Data Mining

Classification
- Part 3 -

Outline

1. What is Classification?

2. K-Nearest-Neighbors

3. Decision Trees

4. Model Evaluation

5. Rule Learning

6. Naïve Bayes

7. Support Vector Machines

8. Artificial Neural Networks

9. Hyperparameter Selection

6. Naïve Bayes

- Probabilistic classification technique based on Bayes theorem
  - widely used and especially successful at classifying texts

- 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 $(A_1, A_2, \ldots, A_n)$,
    the goal is to find the class C that maximizes the conditional probability

  $$P(C | A_1, A_2, \ldots, A_n)$$

- Example: Should we play golf?
  - P(Play=yes | Outlook=rainy, Temperature=cool)
  - P(Play=no | Outlook=rainy, Temperature=cool)

- Question: How to estimate these probabilities given training data?

Bayes Theorem

- Thomas Bayes (1701-1761)
  - British mathematician and priest
  - tried to formally prove the existence of God

- Bayes Theorem

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

- useful in situations where P(C|A) is unknown
  while P(A|C), P(A) and P(C) are known or easy to estimate

Bayes Theorem: Evidence Formulation

- Prior probability of event H:
  - probability of event <u>before</u> evidence is seen
  - we play golf in 70% of all cases ➔ P(H) = 0.7

- Posterior probability of event H:
  - probability of event <u>*after*</u> evidence is seen
  - evidence: It is windy and raining ➔ P(H | E) = 0.2

- Probability of event *H* given evidence *E*:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Applying Bayes Theorem to the Classification Task

Evidence = record

Class-conditional probability of evidence

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

Class

← **Prior probability of class**

← **Prior probability of evidence**

1. Compute the probability P(C | A) for all values of C using Bayes theorem.
   - P(A) is the same for all classes. Thus, we just need to estimate P(C) and P(A|C)

2. Choose value of C that maximizes P(C | A).

Example:

$$P(\text{Play}=\text{yes}/\text{Outlook}=\text{rainy},\text{Temp}=\text{cool}) = \frac{P(\text{Outlook}=\text{rainy},\text{Temp}=\text{cool}/\text{Play}=\text{yes})P(\text{Play}=\text{yes})}{P(\text{Outlook}=\text{rainy},\text{Temp}=\text{cool})}$$

$$P(\text{Play}=\text{no}/\text{Outlook}=\text{rainy},\text{Temp}=\text{cool}) = \frac{P(\text{Outlook}=\text{rainy},\text{Temp}=\text{cool}/\text{Play}=\text{no})P(\text{Play}=\text{no})}{P(\text{Outlook}=\text{rainy},\text{Temp}=\text{cool})}$$

Estimating the Prior Probability P(C)

– The prior probability $P(C_j)$ for each class is estimated by

   1. counting the records in the training set that are labeled with class C_j

   2. dividing the count by the overall number of records

– Example:
   - P(Play=no) = 5/14
   - P(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 $P(A \mid C)$ to be reformulated as the product of the individual probabilities $P(A_i \mid C_j)$:

$$P(A_1, A_2, \ldots, A_n \mid C_j) = \prod P(A_n \mid C_j) = P(A_1 \mid C_j) \times P(A_2 \mid C_j) \times \ldots \times P(A_n \mid C_j)$$

P(Outlook=rainy, Temperature=cool | Play=yes) = P(Outlook=rainy | Play=yes) $\times$
P(Temperature=cool | Play=yes)

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

Estimating the Probabilities $P(A_i \mid C_j)$

Outlook	Yes	No	Temperature	Yes	No	Humidity	Yes	No	Windy	Yes	No	Play	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 $P(A_i \mid C_j)$ are estimated by
1. counting how often an attribute value appears together with class C_j
2. dividing the count by the overall number of records belonging to class C_j

Example:
2 times "Yes" together with "Outlook=sunny" out of altogether 9 "Yes" examples
➜ p(Outlook=sunny|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 Day

Unseen record

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Class-conditional probability of the evidence

$$P(yes \mid E) = P(Outlook = Sunny \mid yes)$$
$$\times P(Temperatur\ e = Cool \mid yes)$$
$$\times P(Humidity = High \mid yes)$$
$$\times P(Windy = True \mid yes)$$
$$\times \frac{P(yes)}{P(E)}$$

Probability of class "yes" given the evidence

Prior probability of class "yes"

Prior probability of evidence

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

Classifying a New Day: Weigh the Evidence!

Outlook	Yes	No	Temperature	Yes	No	Humidity	Yes	No	Windy	Yes	No	Play Yes	Play 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	?

Prior probability
Evidence

Choose Maximum

Likelihood of the two classes

For "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

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 $P(A_i|C_j)$
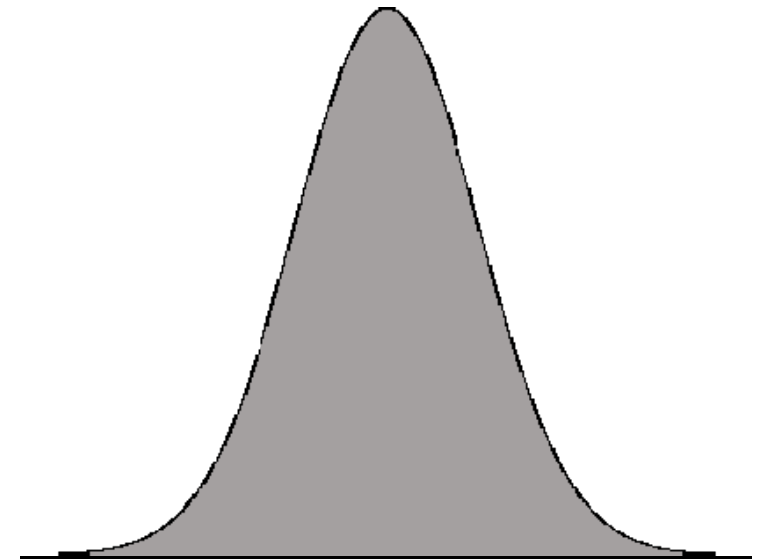
Handling Numerical Attributes

− 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 μ*

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

- *Standard deviation σ*

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2}$$
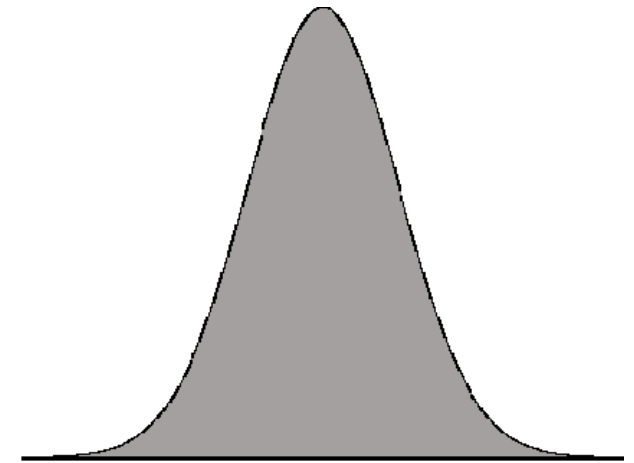
− Both parameters can be estimated from the training data

Statistics for the Weather Data

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:

$$f(temp = 66 \mid yes) = \frac{1}{\sqrt{2\pi}\,6.2}\, e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

Classifying a New Day

Unseen record

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = 2/9 × 0.0340 × 0.0221 × 3/9 × 9/14 = 0.000036

Likelihood of "no" = 3/5 × 0.0291 × 0.0380 × 3/5 × 5/14 = 0.000136

P("yes") = 0.000036 / (0.000036 + 0. 000136) = 20.9%

P("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" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

  Likelihood of "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

  P("yes") = $0.0238 / (0.0238 + 0.0343) = 41\%$

  P("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")

  • class-conditional probability will be zero!

$$P[Out. = overc. \mid no] = \frac{0}{5} = 0$$

− Problem: Posterior probability will also be zero!
No matter how likely the other values are!

$$P[no \mid E] = 0$$

− Remedy: Add 1 to the count for every attribute
value-class combination (*Laplace Estimator*)

− Result: Probabilities will never be zero!
also: stabilizes probability estimates

$$Original : P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$Laplace : P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + |V_i|} \qquad |V_i| \text{ number of values}$$

