

Classification 2

IE500 Data Mining



Outline

- Decision Trees
- Overfitting
- Evaluation Metrics
- **Naïve Bayes**
- **Support Vector Machines**
- **Artificial Neural Networks**
- **Evaluation Methods**
- **Hyperparameter Selection**

Naïve Bayes

- Probabilistic classification technique based on Bayes theorem
 - successful, old-school method for various tasks: NLP, recommendation, ...
- Goal: Estimate the most probable class label for a given record
- Probabilistic formulation of the classification task:
 - consider each attribute and class label as random variables
 - given a record with attributes (A_1, A_2, \dots, A_n)
the goal is to find the class C that maximizes the conditional probability $P(C | A_1, A_2, \dots, A_n)$
- Example: Should we play golf?
 - $P(\text{Play}=\text{yes} \mid \text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool})$
 - $P(\text{Play}=\text{no} \mid \text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool})$
- Question: How to estimate these probabilities given training data?

Bayes Classifier

- Thomas Bayes (1701-1761)
 - British mathematician and priest
 - tried to formally prove the existence of God
- Bayes Theorem
 - important theorem in probability theory
 - was only published after Bayes' death



Bayes Classifier

- Question:
 - How likely is class C, given that we observe attributes A
 - This is called a conditional probability, denoted $P(C|A)$
 - e.g.: Given some attributes A, what is the likelihood of a certain class C?

- Bayes Theorem

$$P(C|A) = \frac{P(A|C) P(C)}{P(A)}$$

- Computes one conditional probability $P(C|A)$ out of another $P(A|C)$
 - given that the base probabilities $P(A)$ and $P(C)$ are known
- Useful in situations where $P(C|A)$ is unknown
 - while $P(A|C)$, $P(A)$ and $P(C)$ are known or easy to determine/estimate?

Bayes Classifier

- **Prior probability** of class C:
 - probability of class C before attributes are seen
 - we play golf in 70% of all cases $\rightarrow P(C) = 0.7$
- **Posterior probability** of class C:
 - probability of class C after attributes A is seen
 - evidence: It is windy and raining $\rightarrow P(C|A) = 0.2$

Estimating the Prior Probability $P(C)$

- The prior probability $P(C_j)$ for each class is estimated by
 - counting the records in the training set that are labeled with class $P(C_j)$
 - dividing the count by the overall number of records
- Example:
 - $P(\text{Play=no}) = 5/14$
 - $P(\text{Play=yes}) = 9/14$

Training Data

| Outlook | Temp | Humidity | Windy | Play |
|----------|------|----------|-------|------|
| Sunny | Hot | High | False | No |
| Sunny | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Rainy | Mild | High | False | Yes |
| Rainy | Cool | Normal | False | Yes |
| Rainy | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Sunny | Mild | High | False | No |
| Sunny | Cool | Normal | False | Yes |
| Rainy | Mild | Normal | False | Yes |
| Sunny | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Rainy | Mild | High | True | No |

Estimating the Class-Conditional Probability $P(A|C)$

- Naïve Bayes assumes that all attributes are **statistically independent**
 - knowing the value of one attribute says nothing about the value of another
 - this independence assumption is almost never correct!
 - but ... this scheme works well in practice
- The **independence assumption** allows the joint probability $P(A|C)$ to be reformulated as the product of the individual probabilities $P(A_i|C_j)$

$$P(A_1, A_2, \dots, A_n | C_j) = P(A_1 | C_j) * P(A_2 | C_j) * \dots * P(A_n | C_j) = \prod_{i=1}^n P(A_i | C_j)$$

$$P(\text{Outlook}=\text{rainy}, \text{Temperature}=\text{cool} | \text{Play}=\text{yes}) = P(\text{Outlook} = \text{rainy} | \text{Play} = \text{yes}) * P(\text{Temperature}=\text{cool} | \text{Play} = \text{yes})$$

- Result: The probabilities $P(A_i|C_j)$ for all A_i and C_j can be estimated directly from the training data

Estimating the Probabilities $P(A_i|C_j)$

| Outlook | | | Temperature | | | Humidity | | | Windy | | | Play | |
|----------|-----|-----|-------------|-----|-----|----------|-----|-----|-------|-----|-----|------|------|
| | Yes | No | | Yes | No | | Yes | No | | Yes | No | Yes | No |
| Sunny | 2 | 3 | Hot | 2 | 2 | High | 3 | 4 | False | 6 | 2 | 9 | 5 |
| Overcast | 4 | 0 | Mild | 4 | 2 | Normal | 6 | 1 | True | 3 | 3 | | |
| Rainy | 3 | 2 | Cool | 3 | 1 | | | | | | | | |
| Sunny | 2/9 | 3/5 | Hot | 2/9 | 2/5 | High | 3/9 | 4/5 | False | 6/9 | 2/5 | 9/14 | 5/14 |
| Overcast | 4/9 | 0/5 | Mild | 4/9 | 2/5 | Normal | 6/9 | 1/5 | True | 3/9 | 3/5 | | |
| Rainy | 3/9 | 2/5 | Cool | 3/9 | 1/5 | | | | | | | | |

- The probabilities $P(A_i|C_j)$ are estimated by
 - Count how often an attribute value co-occurs with class C_j
 - Divide by the overall number of examples belonging to class C_j

Example:

“Outlook=sunny” occurs on 2/9 examples in class “Yes”

$P(\text{Outlook=sunny}|\text{Yes}) = 2/9$

| Outlook | Temp | Humidity | Windy | Play |
|----------|------|----------|-------|------|
| Sunny | Hot | High | False | No |
| Sunny | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Rainy | Mild | High | False | Yes |
| Rainy | Cool | Normal | False | Yes |
| Rainy | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Sunny | Mild | High | False | No |
| Sunny | Cool | Normal | False | Yes |
| Rainy | Mild | Normal | False | Yes |
| Sunny | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Rainy | Mild | High | True | No |

Classifying a New Record

- Unseen record

| Outlook | Temp. | Humidity | Windy | Play |
|---------|-------|----------|-------|------|
| Sunny | Cool | High | True | ? |

Class-conditional probability of the record

$$P(\text{yes}|A) = P(\text{Outlook} = \text{Sunny}|\text{yes})$$

$$\times P(\text{Temperature} = \text{Cool}|\text{yes})$$

$$\times P(\text{Humidity} = \text{High}|\text{yes})$$

$$\times P(\text{Windy} = \text{True}|\text{yes})$$

$$\times \frac{P(\text{yes})}{P(A)}$$

$$= \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{P(A)}$$

Probability of class "yes" given the evidence

Prior probability of class "yes"

Prior probability of record

Classifying a New Record

| Outlook | | | Temperature | | | Humidity | | | Windy | | | Play | |
|----------|-----|-----|-------------|-----|-----|----------|-----|-----|-------|-----|-----|------|------|
| | Yes | No | | Yes | No | | Yes | No | | Yes | No | Yes | No |
| Sunny | 2 | 3 | Hot | 2 | 2 | High | 3 | 4 | False | 6 | 2 | 9 | 5 |
| Overcast | 4 | 0 | Mild | 4 | 2 | Normal | 6 | 1 | True | 3 | 3 | | |
| Rainy | 3 | 2 | Cool | 3 | 1 | | | | | | | | |
| Sunny | 2/9 | 3/5 | Hot | 2/9 | 2/5 | High | 3/9 | 4/5 | False | 6/9 | 2/5 | 9/14 | 5/14 |
| Overcast | 4/9 | 0/5 | Mild | 4/9 | 2/5 | Normal | 6/9 | 1/5 | True | 3/9 | 3/5 | | |
| Rainy | 3/9 | 2/5 | Cool | 3/9 | 1/5 | | | | | | | | |

- A new day:

| Outlook | Temp. | Humidity | Windy | Play |
|---------|-------|----------|-------|------|
| Sunny | Cool | High | True | ? |

Class conditional probability
Prior probability

Likelihood of the two classes

$$\text{For "yes"} = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$$

$$\text{For "no"} = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$$

Conversion into a probability by normalization:

$$P(\text{"yes"}) = 0.0053 / (0.0053 + 0.0206) = 0.205$$

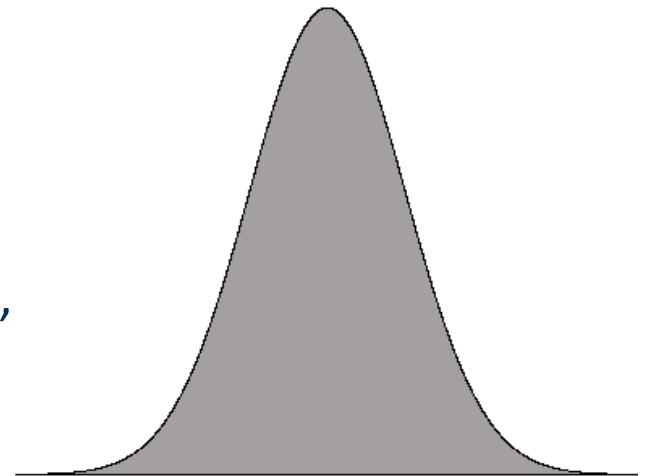
$$P(\text{"no"}) = 0.0206 / (0.0053 + 0.0206) = 0.795$$

Choose Maximum



Handling Numerical Attributes

- Option 1:
Discretize numerical attributes before learning classifier.
 - Temp= 37°C -> “Hot”
 - Temp= 21°C -> “Mild”
- Option 2:
Make assumption that numerical attributes have a **normal distribution** given the class.
 - use training data to estimate parameters of the distribution (e.g., mean and standard deviation)
 - once the probability distribution is known, it can be used to estimate the conditional probability $P(A_i|C_j)$



Handling Numerical Attributes

- The probability density function for the normal distribution is

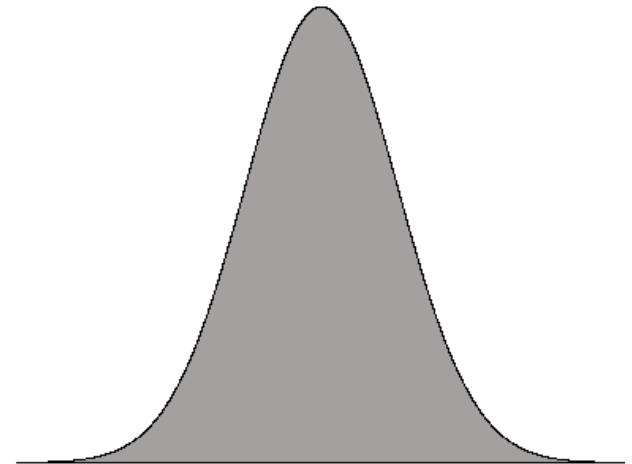
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- It is defined by two parameters:

- Sample mean $\mu = \frac{1}{n} \sum_{i=1}^n x_i$

- Standard deviation $\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2}$

- Both parameters can be estimated from the training data



Handling Numerical Attributes

| Outlook | | | Temperature | | Humidity | | Windy | | | Play | |
|----------|-----|-----|----------------|----------------|-----------------|----------------|-------|-----|-----|------|------|
| | Yes | No | Yes | No | Yes | No | | Yes | No | Yes | No |
| Sunny | 2 | 3 | 64, 68, | 65, 71, | 65, 70, | 70, 85, | False | 6 | 2 | 9 | 5 |
| Overcast | 4 | 0 | 69, 70, | 72, 80, | 70, 75, | 90, 91, | True | 3 | 3 | | |
| Rainy | 3 | 2 | 72, ... | 85, ... | 80, ... | 95, ... | | | | | |
| Sunny | 2/9 | 3/5 | $\mu = 73$ | $\mu = 75$ | $\mu = 79$ | $\mu = 86$ | False | 6/9 | 2/5 | 9/14 | 5/14 |
| Overcast | 4/9 | 0/5 | $\sigma = 6.2$ | $\sigma = 7.9$ | $\sigma = 10.2$ | $\sigma = 9.7$ | True | 3/9 | 3/5 | | |
| Rainy | 3/9 | 2/5 | | | | | | | | | |

- Example calculation:

$$f(\text{temp} = 66 \mid \text{yes}) = \frac{1}{\sqrt{2\pi} * 6.2} e^{-\frac{(66-73)^2}{2*(6.2)^2}} = 0.034$$

Classifying a New Day

- Unseen record

| Outlook | Temp. | Humidity | Windy | Play |
|---------|-------|----------|-------|------|
| Sunny | 66 | 90 | true | ? |

