QUEEN'S UNIVERSITY BELFAST
SCHOOL OF ELECTRONICS, ELECTRICAL ENGNIEERING AND COMPUTER SCIENCE

Data Analysis & Visualisation

CSC3062

BEng (CS & SE), MEng (CS & SE), BIT & CIT

Dr Reza Rafiee

Semester 1 2019

Supervised learning

What we need to know about classification

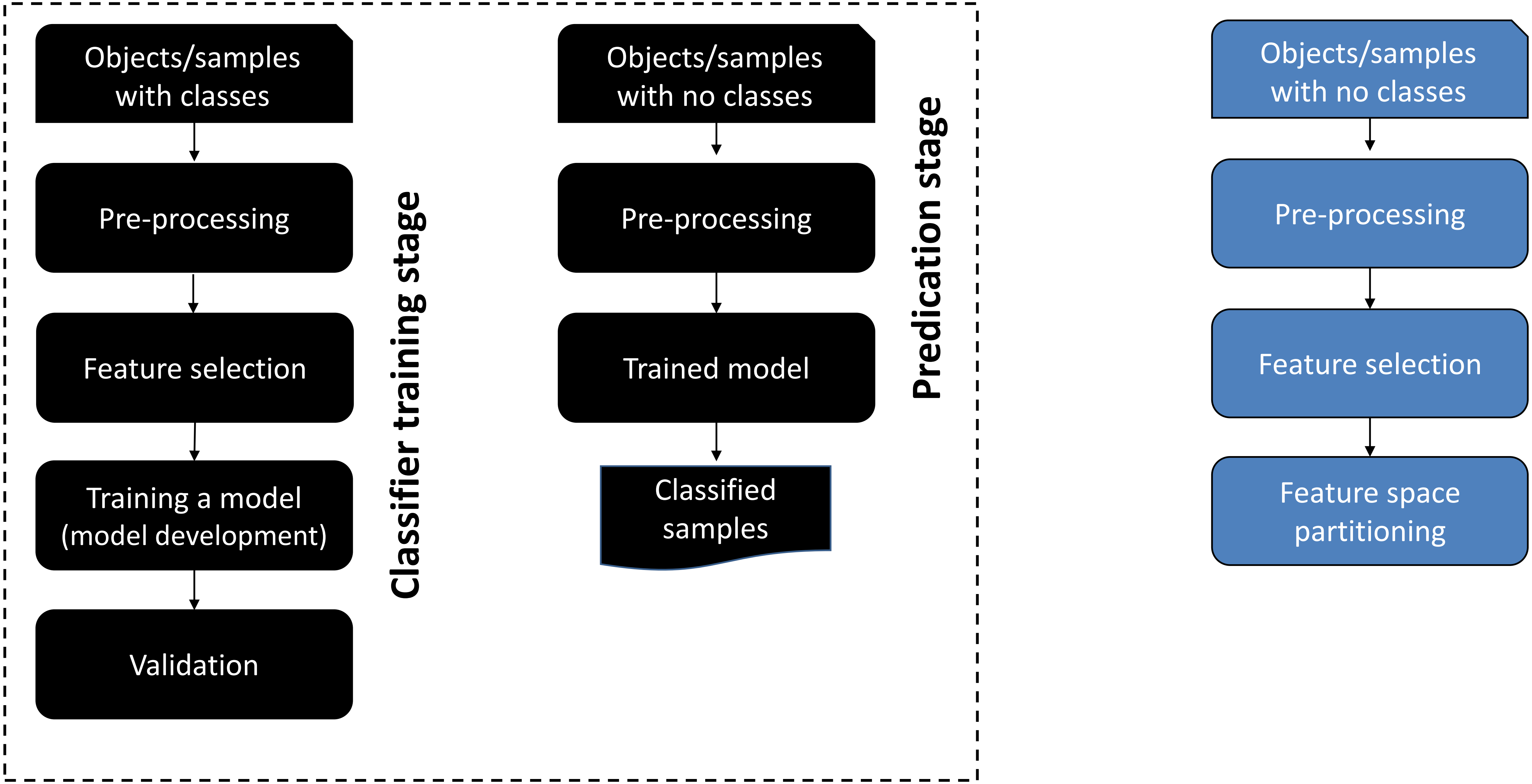
- What is classification?
- What we need as a dataset in classification
- Binary vs. multiclass classification
- Classification models (categories of classifier models)
- How to choose a classification model?
- Support vector machine (SVM) classifier model
- Designing a multiclass SVM model with an example
- How to evaluate the performance of a classifier model?