Naïve Bayes in RapidMiner and Python

RapidMiner

100%

Retrieve Golf

out

Naïve Bayes

tra   mod   res

exa   res

Parameters ✕

💡 Naive Bayes

☑ *laplace correction* ⓘ
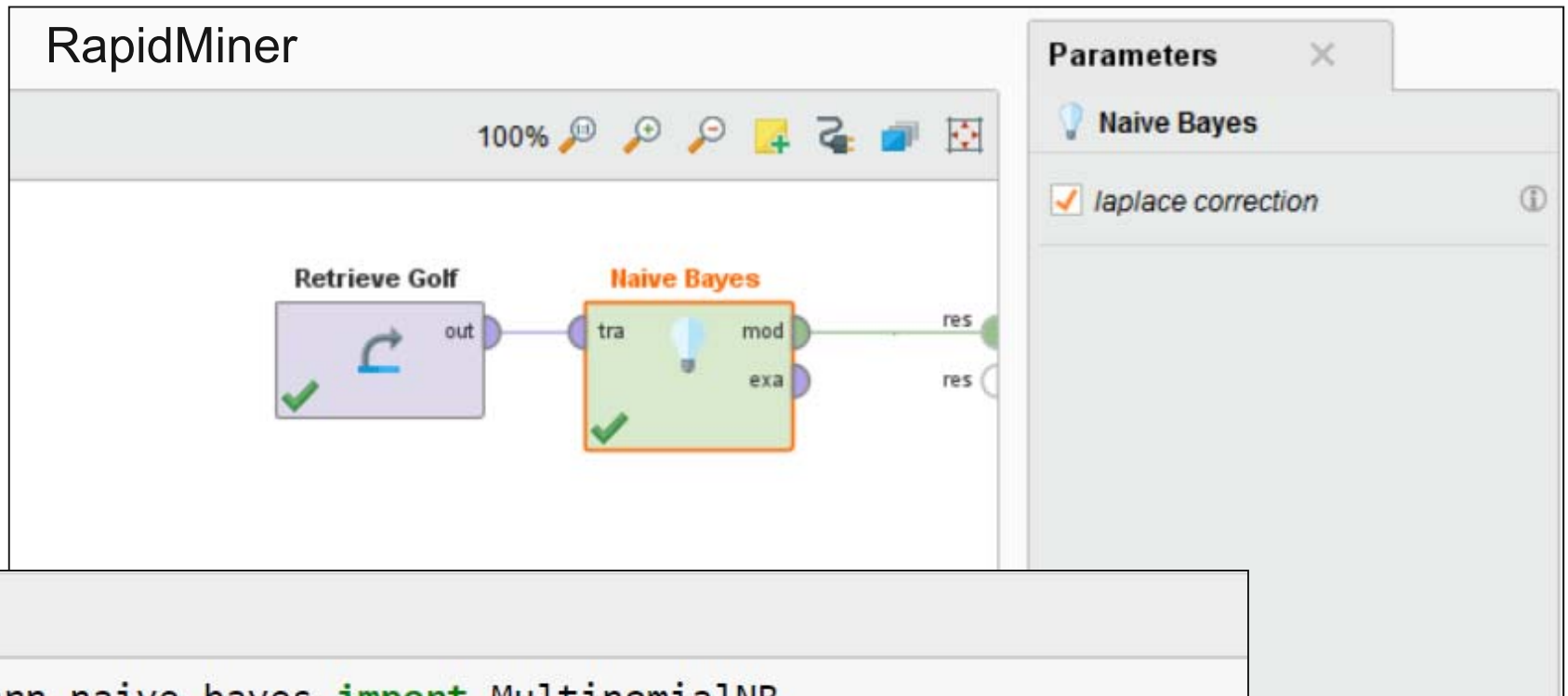
Python

```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)
```

Naïve Bayes in RapidMiner: Probability Distribution Table

Characteristics of Naïve Bayes

– Naïve Bayes works surprisingly well for many classification tasks
- even if independence assumption is clearly violated
- Why? Because 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 $P(A_i \mid C)$ 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 as probabilities can be estimated doing one pass over the training data
- Storing the probabilities does not require a lot on memory

7. 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

- SVMs were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992

Support Vector Machines

- SVMs find a linear hyperplane (decision boundary) that will separate the data
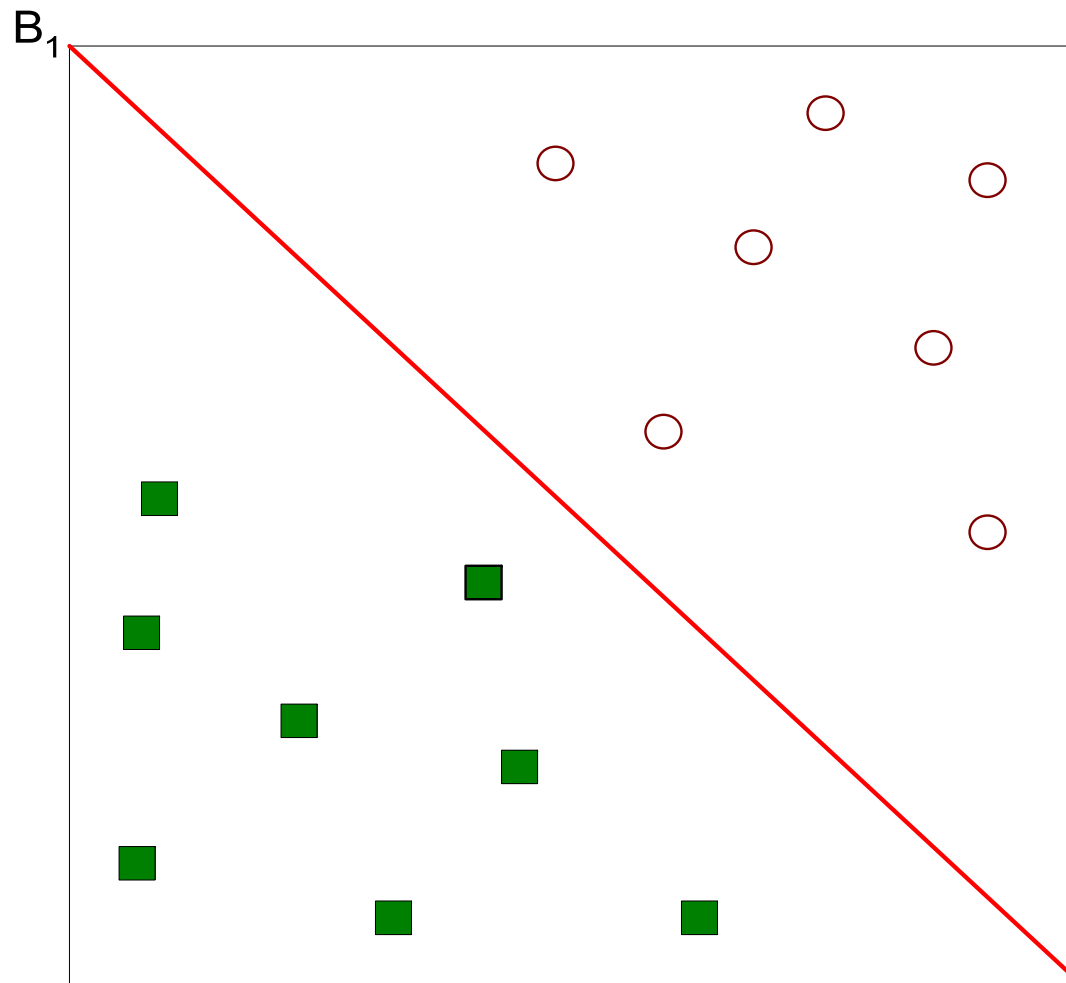
Support Vector Machines
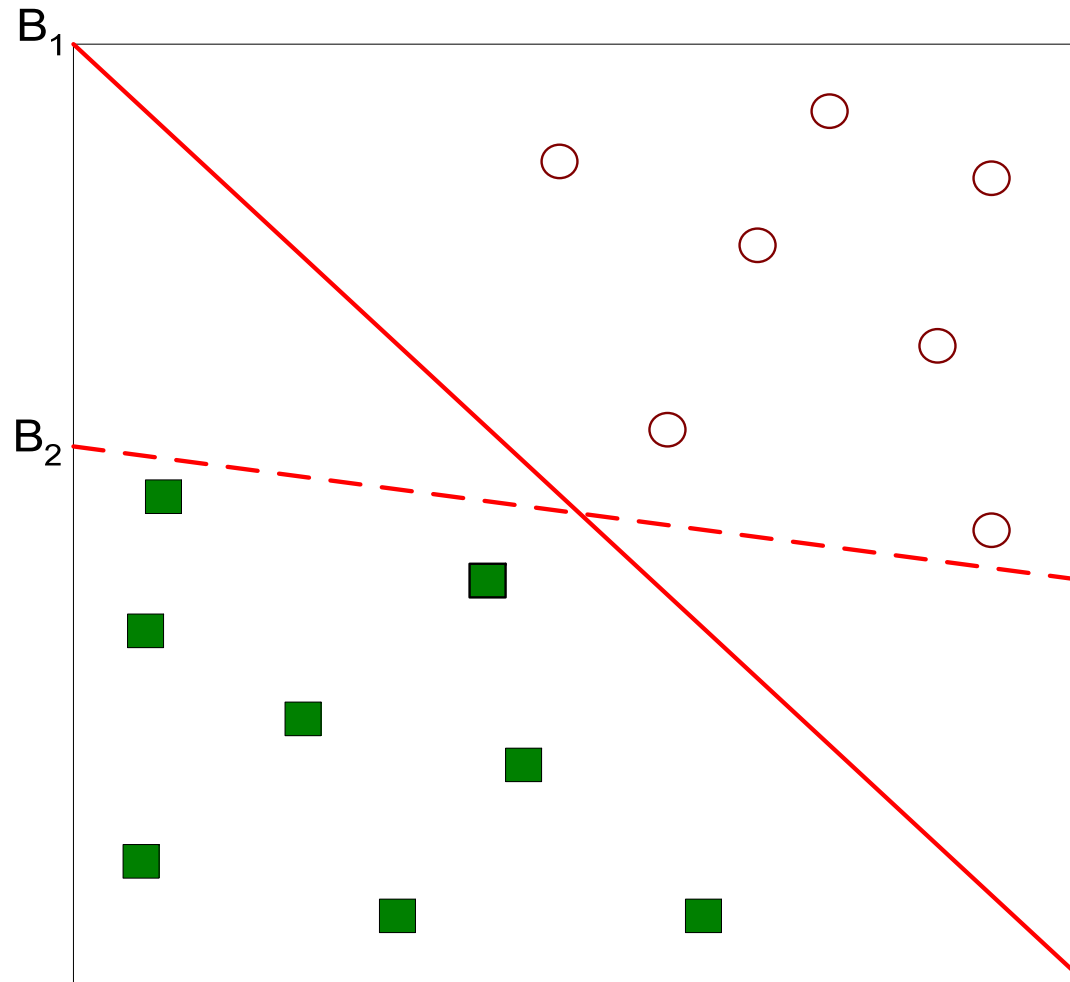
- Which one is better? B1 or B2?
- How do you define "better"?

Which Hyperplane is better?