Likelihood of “yes” = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of “no” = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

$P(\text{“yes”}) = 0.000036 / (0.000036 + 0.000136) = 20.9\%$

$P(\text{“no”}) = 0.000136 / (0.000036 + 0.000136) = 79.1\%$

- But note:** Some numeric attributes are not normally distributed, and you may thus need to choose a different probability density function or use discretization

Handling Missing Values

- Missing values may occur in training and in unseen classification records
- **Training:** Record is not included into frequency count for attribute value-class combination
- **Classification:** Attribute will be omitted from calculation

– Example: Unseen record

| Outlook | Temp. | Humidity | Windy | Play |
|---------|-------|----------|-------|------|
| ? | Cool | High | True | ? |

Likelihood of “yes” = $\frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0238$

Likelihood of “no” = $\frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0343$

$P(\text{“yes”}) = 0.0238 / (0.0238 + 0.0343) = 41\%$

$P(\text{“no”}) = 0.0343 / (0.0238 + 0.0343) = 59\%$

The Zero-Frequency Problem

- What if an attribute value doesn't occur with every class value? (e.g. no "Outlook = overcast" for class "no")

| Outlook | | |
|----------|-----|-----|
| | Yes | No |
| Sunny | 2 | 3 |
| Overcast | 4 | 0 |
| Rainy | 3 | 2 |
| Sunny | 2/9 | 3/5 |
| Overcast | 4/9 | 0/5 |
| Rainy | 3/9 | 2/5 |

| Outlook | Temp. | Humidity | Windy | Play |
|----------|-------|----------|-------|------|
| Overcast | Cool | High | True | ? |

Likelihood of the two classes

For "yes" = $4/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0105$

For "no" = $0/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0$

- class-conditional probability will be zero! $P(\text{Outlook} = \text{Overcast} | \text{no}) = \frac{0}{5} = 0$
- Problem: Posterior probability will also be zero! $P(\text{no} | A) = 0$
No matter how likely the other values are!

The Zero-Frequency Problem

- Remedy: Add 1 to the count for every attribute value-class combination (**Laplace Estimator**)
- Result: Probabilities will never be zero!
also: stabilizes probability estimates

- Original: $P(A_i|C) = \frac{N_{ic}}{N_c}$ Laplace: $P(A_i|C) = \frac{N_{ic}+1}{N_c+c}$

| Outlook | | |
|----------|-----|------------|
| | Yes | No |
| Sunny | 2 | 3 |
| Overcast | 4 | 0 |
| Rainy | 3 | 2 |
| Sunny | 2/9 | 3/5 |
| Overcast | 4/9 | 0/5 |
| Rainy | 3/9 | 2/5 |



| Outlook | | |
|----------|------|------------|
| | Yes | No |
| Sunny | 2 | 3 |
| Overcast | 4 | 0 |
| Rainy | 3 | 2 |
| Sunny | 3/12 | 4/8 |
| Overcast | 5/12 | 1/8 |
| Rainy | 4/12 | 3/8 |

c = number of attribute values of A

Using Conditional Probabilities for Naïve Bayes

Result Overview ExampleSet (Retrieve Golf-Testset)

Data View Meta Data View Plot View Advanced Charts Annotations

ExampleSet (14 examples, 4 special attributes, 4 regular attributes) View Filter (14 / 14):

| Row No. | Play | confidence(no) | confidence(yes) | prediction(Play) | Outlook | Temperature | Humidity | Wind |
|---------|------|----------------|-----------------|------------------|----------|-------------|----------|-------|
| 1 | yes | 0.711 | 0.289 | no | sunny | 85 | 85 | false |
| 2 | no | 0.058 | 0.942 | yes | overcast | 80 | 90 | true |
| 3 | yes | 0.014 | 0.986 | yes | sunny | 83 | 78 | false |
| 4 | yes | 0.412 | 0.588 | yes | overcast | 70 | 96 | false |
| 5 | yes | 0.460 | 0.540 | yes | overcast | 68 | 80 | true |
| 6 | no | 0.336 | 0.664 | yes | rain | 65 | 70 | true |
| 7 | yes | 0.010 | 0.990 | yes | overcast | 64 | 65 | true |
| 8 | no | 0.596 | 0.404 | no | sunny | 72 | 95 | false |
| 9 | yes | 0.248 | 0.752 | yes | sunny | 69 | 70 | false |
| 10 | no | 0.407 | 0.593 | yes | sunny | 75 | 80 | false |
| 11 | yes | 0.496 | 0.504 | yes | sunny | 68 | 70 | true |
| 12 | yes | 0.038 | 0.962 | yes | overcast | 72 | 90 | true |
| 13 | no | 0.027 | 0.973 | no | overcast | 81 | 75 | true |
| 14 | yes | 0.453 | 0.547 | yes | overcast | 71 | 80 | true |

Classifier is quite sure

Classifier is not sure

Python

```

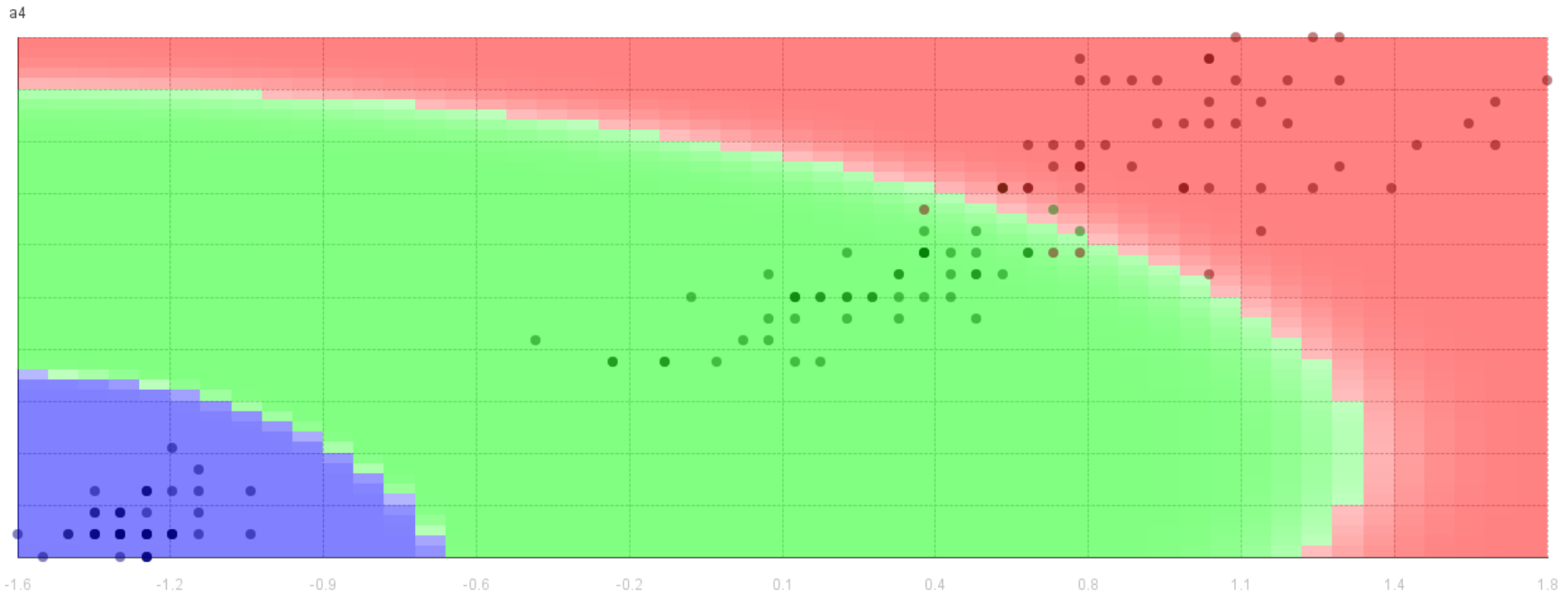
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB

# Train classifier
estimator = MultinomialNB(alpha=1.0)
estimator.fit(preprocessed_training_data, training_labels)

```

Decision Boundary of Naive Bayes Classifier

- Usually larger coherent areas
- Soft margins with uncertain regions
- Arbitrary (often curved) shapes



Naïve Bayes Discussion

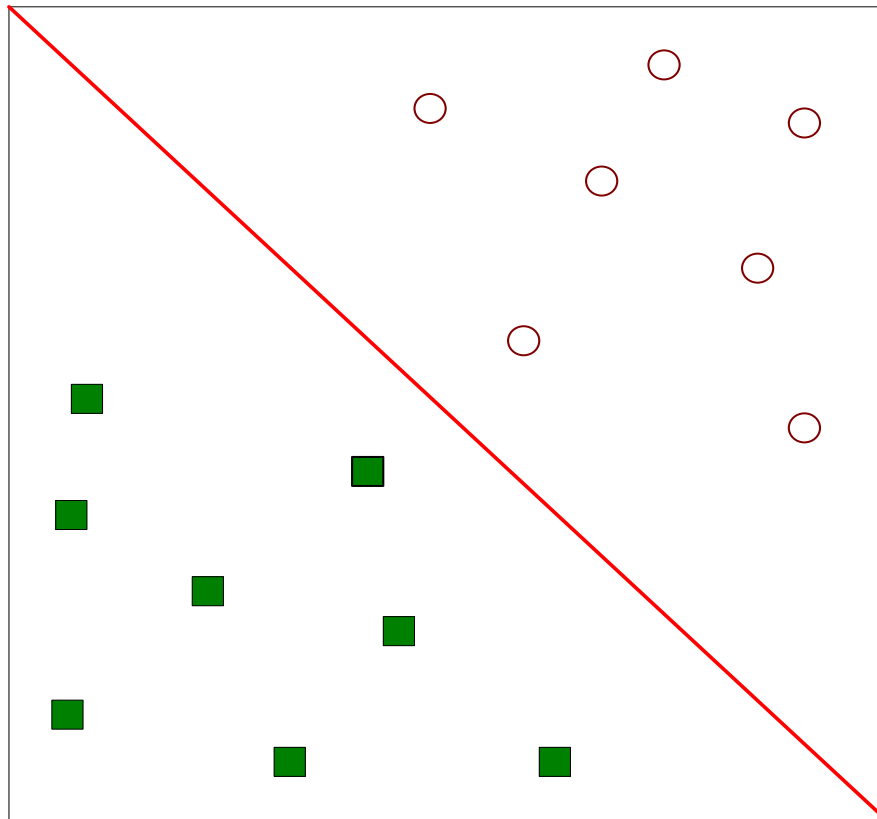
- Naïve Bayes **works surprisingly well**
 - Even if independence assumption is clearly violated
 - Classification doesn't require accurate probability estimates as long as maximum probability is assigned to correct class
- Robust to **isolated noise points** as they will be averaged out
- Robust to **irrelevant attributes** as $P(A_i|C)$ distributed uniformly for A_i
- Adding too many redundant attributes can cause problems
 - Solution: Select attribute subset as Naïve Bayes often works better with just a fraction of all attributes
- Technical advantages
 - learning Naïve Bayes classifiers is computationally cheap (probabilities can be estimated doing one pass over the training data)
 - Storing the probabilities does not require a lot of memory

Support Vector Machines

- Support vector machines (SVMs) are algorithms for learning linear classifiers for
 - **Two class problems** (a positive and a negative class)
 - From examples described by continuous attributes
- SVMs
 - achieve **very good results** especially for high dimensional data
 - invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992