QUEEN'S UNIVERSITY BELFAST | SCHOOL OF ELECTRONICS, ELECTRICAL ENGNIEERING AND COMPUTER SCIENCE

Classification applications

- Text categorisation (e.g., spam filtering)
- Optical character recognition (OCR) (e.g., a computerised system transferring hard documents into word doc.)
- Machine vision (e.g., face detection)
- Natural-language processing (NLP) (e.g., spoken language understanding)
- Market segmentation (e.g., predict if customer will respond to promotion)
- Bioinformatics and medicine (e.g., classify cancer patients into different immune subtypes)
- …

Classification vs. clustering



Feature selection for high separability

QUEEN'S UNIVERSITY BELFAST

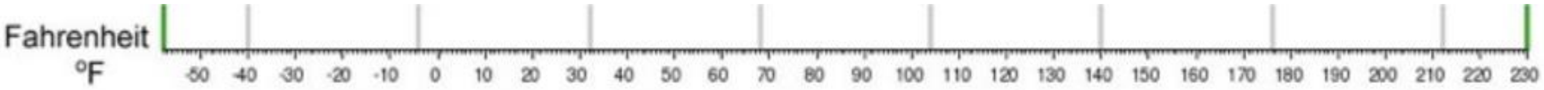
SCHOOL OF ELECTRONICS, ELECTRICAL ENGNIEERING AND COMPUTER SCIENCE

Prediction in regression & classification

What is the temperature going to be tomorrow?

Based on previous data

35°

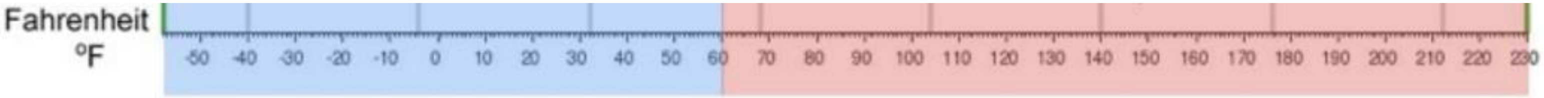


Will it be cold or hot weather?