- In order to avoid overfitting and to generalize for unseen data, SVMs find the hyperplane that maximizes the margin to the closest points (support vectors)

- Visual solution:
  - B1 is better than B2

- Mathematical solution:
  - constrained optimization that can be solved using quadratic programming
  - See Tan/Steinbach/ Kumar, Chapter 6.9
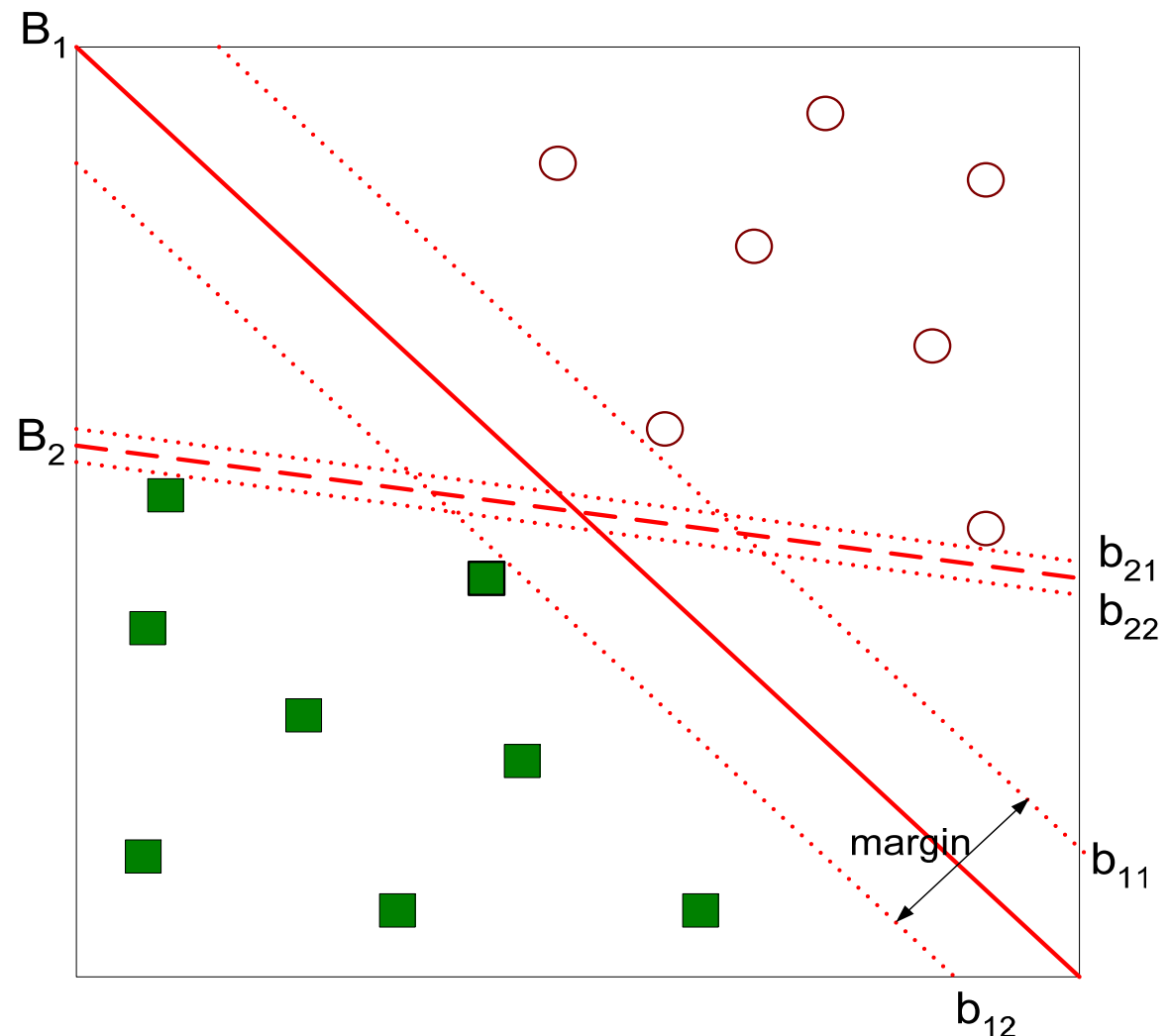
Dealing with Not Linearly Separable Data

– What if the problem is not linearly separable due to noise points?
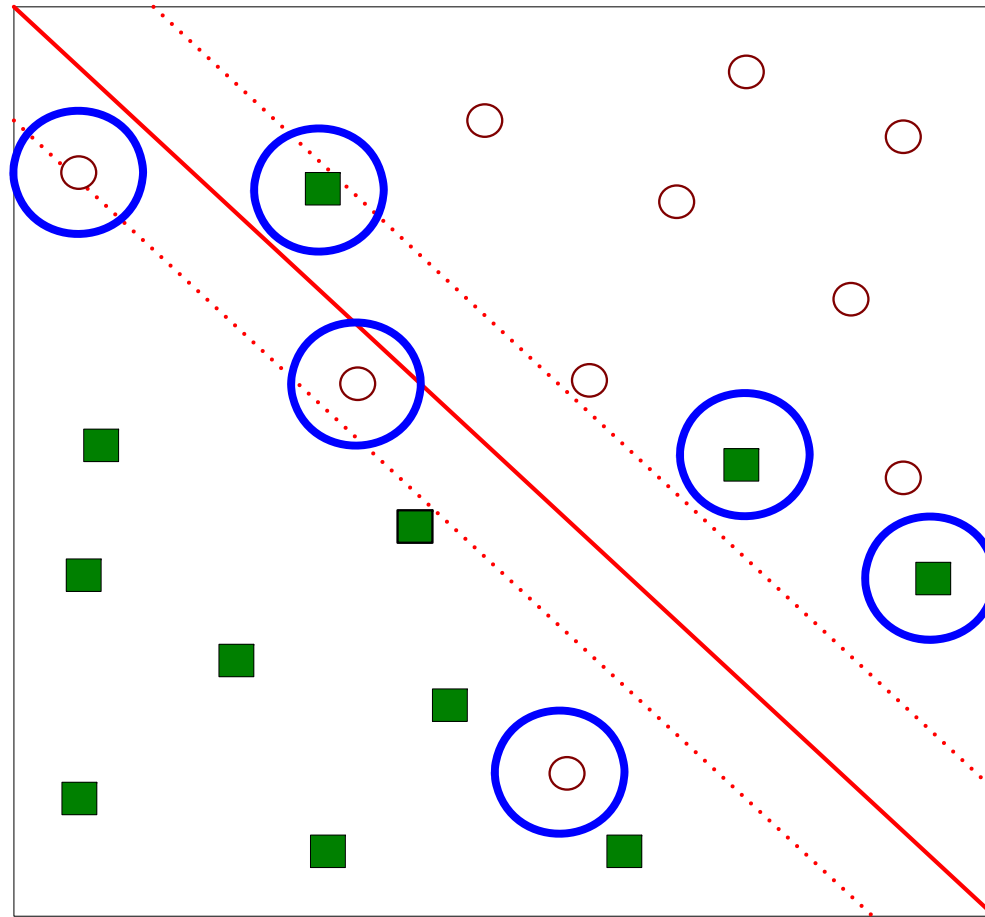
– Solution: Introduce slack variables in margin computation which result in a penalty for each data point that violates decision boundary

Dealing with Non-Linear Decision Boundaries

- Problem: What if decision boundary is not linear?

- Solution: Transform data into higher dimensional space where there is a linear separation

  - details: see Tan/Steinbach/Kumar, Chapter 6.9

  - different types of kernel functions are used for this transformation
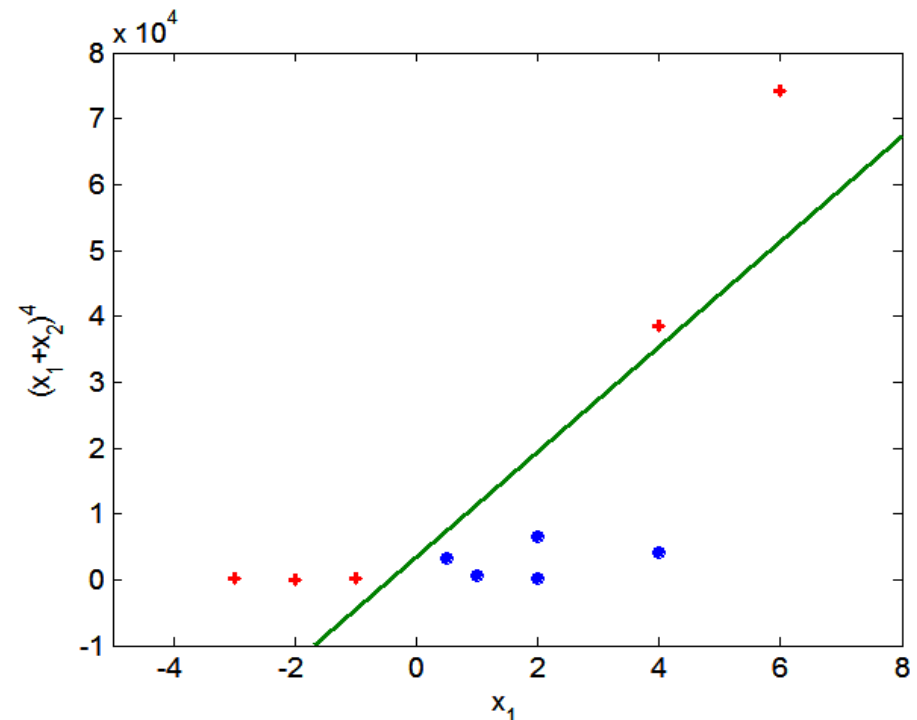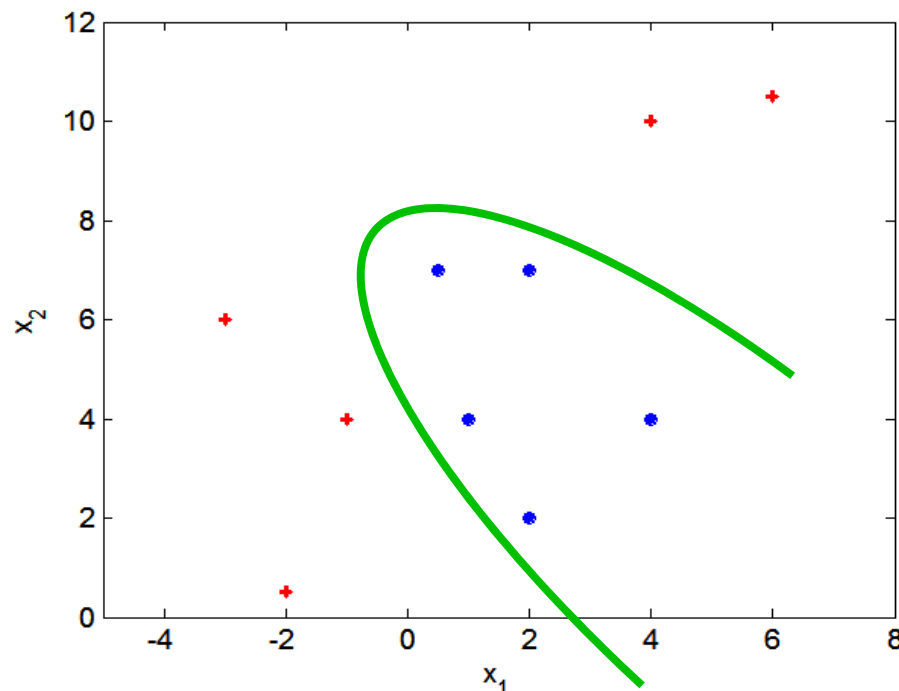
Characteristics of Support Vector Machines