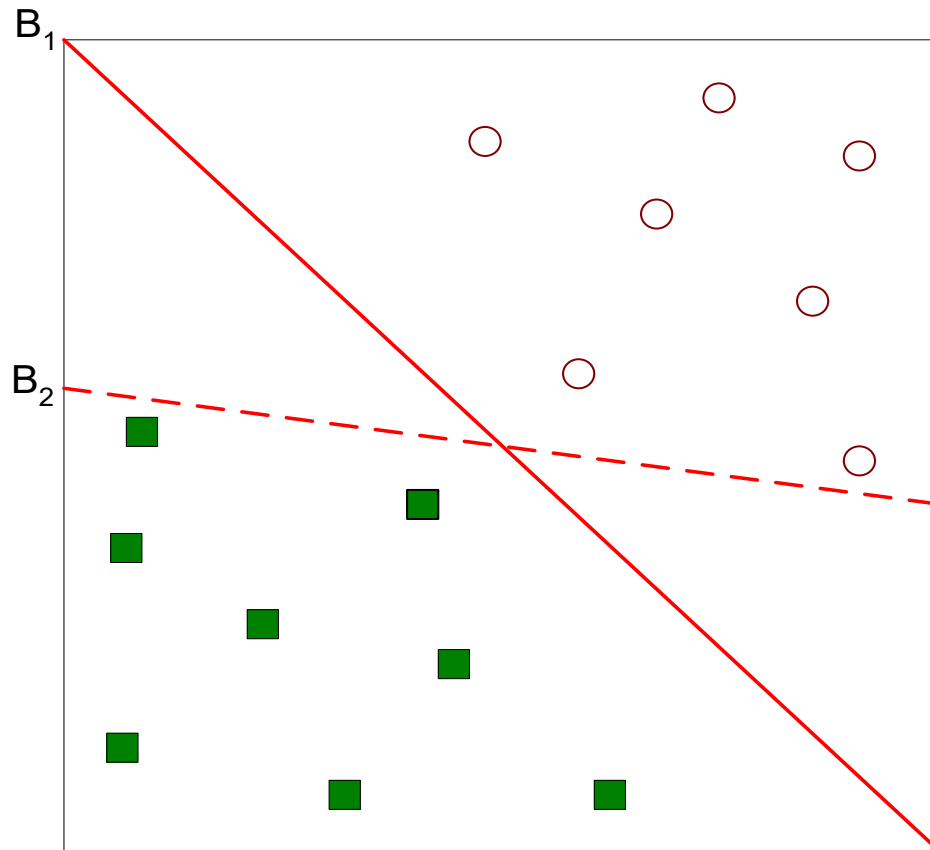
Support Vector Machines

- Find a linear hyperplane (decision boundary) that will separate the data



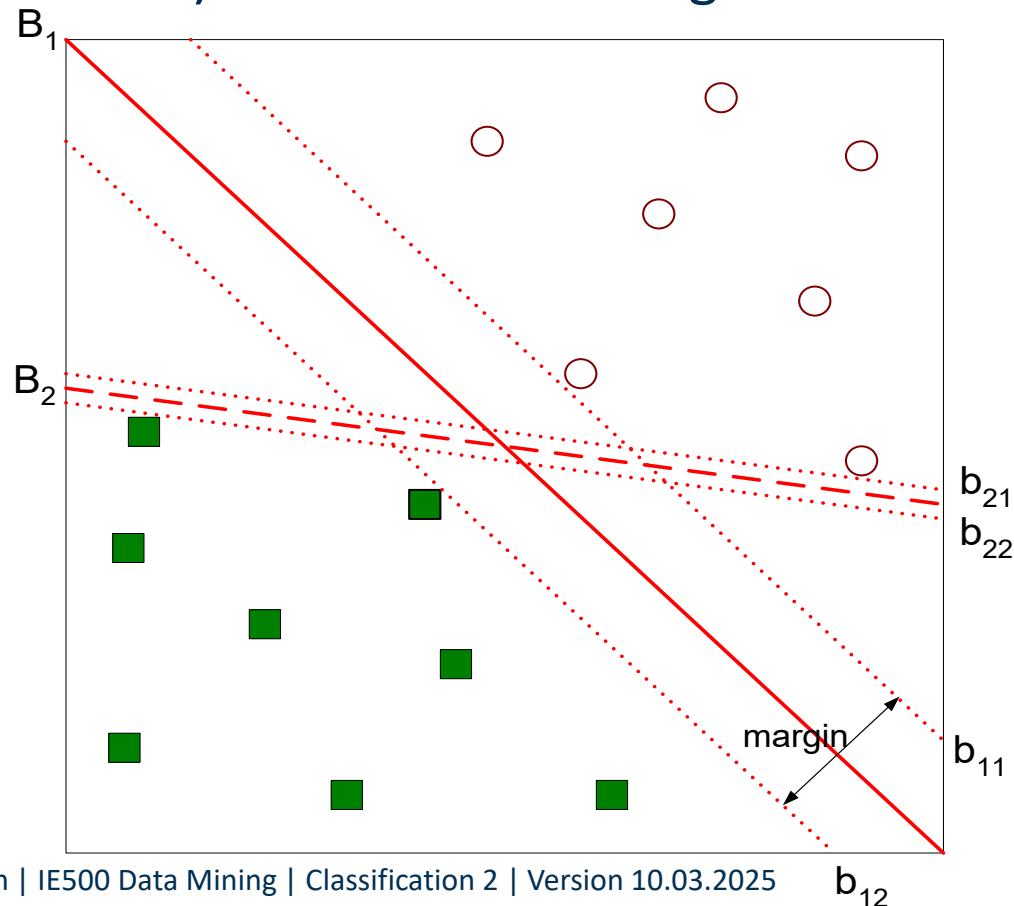
Which Hyperplane is better?

- Which one is better? B_1 or B_2 ? How do you define “better”?



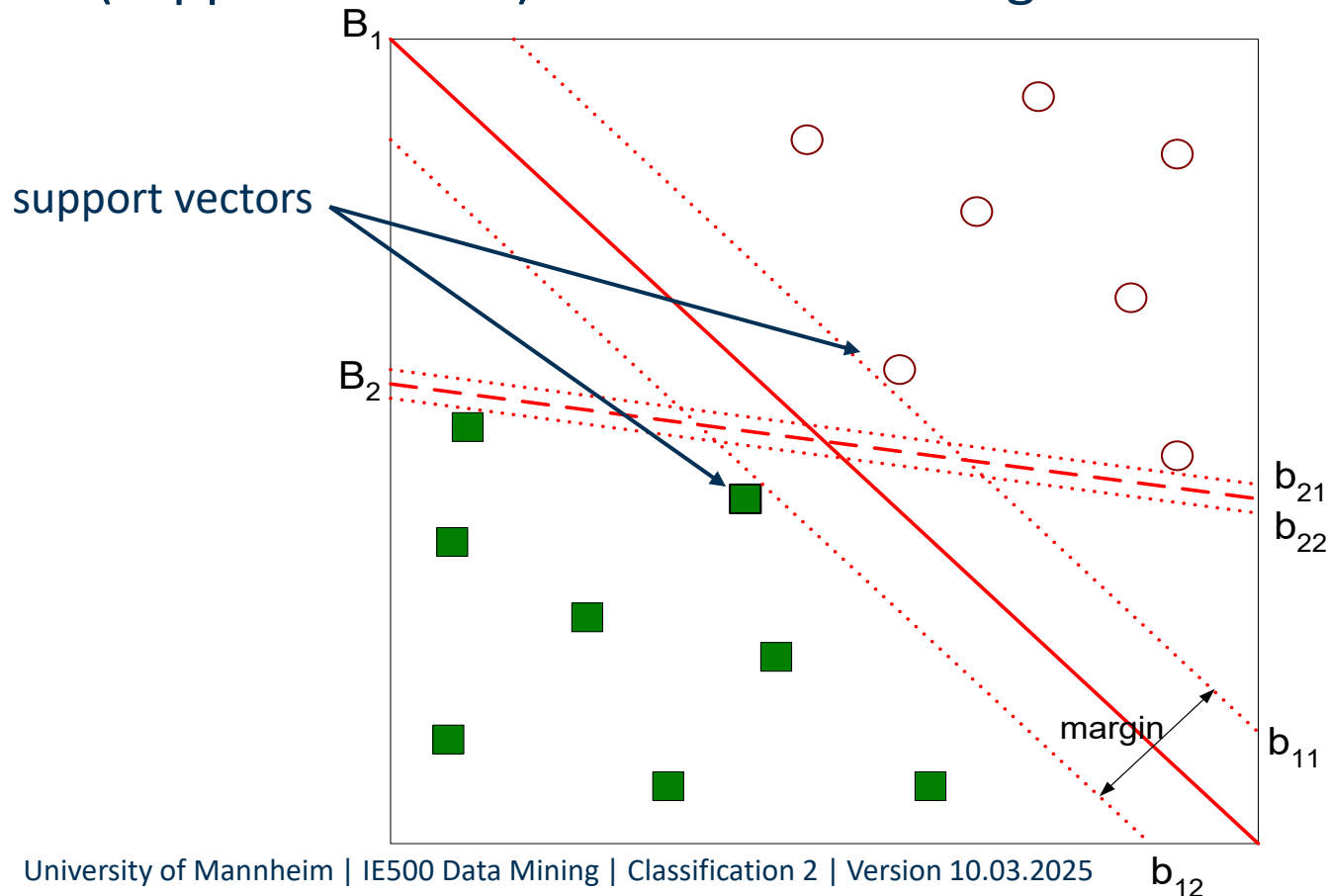
Which Hyperplane is better?

- Find hyperplane **maximizes** the margin to the closest points (support vectors) to avoid overfitting => B1 is better than B2



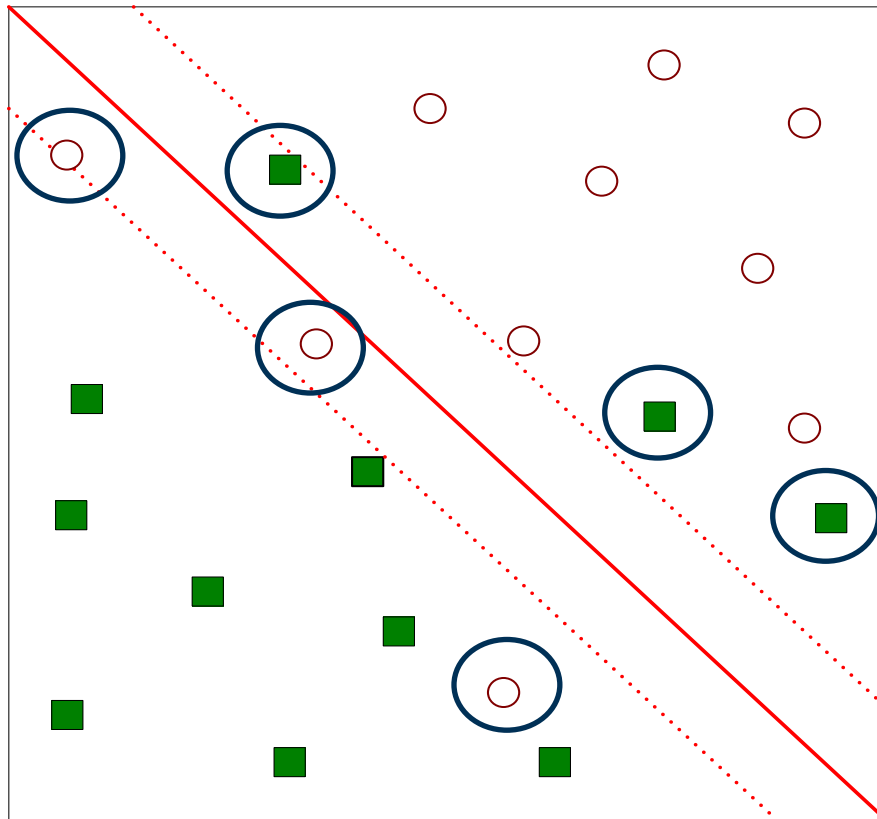
Which Hyperplane is better?