Based on previous data

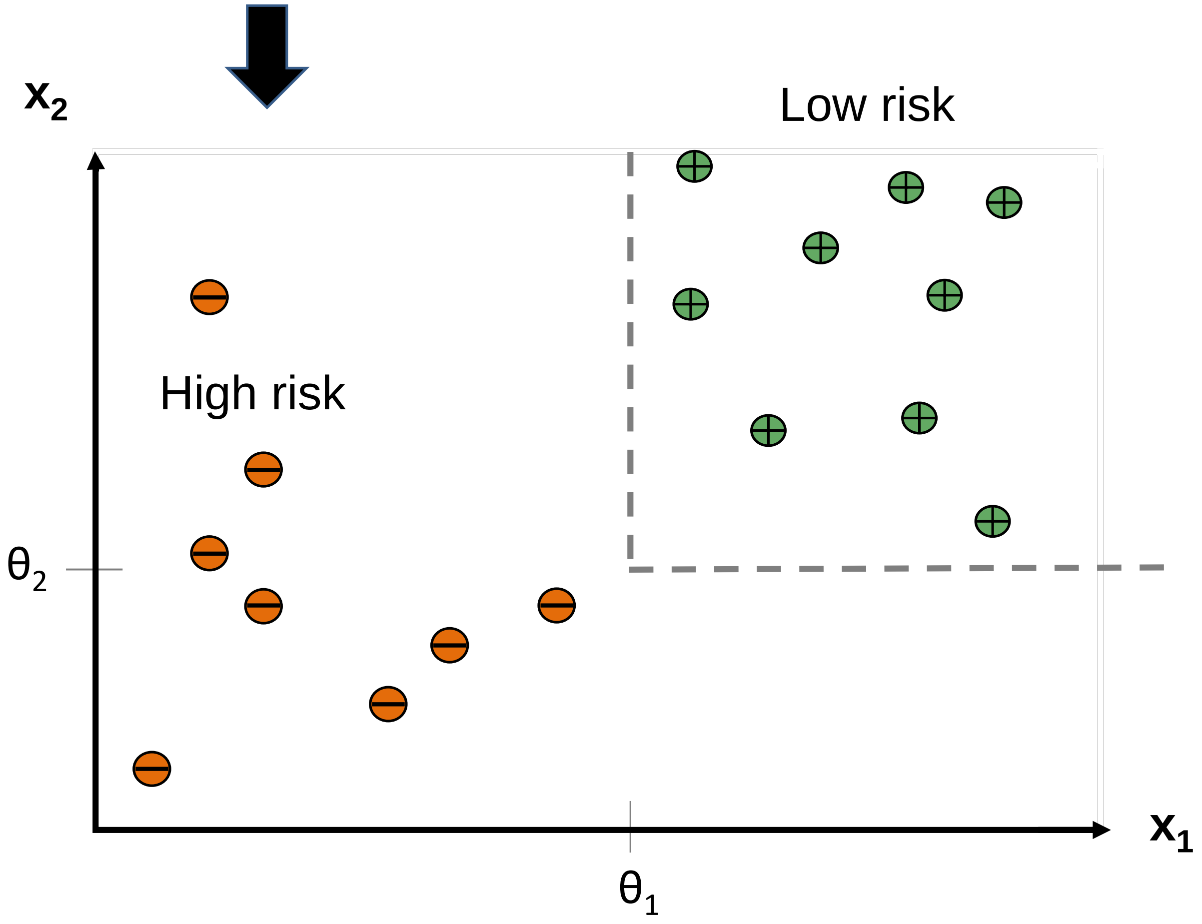
Cold

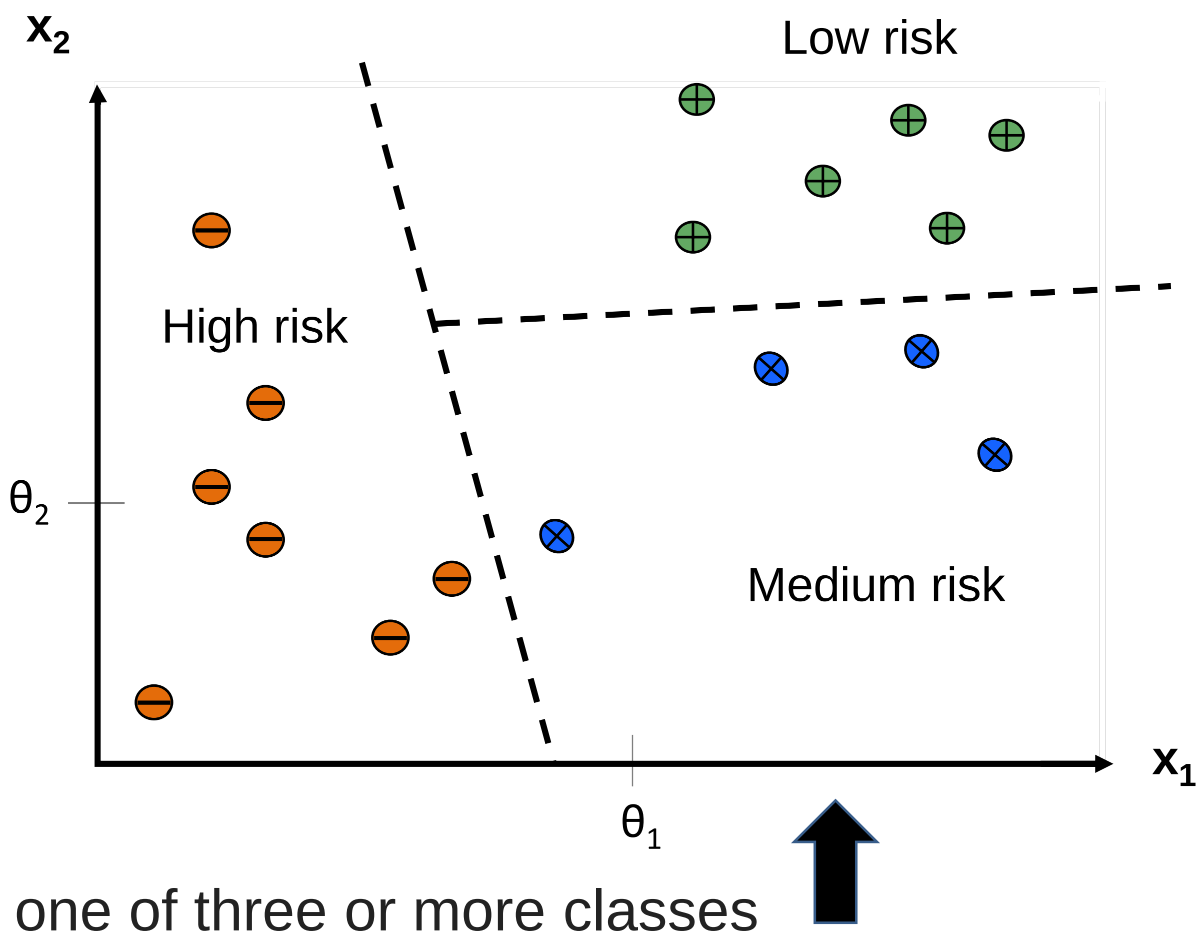
Hot



Binary vs. multiclass classification

Binary classifier classifies data points into one of two classes





Multiclass classifier: classifies data points into one of three or more classes

Classification algorithms

- K-Nearest Neighbour
- Naive Bayes Classifier
- Support Vector Machines (the basic SVM supports only binary classification); linear or with Gaussian kernels
- Decision Trees (e.g., Random Forest)
- Artificial Neural Networks (ANN)
- Hierarchal classifier