– SVMs were often the most successful classification technique for high dimensional data before DNNs appeared

– Application areas of SVMs include

- Text classification

- Computer vision, e.g face identification

- Handwritten digit recognition

- SPAM detection

- Bioinformatics

– Hyperparameter selection often has a high impact on the performance of SVNs

- see next slide

SVMs in RapidMiner and Python

Tuning a SVM

1. Transform all attributes to numeric scale
2. Normalize all value ranges to [0,1]
3. Use the RBF kernel function
4. Use nested cross-validation to find the best values for the parameters
   1. C = weight of slack variables (Range: 0.03 to 30000)
   2. gamma = kernel parameter (Range: 0.00003 to 8)

More details on tuning: Hsu, et al: A Practical Guide to Support Vector Classification.
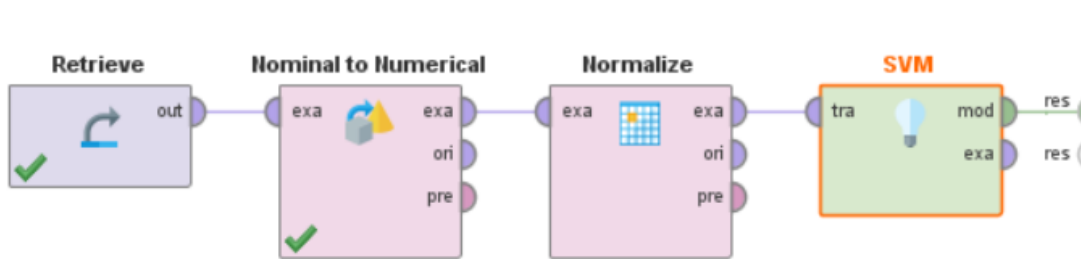
RapidMiner

Parameters ×

💡 **SVM (Support Vector Machine (LibSVM))**

svm type	C-SVC ▼ ⓘ
kernel type	rbf ▼ ⓘ
gamma	0.0 ⓘ
C	0.0 ⓘ
cache size	80 ⓘ
epsilon	0.001 ⓘ

