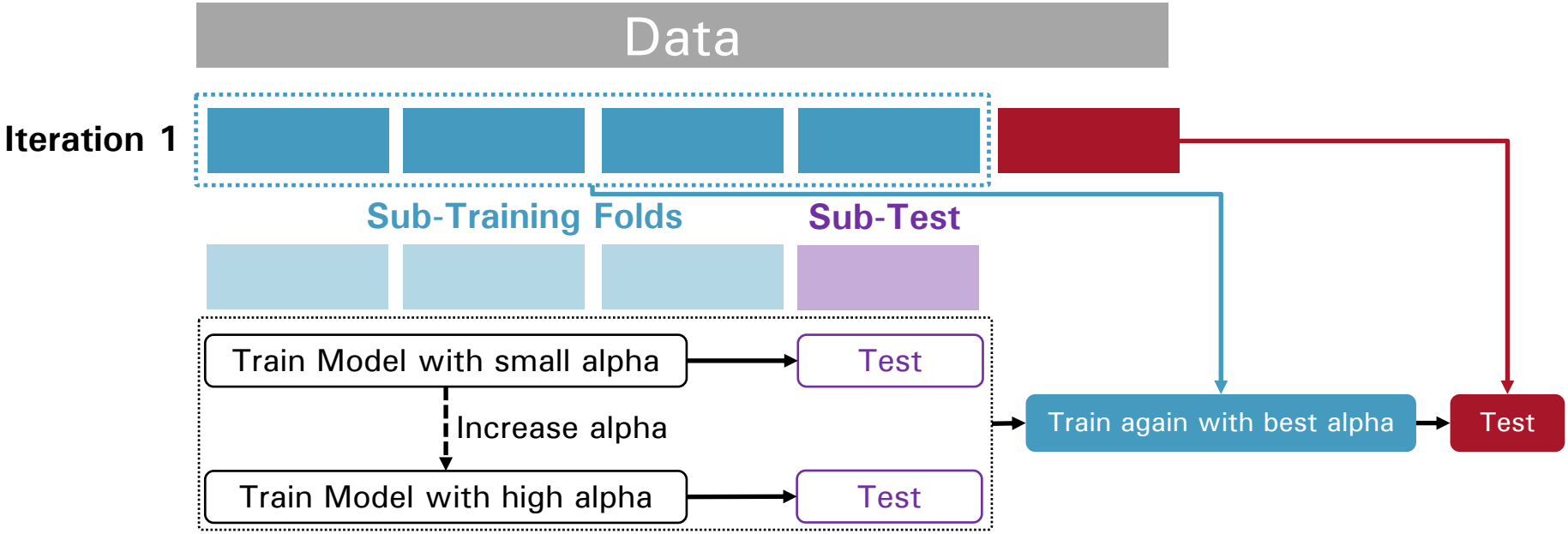
How to train a model and how to find the optimal hyperparameter?

Nested K-Fold Cross-Validation for Hyperparameter Optimization



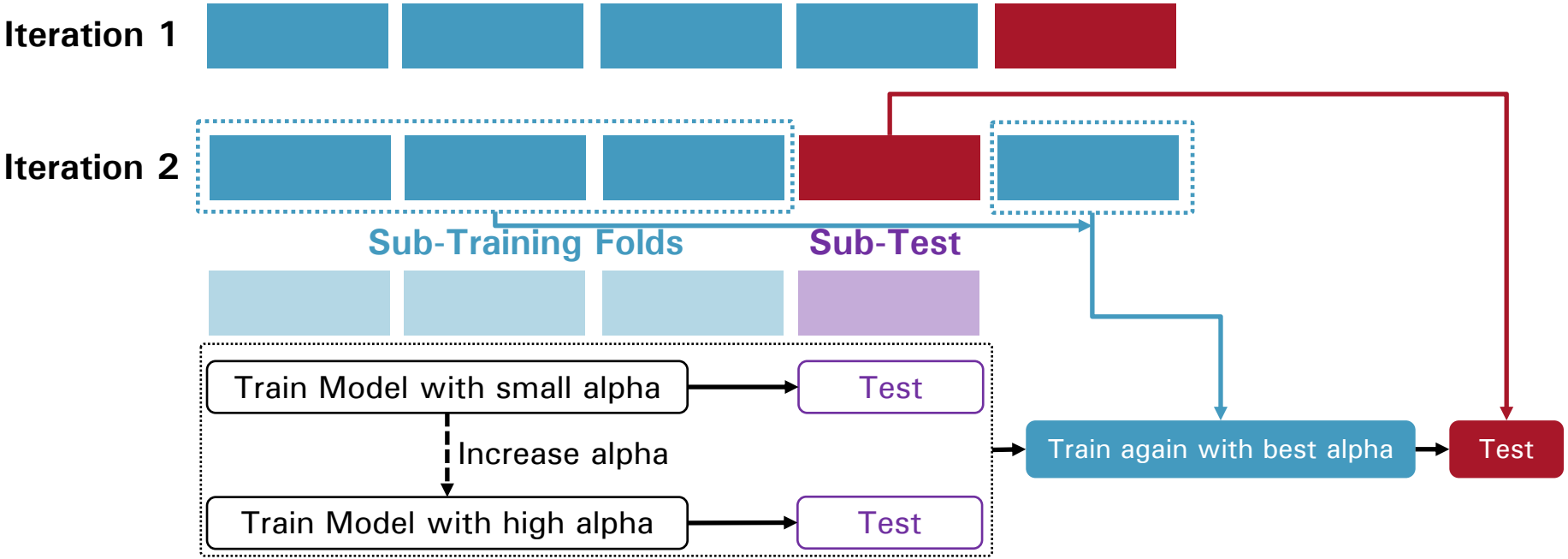
How to train a model and how to find the optimal hyperparameter?

Nested K-Fold Cross-Validation for Hyperparameter Optimization

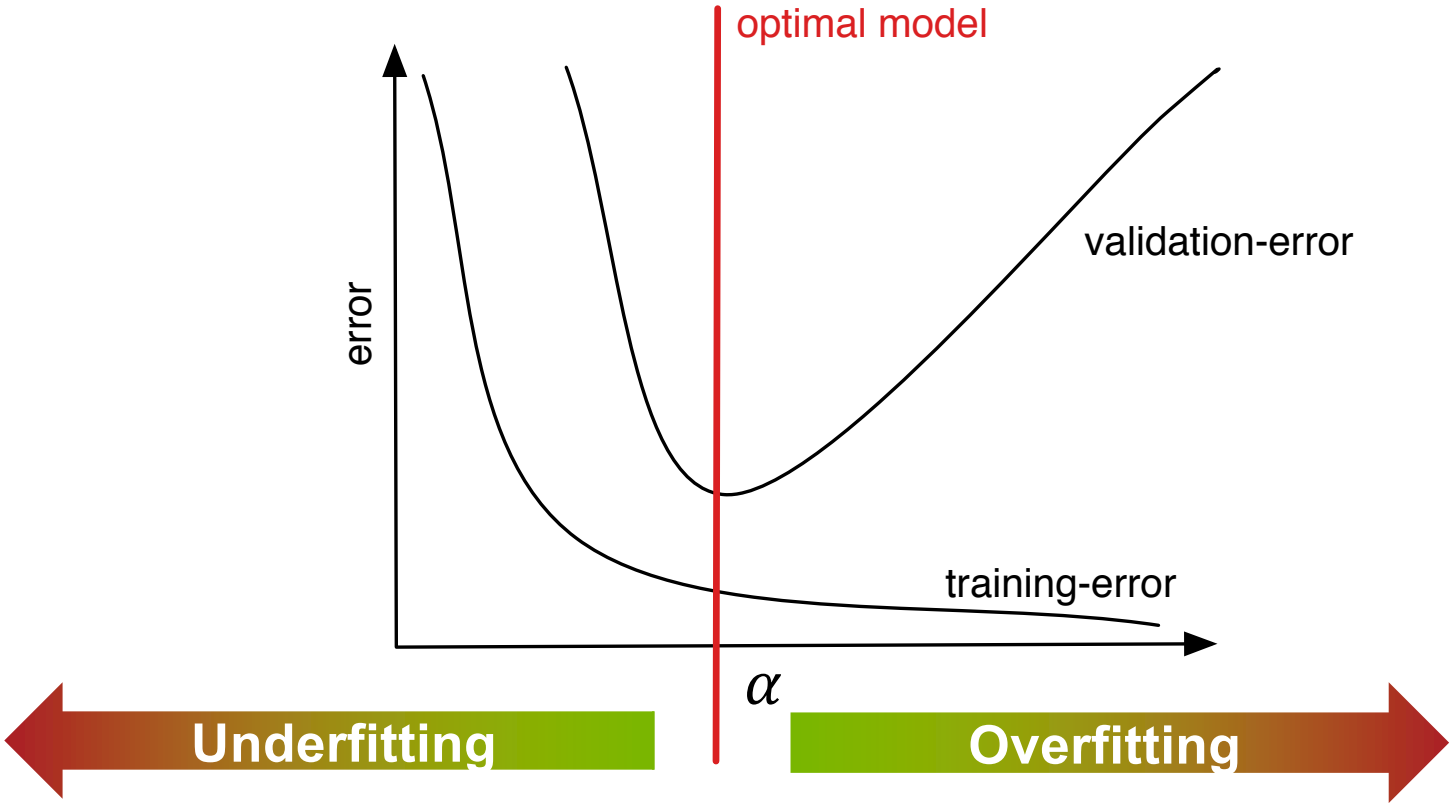


How to train a model and how to find the optimal hyperparameter?

Nested K-Fold Cross-Validation for Hyperparameter Optimization



Bias-Variance Tradeoff (Model Complexity)



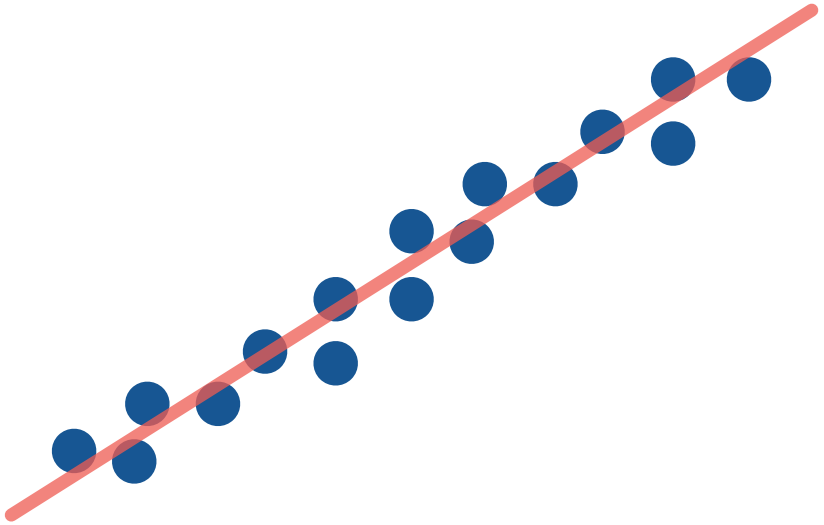
Summary Regression

- **Regression** is used for continuous target variables y
- To many features can lead to overfitting
- **L2-Regularization** can be used to avoid overfitting
- **Regression** is a supervised learning method
- **Cross-validation** can be used to validate the performance of the model and its generalization abilities
- **Nested Cross-validation** can be used to find the optimal hyperparameter

Classification

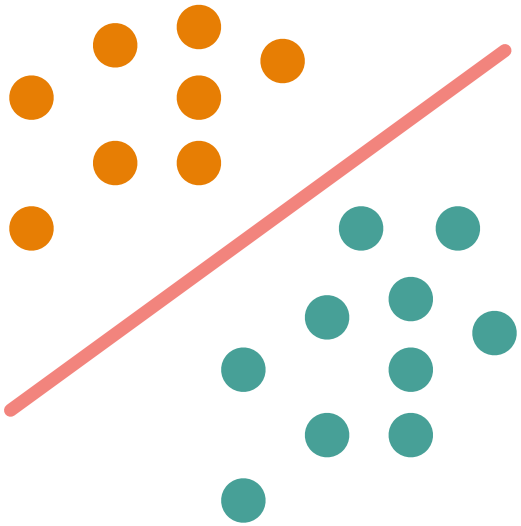
Regression

$\boldsymbol{y} \in \mathbb{R}^n$: *Target*

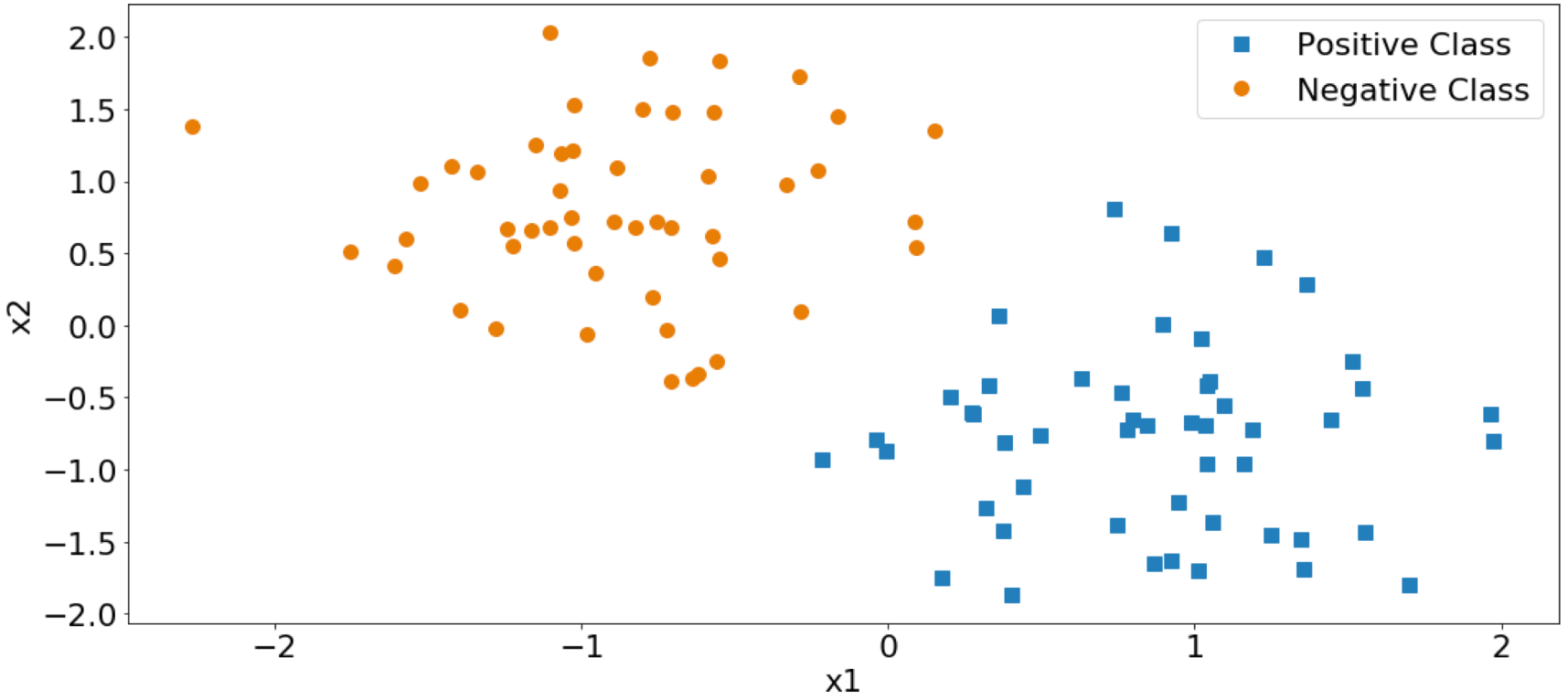


Classification

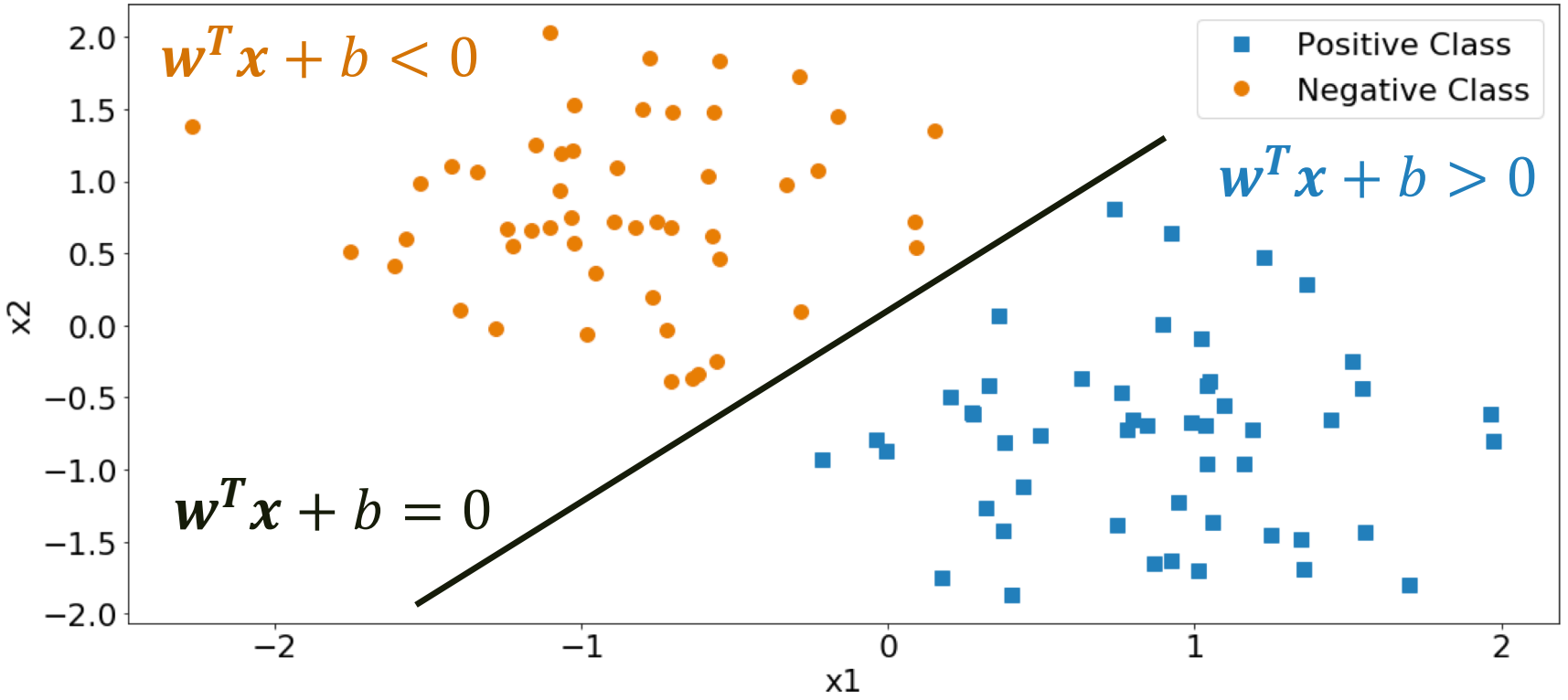
$\boldsymbol{y} \in \{-1,1\}^n$: *Label*



