GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

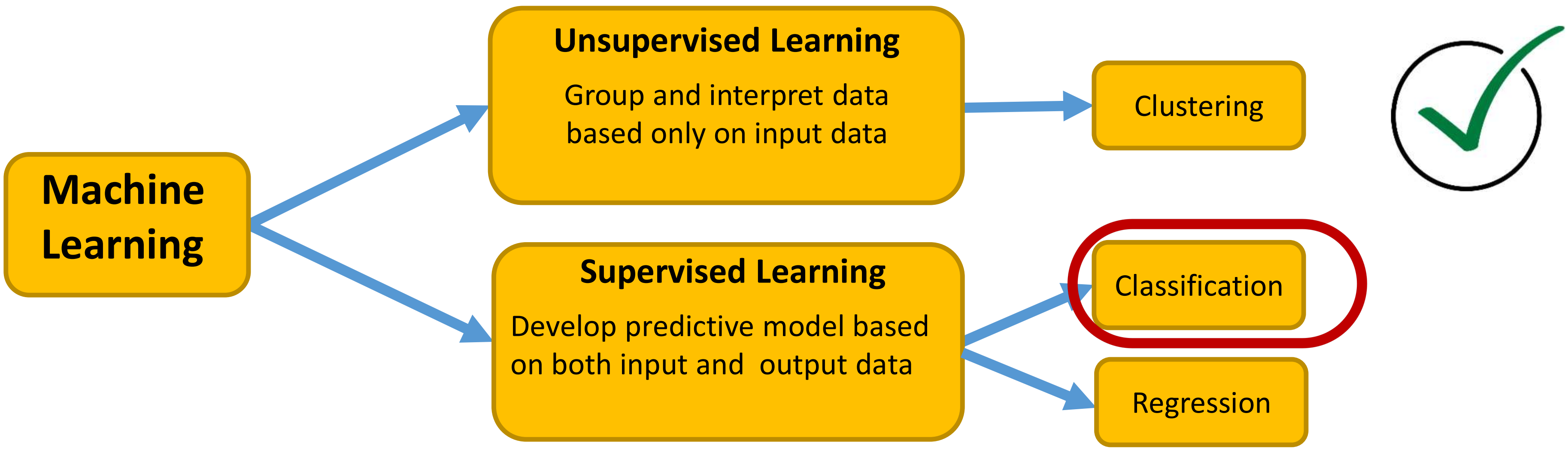
Applied Machine Learning with R

Felix Heinrich

Breeding Informatics Group
Department of Animal Science
Georg-August University Göttingen

Algorithms

- The success of machine learning systems also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.