Python

```python
from sklearn.svm import SVC

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

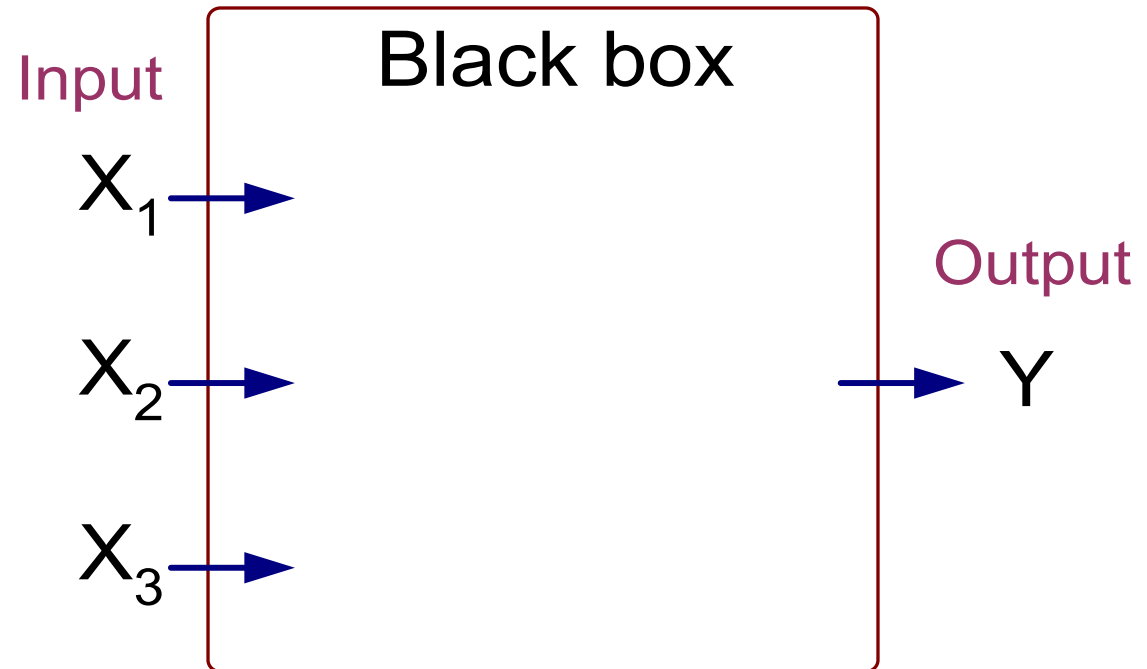# 8. Artificial Neural Networks (ANN)

- Inspiration
  - one of the most powerful super computers in the world
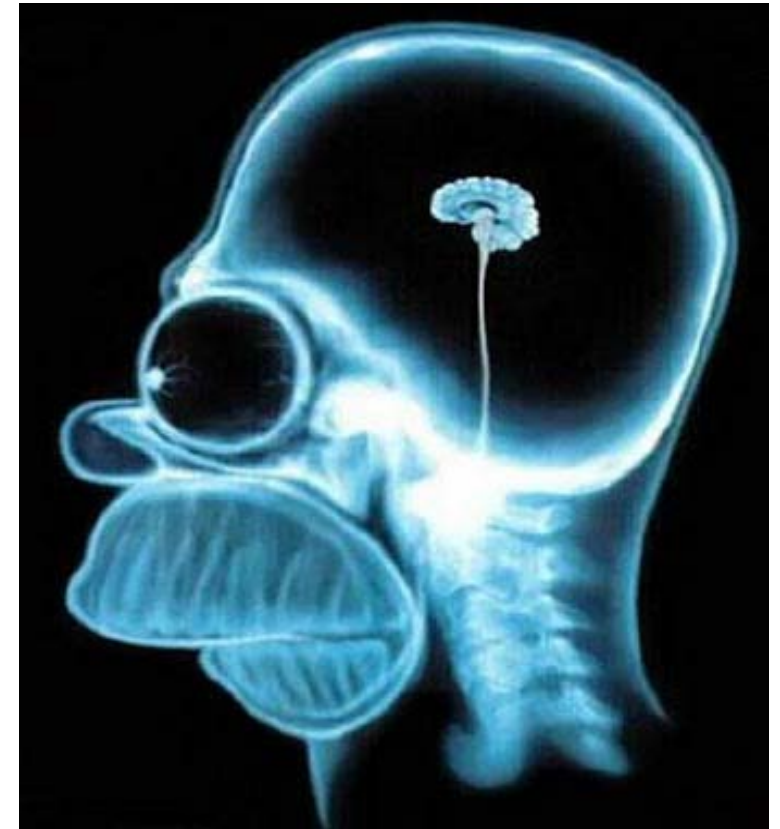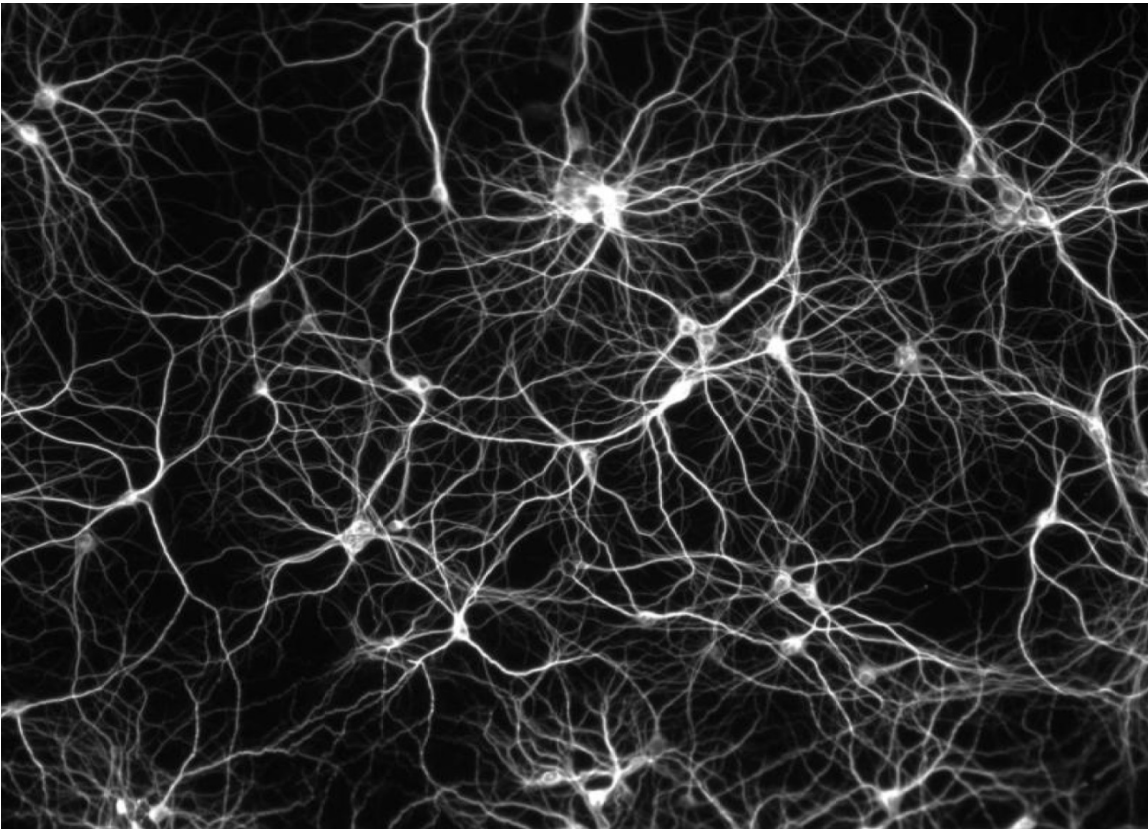
Artificial Neural Networks (ANN)

Training Data

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

Input

Black box

X$_1$ →

Output

X$_2$ → → Y

X$_3$ →

Example:
Output Y is 1 if at least two of the three inputs are equal to 1

Artificial Neural Networks (ANN)

Training Data

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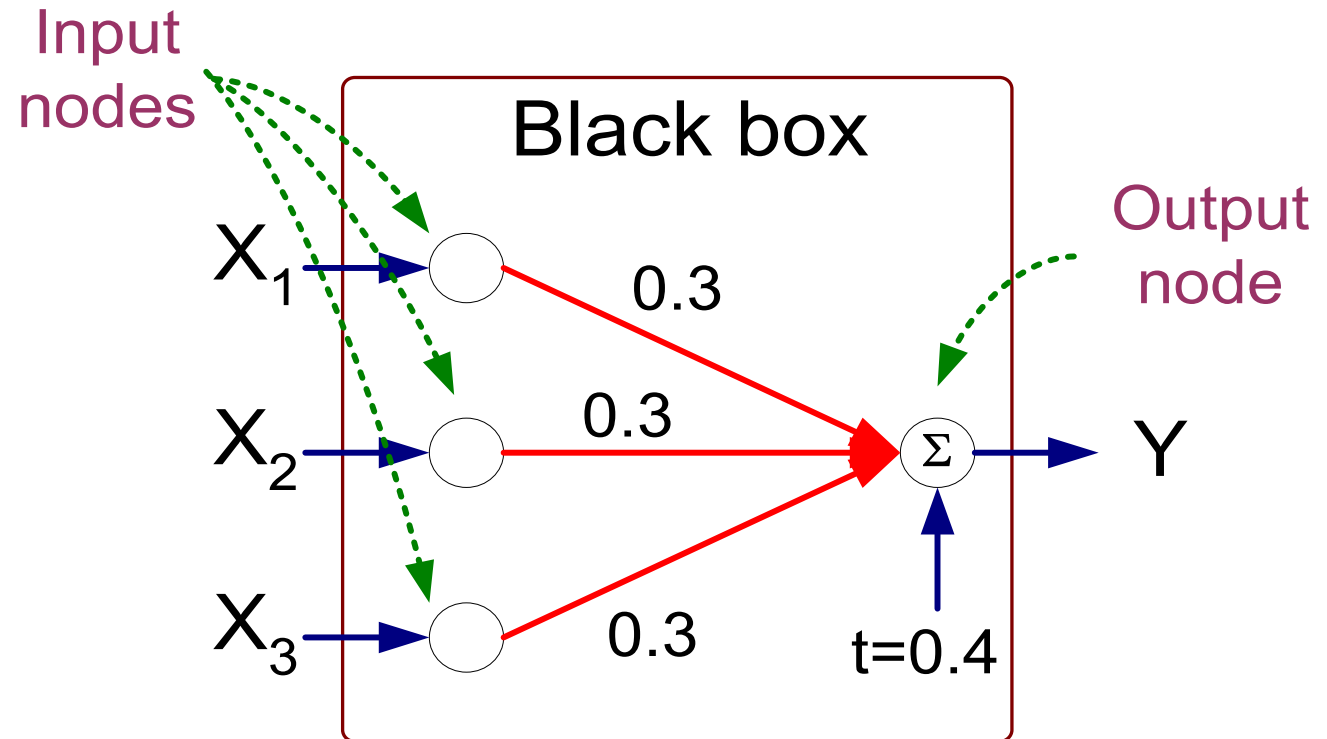
$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

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

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
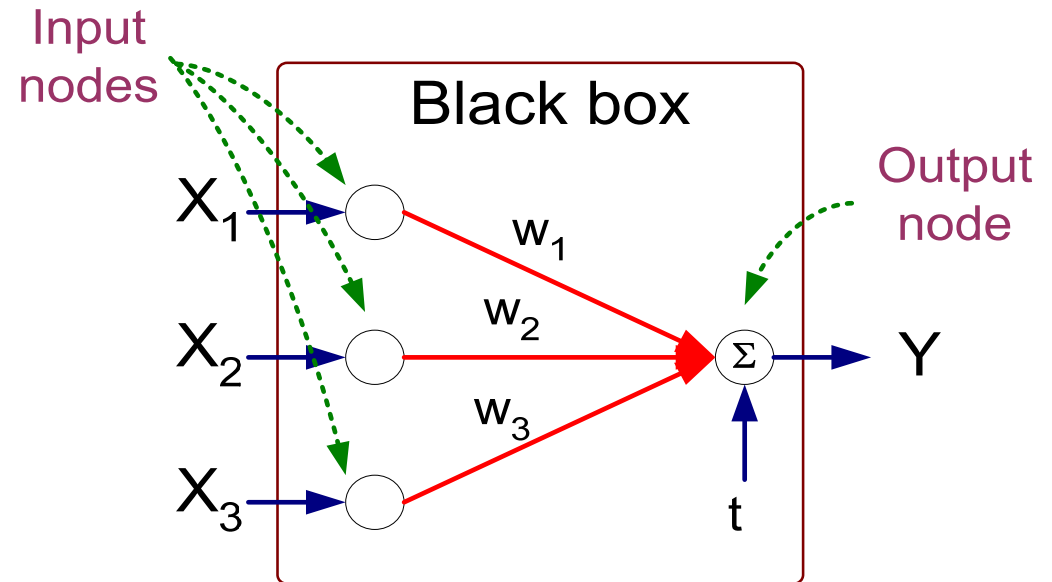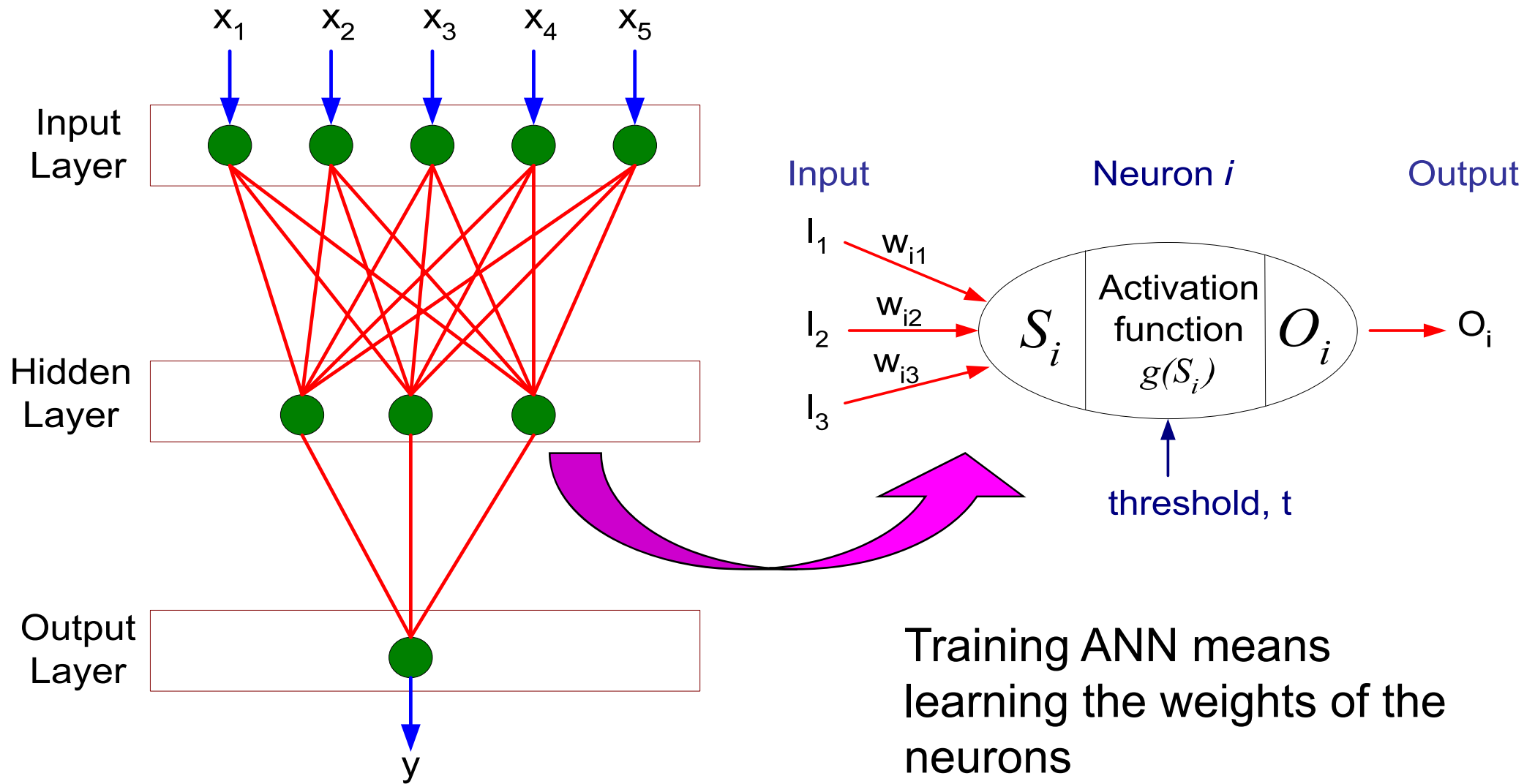

Perceptron Model

$$Y = I(\sum_i w_i X_i - t > 0) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

Multi-Layer Artificial Neural Networks

x_1 x_2 x_3 x_4 x_5

Input Layer

Hidden Layer

Output Layer

y

Input Neuron i Output

I_1 w_{i1}

I_2 w_{i2} S_i Activation function $g(S_i)$ O_i O_i

I_3 w_{i3}

threshold, t

Training ANN means learning the weights of the neurons

Algorithm for Training ANNs

1. Initialize the weights (w_0, w_1, …, w_k), e.g., all with 1 or random

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

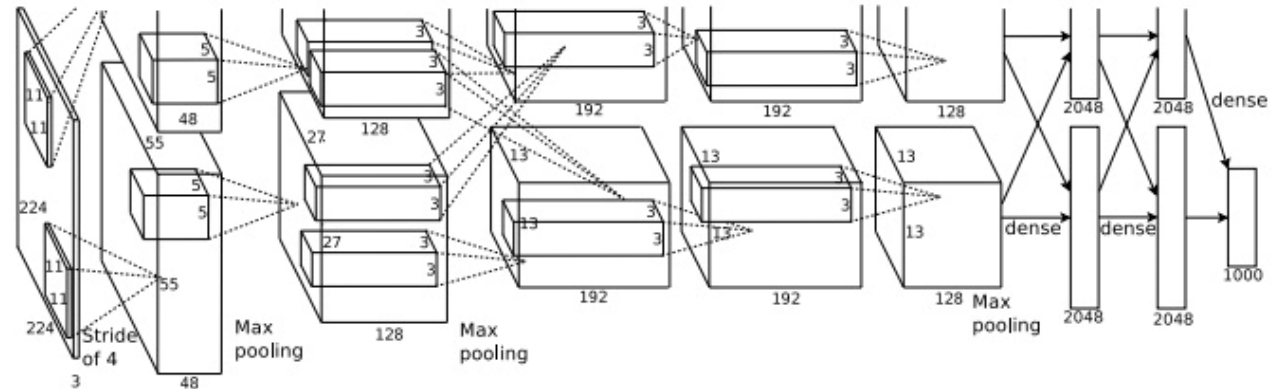- Objective function: $$E = \sum_i [Y_i - f(w_i, X_i)]^2$$

- Find the weights w_i's that minimize the error E

- using for example the **back propagation algorithm** (see Tan/Steinbach, Chapter 6.7)

- Adjustment factor: learning rate

Input **x**   output $\widehat{y}$ ← target **y**