Using a decision function for classification



Using a decision function for classification



Using a decision function for classification

Decision Function

$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}^T \boldsymbol{x} + b)$$

Sign Function

$$sgn(z) = \begin{cases} +1 & \text{falls} \quad z > 0 \\ 0 & \text{falls} \quad z = 0 \\ -1 & \text{falls} \quad z < 0 \end{cases}$$

Quiz

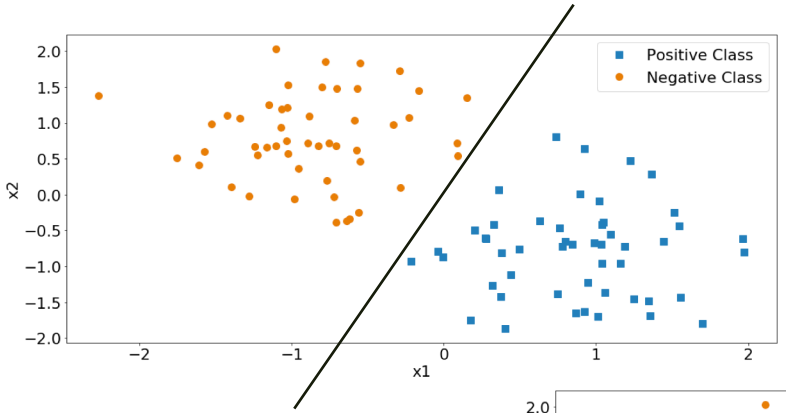
http://pingo.upb.de

ID: 781448

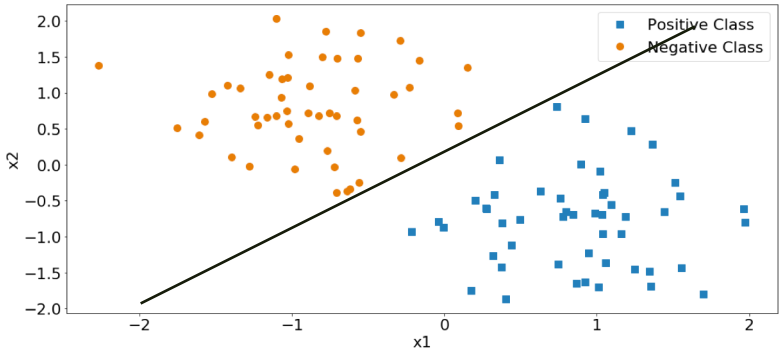
Quiz

What is the correct decision function?

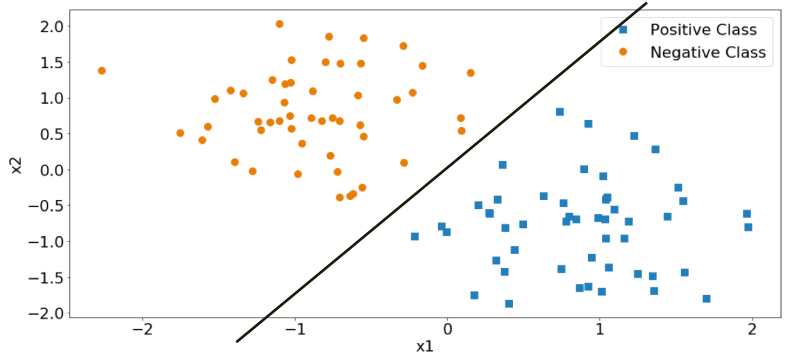
A



B



C

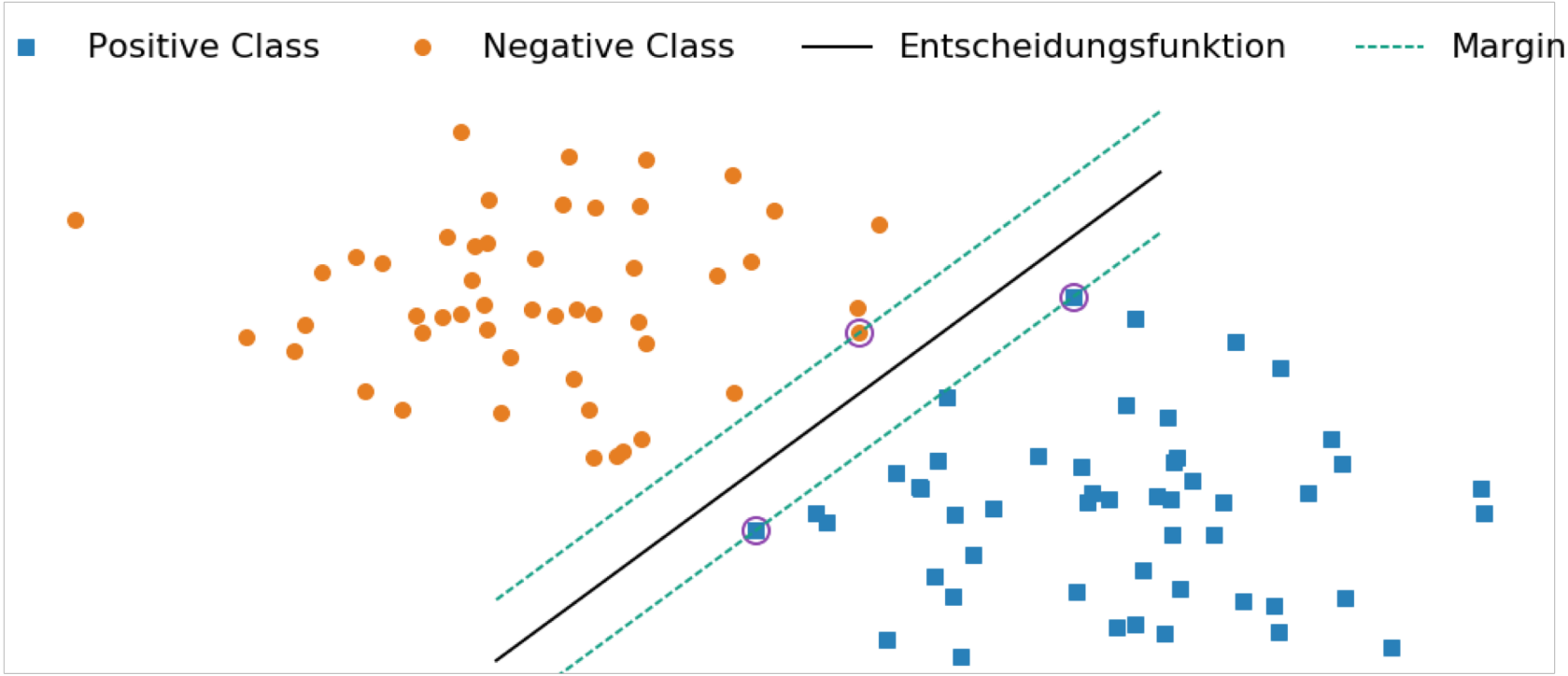


Quiz

What is the correct decision function?

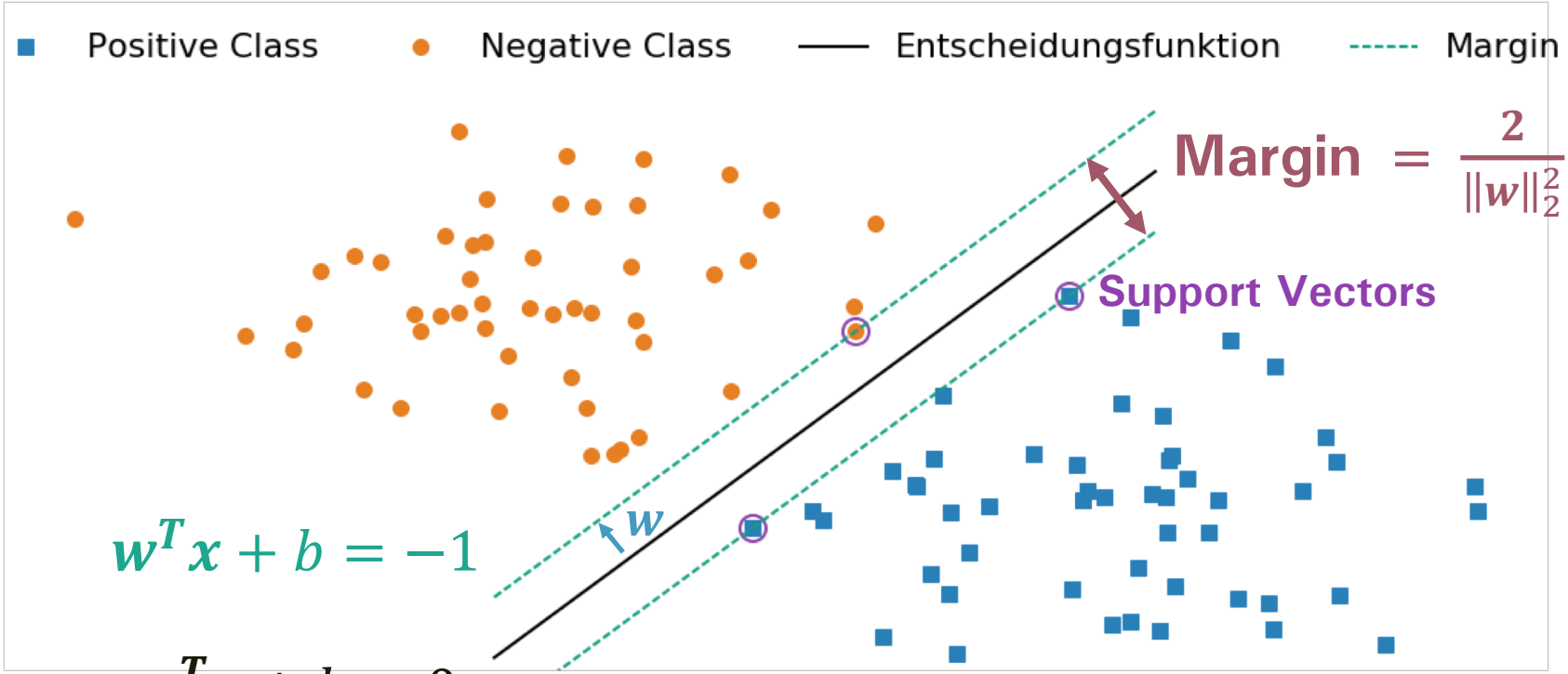
There are infinitely many decision functions, if two classes are separable with a linear decision function!

Hard-Margin Support Vector Machine (SVM)



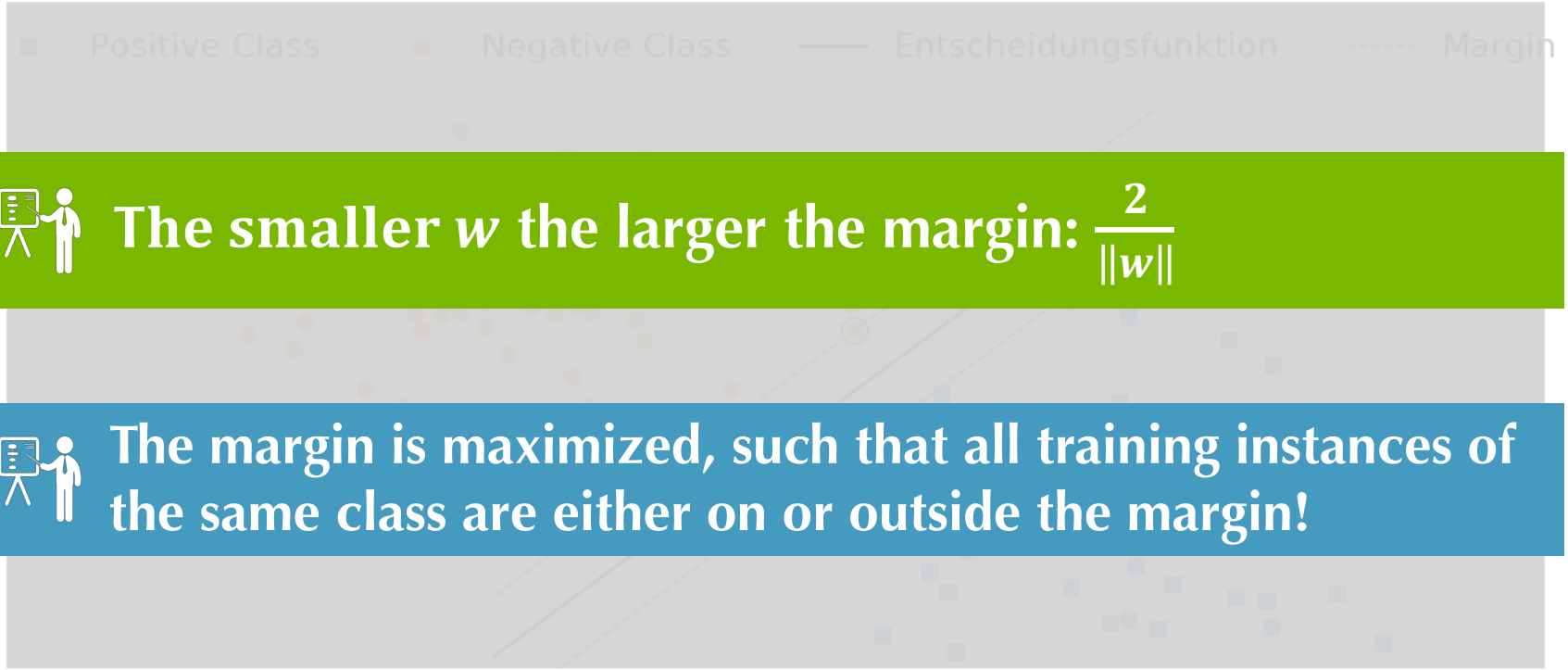
Vapnik und Chervonenkis, 1974

Hard-Margin Support Vector Machine (SVM)



Vapnik und Chervonenkis, 1974

Hard-Margin Support Vector Machine (SVM)

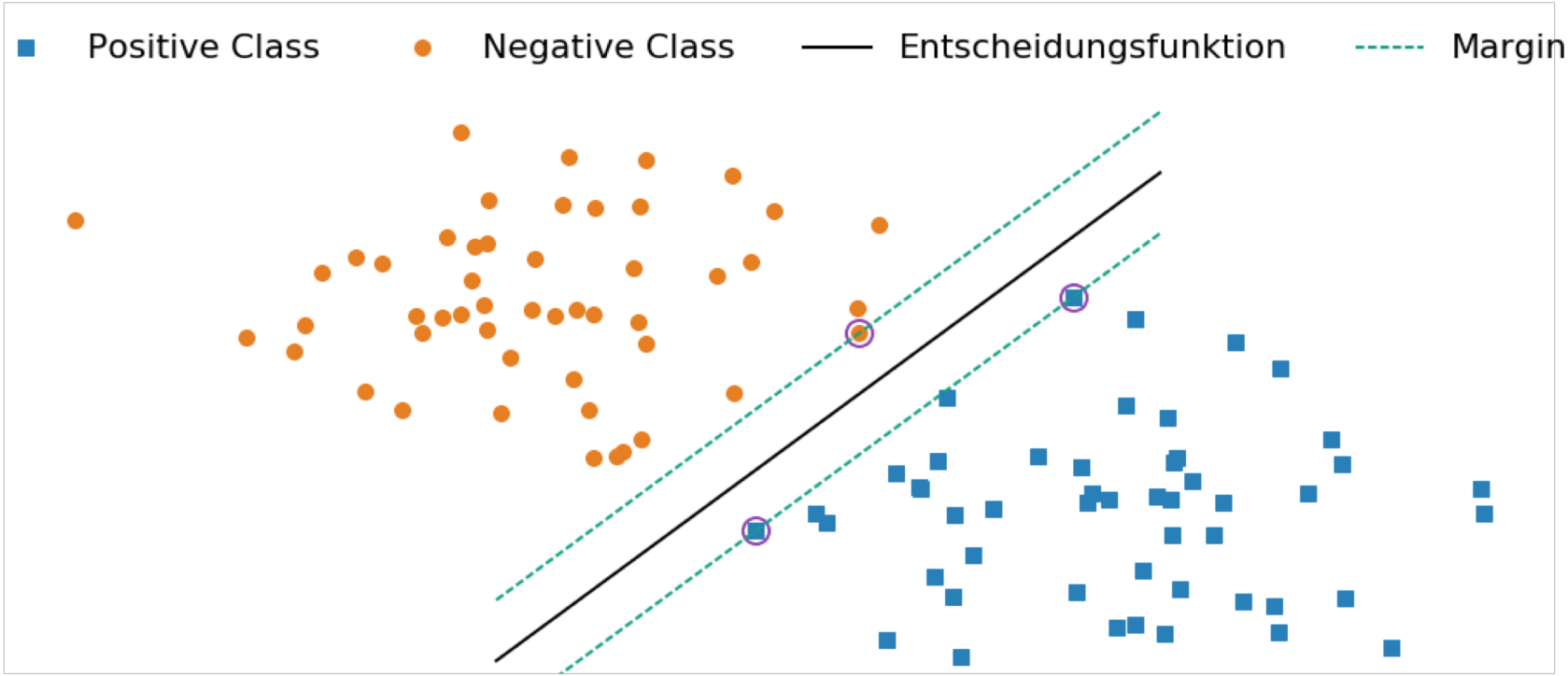


The smaller w the larger the margin: $\frac{2}{\|w\|}$

The margin is maximized, such that all training instances of the same class are either on or outside the margin!