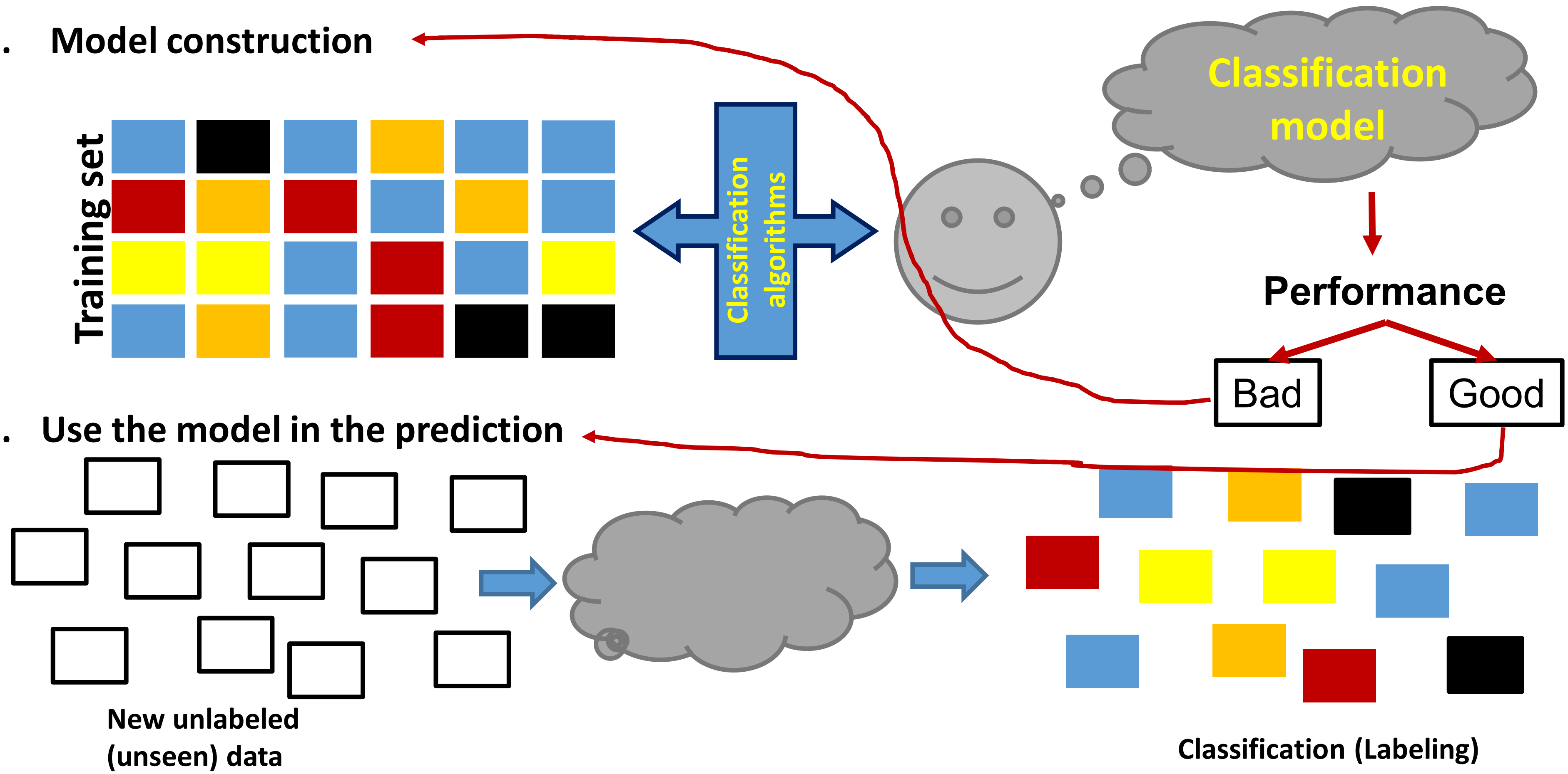
The aim of classification

- The goal of data classification is to organize and categorize data into distinct classes
  - A model is first created based on the training data (learning)
  - The model is then validated on the testing data
  - Finally, the model is used to classify the new data

Clustering vs. Classification

Clustering	Classification
Unknown number of groups	**Known number of classes**
No prior knowledge	**Training data is needed**
Used to explore data	**Used to make new predictions**
A form of unsupervised learning	**A form of supervised learning**

Classification process: Learning the model



GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Example classification process: Prediction of future cases

Supervised learning

Observation	Outcome
Winter of 2014 was **warm**	Summer of 2015 was **cold**
Winter of 2015 was **cold**	Summer of 2016 was **cold**
Winter of 2016 was **cold**	Summer of 2017 was **cold**
Winter of 2017 was **warm**	Summer of 2018 was **hot**
Winter of 2018 was **cold**	Summer of 2019 was **cold**
Winter of 2019 was **warm**	Summer of 2020 was **warm**
Winter of 2020 was **cold**	Summer of 2021 was **cold**
Winter of 2021 was **cold**	Summer of 2022 was **hot**
Winter of 2022 was **warm**	**Summer of 2023 will be ?**

Example classification process: Prediction of future cases

Supervised learning

Observation	Outcome
Winter of 2014 was **warm**	Summer of 2015 was **cold**
Winter of 2015 was **cold**	Summer of 2016 was **cold**
Winter of 2016 was **cold**	Summer of 2017 was **cold**
Winter of 2017 was **warm**	Summer of 2018 was **hot**
Winter of 2018 was **cold**	Summer of 2019 was **cold**
Winter of 2019 was **warm**	Summer of 2020 was **warm**
Winter of 2020 was **cold**	Summer of 2021 was **cold**
Winter of 2021 was **cold**	Summer of 2022 was **hot**
Winter of 2022 was **warm**	**Summer of 2023 will be ?**

Observation	Outcome
1	0
0	0
0	0
1	2
0	0
1	1
0	0
0	2
1	?