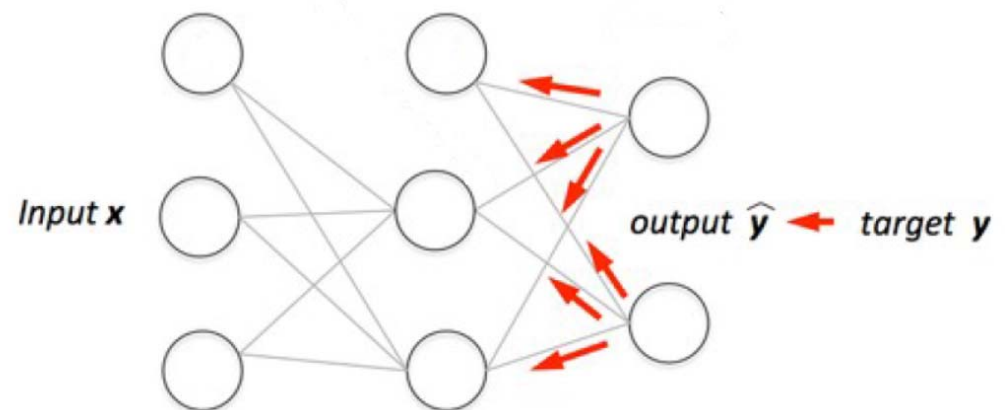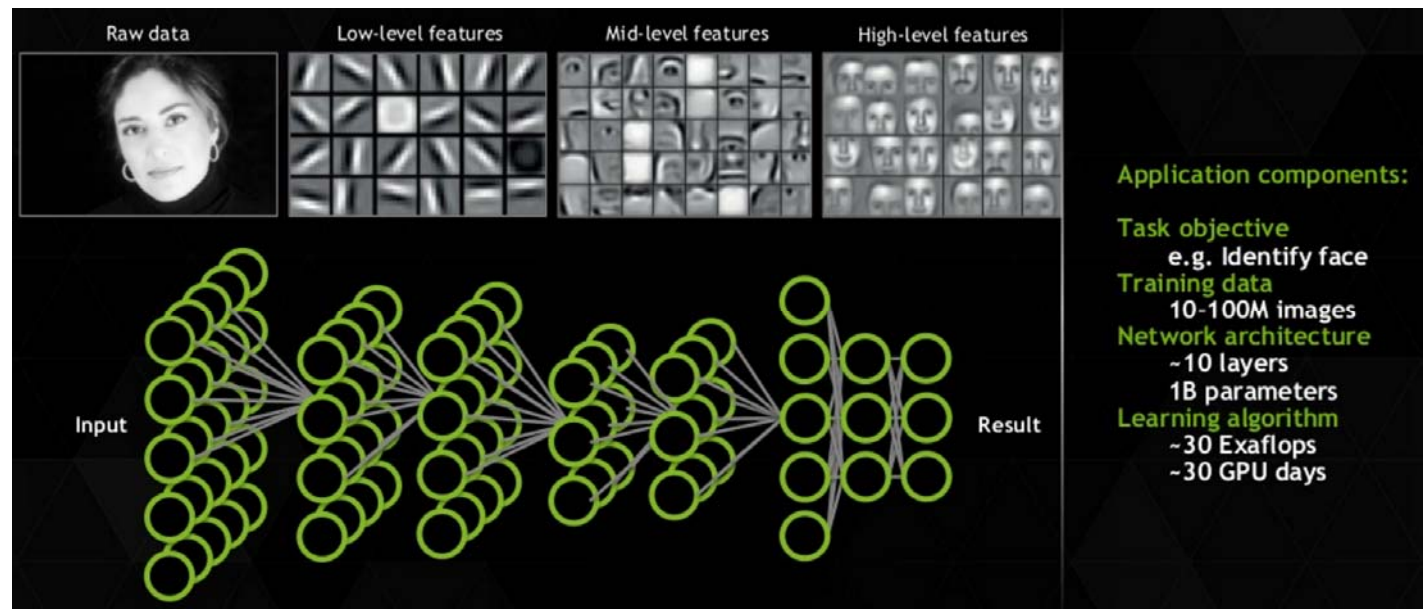
Deep Neural Networks (DNN)

– Hype topic as DNN are very successful in

- computer vision

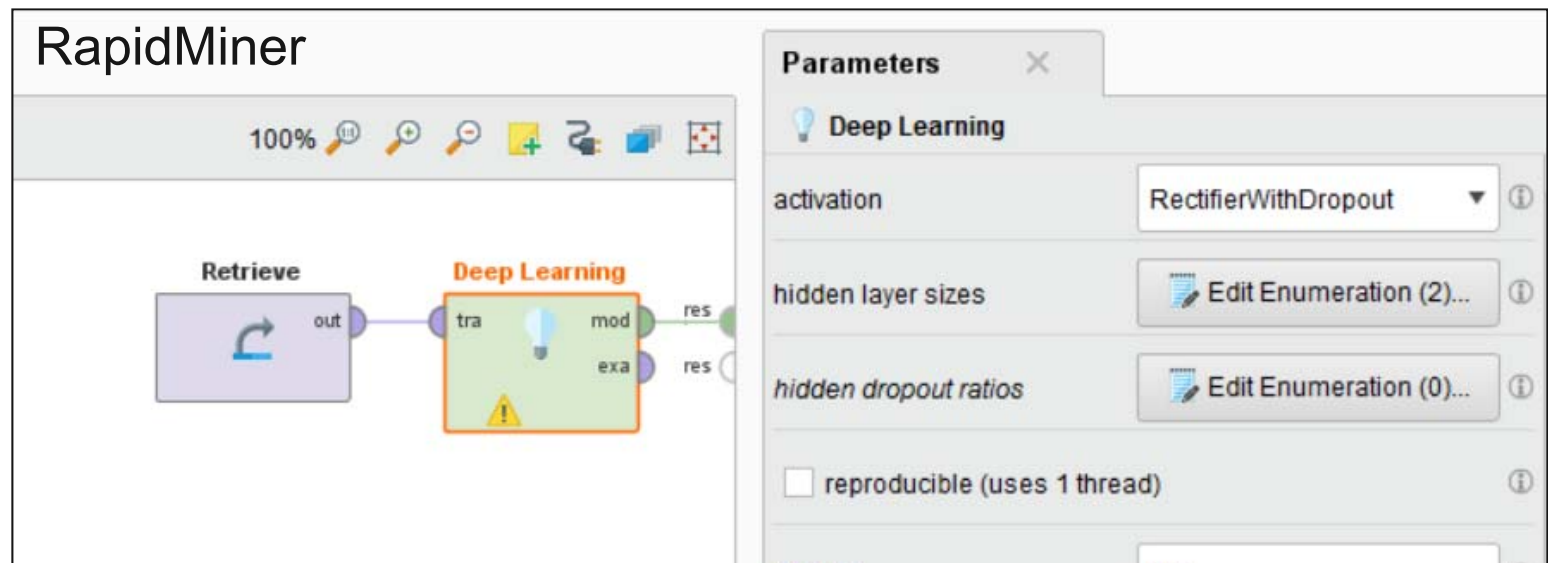- speech recognition

- natural language processing

– Require

- lots of training data

- lots of GPUs to

  - adjust weights

  - test different network architectures

Source: NVIDIA

Artificial Neural Networks in RapidMiner and Python

– both platforms provide their own neural network implementations
– as well as interfaces to state-of-the-art deep learning libraries

Characteristics of Artificial Neural Networks

- ANNs can be used for classification as well as numerical regression tasks (more on this next week)

- Multi-layer neural networks are universal approximators
  - meaning that the can approximate any target function

- Very important but difficult to choose the right network topology
  - Expressive hypothesis space often leads to overfitting
  - Possible approaches to deal with overfitting:
    - Use more training data (a lot more might be necessary)
    - Step-by-step simplify the topology (regularization)
      1. Start with several hidden layers and larger number of nodes
      2. Estimate generalization error using validation dataset
      3. Step by step remove nodes as long as generalization error improves

- Model building is very time consuming, model application is fast

- Can handle redundant attributes, difficult to handle missing values
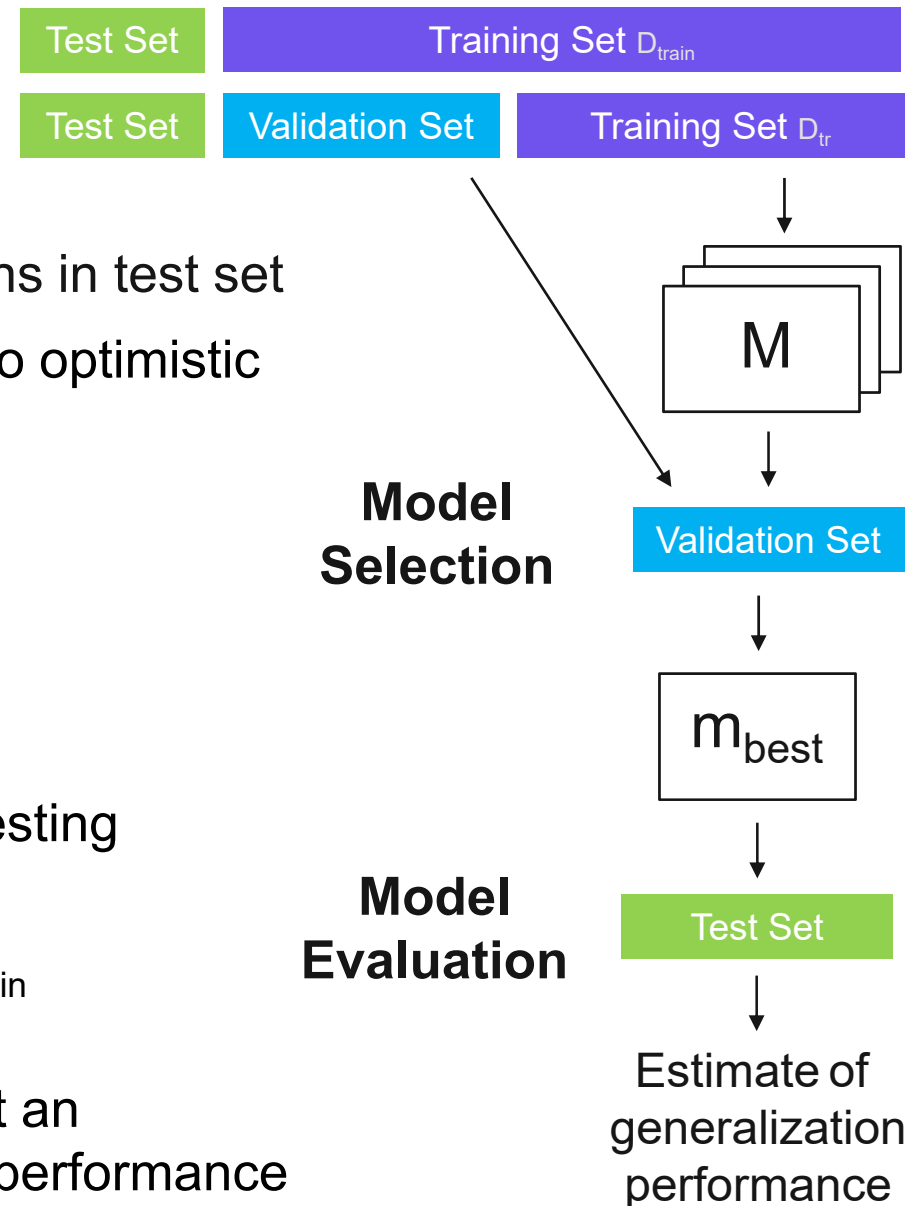
9. Hyperparameter Selection

– 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 from the 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
  - Evolutionary Search: Keep specific parameter values that worked well

– Often hundreds of combinations are tested
  - reason for cloud computing