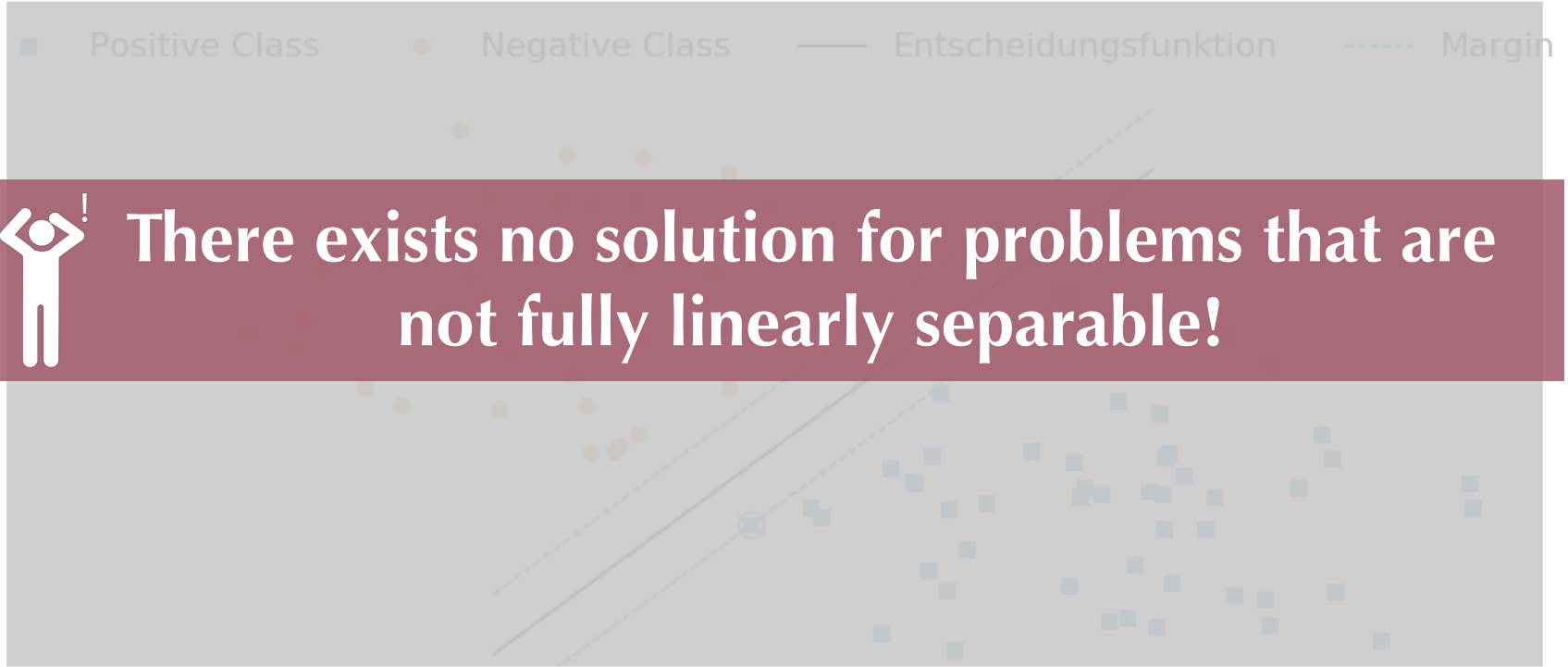
Hard-Margin Support Vector Machine (SVM)

Might there be any problems when looking at this plot?



Hard-Margin Support Vector Machine (SVM)

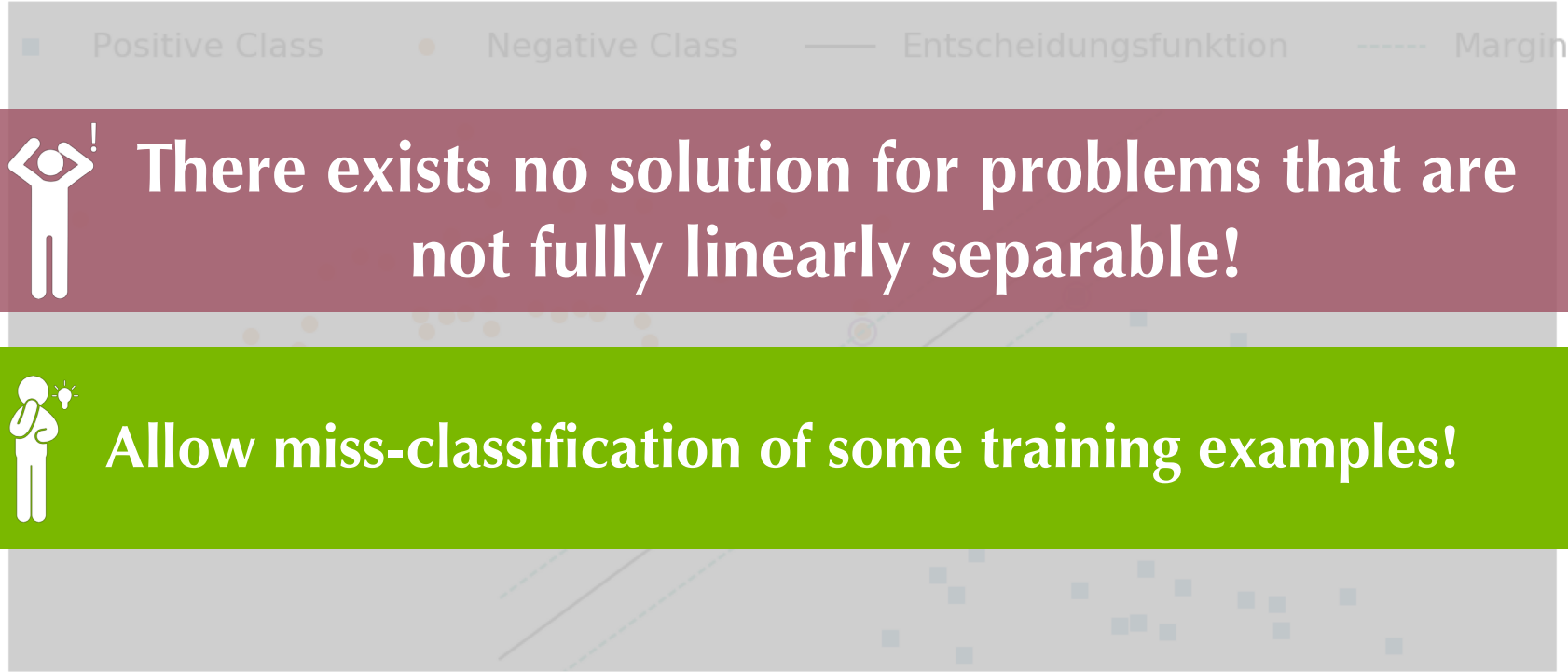
Might there be any problems when looking at this plot?



There exists no solution for problems that are not fully linearly separable!

Hard-Margin Support Vector Machine (SVM)

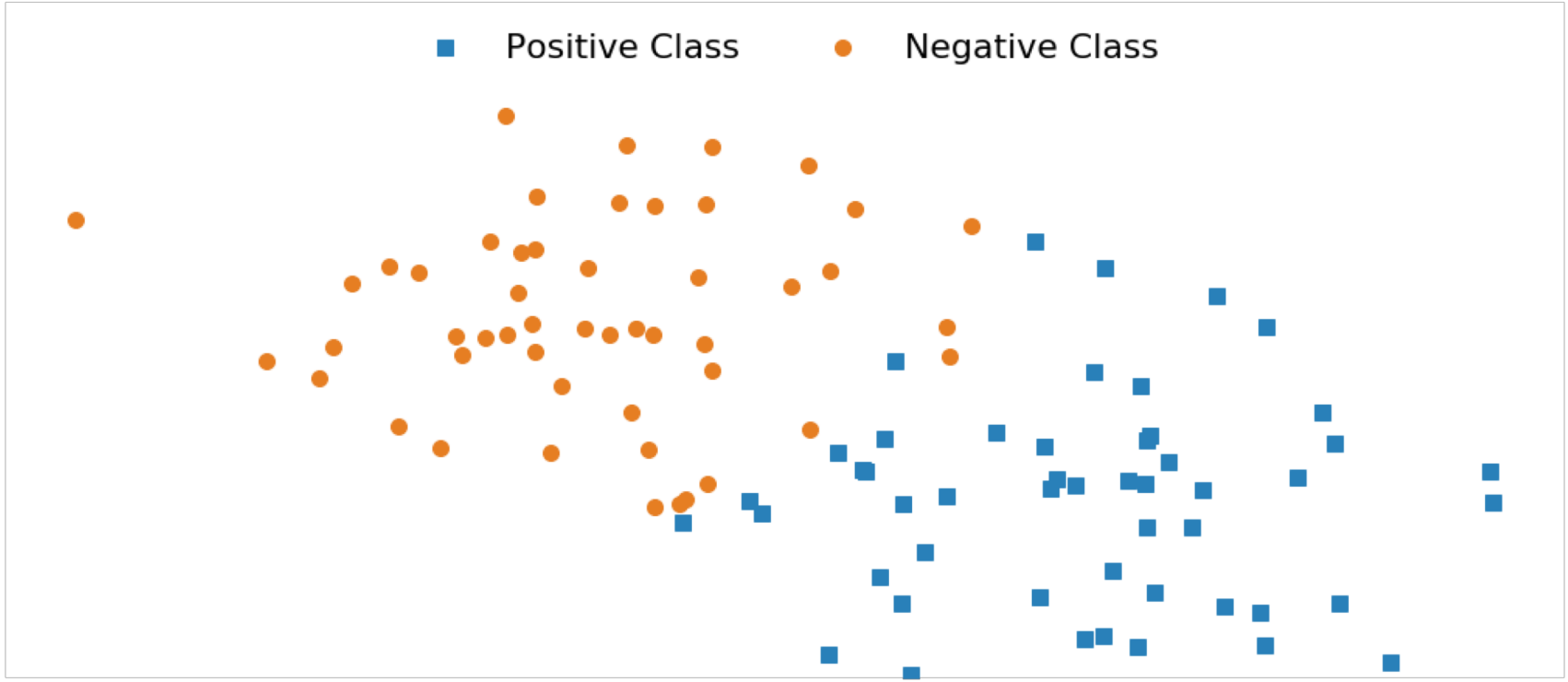
Might there be any problems when looking at this plot?



There exists no solution for problems that are not fully linearly separable!

Allow miss-classification of some training examples!

Soft-Margin Support Vector Machine (C-SVM)



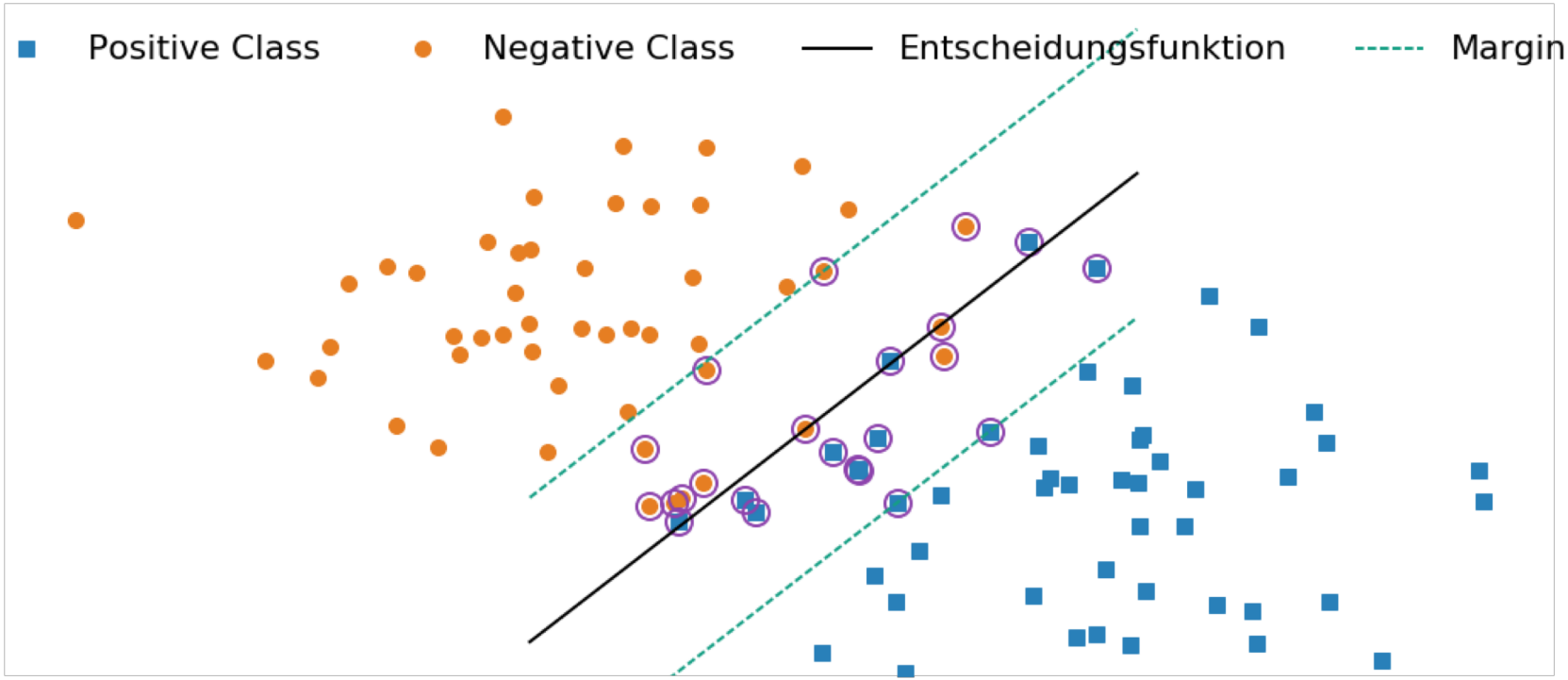
Cortes & Vapnik, 1995

Soft-Margin Support Vector Machine (C-SVM)

Add regularization parameter (C) to regularize the degree of allowed miss-classifications

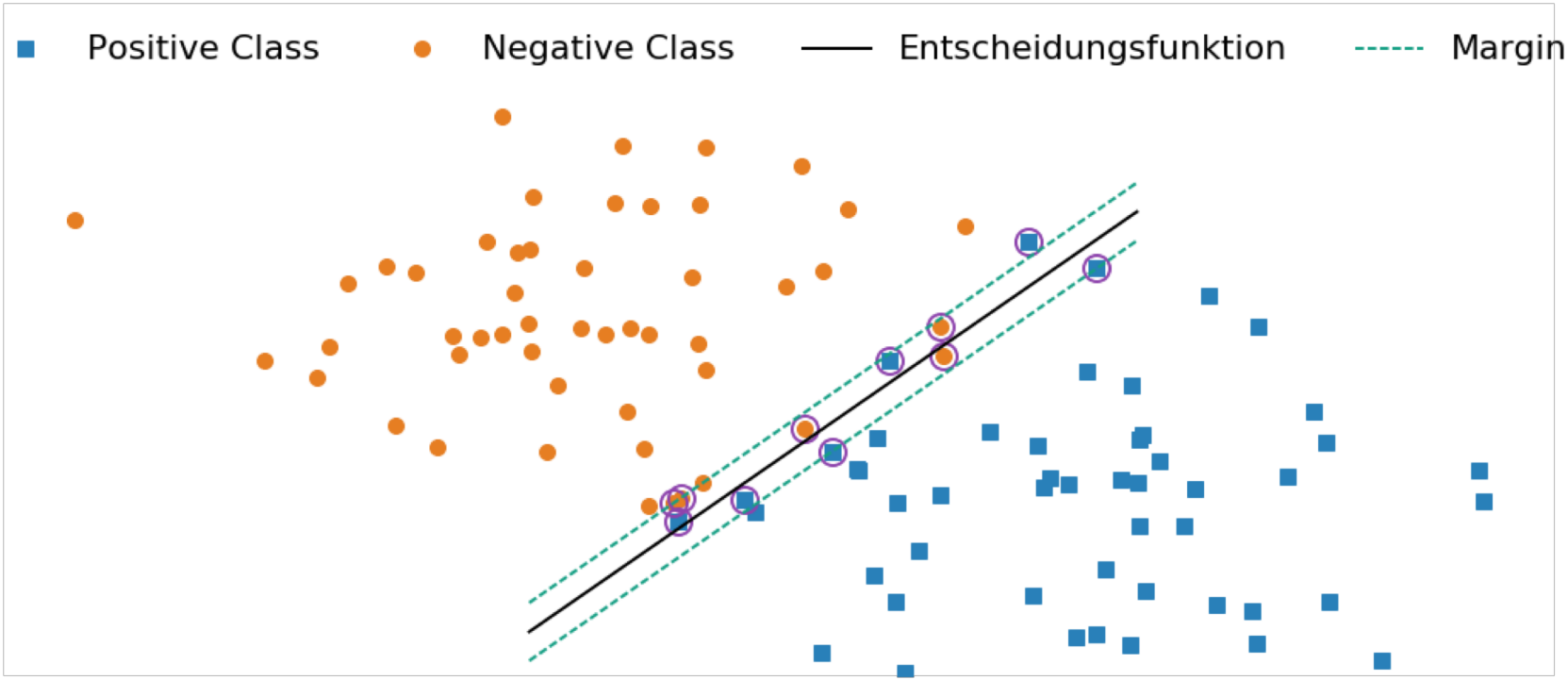
Cortes & Vapnik, 1995

Soft-Margin Support Vector Machine (C-SVM)