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Example classification process: Prediction of future cases

Supervised learning

Observation	Outcome
Winter of 2014 was **warm**	Summer of 2015 was **cold**
Winter of 2015 was **cold**	Summer of 2016 was **cold**
Winter of 2016 was **cold**	Summer of 2017 was **cold**
Winter of 2017 was **warm**	Summer of 2018 was **hot**
Winter of 2018 was **cold**	Summer of 2019 was **cold**
Winter of 2019 was **warm**	Summer of 2020 was **warm**
Winter of 2020 was **cold**	Summer of 2021 was **cold**
Winter of 2021 was **cold**	Summer of 2022 was **hot**
Winter of 2022 was **warm**	**Summer of 2023 will be ?**

Observation	Outcome
1	0
0	0
0	0
1	2
0	0
1	1
0	0
0	2
1	?

Supervised learning: Limitations and opportunities

Observation	Outcome
1	0
0	0
0	0
1	2
0	0
1	1
0	0
0	2
1	?

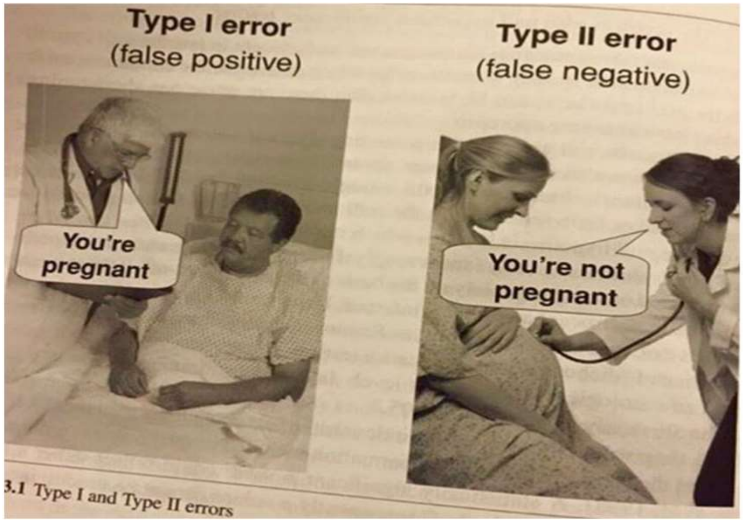
Mathematical algorithms good at pattern identification, but bad at generalization.

Human brain good at generalization, but bad at pattern identification.

Performance of a classification algorithm:

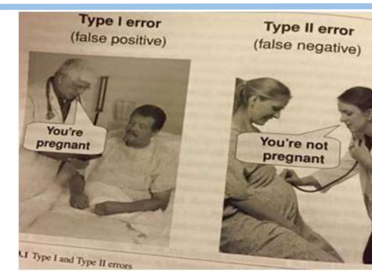
Typically for supervised learning algorithm a **<u>confusion matrix</u>** is used to measure the performance of the training process

- **<u>Confusion matrix</u>**
  - known as an error matrix
  - a specific table layout that allows visualization of the performance of an algorithm
  - a special kind of contingency table, with two dimensions ("actual" and "predicted")
  - reports the number of **false positives, false negatives, true positives**, and **true negatives**.



GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Performance of a classification algorithm:



- **Confusion matrix**

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$	F_1 score = $\frac{1}{\frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

Terminology and derivations from the confusion matrix

condition positive (P)	**condition negative (N)**
the number of real positive cases in the data	the number of real negative cases in the data
true positive (TP)	**true negative (TN)**
eqv. with hit	eqv. with correct rejection
false positive (FP)	**false negative (FN)**
eqv. with false alarm, Type I error	eqv. with miss, Type II error

wikipedia.org

Terminology and derivations from the confusion matrix

sensitivity, recall, hit rate, or **true positive rate (TPR)**

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or **true negative rate (TNR)**

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or **positive predictive value (PPV)**

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

negative predictive value (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}}$$

miss rate or **false negative rate (FNR)**

$$\mathrm{FNR} = \frac{\mathrm{FN}}{P} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}} = 1 - \mathrm{TPR}$$

fall-out or **false positive rate (FPR)**

$$\mathrm{FPR} = \frac{\mathrm{FP}}{N} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} = 1 - \mathrm{TNR}$$

false discovery rate (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TP}} = 1 - \mathrm{PPV}$$

accuracy (ACC)

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

Matthews correlation coefficient (MCC)

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN} - \mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

True class

Predicted class	Positive (class = 1)	Negative (class = 0)
Positive (class = 1)		
Negative (class = 0)		

wikipedia.org

Performance of a regression algorithm:

- Now consider the outcome Y not as an instance of a distinct class, but as a continous value
  - Example: - Temperature
    - Stock price
    - Gene expression
- The computation of a confusion matrix is not that easy (no TP, FP, TN, FN)
- Alternative metrics are more informative
- Error-based metrics: deviation of the model output (Ŷ) from the target (Y)
  - Mean squared error (MSE): $\frac{1}{n}\sum_{i=0}^{n}(\widehat{y}_i - y_i)^2$
  - Root Mean Squared Error (RMSD): $\sqrt{\frac{1}{n}\sum_{i=0}^{n}(\widehat{y}_i - y_i)^2}$
  - Mean absolute error (MAE): $\frac{1}{n}\sum_{i=0}^{n}|\widehat{y}_i - y_i|$
- Correlation-based metrics: $\rho(\widehat{Y}, Y) = \frac{Cov(\widehat{Y}, Y)}{\sqrt{Var(\widehat{Y})Var(Y)}}$

Support Vector Machine (SVM):

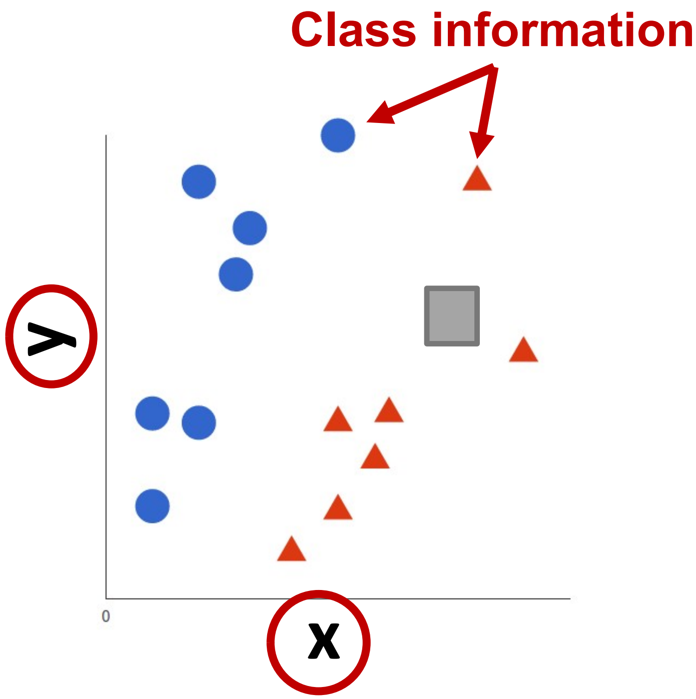
Support Vector Machine (SVM):

- Supervised learning models primarily used for binary classification (but extendable to multiple classes)
- Analyzing data sets used for classification

SVM Algorithm

- **Data**
  - Imagine that we have two tags: red and blue,
  - Our data has two features: x and y
- **Aim:** - Learn a classifier to predict the class based on (x,y)

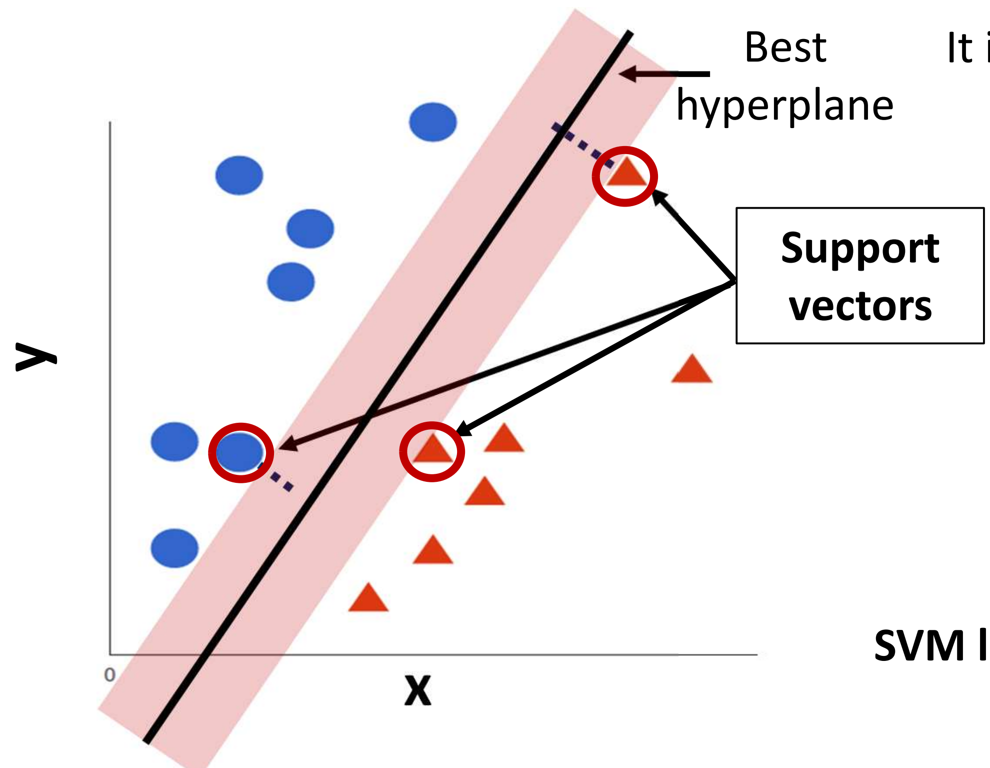
  - It can identify whether a new data point is **red** or **blue**