– Model Selection: From all learned models M, select the model m_{best} that is expected to generalize best to unseen records
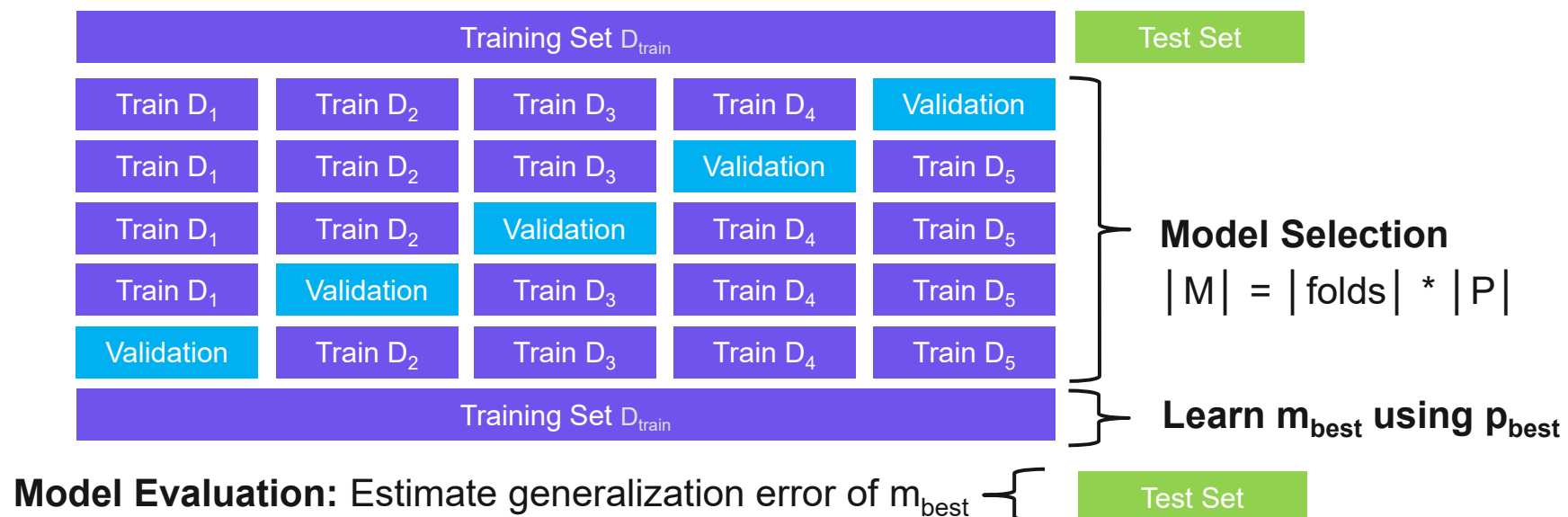
Model Selection Using a Validation Set

– Keep data used for model selection strictly separate from data used for model evaluation, otherwise:

- selected model m_{best} will overfit to patterns in test set
- estimate of generalization error will be 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 on D_{tr} using different hyperparameter value combinations P

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

Test Set	Training Set D_{train}
Test Set	Validation Set	Training Set D_{tr}

Model Selection

M

Validation Set

m_{best}

Model Evaluation

Test Set

Estimate of generalization performance

Model Selection using Cross-Validation

- But wait, we want to

  1. make sure that all examples are used for validation once (remember: red tree)

  2. use as much labeled data as possible for training (remember: learning curve)

- Both goals are met by using cross-validation for model selection

Training Set D_{train}					Test Set
Train D_1	Train D_2	Train D_3	Train D_4	Validation	
Train D_1	Train D_2	Train D_3	Validation	Train D_5	
Train D_1	Train D_2	Validation	Train D_4	Train D_5	
Train D_1	Validation	Train D_3	Train D_4	Train D_5	
Validation	Train D_2	Train D_3	Train D_4	Train D_5	

Model Selection
$$|M| = |folds| * |P|$$

Learn m_{best} using p_{best}

Model Evaluation: Estimate generalization error of m_{best}

Test Set

- 5 folds, 100 parameter value sets ➔ 501 models learned

Model Evaluation using Nested Cross-Validation

- Nest two cross-validation loops into each other in order to:
  1. find the best hyperparameter setting (model selection)
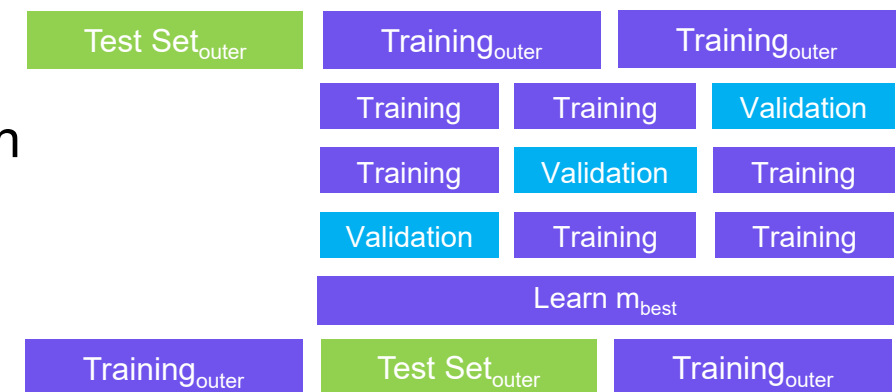 2. get a reliable estimate of the generalization error (model evaluation)