Small C leads to large margin → Danger of underfitting

Soft-Margin Support Vector Machine (C-SVM)

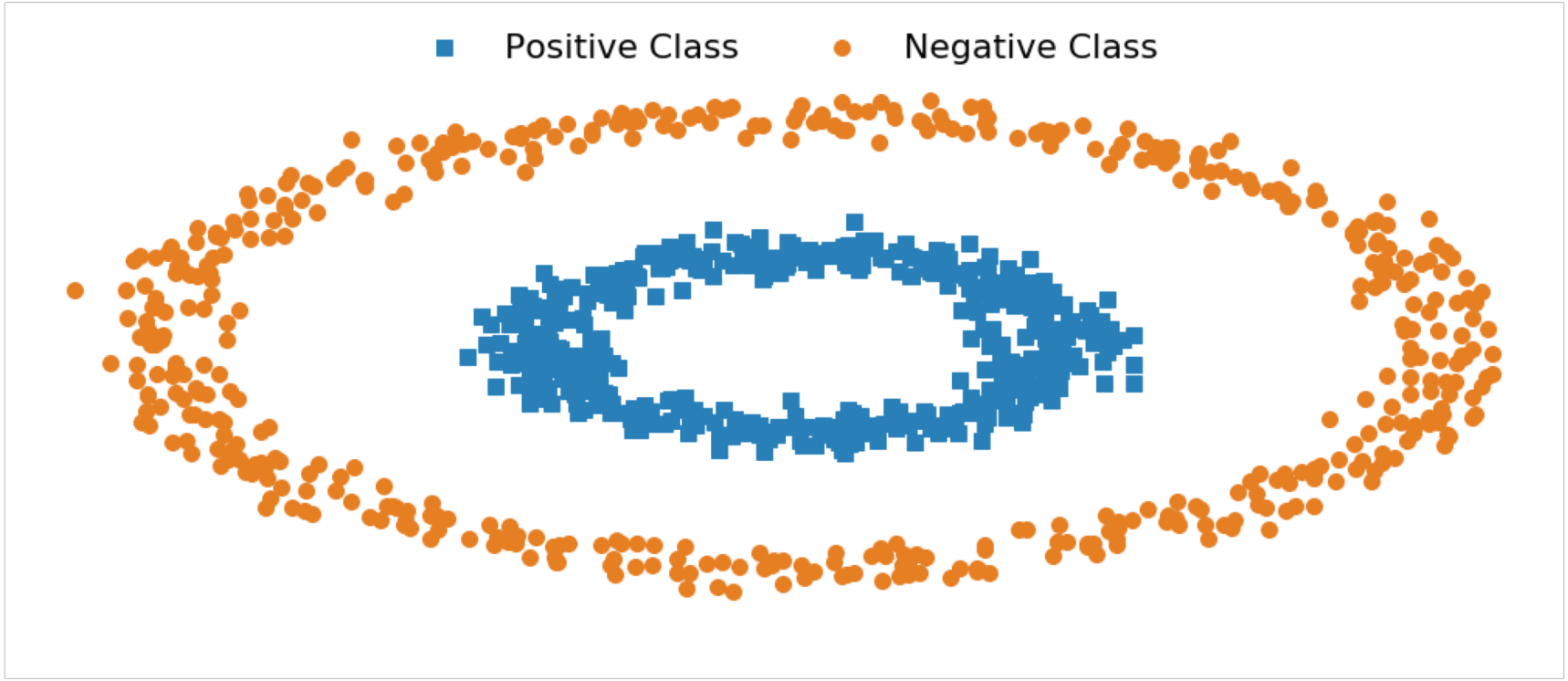


Large C leads to small margin → Danger of overfitting

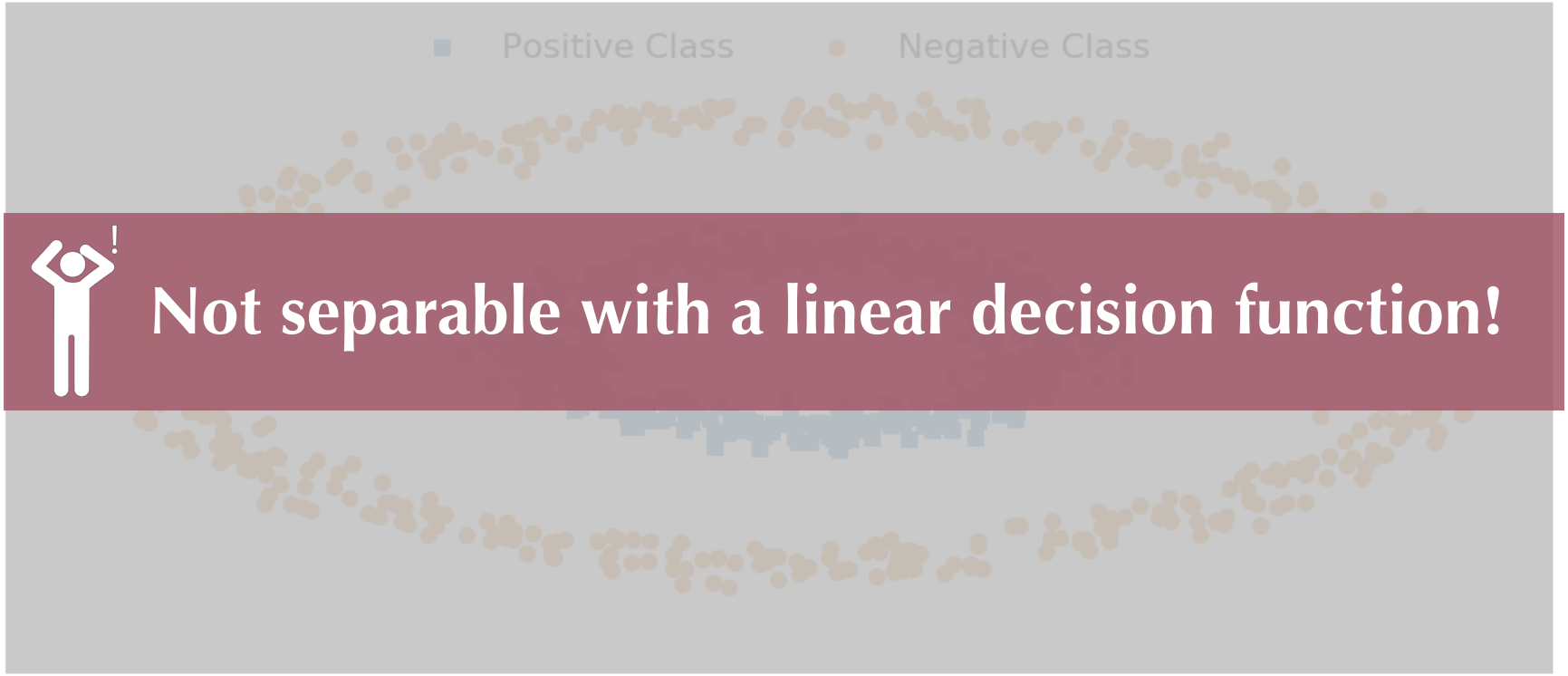
Soft-Margin Support Vector Machine (C-SVM)

Tradeoff between maximizing the margin and minimizing the training-error

How can we separate this two classes with a linear function?



How can we separate this two classes with a linear function?



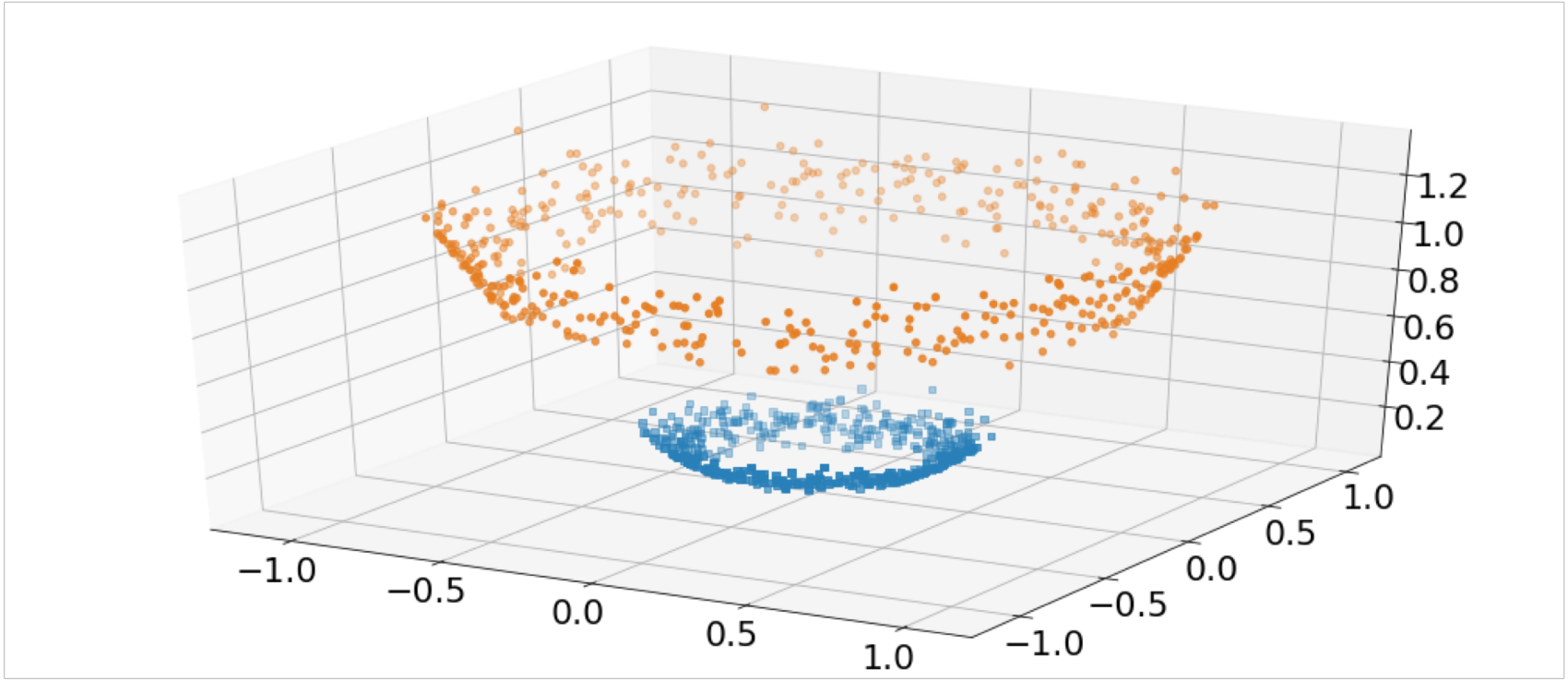
Not separable with a linear decision function!

Transformation of the data into a higher dimensional space

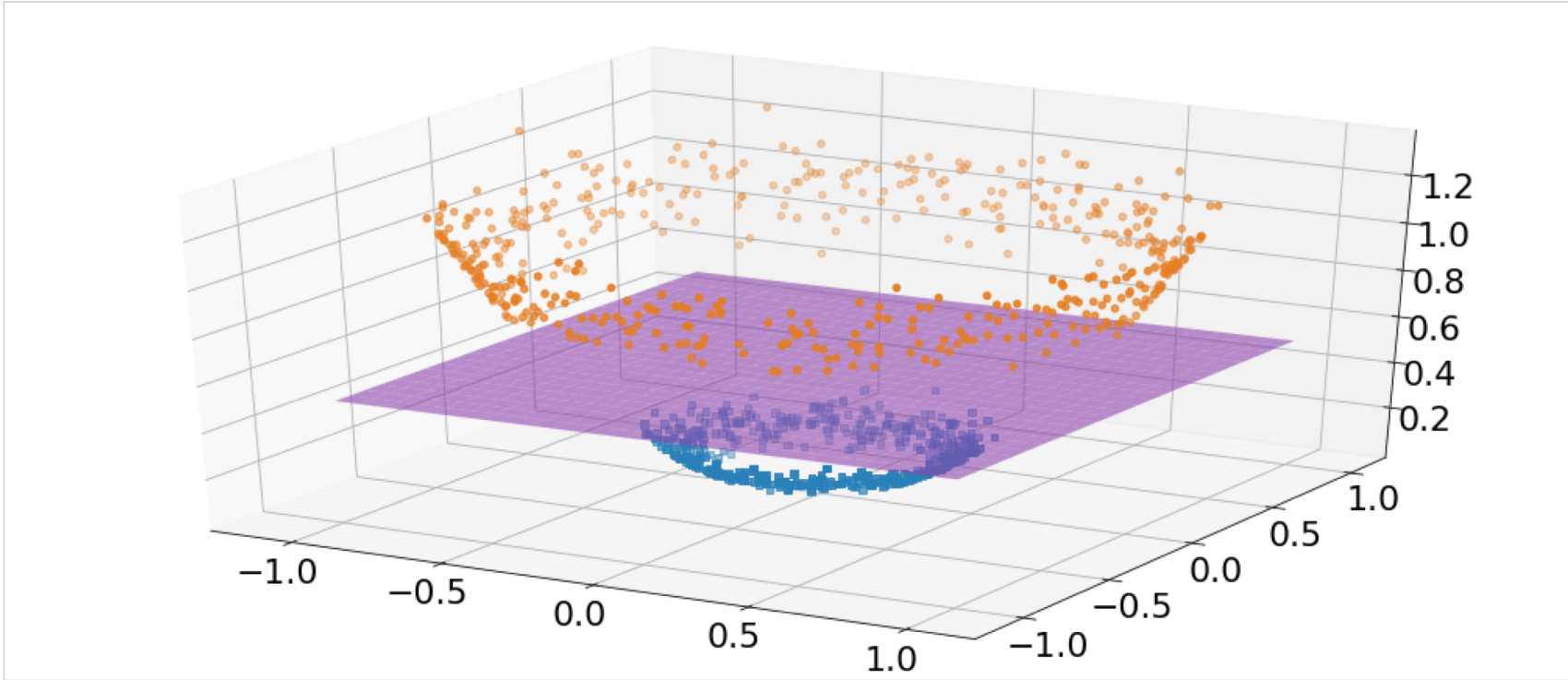
Transformation into 3-dimensional space

$$\mathbb{R}^2 \quad \rightarrow \mathbb{R}^3$$
$$(x_1, x_2) \rightarrow (z_1, z_2, z_3) := (x_1, x_2, x_1^2 + x_2^2)$$

Transformation of the data into a higher dimensional space



Transformation of the data into a higher dimensional space



Transformation of the data into a higher dimensional space

Problem: Feature transformation has to be known and explicitly computed → Not feasible for infinitely dimensional space

Transformation of the data into a higher dimensional space

Problem: Feature transformation has to be known and explicitly computed → Not feasible for infinitely dimensional space

With a mathematical trick we can circumvent this problem → The kernel trick!

The kernel trick

Note: In this lecture we will not explain kernels and its mathematical details; We will only explain the concept!

For more information and details you can have a look at:

"Learning with Kernels", Bernhard Schölkopf & Alex Smola

MIT Press, Cambridge, MA, 2002

What do kernels do and why can they help?

- We do **not** need to know the **exact feature space** when using kernel functions → we only need a function, that computes a **similarity measure** between all the features

- An **optimal** kernel assigns training instances
  - a higher similarity value, if they belong to the same class
  - a smaller similarity value, if they belong to different classes

Kernels

Linear Kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$$

Polynomial Kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + c)^d$$

Gaussian Radial Basisfunction (RBF) Kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$$

Kernels

Linear Kernel (Standard Kernel)

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$$

Cannot separate non-linear problems

Polynomial Kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + c)^d$$

Gaussian Radial Basisfunction (RBF) Kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$$

How to measure performance of classification models?

Confusion Matrix

True Positive (TP): **Label** is **positive** and **Prediction** is **positive**
False Positive (FP): **Label** is **negative** but **Prediction** is **positive**
True Negative (TN): **Label** is **negative** and **Prediction** is **negative**
False Negative (FN): **Label** is **positive** but **Prediction** is **negative**

		Ground Truth or Gold Standard	
		Label is positive	**Label is negative**
Prediction	**Prediction positive**	TP	FP
	Prediction negative	FN	TN

How to measure performance of classification models?