datacamp.com

Support Vector Machine (SVM):

- Based on these data points, a SVM outputs the hyperplane
- Best separation of the tags in the analyzed data sets used for classification



It is our **decision boundary**

What exactly is the best hyperplane?

It is the line that maximizes the margins from both tags.

In other words: the hyperplane whose distance to the nearest element of each tag is the largest.

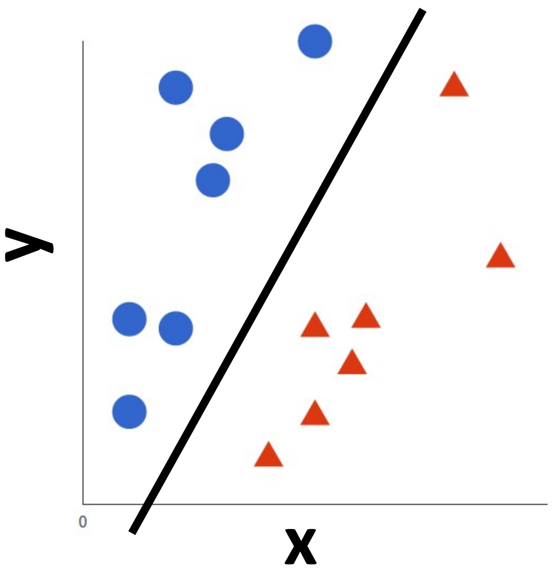
SVM looks at the data and sorts it into one of the two categories: **red** or **blue**

datacamp.com

Support Vector Machine (SVM):

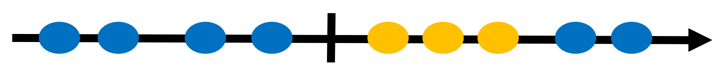
Linear Data:

- Determine best hyperplane



What happens, if my data is not like this ?