- Find hyperplane **maximizes** the margin to the closest points (support vectors) to avoid overfitting => B1 is better than B2



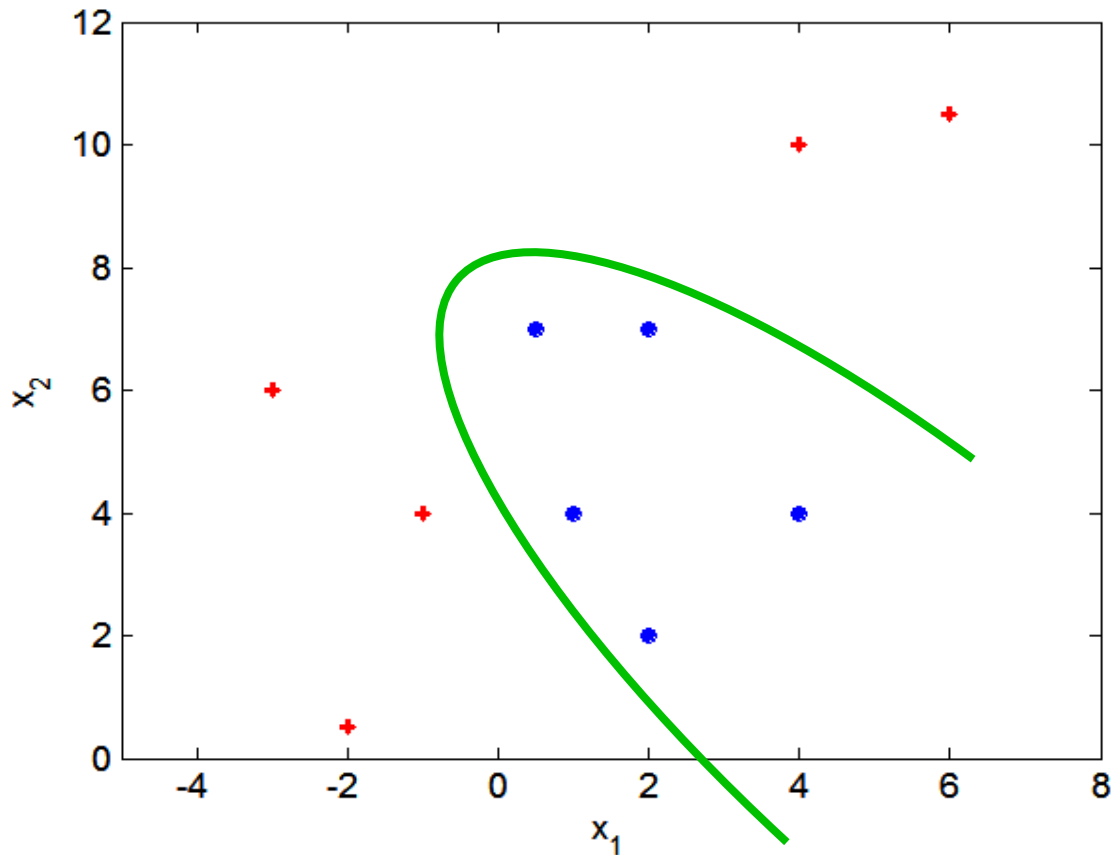
Dealing with Not Linearly Separable Data

- Introduce **slack variables** in margin computation which result in a penalty for each data point that violates decision boundary



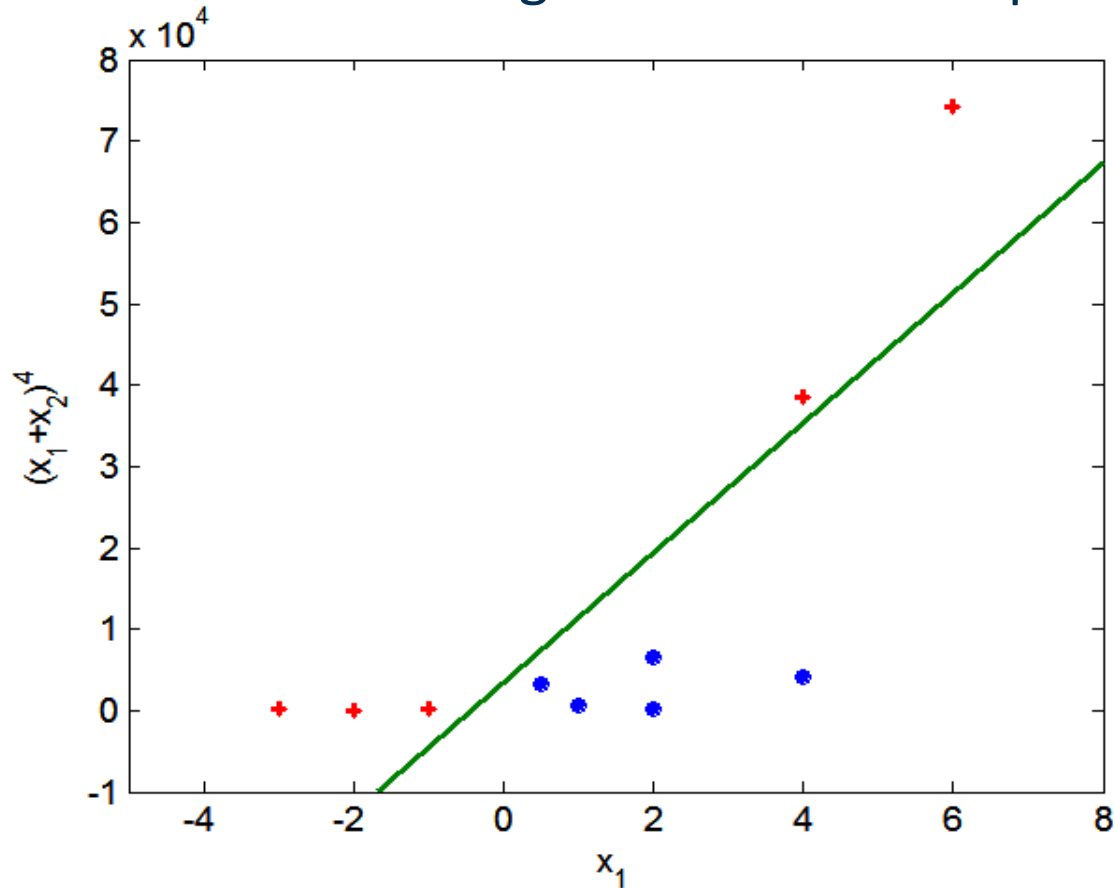
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



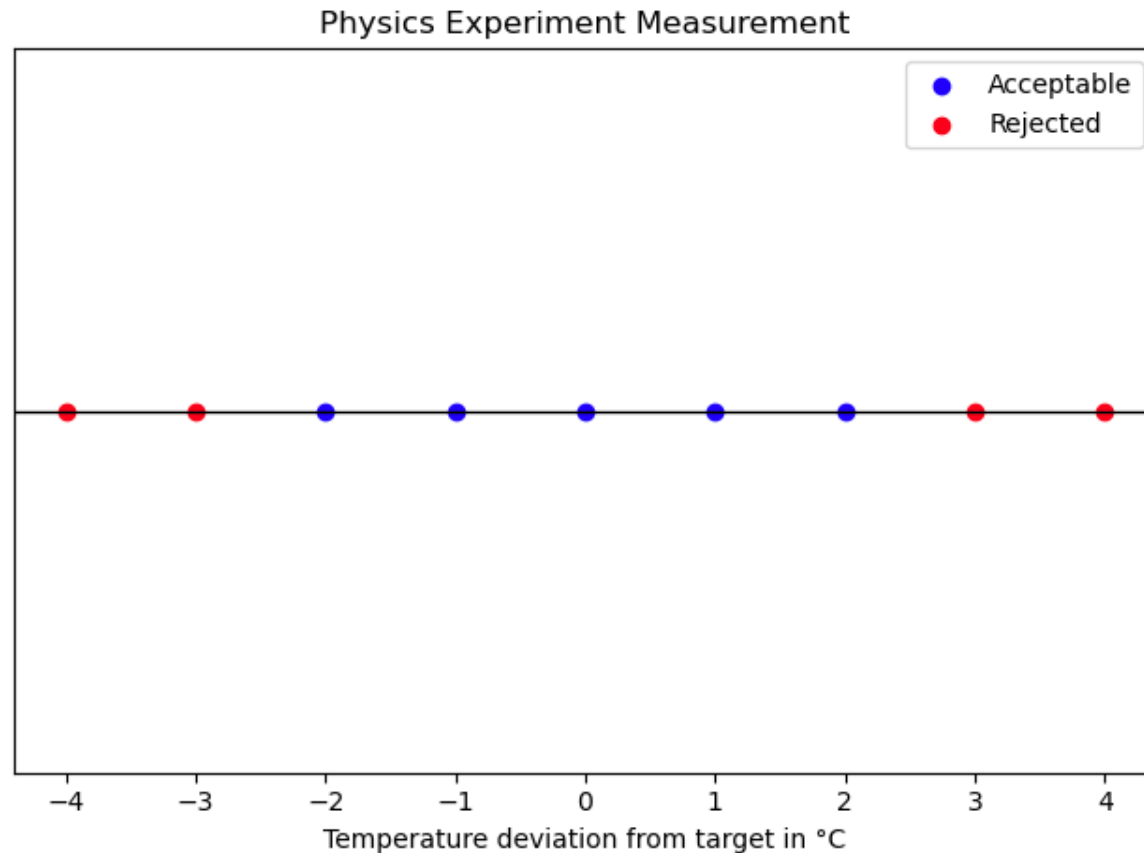
Nonlinear Support Vector Machines

- Transform data into higher dimensional space



Transform data into higher dimensional space - Example

- Transform from 1-D into 2-D
(by adding a y axis which is computed by a simple function)



Nonlinear Support Vector Machines

- Transformation in higher dimensional space
 - Uses so-called Kernel function
 - Different variants: polynomial function, radial basis function, ...
- Finding a hyperplane in higher dimensional space
 - is computationally expensive
 - Kernel trick: expensive parts of the calculation can be performed in lower dimensional space
- Python:

```
from sklearn.svm import SVC

# Train classifier
estimator = SVC(C=1.0, kernel='rbf')
estimator.fit(scaled_training_data, training_labels)
```

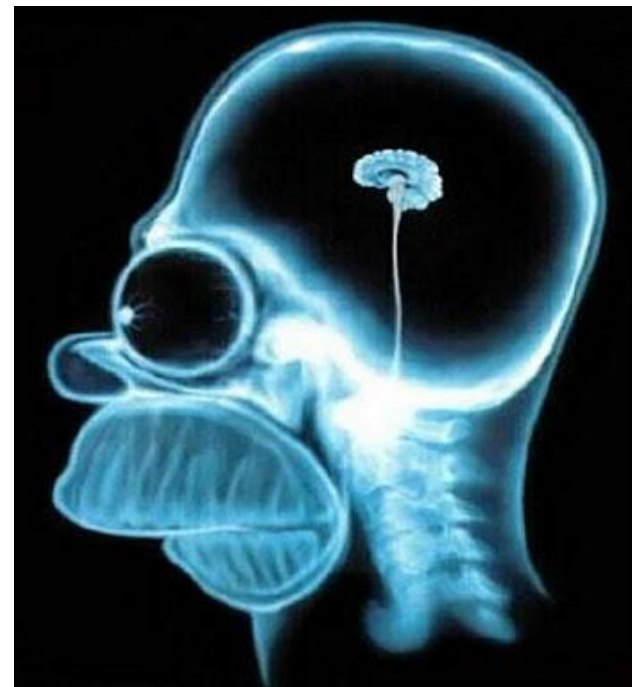
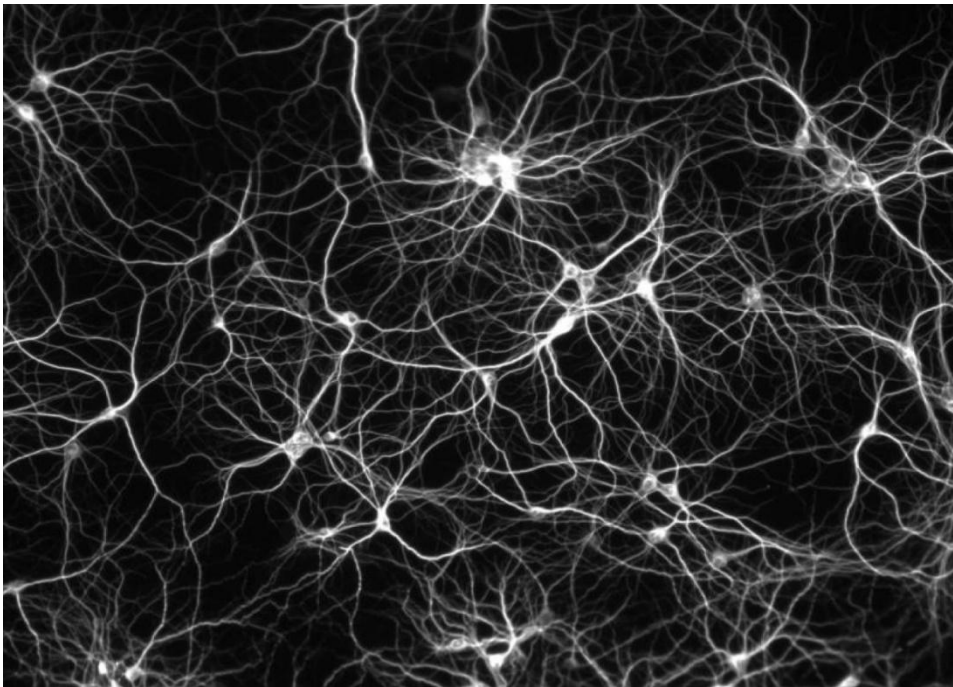
Tuning of SVM