Confusion Matrix

True Positive (TP): **Label** is **positive** and **Prediction** is **positive**
False Positive (FP): **Label** is **negative** but **Prediction** is **positive**
True Negative (TN): **Label** is **negative** and **Prediction** is **negative**
False Negative (FN): **Label** is **positive** but **Prediction** is **negative**

		Ground Truth or Gold Standard: Label is positive	Ground Truth or Gold Standard: Label is negative
Prediction	Prediction positive	TP	FP
	Prediction negative	FN	TN
		Set of all positives (P)	Set of all negatives (N)

How to measure performance of classification models?

Performance Measures

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} = \frac{TP + TN}{P + N}$$

How to measure performance of classification models?

Performance Measures

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} = \frac{TP + TN}{P + N}$$

$$Precision = \frac{TP}{TP + FP}$$

How to measure performance of classification models?

Performance Measures

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} = \frac{TP + TN}{P + N}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$ (also called **Sensitivity, True Positive Rate**)

How to measure performance of classification models?

Performance Measures

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} = \frac{TP + TN}{P + N}$$

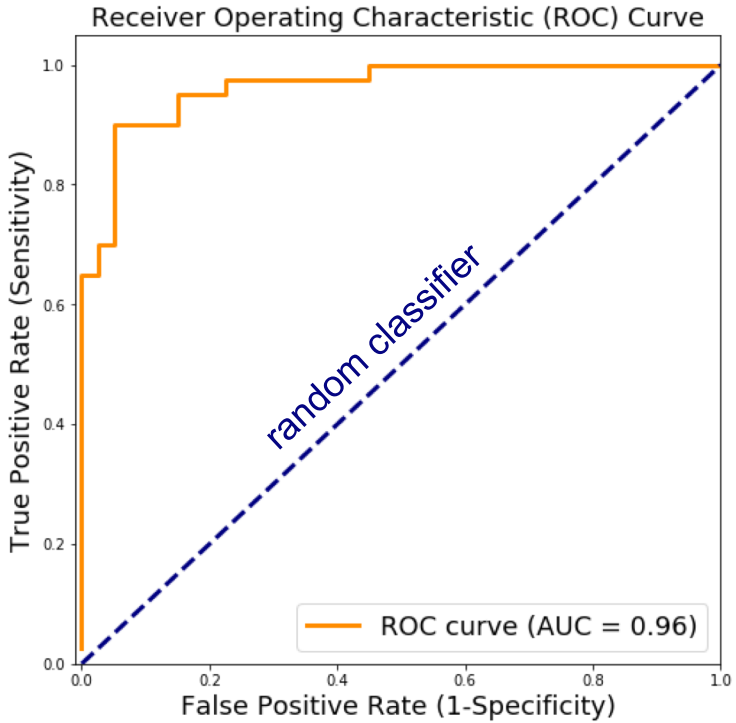
$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$ (also called **Sensitivity, True Positive Rate**)

$$Specificity = \frac{TN}{TN + FP}$$ (also called **True Negative Rate**)

How to measure performance of classification models?

Receiver Operating Characteristic (ROC) Curve



- The **ROC-Curve** is the fraction of the TP over all positives (TP+FN) against the fraction of TN over all negatives (TN+FP)
- The **Area Under The Curve (AUC)** is the area under the ROC-Curve and is a measure of the classifiers performance
- If AUC=1 → Perfect classifier
- If AUC=0.5 → Random classifier

Summary Support Vector Machine

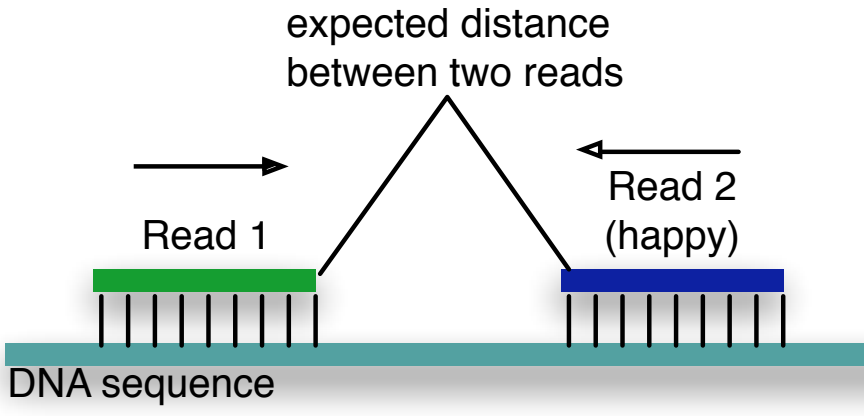
- **Support Vector Machines** are (usually) used for classification problems
- The **hard-margin SVM** is a supervised learning method, which can be used for fully linear separable problems (does usually not exist in reality)
- The **soft-margin SVM (C-SVM)** uses a regularization term, to allow a tradeoff between maximizing the **margin** and minimizing the **training-error**
- The kernel-trick helps to find solutions for non-separable problems in an implicit higher dimensional space (we do not need to explicitly map all instances into the higher dimensional space)

Part 3

Real-World Machine Learning Applications in Genomics and Genetics

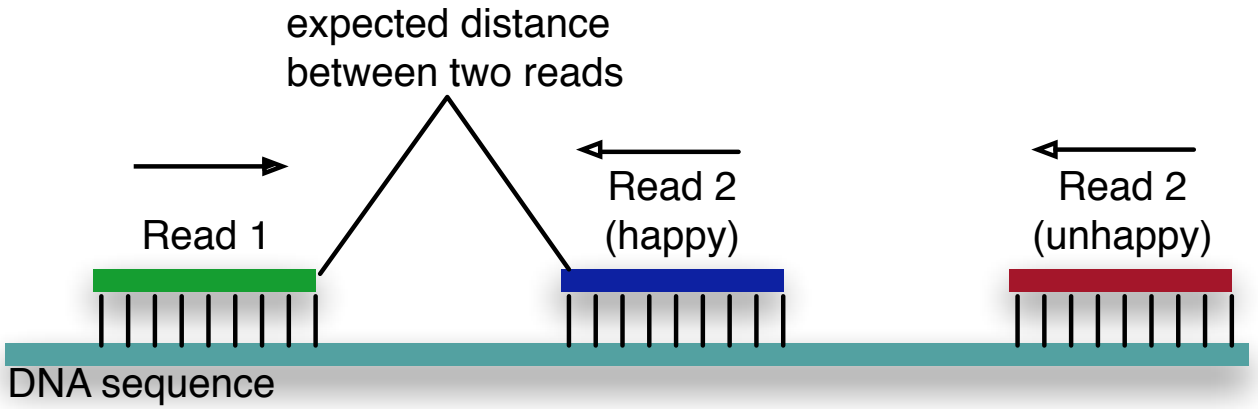
Example 1: Structural Variant Calling using Short Reads

Paired-End Reads



Example 1: Structural Variant Calling

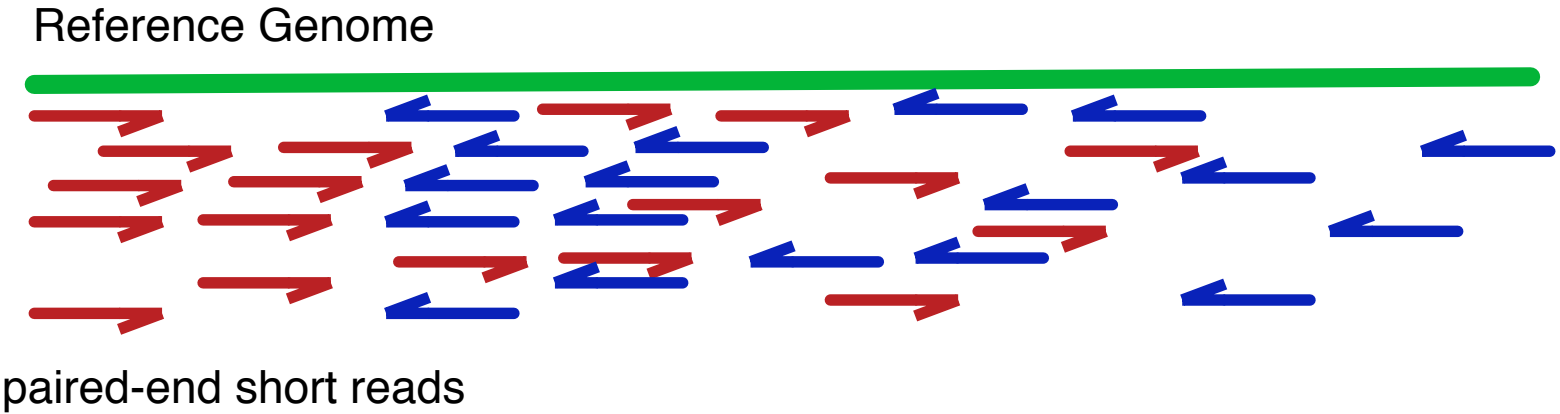
Paired-End Reads



Example 1: Structural Variant Calling

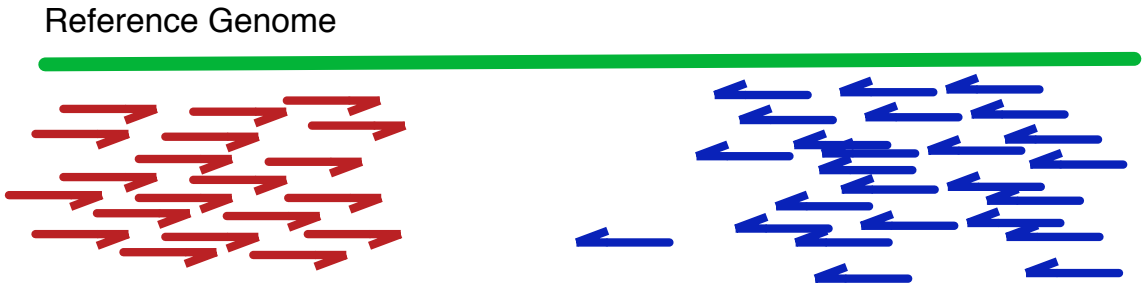
Reference Guided Genome Mapping

- Millions of short reads (read-pairs with length between ~30 bp – 400bp) are produced with Next Generation Sequencing technologies
- Reconstruction of genome is challenging → Align (**map**) reads against a known genome of the same species (**reference genome)**



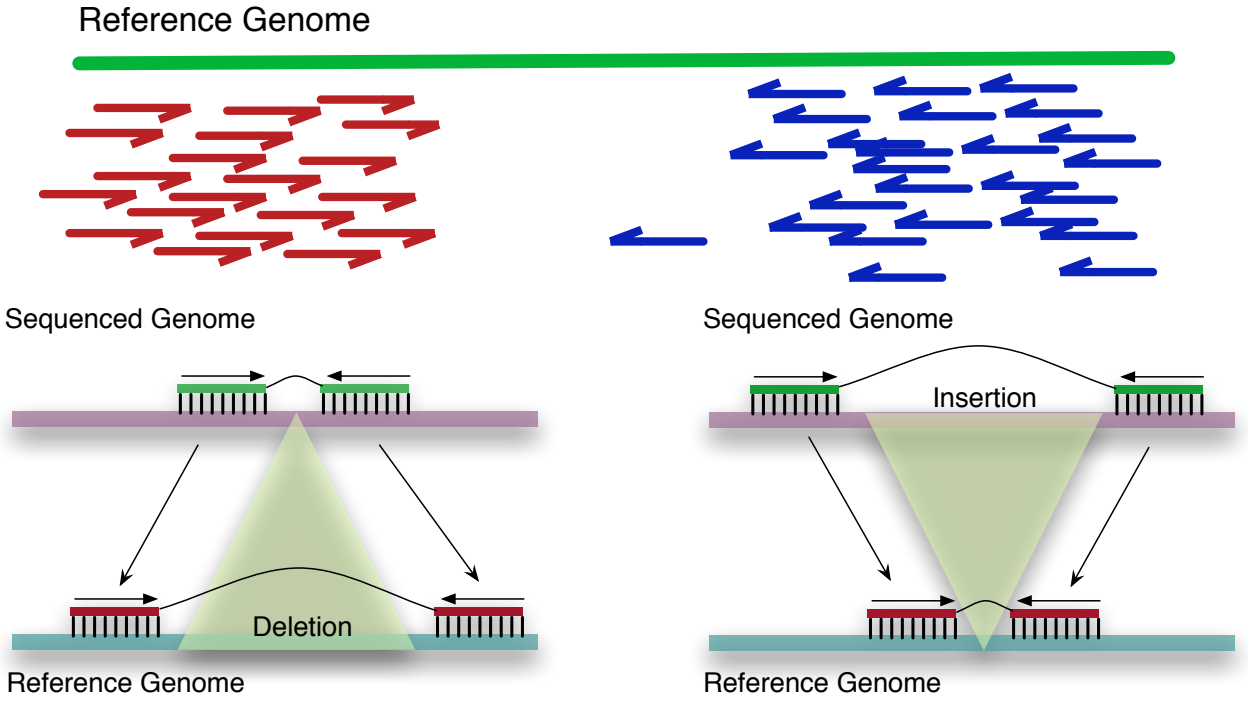
Example 1: Structural Variant Calling

Reconstruction and mapping is not trivial



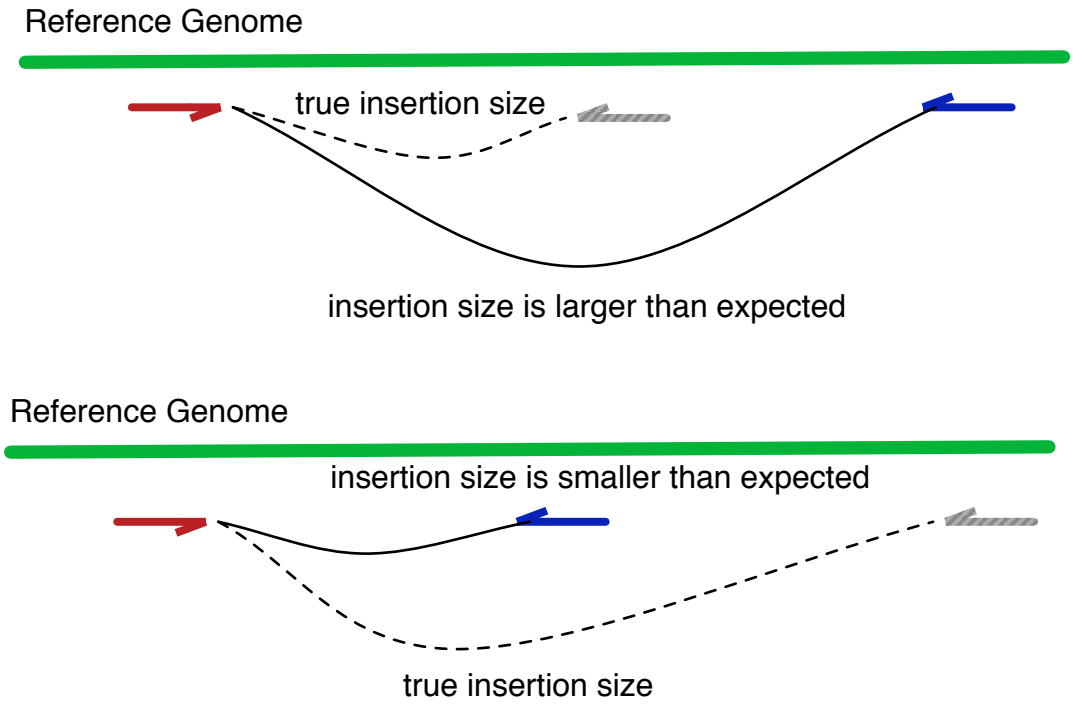
Example 1: Structural Variant Calling

What are Deletions and Insertions?