Non-Linear Data



Two sets of data:
One of them occurs in the middle of another set

Hyperplane cannot be used ☹

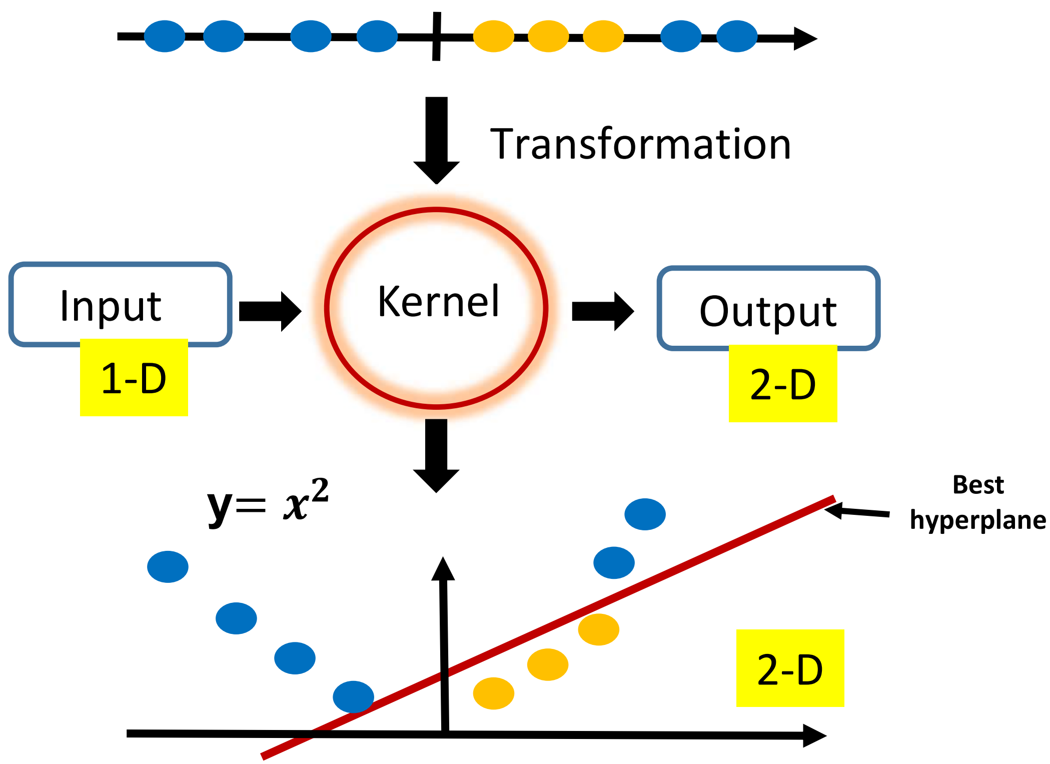
datacamp.com

Support Vector Machine (SVM):

Non-Linear Data

It is necessary to move away from a 1-D view of the data to a 2-D view

➔ Transformation using a Kernel Function

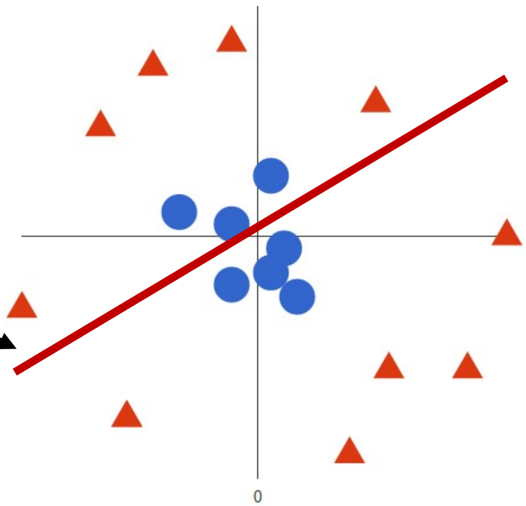


datacamp.com

Support Vector Machine (SVM):

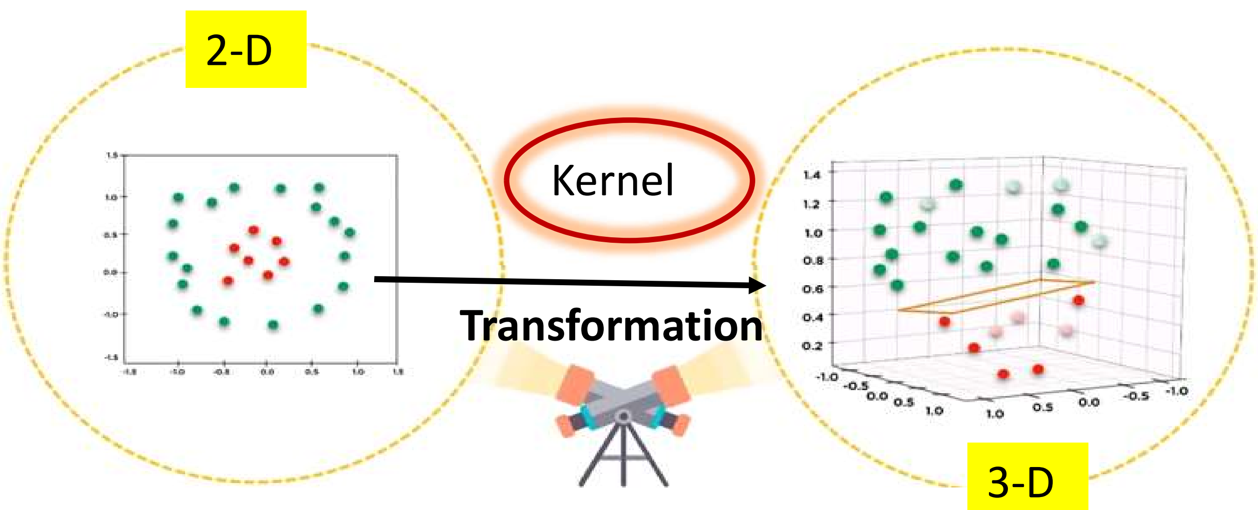
Non-Linear Data

How to perform SVM for this type of data set?