- Instead of randomly trying a few kernel and parameters Hsu et al. proposes a systematic method
 1. Linearly **scaling** each attribute to the range $[-1, +1]$ or $[0, 1]$
 2. Use **RBF Kernel**
 3. Use **cross-validation** to find the best parameter C and γ
 - trying exponentially growing sequences of C and γ
e.g. $C = 2^{-5}, 2^{-3}, \dots, 2^{15}$ $\gamma = 2^{-15}, 2^{-13}, \dots, 2^3$
 4. Use the best parameter C and γ to **train the whole training set**
 5. **Test**

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin:
A Practical Guide to Support Vector Classification

Artificial Neural Networks (ANN)

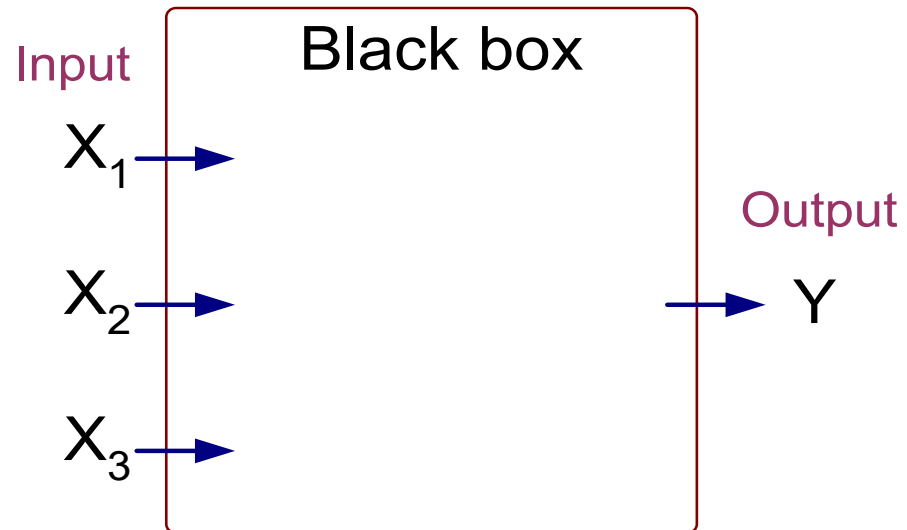
- Inspiration
 - one of the most powerful super computers in the world



Artificial Neural Networks (ANN)

- Function fitting the training data:
Output Y is 1 if at least two of the three inputs are equal to 1

| X_1 | X_2 | X_3 | Y |
|-------|-------|-------|-----|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |

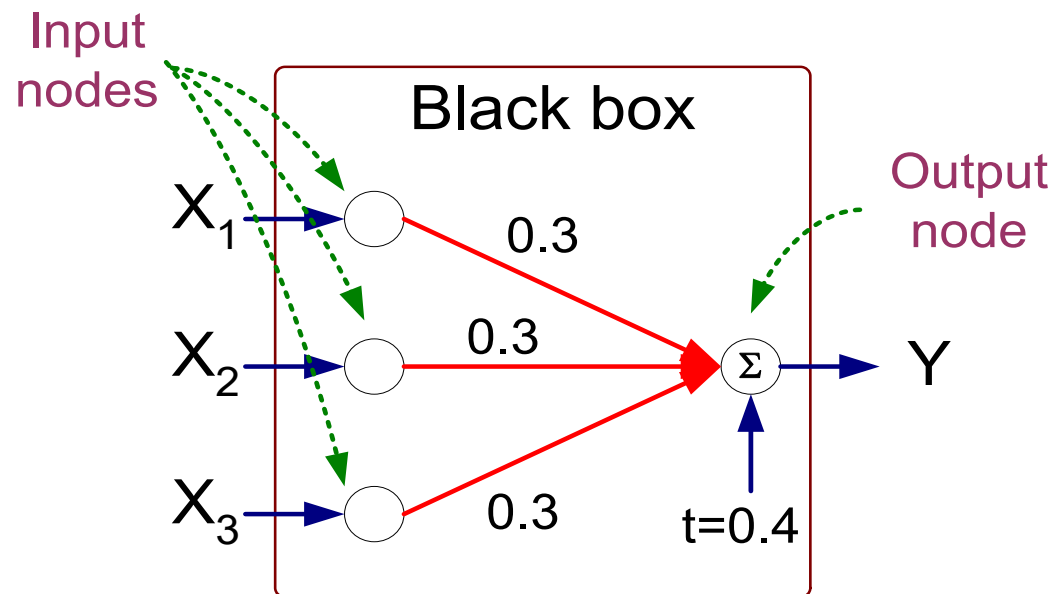


Artificial Neural Networks (ANN)

$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

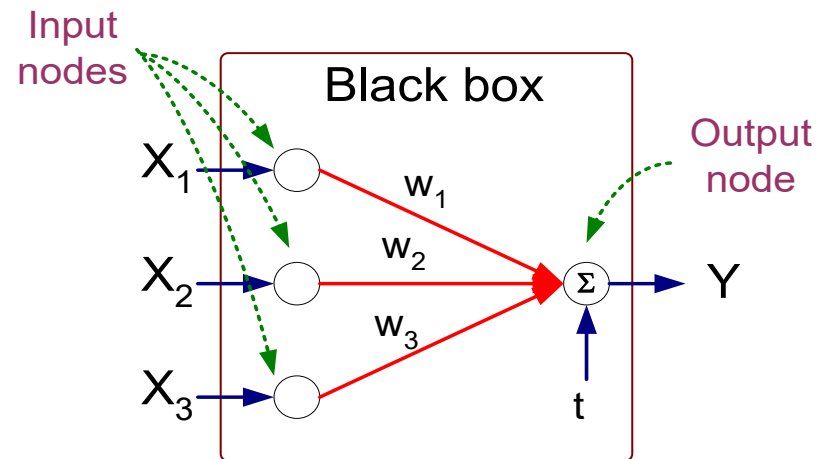
$$\text{Where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

| X_1 | X_2 | X_3 | Y |
|-------|-------|-------|-----|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |



Artificial Neural Networks (ANN)

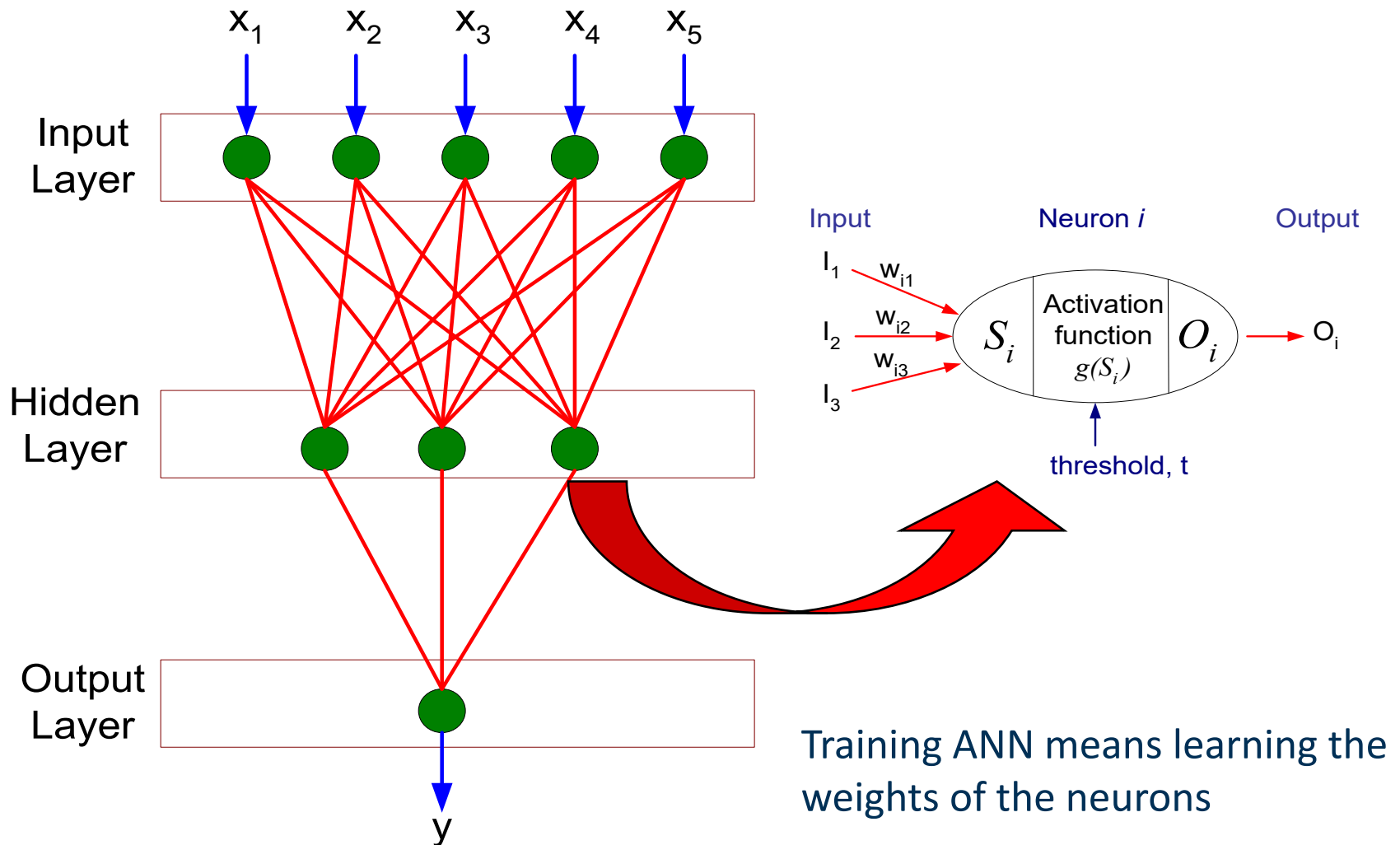
- Model is an assembly of inter-connected nodes (called neurons) and weighted links
- Output node sums up each of its input values according to the weights of its links
- Classification decision:
Compare output node against some threshold t



Perceptron Model

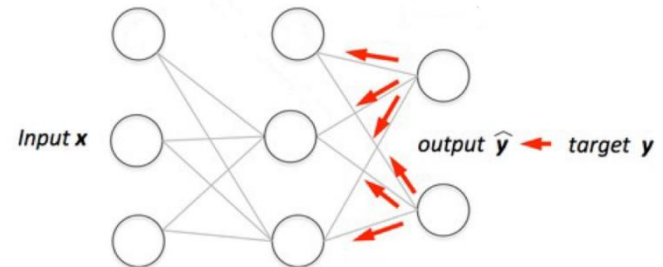
$$Y = I \left(\sum w_i X_i - t > 0 \right)$$

Multi-Layer Artificial Neural Networks



Algorithm for Training ANNs

1. Initialize the weights (w_0, w_1, \dots, w_k), e.g., random or pre-trained
2. Adjust the weights in such a way that the output of ANN is as consistent as possible with class labels of the training examples
 - Objective function: $E = \sum_i [Y_i - f(w_i, X_i)]^2$
 - Find the weights w_i 's that minimize the sum of squared error E
 - using the **back propagation algorithm**
(see Tan/Steinbach: Chapter 6.7,
Gemulla: Machine Learning)
 - Adjustment factor: learning rate



Python

```
from sklearn.neural_network import MLPClassifier

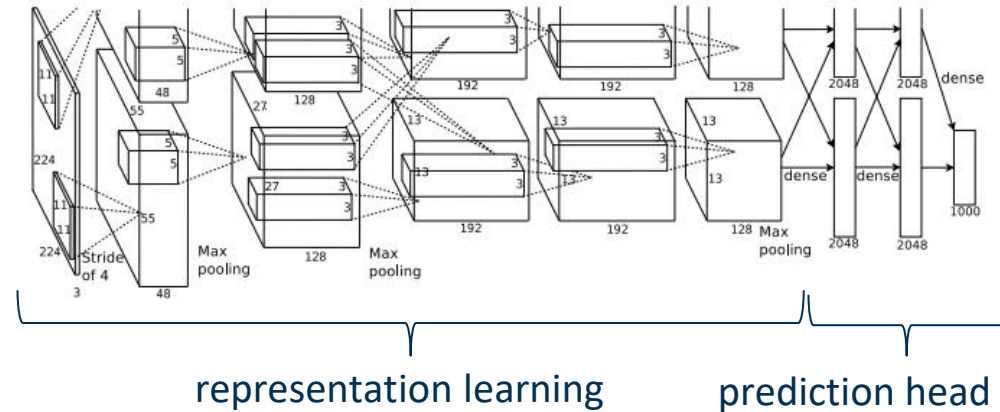
# Train classifier
neuralnet = MLPClassifier(hidden_layer_sizes=(100, ), learning_rate_init=.1)
neuralnet.fit(training_data, training_labels)
```