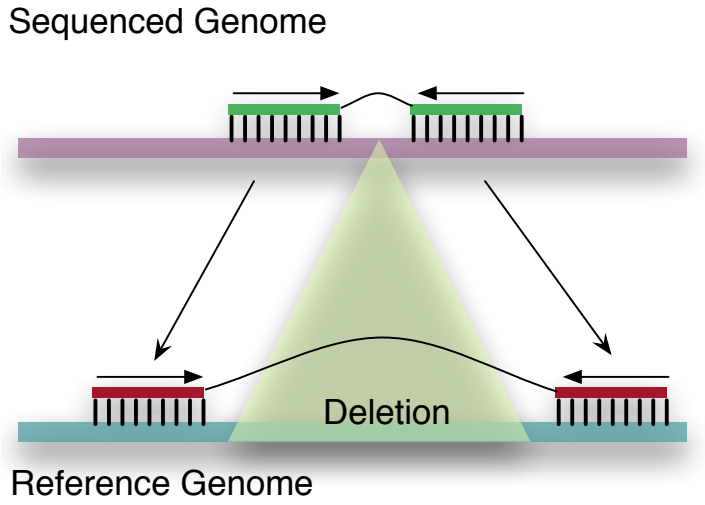
Example 1: Structural Variant Calling

Reconstruction and mapping is not trivial



Example 1: Inferring Deletions and Insertions

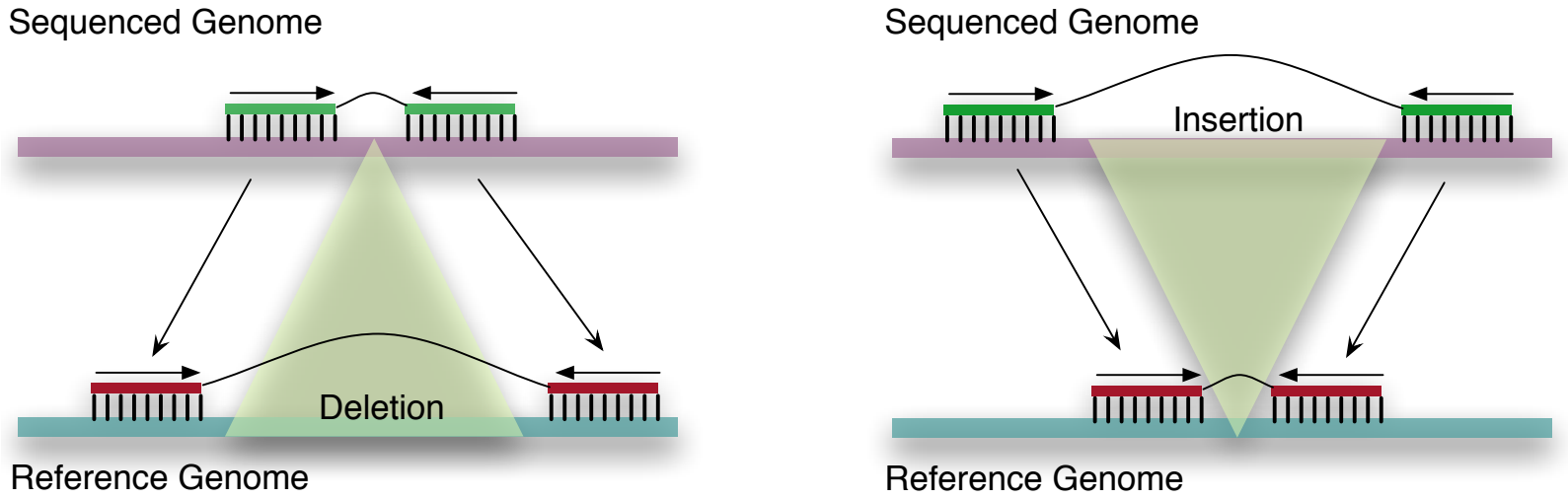
Inferring Insertions & Deletions (InDels) with discordant paired-end reads (Tuzun *et. al.* 2005)



» If the distance between two mapped reads is significantly **larger** than expected it is an indicator for a **deletion**

Example 1: Inferring Deletions and Insertions

Inferring Insertions & Deletions (InDels) with discordant paired-end reads (Tuzun *et. al.* 2005)

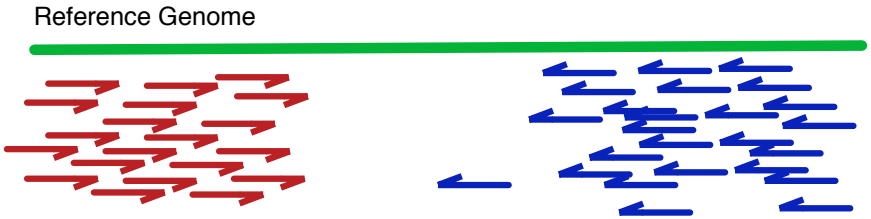


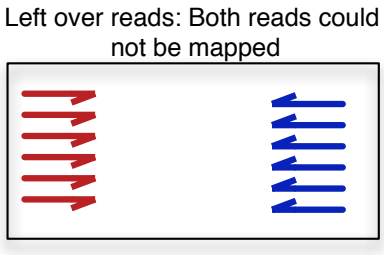
» If the distance between two mapped reads is significantly **larger** than expected it is an indicator for a **deletion**

» If the distance between two mapped reads is significantly **smaller** than expected it is an indicator for an **insertion**

Example 1: Inferring Deletions and Insertions

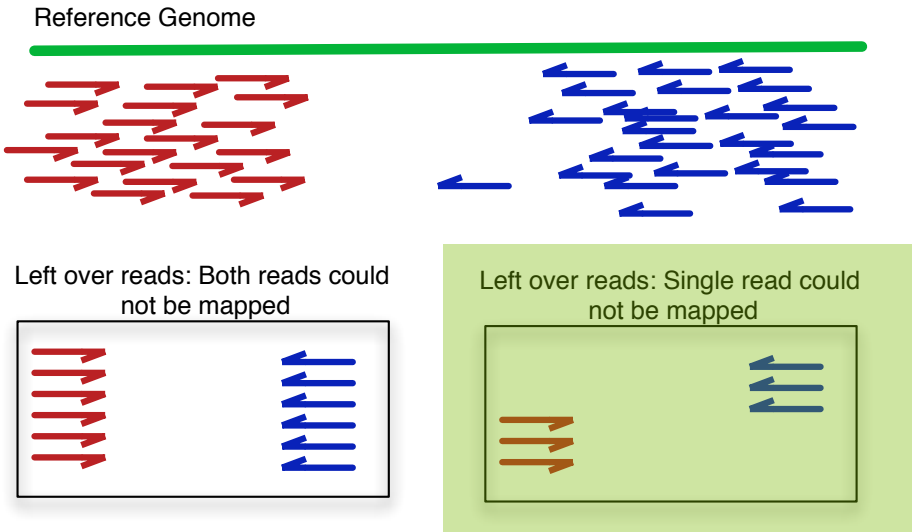
The method only uses mapped reads; it ignores the large proportion of unmapped reads!





Example 1: Inferring Deletions and Insertions

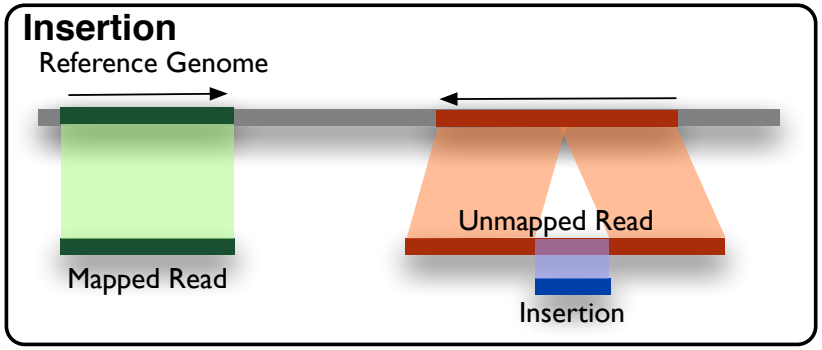
The method only uses mapped reads; it ignores the large proportion of unmapped reads!

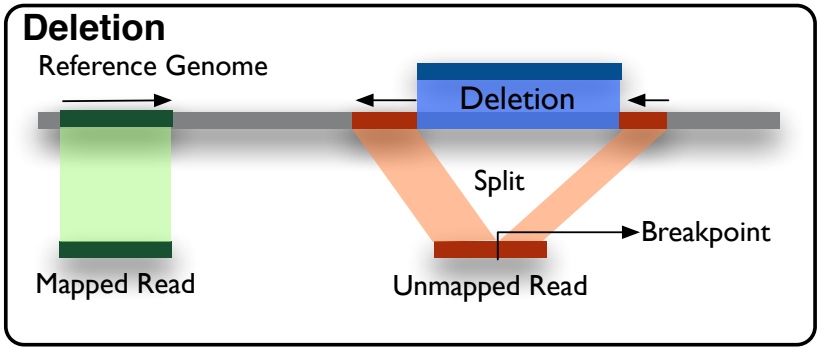


Read pairs for which only one partner could be mapped

Example 1: Structural Variant Calling

Pindel: A pattern growth approach for InDel detection (Ye *et. al.* 2009)

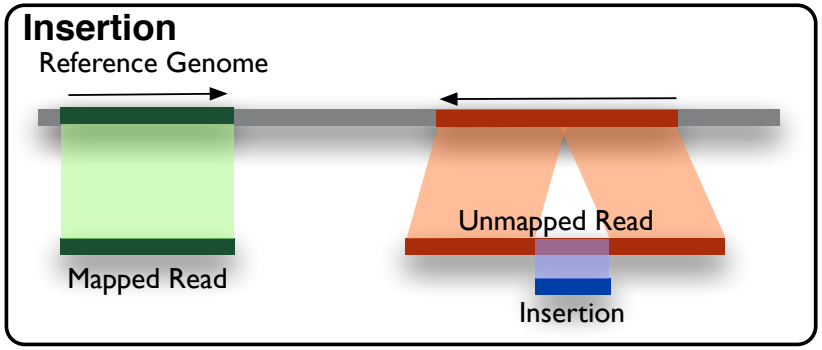


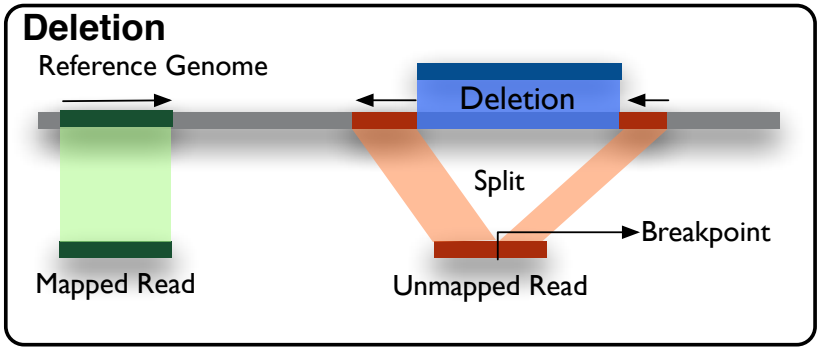


» Keep paired-end reads for which **only one read** could be mapped uniquely and exactly (no mismatches) against the reference genome

Example 1: Structural Variant Calling

Pindel: A pattern growth approach for InDel detection (Ye *et. al.* 2009)

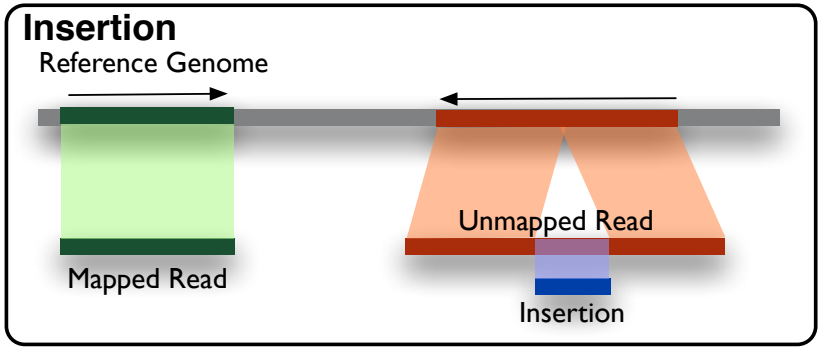


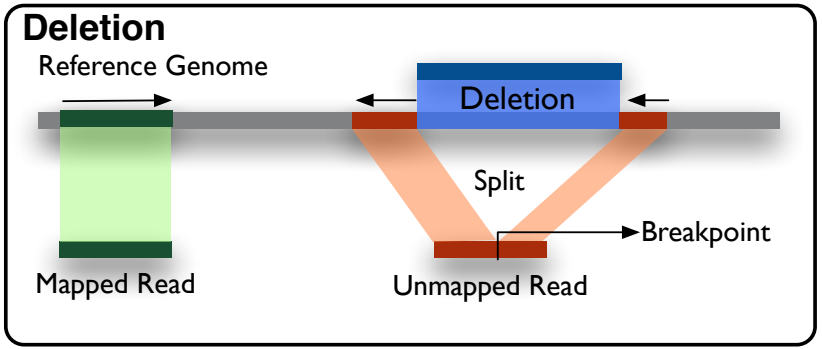


- Keep paired-end reads for which **only one read** could be mapped uniquely and exactly (no mismatches) against the reference genome
- Use pattern growth approach to **align the unmapped partner** against the reference genome (within a window of **two times the average insertion size**)

Example 1: Structural Variant Calling

Pindel: A pattern growth approach for InDel detection (Ye *et. al.* 2009)





- Keep paired-end reads for which **only one read** could be mapped uniquely and exactly (no mismatches) against the reference genome
- Use pattern growth approach to **align the unmapped partner** against the reference genome (within a window of **two times the average insertion size**)
- Report InDel if **at least two split re**ads support the **exact same split-alignment**

Example 1: Structural Variant Calling

Pindel: A pattern growth approach for InDel detection (Ye *et. al.* 2009)

Advantages

- Makes use of left-over reads
- Fast algorithm, which can be applied to large genomes
- It can detect indels at the base-pair level (not possible with discordant reads)

Disadvantages

- Only exactly and uniquely mapped reads are allowed → Large number of reads are excluded due to sequencing errors, repetitive regions or multiple mapping positions
- The second partner needs to be mapped within two times of the average insertion size
- Only one feature (at least 2 reads, which support the same indel position) is used to call an indel a true indel

Example 1: Accurate Indel Prediction

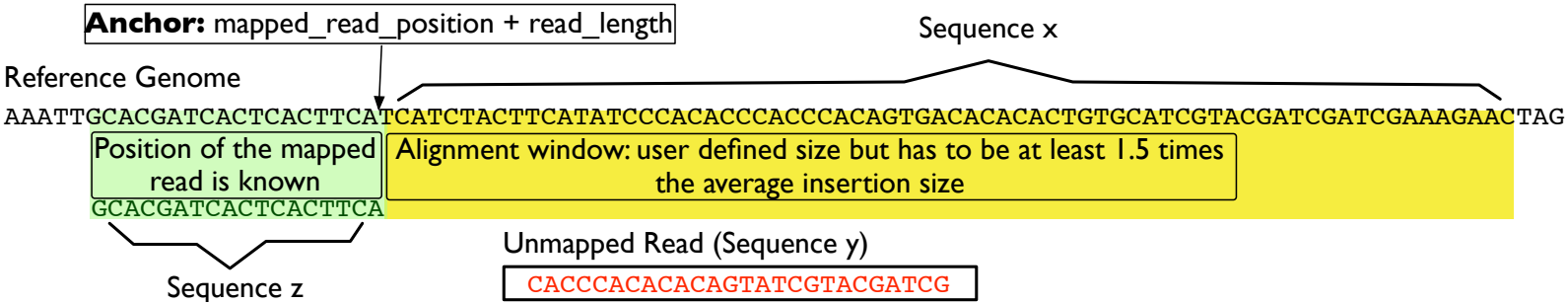
Improving InDel prediction (Grimm *et. al.* 2013)

Objectives to accomplish

- Use **non-error free** and **non-unique reads** (reads with multiple mapped positions)
- **Allow mismatches** and **gaps** in the re-alignment step (split-alignment) of unmapped partner
- **Allow** re-alignments **larger than two times the average insertion size**
- Use more **comprehensive** set of features and a **discriminative classifier** to predict if an InDel is a true or false one

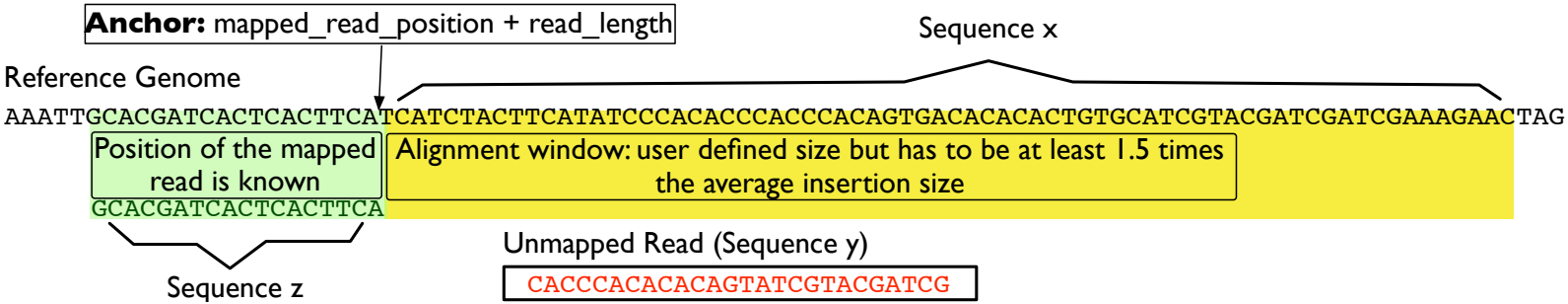
Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Perform split-alignment of unmapped read partner using Gotoh alignment (Gotoh, 1982)

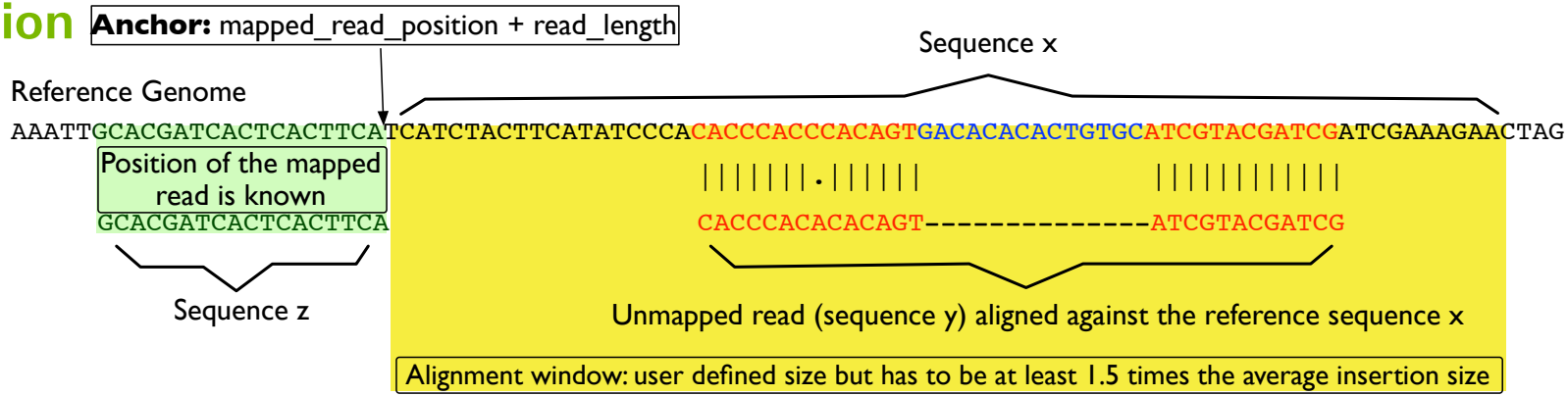


Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Perform split-alignment of unmapped read partner using Gotoh alignment (Gotoh, 1982)