The hyperplane is not optimal ☹

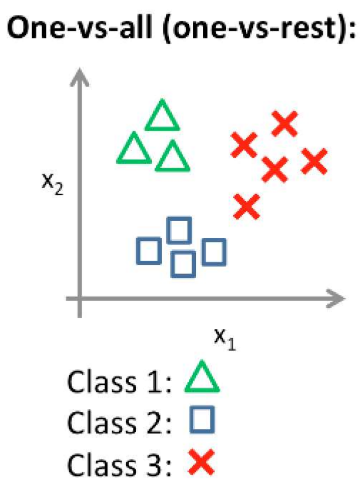
➔ Transformation using a Kernel Function



datacamp.com

Support Vector Machine (SVM):

Data with more than two classes



- For more than two classes reduce the classification problem into multiple binary classification problems
- To classify **k** classes we need **k** different SVMs
- R handles multiple classes automatically

Support Vector Machine (SVM): What is a kernel?

Idea of the kernel trick:

- Transform your data to a higher dimensional space

Simple examples:

- 1D -> 2D: Data is one-dimensional; e.g. x = 2; 6; 3
  - Introduce a function f(x) to create a new variable y: $y = f(x) = x^2$ ➡ y = 4; 36; 9
- 2D -> 3D: Data is two-dimensional; e.g. x = 2; 6; 3 and y = 4; 36; 9
  - Introduce a function f(x,y) to create a new variable z: $z = f(x, y) = x + y$ ➡ z = 6; 42; 12

A kernel is mathematical function

Support Vector Machine (SVM):

Advantages

- **High Dimensionality:** SVM is an effective tool in high-dimensional spaces, which is particularly applicable to document classification and sentiment analysis where the dimensionality can be extremely large.
- **Memory Efficiency:** Since only a subset of the training points are used in the actual decision process of assigning new members, just these points need to be stored in memory (and calculated upon) when making decisions.
- **Versatility:** Class separation is often highly non-linear. The ability to apply new kernels allows substantial flexibility for the decision boundaries, leading to greater classification performance.

Disadvantages

- **Kernel Parameters Selection:** SVMs are very sensitive to the choice of the kernel parameters. In situations where the number of features for each object exceeds the number of training data samples, SVMs can perform poorly.

- **Non-Probabilistic:** Since the classifier works by placing objects above and below a classifying hyperplane, there is no direct probabilistic interpretation for group membership. However, one potential metric to determine the "effectiveness" of the classification is how far from the decision boundary the new point is.

datacamp.com

Support Vector Machines

Example:

Import the data file called **“catsData.csv”** into the variable **mydata**

> mydata= read.table(“catsData.csv”, header=TRUE, sep=“,”, stringsAsFactors = TRUE)

\# Check the dimensions of the data

\# View statistical summary of dataset

\# View the complete data

#bwt: body weight (in kg)

#hwt: heart weight (in g)

Process the dataset

- Shuffle and divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)
- Perform a **svm** to classify the data based on the **sex** with **dfTraining** (function in R: **svm**)
- Check the number of support vectors that are used
- Verify results of the svm by plotting them
- Create a **confusion matrix** for the predicted and true sex on **dfTest**
- Calculate the following values using your own functions:
  - **Sensitivity , Specificity**
  - **Precision, Accuracy**

Support Vector Machines

Example:

Process the dataset

– Divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)

```
>countTraining = round(nrow(mydata)*0.7)
>randomRows=sample(1:nrow(mydata), size= countTraining, replace=F)
>dfTraining = mydata[randomRows,]
>dfTest = mydata[- randomRows,]
```

– Perform a **svm** to classify the data based on the **sex** with **dfTraining** (function in R: **svm**)

```
>library(e1071)  #library containing the svm function
>mySVM = svm(Sex~., dfTraining, scale = TRUE)
#Create a svm with Sex as class attributes and all other attributes as variables
```

– Check the number of support vectors that are used

```
>summary(mySVM)
```

Support Vector Machines

Example:

Process the dataset

- Verify results of the svm by plotting them

>plot(mySVM, data = dfTraining)
#Crosses are the support vectors while circles are normal data points
#Using ggplot2 for svm is possible but far more difficult than the normal plot() function

- Use the training model to make predictions based on **dfTest**(function: **predict**)

>myPred = predict(mySVM,dfTest) #Create vector of predicted sex based on the svm

- Create a **confusion matrix** for the predicted and true sex on **dfTest** (function**: table)**