- …

Classification algorithms

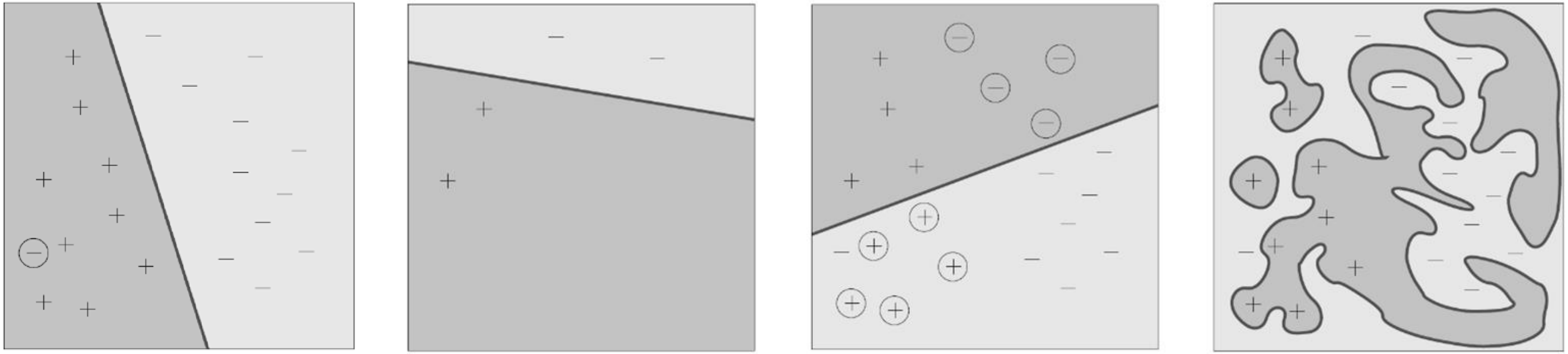
- ❑ K-Nearest Neighbour
- ❑ Naive Bayes Classifier
- ❑ Support Vector Machines (the basic SVM supports only binary classification); linear or with Gaussian kernels
- ❑ Decision Trees (e.g., Random Forest)
- ❑ Artificial Neural Networks (ANN)
- ❑ Hierarchal classifier
- ❑ …

Parametric vs. nonparametric models

Parametric	Nonparametric
Linear regression	Decision Trees
Naive Bayes	KNN
Linear SVMs	SVMs (nonlinear kernels)
Logistic regression	ANNs
Less flexibility	More flexibility

Good vs. bad classifiers?