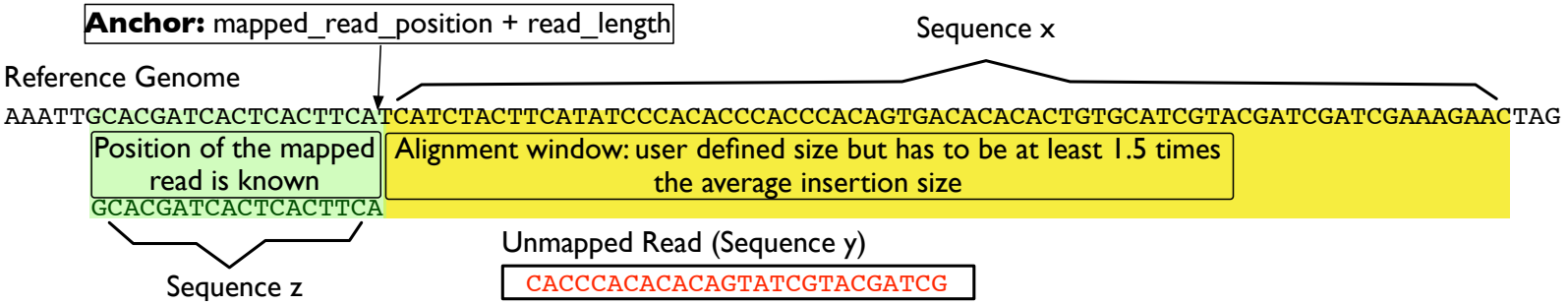


Deletion

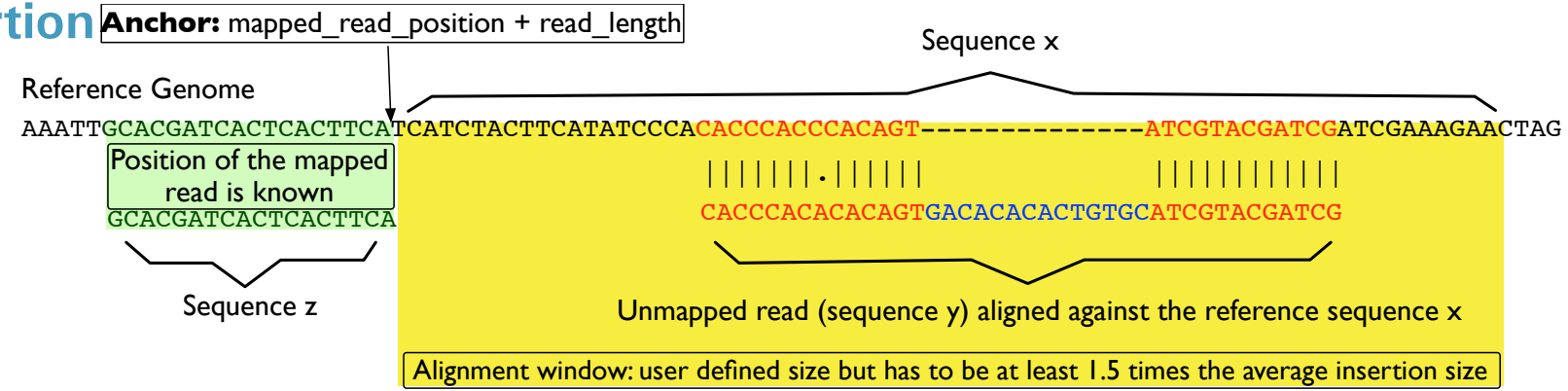


Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Perform split-alignment of unmapped read partner using Gotoh alignment (Gotoh, 1982)



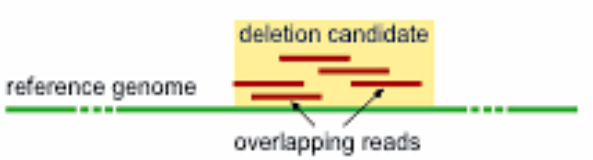
Insertion

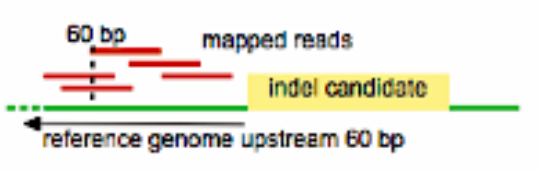


Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Alignment Features

	Category 1 (only for deletions)
1	Number of *uniquely mapped reads (UMRs) overlapping* the deletion candidate
2	Number of *error-free UMRs overlapping* the deletion candidate
3	Number of *non uniquely mapped reads (N-UMRs) overlapping* the deletion candidate
4	Number of *error-free N-UMRs overlapping* the found deletion
	Category 2 (for all indels)
5	Number of *UMRs* mapping 60bp *upstream* of the indel candidate
6	Number of *error-free UMRs* mapping 60bp *upstream* of the indel candidate
7	Number of *N-UMRs* mapping 60bp *upstream* of the indel candidate
8	Number of *error-free N-UMRs* mapping 60bp *upstream* of the indel candidate
9	Number of *UMRs* mapping 60bp *downstream* of the indel candidate
10	Number of *error-free UMRs* mapping 60bp *downstream* of the indel candidate
11	Number of *N-UMRs mapping* 60bp *downstream* of the indel candidate
12	Number of *error-free N-UMRs* mapping 60bp *downstream* of the indel candidate





Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

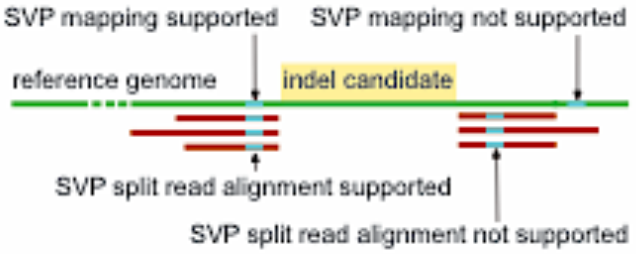
Alignment Features

Category 3 (for all indels)

13 Single position variation (SVP) from split read alignment confirmed by SVP from the mapping algorithm

14 SVP from split read alignment not confirmed by SVP from the mapping algorithm

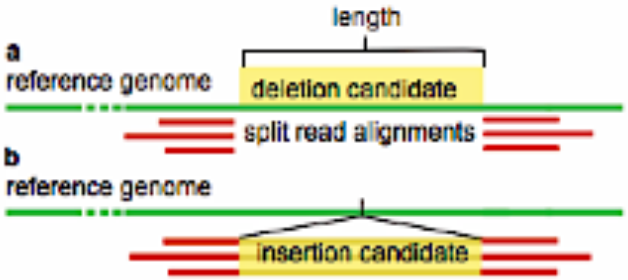
15 SVP from mapping algorithm not conformed by SVP from the split read alignment



Category 4 (for all indels)

16 Deletion/Insertion *length*

17 Number of splitted reads *supporting* the *same* indel location (split read alignment support)



Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Learn Discriminative Classifier (C-SVM)

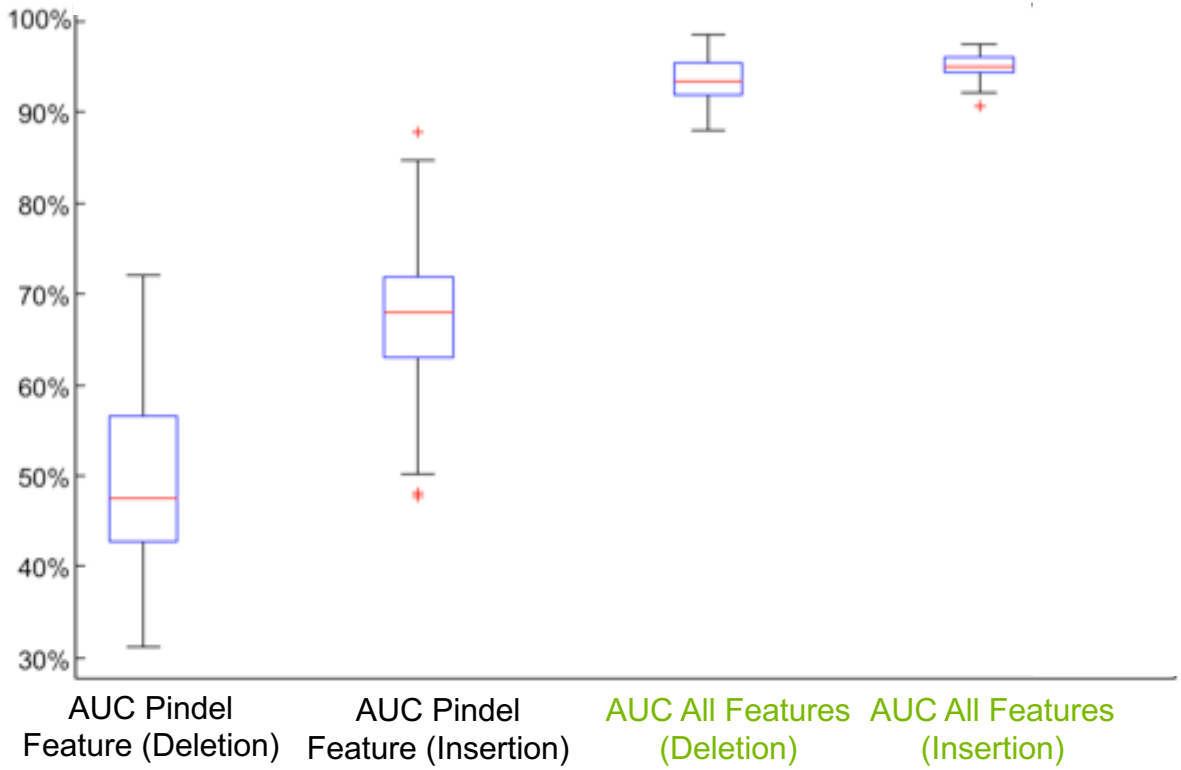
- **Data**

  219 Sanger validated **deletion** candidates (172 true, 47 false candidates)

  43 Sanger validated **insertion** candidates (33 true, 10 false candidates)
- Trained C-SVM using **10-fold cross-validation** on **deletion** candidates
- Trained C-SVM using **5-fold cross-validation** on **insertion** candidates

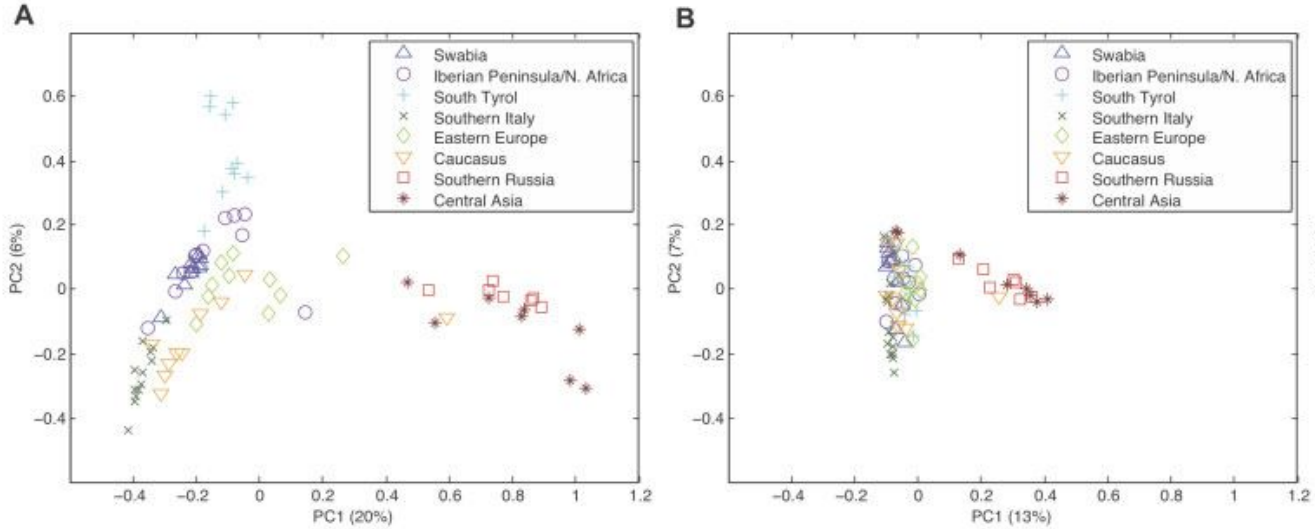
- Comparison with feature from Pindel (single feature, number of supporting reads for detected indel candidate)

Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)



Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Validating InDel Results with Population Structure



» **(A)** PCA on all positively classified indels from the discriminative classification approach (PC1 vs. PC2)

» **(B)** PCA for all detected indels from the tool PinDel (v0.1; PC1 vs. PC2)

Example 1: Accurate Indel Prediction (Grimm *et. al.* 2013)

Summary Indel Prediction

- Using exact alignment algorithms for the split-read-alignment leads to a larger set of potential indel candidates compared to the pattern growth approach by Pindel (however, also needs more computational resources)

- Using more than a single feature and machine learning techniques lead to less false positive candidates (→ less spurious biological interpretations)

- Set of features might look different for other species, datasets or different sequencing technologies

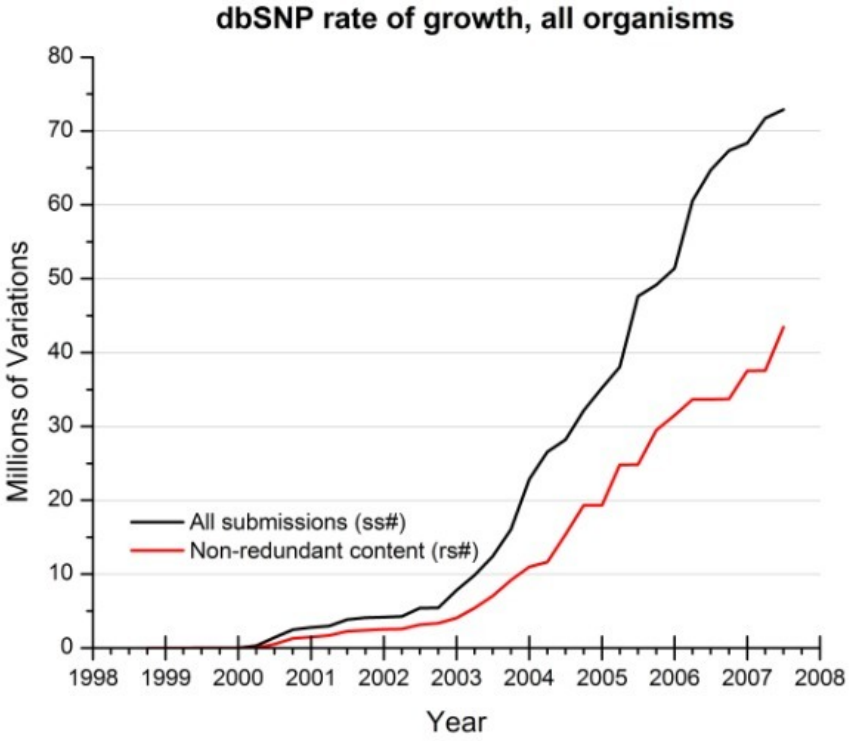
Reference: Grimm *et. al.,* "Accurate indel prediction using paired-end short reads" *BMC Genomics, 2013*

Example 2: Pathogenicity (Deleteriousness) Prediction

Impact of missense variants

Given the availability of more and more sequencing data on individual patients, one might ask the following fundamental question:

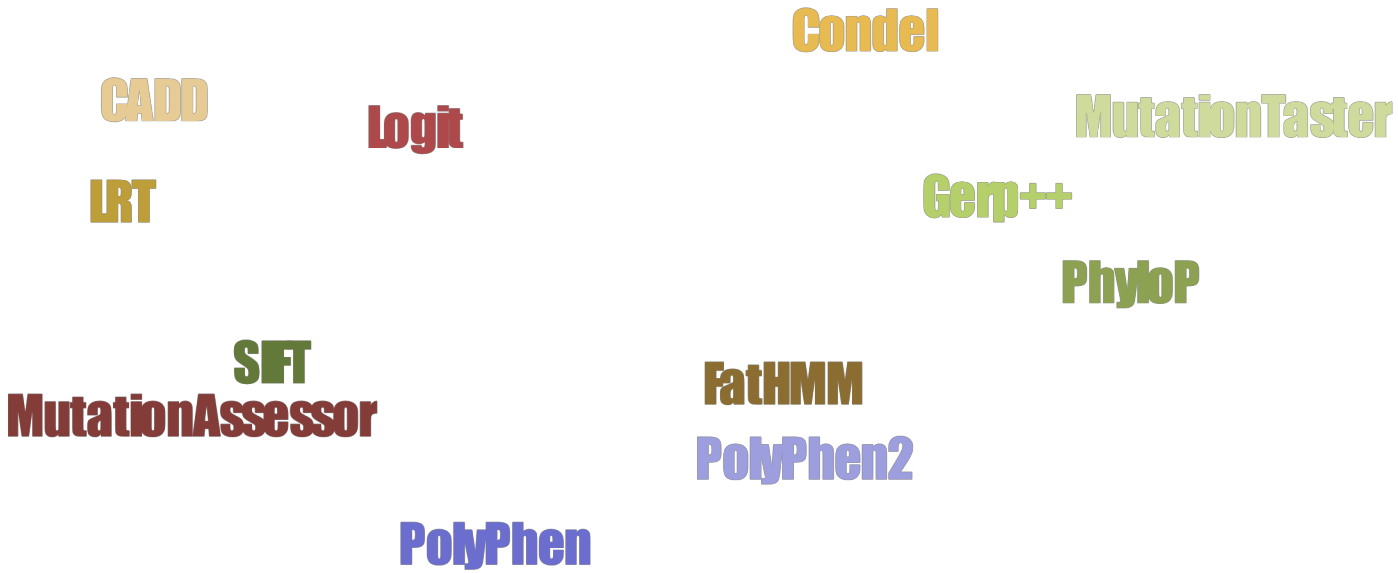
Is a specific mutation (variant) at a specific position in the genome damaging/deleterious?



https://www.ncbi.nlm.nih.gov/books/NBK44423/

Example 2: Pathogenicity (Deleteriousness) Prediction

Many tools have been developed to discriminate between deleterious and neutral variants!