Overview: Types of Deep Learning Models

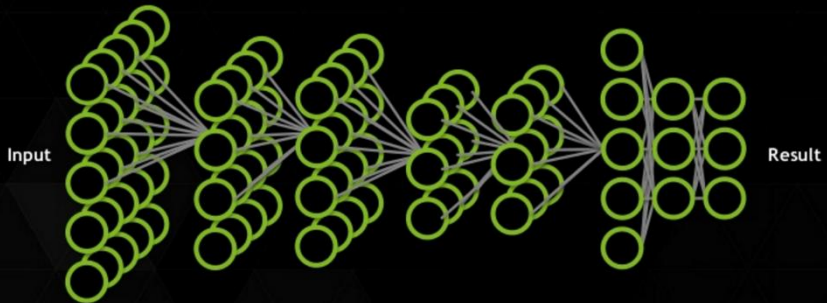
- Convolutional Neural Networks
- Pre-Trained Language Models: BERT
- Generative Models: T5, GPT3, DALL·E
- Instruct Models: ChatGPT, LaMDA

Convolutional Neural Networks (CNNs)

- Invented in computer vision
- Combine
 1. Representation learning (convolutions and pooling)
 2. Prediction head (densely connected layers)
- Reduce number of input features via convolutions and pooling
- High capacity of models requires
 - Lots of training data
 - Lots of GPU time



| Raw data | Low-level features | Mid-level features | High-level features |
|---|--|---|---|
|  |  |  |  |



Application components:

Task objective
e.g. Identify face

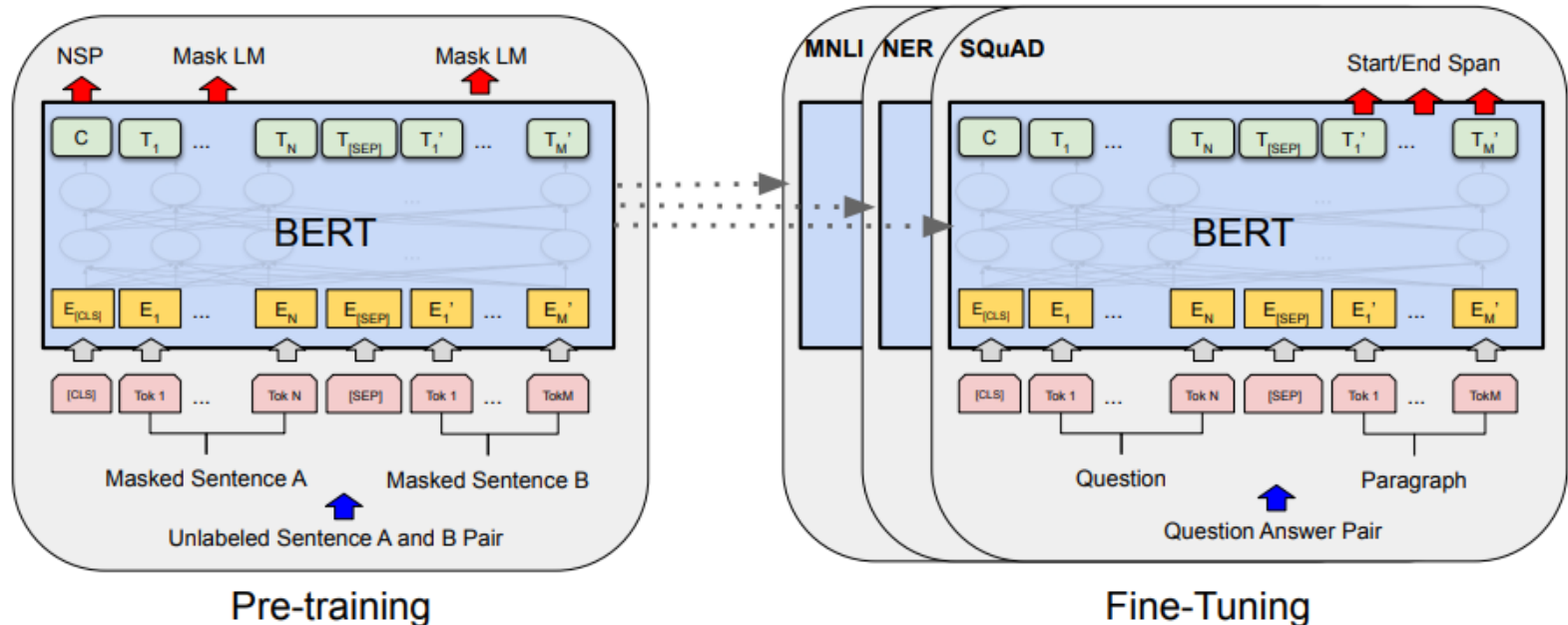
Training data
10-100M images

Network architecture
~10 layers
1B parameters

Learning algorithm
~30 Exaflops
~30 GPU days

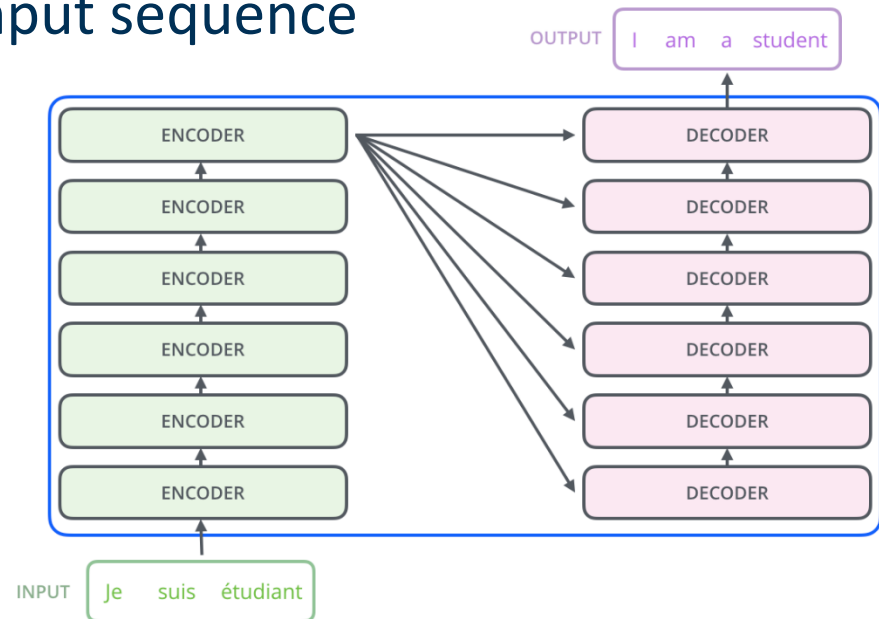
Pre-Trained Language Models

- Introduce pre-training, fine-tuning paradigm
 - Pre-trained on large text corpora
 - Model size: BERT-base 110 million parameters
- Outperform previous models on most NLP tasks



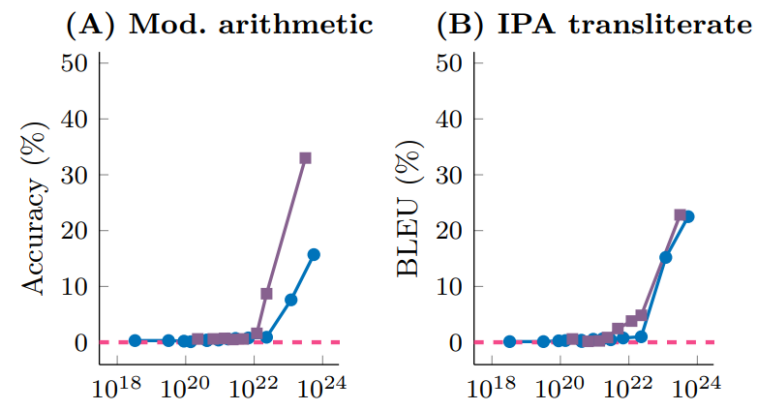
Generative Models

- Use transformer architecture to generate text or images based on embeddings of input sequence
- Models for Text
 - T5, GPT3
- Models for Images
 - DALL·E, Stable Diffusion
- Pretrained on large text and image corpora
 - Web crawls
 - ImageNet, LAION-5B
- Model sizes: 5 to 175 billion parameters
 - accessible mostly via APIs



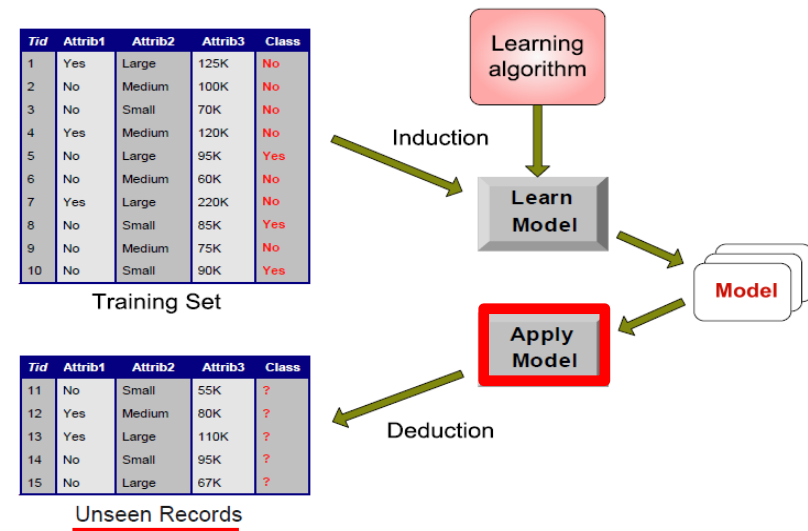
Instruct Language Models

- After being pre-trained on large text corpora, instruct models are fine-tuned with instruction/output pairs
- Show good few-shot performance on wide range of task
 - BIG-bench collects 200+ tasks
- Models show emergent abilities
 - Can perform tasks they were not directly trained for
- Prompt design and in-context learning determine performance of frozen models



Model Evaluation

- Central Question:
 - How good is a model at classifying unseen records?
(generalization performance)
- Last week: **Evaluation Metrics**
 - How to measure the performance of a model?
- This week: **Evaluation Methods**
 - How to obtain reliable estimates?

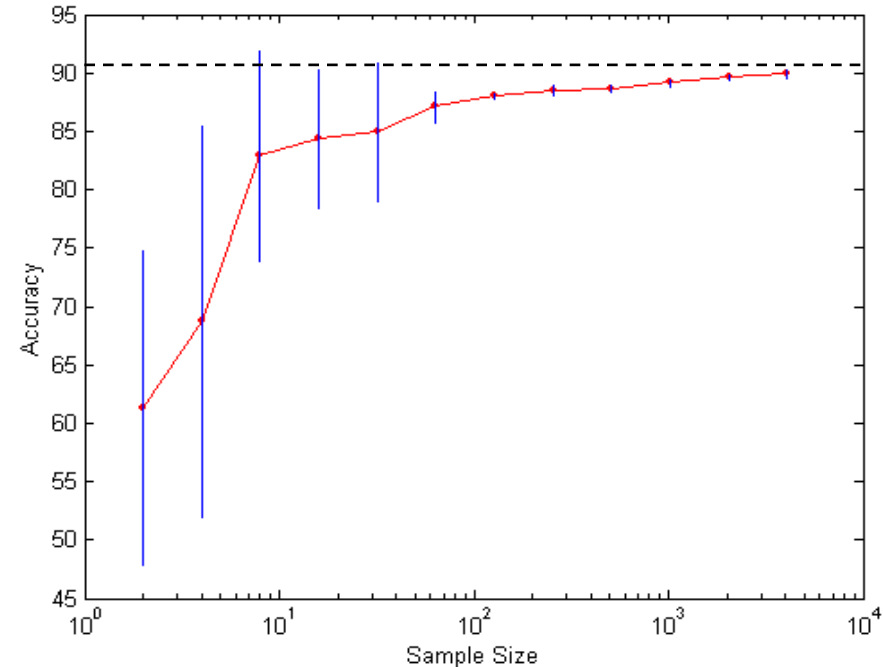


Model Evaluation

- How to obtain a reliable estimate of the generalization performance?
- General approach: Split labeled records into a training set and a test set
- **Never ever test a model on data that was used for training!**
 - Because model has been fit to training data, evaluating on training data does not result in a suitable estimate of the performance on unseen data
 - We need to keep training set and test set strictly separate
- Which labeled records to use for training and which for testing?
- Alternative splitting approaches:
 - Holdout Method
 - Random Subsampling
 - Cross Validation