- Outer Cross-Validation
  - estimates generalization error of m_{best}
  - training set is passed on to inner cross-validation in each iteration

- Inner Cross-Validation
  - searches for best parameter combination
  - splits outer training set into inner training and validation set
  - learns model m_{best} using all outer training data

Training$_{outer}$	Training$_{outer}$	Test Set$_{outer}$
Training	Training	Validation
Training	Validation	Training
Validation	Training	Training
Learn m_{best}		

Test Set$_{outer}$	Training$_{outer}$	Training$_{outer}$
	Training	Training	Validation
	Training	Validation	Training
	Validation	Training	Training
	Learn m_{best}	

Training$_{outer}$	Test Set$_{outer}$	Training$_{outer}$

… …

- 5 folds$_{Outer}$ *((5 folds$_{Inner}$ * 100 para.sets)+1) ➔ 2505 models learned

Nested Cross-Validation in RapidMiner

Outer Cross-Validation

Optimize Parameters

Inner Cross-Validation

https://rapidminer.com/resource/correct-model-validation/
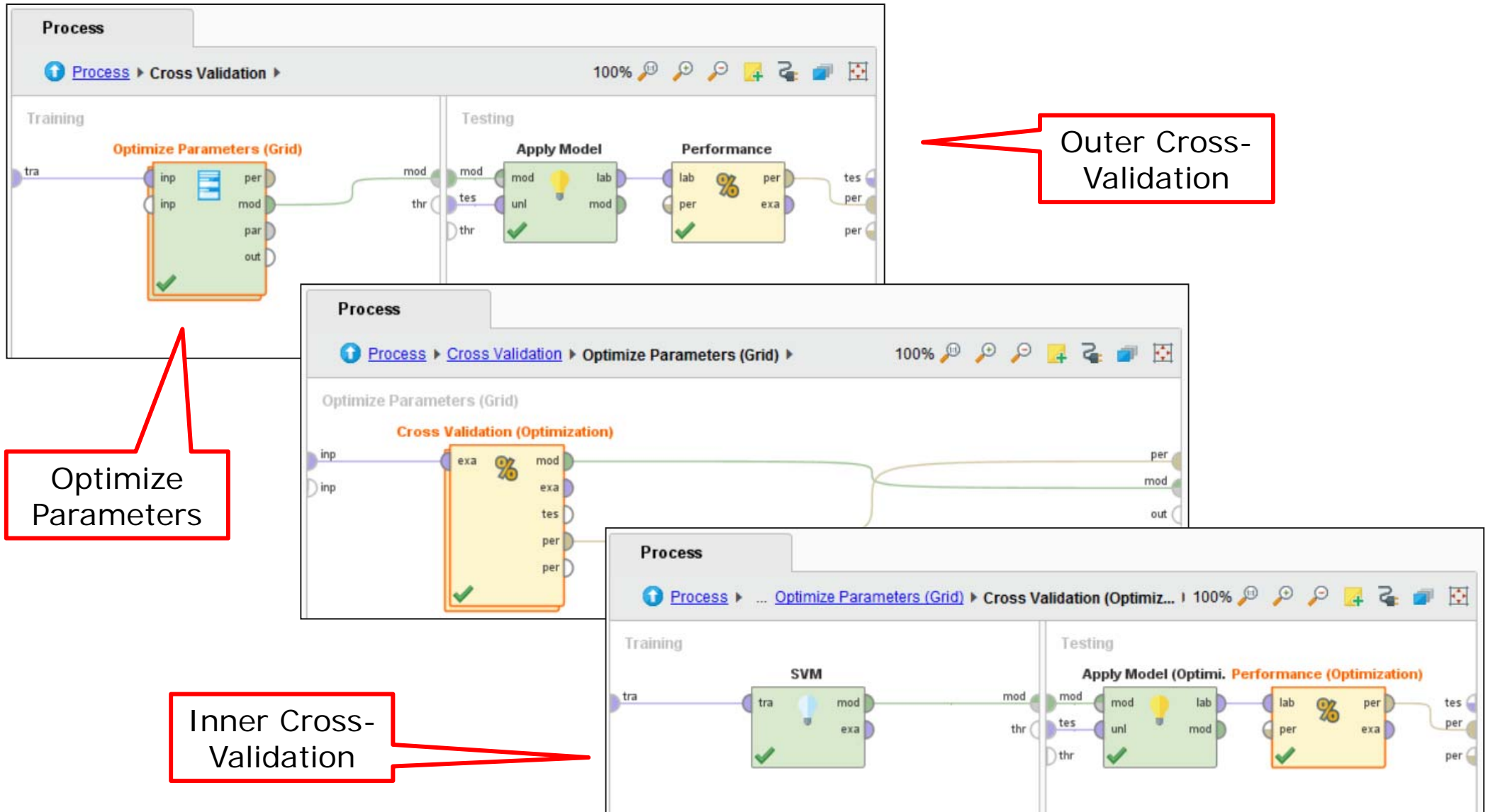
Hyperparameter Optimization in RapidMiner

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

Feature Selection

– 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

Selected   Not used

– Automated feature selection approaches

  • **Forward selection:** Find best single feature, add further features, test again

  • **Backward selection:** start using all features, remove features, test again

– Use nested cross-validation to estimate the generalization error
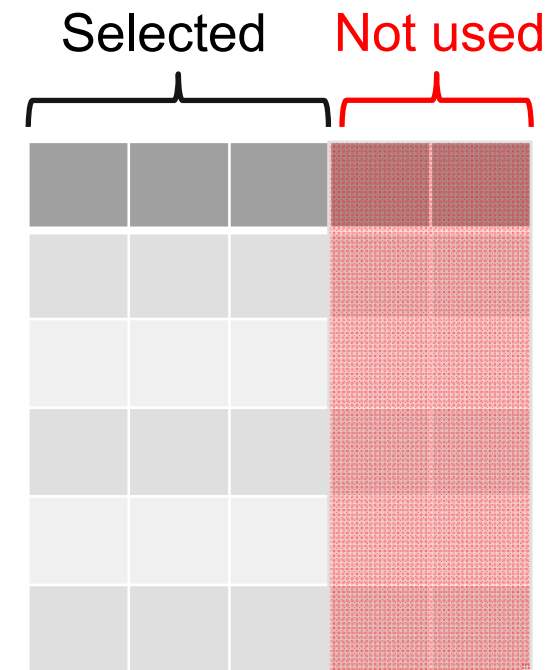
Feature Selection in RapidMiner using Nested X-Validation
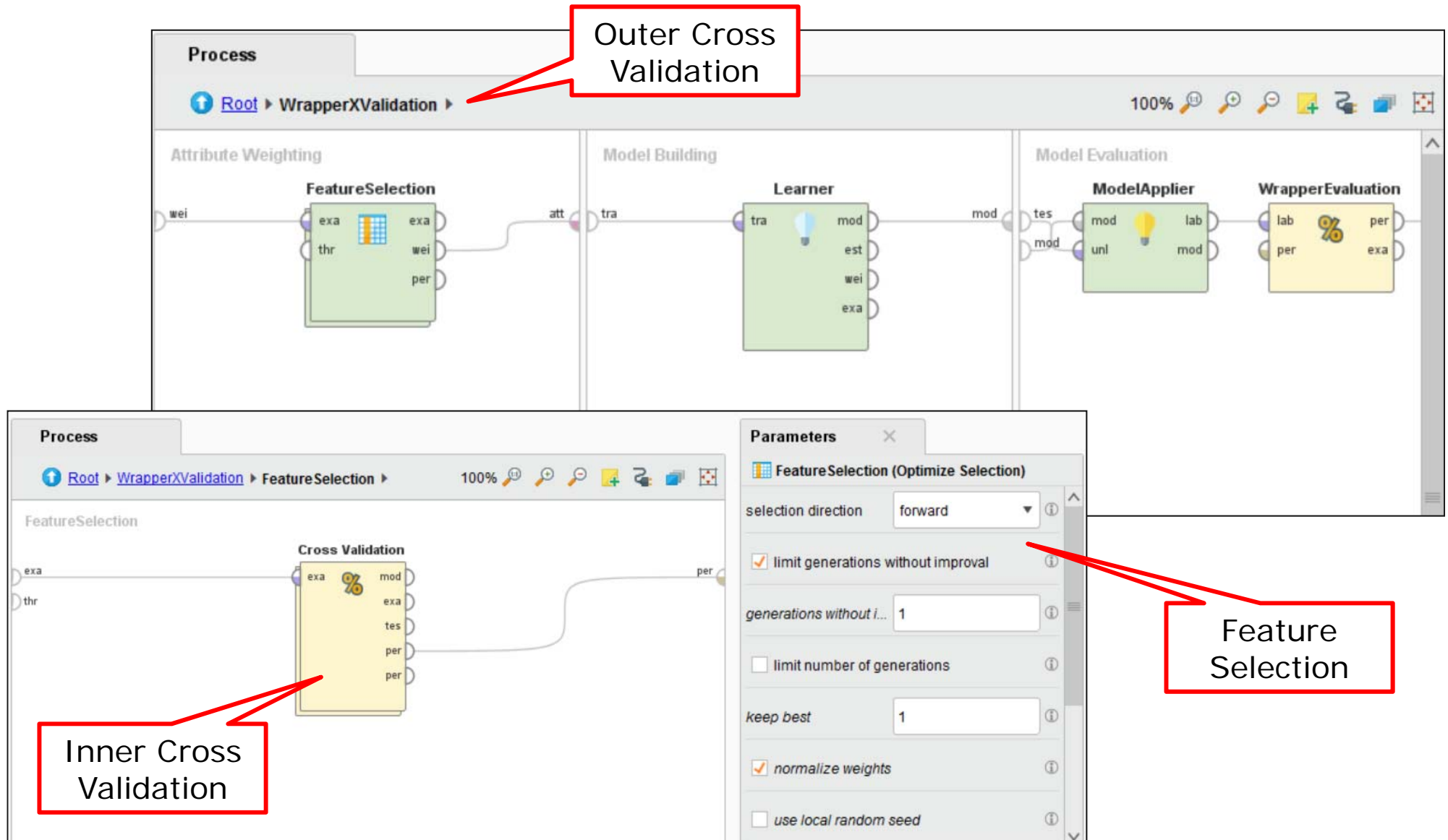
Summary: Hyperparameter and Feature Selection

- 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!

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