Example 2: Pathogenicity (Deleteriousness) Prediction

Which features are used for training a deleteriousness predictor?

- Alignment related features, e.g.

  PSIC Score of wild-type amino-acid residue, PSIC Score of mutant amino-acid residue, conservation score of amino-acid residue (wild-type/mutant), etc…

- UniProtKB/Swissport derived protein sequence annotation, e.g.

  substitution site information, region annotation etc…

- Protein 3D features, e.g.

  accessible surface area, change in residue side chain volume, closest residue contact with other chain, etc…

- Protein Family related features
- Nucleotide sequence context features
- Any many more…

Example 2: Pathogenicity (Deleteriousness) Prediction

Which machine learning methods where used to train these predictors?

Random Forest Classifier

Neural Networks

Hidden-Markov-Models

Support Vector Machine

Logistic Regression

Bayesian Methods

Ensemble Learning Methods

Example 2: Pathogenicity (Deleteriousness) Prediction

Which predictor performs best?

Objective: Provide the cleanest and most comprehensive comparison of different deleteriousness prediction tools on a wide range of different datasets

Reference: Grimm *et. al.,* "The evaluation of tools used to predict the impact of missense variants is hindered by two types of circularity"*, Human Mutation*, 2015

Example 2: Pathogenicity (Deleteriousness) Prediction

Two major types of circularity have been discovered!

Type-1-Circularity

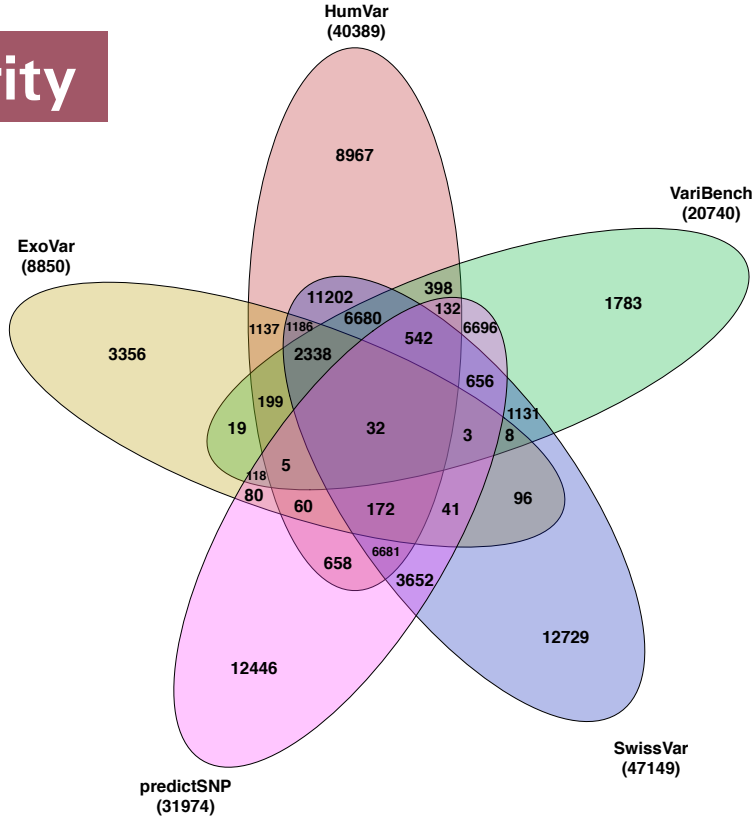
Common benchmark datasets for training and testing the tools overlap to a large degree

Type-2-Circularity

Most proteins in the benchmark datasets contain only deleterious **or** neutral variants, which leads to the problem that a naïve majority vote on the protein gives (artificially) excellent results

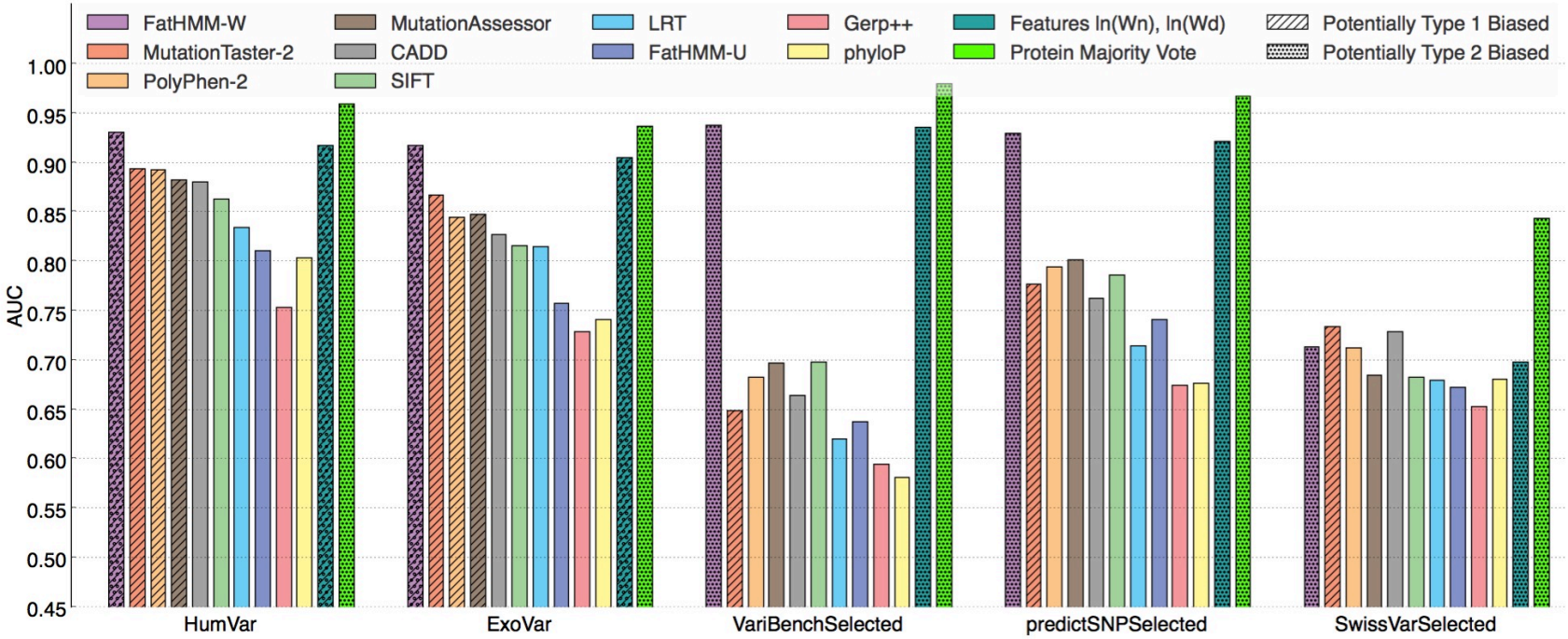
Example 2: Pathogenicity (Deleteriousness) Prediction

Type-1-Circularity



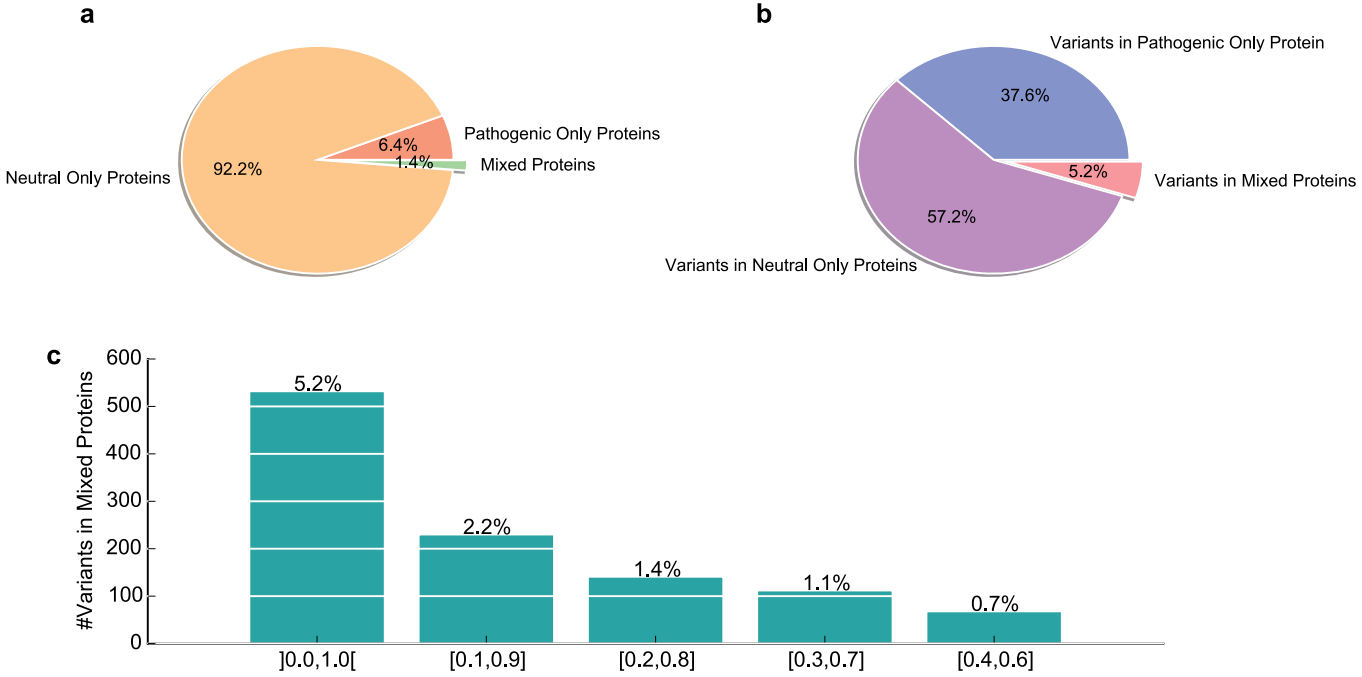
Example 2: Pathogenicity (Deleteriousness) Prediction

Type-1-Circularity: Comparative Evaluation



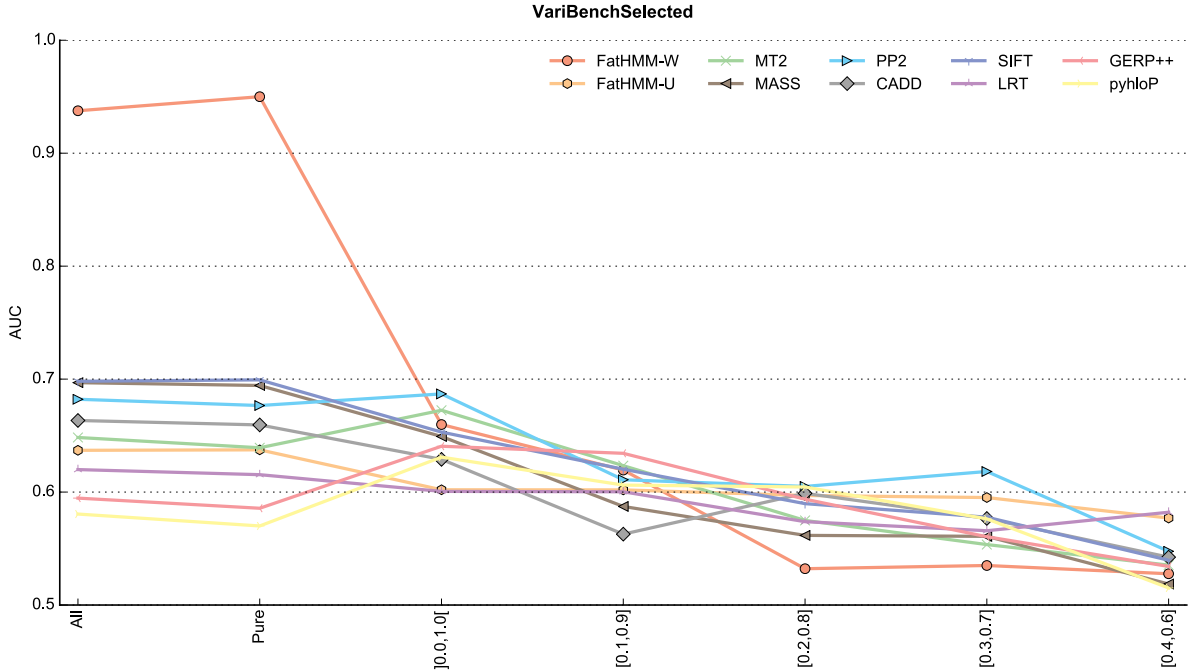
Example 2: Pathogenicity (Deleteriousness) Prediction

Fraction of mixed proteins in VaribenchSelected



Example 2: Pathogenicity (Deleteriousness) Prediction

Type-2-Circularity: Predictive performance on different deleterious/neutral ratios



Example 2: Pathogenicity (Deleteriousness) Prediction

Summary: Comparison of deleteriousness prediction tools

A comparison and accurate evaluation of deleteriousness prediction tools is complicated by two types of circularity

Type-1-circularity can only be avoided by cleanly separating training and testing data

Type-2-circularity can only be avoided by stratifying training and testing data with respect to protein membership

Reference: Grimm *et. al.,* “The evaluation of tools used to predict the impact of missense variants is hindered by two types of circularity”*, Human Mutation*, 2015

Tutorial: Applied ML Tutorial

Part 4

Tutorial: Applied Machine Learning with Python & Scikit-Learn

Toy-Example: Ridge Regression and SVM

New drug for the therapy of
a special disease

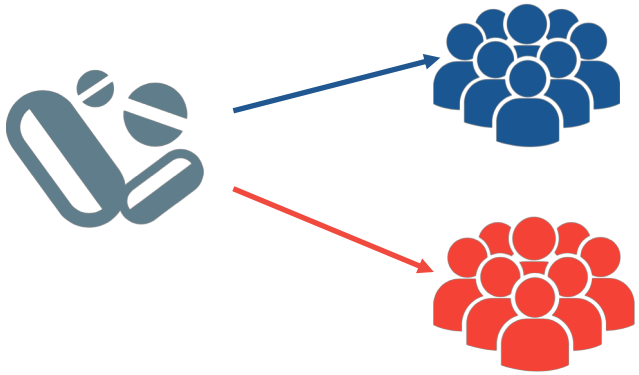
Toy-Example: Ridge Regression and SVM

New drug for the therapy of a special disease

Drug shows positive response within the first 50 days of treatment

Drug shows positive response after first 50 days of treatment, however with severe side-effects

Toy-Example: Ridge Regression and SVM

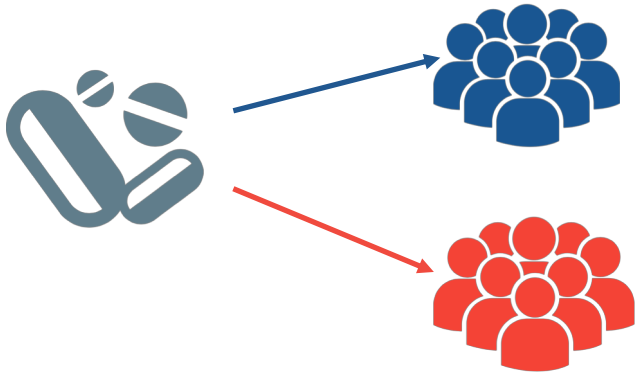


Is it possible to predict the the date from which (number of days) the drug shows a positive response, based on the genetic differences between the patients?

Toy-Example: Ridge Regression and SVM

Number of patients: 400

Number of mutations (features): 600



Target variable y: Number of days until drug shows positive response

Is it possible to predict the the date from which (**number of days**) the drug shows a positive response, based on the genetic differences between the patients?

Toy-Example: Ridge Regression

Data and Code is available on GitHub

GitHub https://github.com/grimmlab/lecture_ml4genomics

TUMCS

HOCHSCHULE
WEIHENSTEPHAN-TRIESDORF
UNIVERSITY OF APPLIED SCIENCES

Thanks for your attention!

Contact details

Prof. Dr. Dominik Grimm

dominik.grimm@hswt.de

http://www.cs.tum.de/de/bioinformatik/

Lecture Material & Code

GitHub https://github.com/grimmlab/lecture_ml4genomics