>confTable = table(myPred,dfTest$Sex) #Create confusion matrix from predicted and true values
>confTable

- Calculate the following values using your own functions
  - **Sensitivity, Specificity**
  - **Precision, Accuracy**

Support Vector Machines

Example:

Process the dataset

- Calculate the following values using your own functions

#We need the values for **True_Positives, True_Negatives, False_Positives and False_Negatives**
#These can be read from the confusion matrix
#In this case we assume that female equals **true** and male equals **false**

```
>truePos = confTable[1,1]     #True_Positives
>trueNeg = confTable[2,2]     #True_Negatives
>falsePos = confTable[1,2]    #False_Positives
>falseNeg = confTable[2,1]    #False_Negatives
```

sensitivity, **recall**, **hit rate**, or **true positive rate** (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, **selectivity** or **true negative rate** (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or **positive predictive value** (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

accuracy (ACC)

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

Support Vector Machines

Example:

Process the dataset

– Calculate the following values using your own functions

```
#Sensitivity
>calcSens = function(truePos,falseNeg){
        sens = truePos / (truePos + falseNeg)
        return(sens)
}
#Specificity
>calcSpec = function(trueNeg,falsePos){
        spec = trueNeg / (trueNeg + falsePos)
        return(spec)
}
```

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = 1 - \mathrm{FPR}$$

Support Vector Machines

Example:

Process the dataset

– Calculate the following values using your own functions

```
#Precision
>calcPrec = function(truePos,falsePos){
        prec = truePos / (truePos + falsePos)
        return(prec)
}
#Accuracy
>calcAcc = function(truePos,trueNeg,falsePos,falseNeg){
        acc = (truePos + trueNeg) / (truePos + trueNeg + falsePos + falseNeg)
        return(acc)
}
```

precision or **positive predictive value** (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

accuracy (ACC)

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

Support Vector Machines

Example:

Process the dataset

- Calculate the following values using your own functions

```
>sens = calcSens(truePos,falseNeg)
>cat("Sensitivity:", sens)

>spec = calcSpec(trueNeg,falsePos)
>cat("Specificity:", spec)

>prec = calcPrec(truePos,falsePos)
>cat("Precision:", prec)

>acc = calcAcc(truePos,trueNeg,falsePos,falseNeg)
>cat("Accuracy:", acc)
```

Support Vector Machines

Exercise:

Import the data file called **“plantData.csv**” into the variable **mydata**

Process the dataset

- Shuffle and divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)
- Create a **svm** for classifying the data based on the species with **dfTraining** (function in R: **svm**)
- Check the number of support vectors that are used
- Verify results of the svm by plotting them
- Create a **confusion matrix** for the predicted and true species on **dfTest**
- Calculate the following values using your own functions:
  - **Accuracy**

Support Vector Machines

Exercise:

Import the data file called **"plantData.csv"** into the variable **mydata**

```
>mydata = read.table("plantData.csv", header = TRUE, sep = ",", stringsAsFactors =  TRUE)
```

Process the dataset

- Divide **mydata** in two data frames as **dfTraining** (70%) and **dfTest** (30%)

```
>countTraining = round(nrow(mydata)*0.7)
>randomNumbers= sample(1:nrow(mydata), size = countTraining, replace = F) #randomSamples
>dfTraining =mydata[randomNumbers, ]
>dfTest =  mydata[-randomNumbers, ]
```

- Create a **svm** for classifying the data based on the species with **dfTraining** (function in R: **svm**)

```
>mySVM = svm(Species~., dfTraining)
```

- Check the number of support vectors that are used

```
>summary(mySVM)
```

Support Vector Machines

Exercise:

Process the dataset

- Verify results of the **svm** by plotting them

```
#The plot has become a bit more difficult since we have more than 2 dimensions in our data
#Here we plot the points based on 2 dimensions while we fix the values for the other dimensions
>plot(mySVM, data = mydata, Petal.Width~Petal.Length,
+       slice = list(Sepal.Width = 3, Sepal.Length = 4))
#The points are plotted according to their values in Petal.Width and Petal.Length
#The values of Sepal.Width and Sepal.Length are fixed to 3 respectively  4
```

- Create a **confusion matrix** for the predicted and true species on **dfTest**

```
# perform a svm classification using the training model
>myPred = predict(mySVM,dfTest)
# confusion matrix
>confTable = table(myPred,dfTest$Species)
```