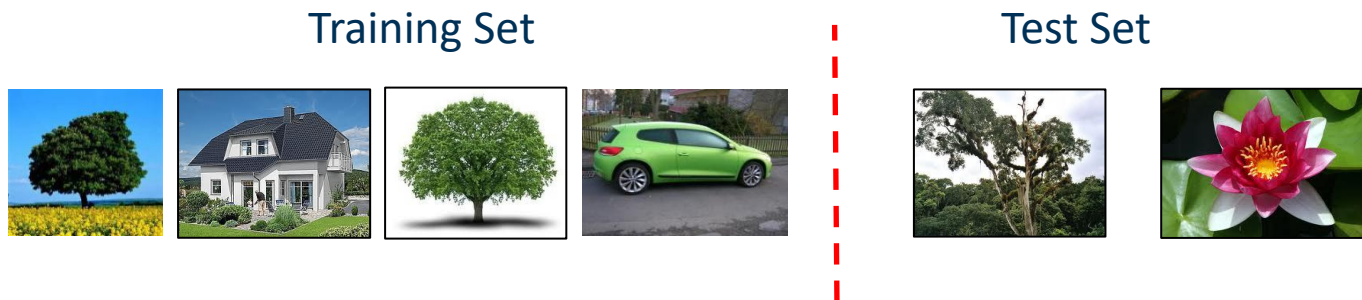
Learning Curve

- The learning curve shows how accuracy changes with growing training set size
- Conclusion:
 - If model performance is low and unstable, get more training data
 - Use labeled data rather for training than testing
- Problem:
 - Labeling additional data is often expensive due to manual effort involved



Holdout Method

- The **holdout method** reserves a certain amount of the labeled data for testing and uses the remainder for training
 - applied when **lots of sample data** is available
- Usually: 2/3 for training , 1/3 for testing (or even better 80% / 20%)

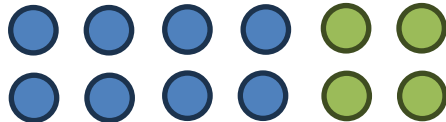


- For imbalanced datasets, random samples might not be representative
 - few or no records of the minority class (aka positive class) in training or test set

Stratified Sampling

- **Stratified sample:** Sample each class independently, so that records of the minority class are present in each sample
 - Make sure that each class is represented with approximately equal proportions in both subsets
 - Other attributes may also be considered for stratification
 - e.g., gender, age, ...

Whole data



Training (3/4)



Testing (1/4)



Random Subsampling

- Holdout estimate can be made more reliable by repeating the process with different subsamples
 - In each iteration, a certain proportion is randomly selected for training
 - The performance of the different iterations is averaged



- Still not optimal as the different test sets may overlap
 - Problem: some outliers might always end up in the test sets
 - Problem: important records for learning (red tree) might always be in test sets

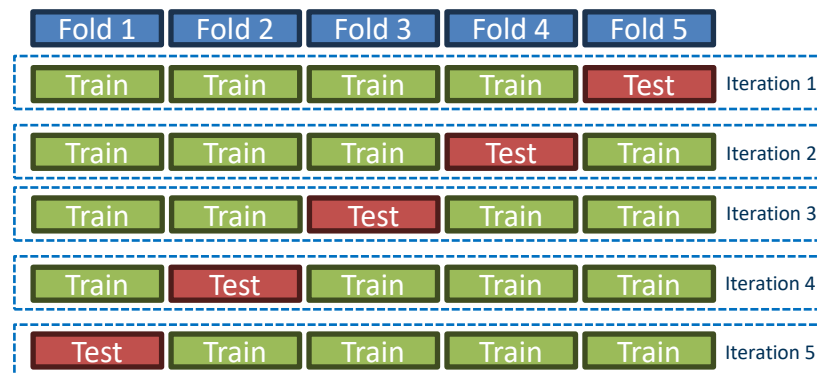
Leave One Out

- Iterate over all examples
 - Train a model on all examples but the current one
 - Evaluate on the current one
- Yields a very accurate estimate
- Uses as much data for training as possible
 - But is computationally infeasible in most cases
- Imagine: a dataset with a million instances
 - One minute to train a single model
 - Leave one out would take almost two years

Cross-Validation

- Compromise of Leave One Out and decent runtime
- Cross-validation avoids overlapping test sets
 - First step: data is split into k subsets of equal size
 - Stratification may be applied
 - Second step: each subset in turn is used for testing and the remainder for training
- This is called k -fold cross-validation
- The error estimates are averaged to yield an overall error estimate

5-fold cross-validation



Cross-Validation

- Frequently used value for k : 10
 - Why ten? Extensive experiments have shown that this is a good choice to get an accurate estimate
 - Often the subsets are generated using stratified sampling (to deal with class imbalance)
 - Recent works on very large models have lead to a tendency of lowering that value (default value in scikit-learn is 5)

Python

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score

# Specify how examples are split
cross_val = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

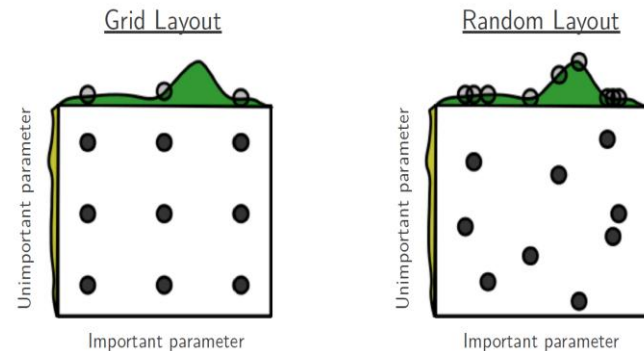
# Run cross-validation and calculate performance metric
accuracy = cross_val_score(estimator, data, target, cv=cross_val, scoring='accuracy')
```

Hyperparameter Selection

- A **hyperparameter** is a parameter which influences the learning process and whose value is set **before the learning begins**
 - pruning thresholds for trees and rules
 - gamma and C for SVMs
 - learning rate, hidden layers for ANNs
- By contrast, **parameters** are learned during training / from training data
 - weights in an ANN, probabilities in Naïve Bayes, splits in a tree
- Many methods work poorly with the default hyperparameters 😞
- How to determine good hyperparameters?
 - Manually play around with different hyperparameter settings
 - Have your machine automatically test many different settings (hyperparameter optimization)

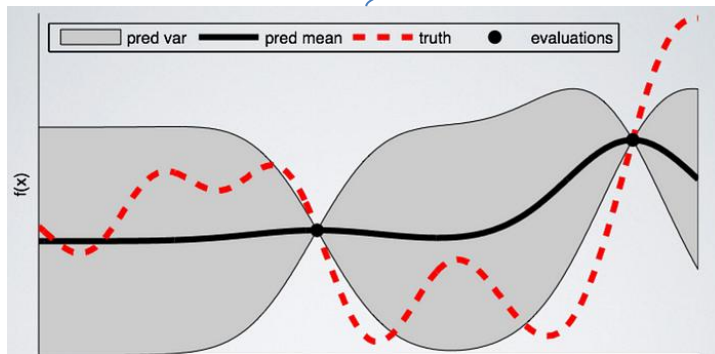
Hyperparameter Optimization

- Goal: Find the combination of hyperparameter values that results in learning the model with the lowest generalization error
- How to determine the parameter value combinations to be tested?
 - **Grid Search:** Test all combinations in user-defined ranges
 - **Random Search:** Test combinations of random parameter values
- Paper from 2012 (Bergstra and Bengio):
 - Grid search may easily miss best parameters
 - some hyperparameters are pretty sensitive e.g., 0.02 is a good value, but 0 and 0.05 are not
 - Random search often yields better results



Hyperparameter Optimization

- **Evolutionary Search**
 - Keep specific parameter values that worked well
- **Bayesian optimization**
 - Hyperparameter tuning as a learning problem:
 - Given a set of hyperparameters p , predict evaluation score s of model
 - The prediction model is referred to as a **surrogate model** or oracle
 - Training and evaluating an actual model is costly



Surrogate Model

“test these
hyperparameters, please”

“here’s the performance
of those hyperparameters”



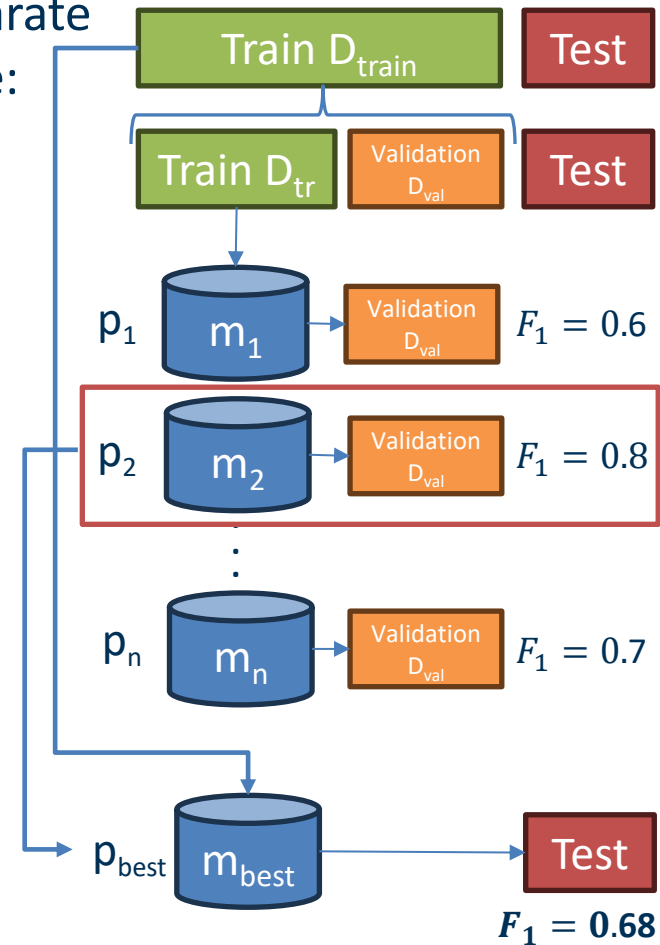
Actual Model

Hyperparameter Optimization

- Done:
 - Which hyperparameter combinations should be tested
 - Often hundreds of combinations are tested
 - reason for cloud computing
- Now:
 - **Model Selection:** From all learned models M , select the model m_{best} that is expected to generalize best to unseen records
 - On which data should the model be tested?

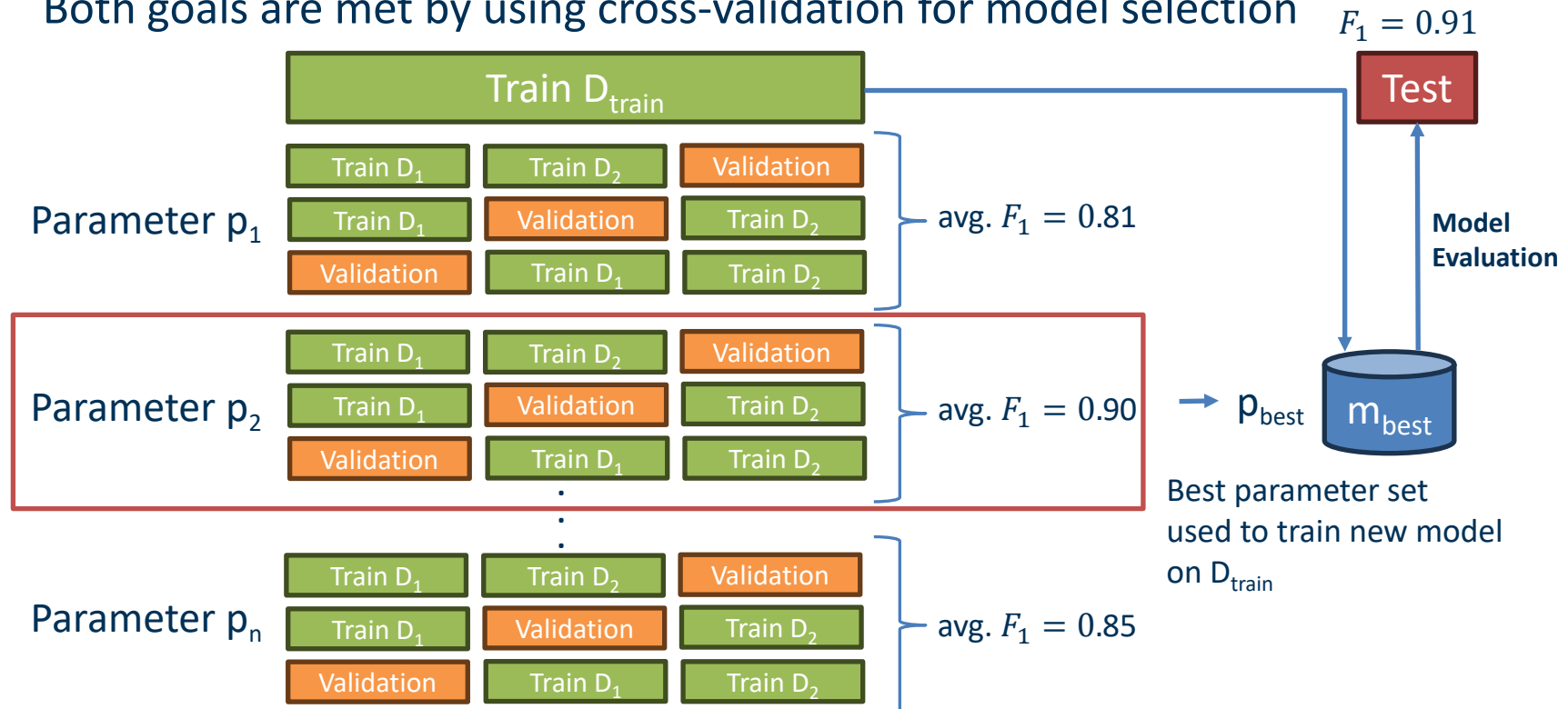
Model Selection Using a Validation Set

- Keep data used for model selection strictly separate from data used for model evaluation, otherwise:
 - Selected model m_{best} will overfit to test set
 - Estimate of generalization error is too optimistic
- Method to find the best model:
 1. Split training set D_{train} into validation set D_{val} and training set D_{tr}
 2. Learn models m_i on D_{tr} using different hyperparameter value combinations p_i
 3. Select best parameter values p_{best} by testing each model m_i on the validation set D_{val}
 4. Learn final model m_{best} on complete D_{train} using the parameter values p_{best}
 5. Evaluate m_{best} on test set in order to get an unbiased estimate of its generalization performance



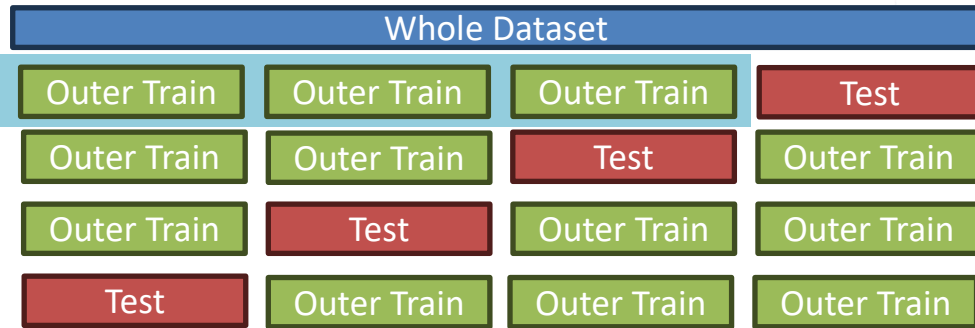
Model Selection Using Cross-Validation

- But wait, we want to
 1. Make sure that all examples are used for validation once
 2. Use as much labeled data as possible for training
- Both goals are met by using cross-validation for model selection



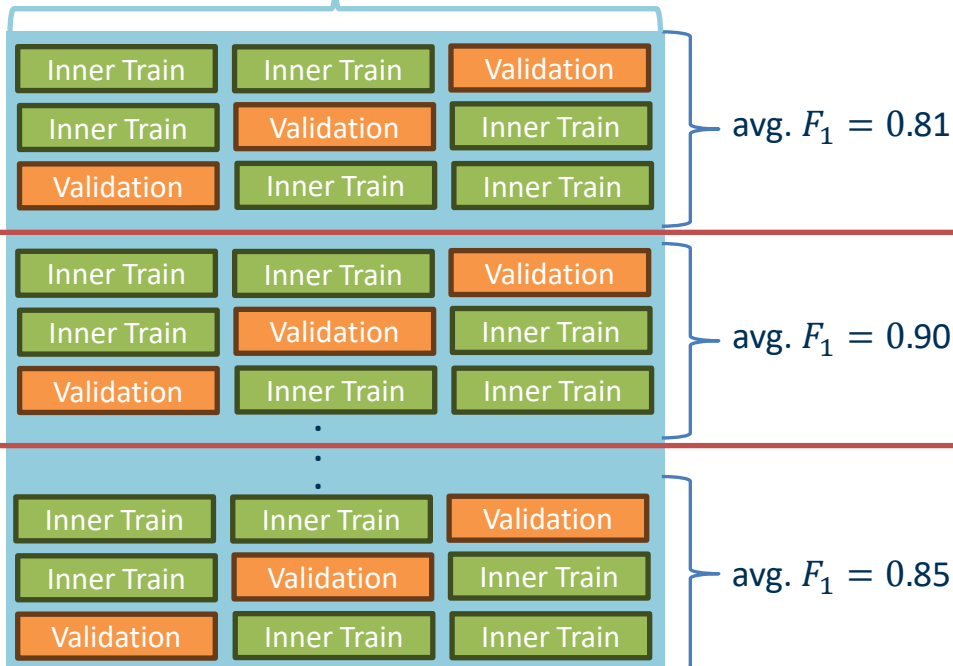
Nested Cross-Validation

Outer Cross Validation:
Model evaluation
(generalization error is estimated)



avg. $F_1 = 0.87$

Inner Cross Validation(s):
Model selection (best hyperparameters)



avg. $F_1 = 0.81$

avg. $F_1 = 0.90$

avg. $F_1 = 0.85$



Best parameter set used to train new model on outer train

Nested Cross-Validation in Python

Python

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC

# Specify hyperparameter combinations for search
parameter_grid = {"C": [1, 10, 100, 1000], "gamma": [.001, .01, .1, 1]}

# Create SVM
estimator_svm = SVC(kernel='rbf')

# Create the grid search for model selection
estimator_gs = GridSearchCV(estimator_svm, parameter_grid, scoring='accuracy', cv=5)

# Run nested cross-validation for model evaluation
accuracy_cv = cross_val_score(estimator_gs, dataset, labels, cv=5, scoring='accuracy')
```

scikit-learn Documentation: Tuning the hyper-parameters of an estimator

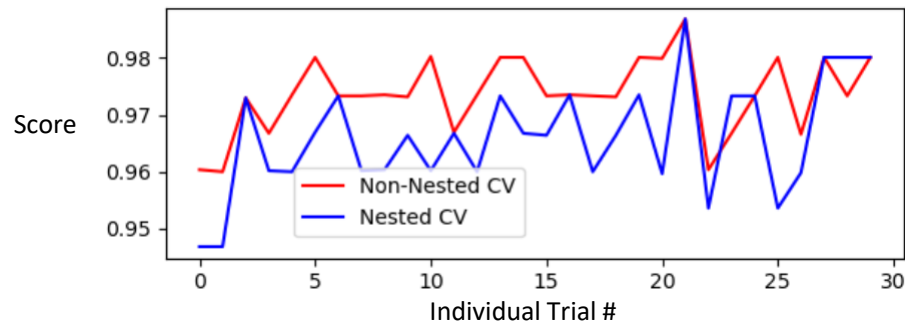
https://scikit-learn.org/stable/modules/grid_search.html

scikit-learn Documentation: Nested versus non-nested cross-validation

https://scikit-learn.org/stable/auto_examples/model_selection/plot_nested_cross_validation_iris.html

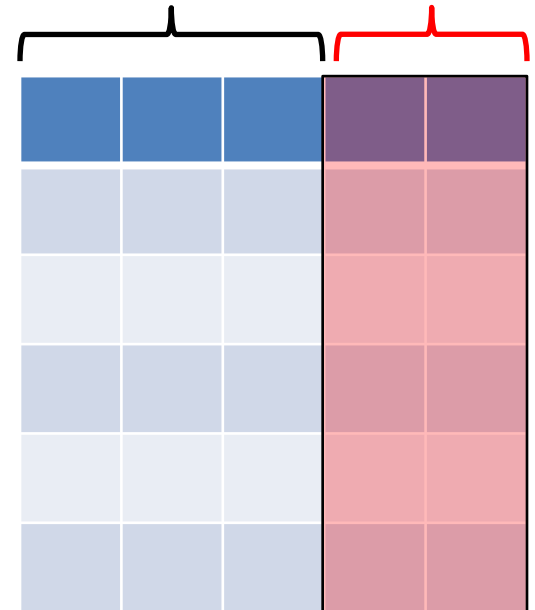
Model Selection - Overview

- For model evaluation with validation set and cross validation
 - Use the test set only once to get one final estimate of the error!
- The more models you train, the better estimate of the error



- Setting: 100 parameter combinations, 5 fold cross validation (inner and outer)
 - Validation set: $|P| + 1 = 101$ models learned
 - Cross Validation: $|folds| * |P| + 1 = 5 * 100 + 1 = 501$ models learned
 - Nested Cross Validation: $|folds_{Outer}| * (|folds_{Inner}| * |P| + 1) = 5 * ((5 * 100) + 1) = 2505$ models learned

Selected Not used



- Some classification methods automatically select the relevant feature subset as part of the learning process
 - e.g. Decision Trees, Random Forests, ANNs, SVMs
- The performance of other methods depends on the subset of the features provided
 - e.g. KNN, Naïve Bayes
- Automated feature selection approaches
 - **Backward selection:** start using all features, remove features, test again
 - **Forward selection:** Find best single feature, add further features, test again
- Use nested cross-validation to estimate the generalization error

Summary: Hyperparameter and Feature Selection

- Hyperparameter selection
 - Default: **Always run hyperparameter optimization!**
 - Otherwise you cannot say that a method does not work for a task
- Feature selection
 - Default: **Check if classification method requires feature selection**
 - If yes, run automated feature selection
- Model selection
 - Default: **Use nested cross-validation**
 - If computation takes too long: use better hardware, reduce number of folds, reduce parameter search space, sample data to reduce size
 - If exact replicability of results is required: Use single train, validation, test split
- If your dataset is imbalanced
 - don't forget to **balance your training set**, not your test set!

Online Lectures

- This week additional material is about comparing classifiers
- Online lectures are exercise and exam relevant

| Week | Monday(Offline Lecture) | Online Lecture (see Ilias Course) | Thursday (Exercise) |
|------------|---|--------------------------------------|--|
| 01.09.2025 | no lecture | | Introduction to Python (13:45–15:15) |
| 08.09.2025 | Introduction to Data Mining (PDF, 3 MB) | | Intro |
| 15.09.2025 | Preprocessing (PDF, 2 MB) | | Preprocessing |
| 22.09.2025 | Classification 1 (PDF, 2 MB) + Intro to Student Project | Nearest Centroids | Classification 1 |
| 29.09.2025 | Classification 2 | Comparing Classifiers | Classification 2 |
| 06.10.2025 | Regression | Ensembles | Regression |
| 13.10.2025 | Clustering and Anomalies | Hierarchical Clustering | Clustering |
| 20.10.2025 | Feedback on project outlines | Time Series | Time Series |
| 27.10.2025 | Association Analysis and Subgroup Discovery | Multi Modal Data | Association Analysis |
| 03.11.2025 | Project feedback session | | Project Work |
| 10.11.2025 | Project feedback session | | Project Work |
| 17.11.2025 | Project feedback session | | Project Work |
| 24.11.2025 | Project feedback session | | Project Work |
| 01.12.2025 | Q&A | | Project Presentations |

Questions?



Literature for this Slideset

- Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar: Introduction to Data Mining. 2nd Edition. Pearson.
- Chapter 6.4: Naïve Bayes
- Chapter 6.9: Support Vector Machines
- Chapter 6.7 and 6.8: Artificial Neural Networks
- Chapter 3.5: Model Selection
- Chapter 3.7: Presence of Hyper-Parameters