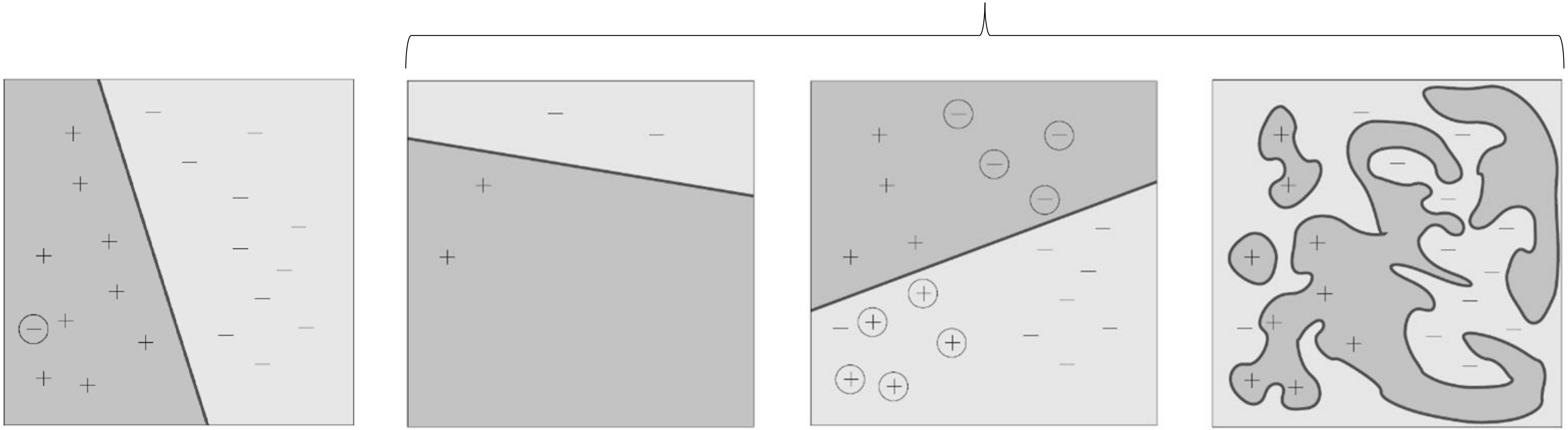
Assume, we have four classifiers developed on four training datasets.



Good vs. bad classifiers

Good

Bad



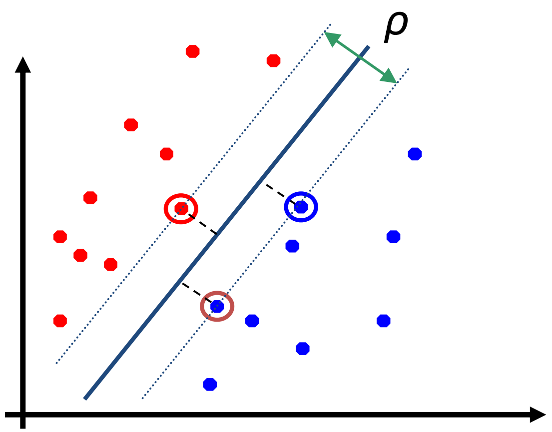
Sufficient data
Low training error
Simple classifier

Insufficient data

Training error too high

Classifier too complex

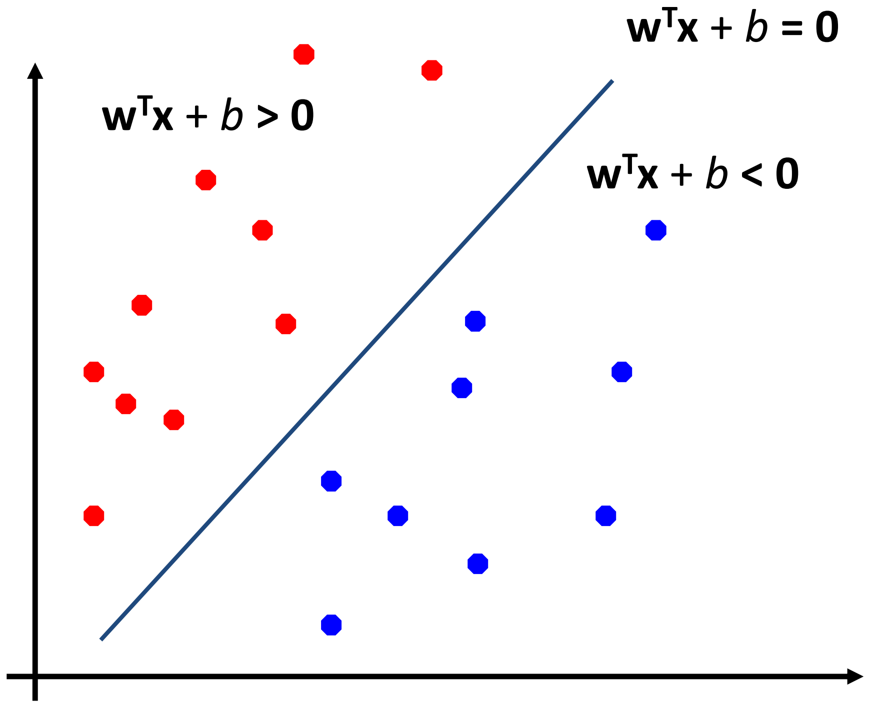
Support vector machines (SVM)



A linear separator

Given a set of training samples, an SVM training algorithm builds a model that assigns new samples to one of the two classes (binary classifier).

Training samples
Two classes: +1 & -1

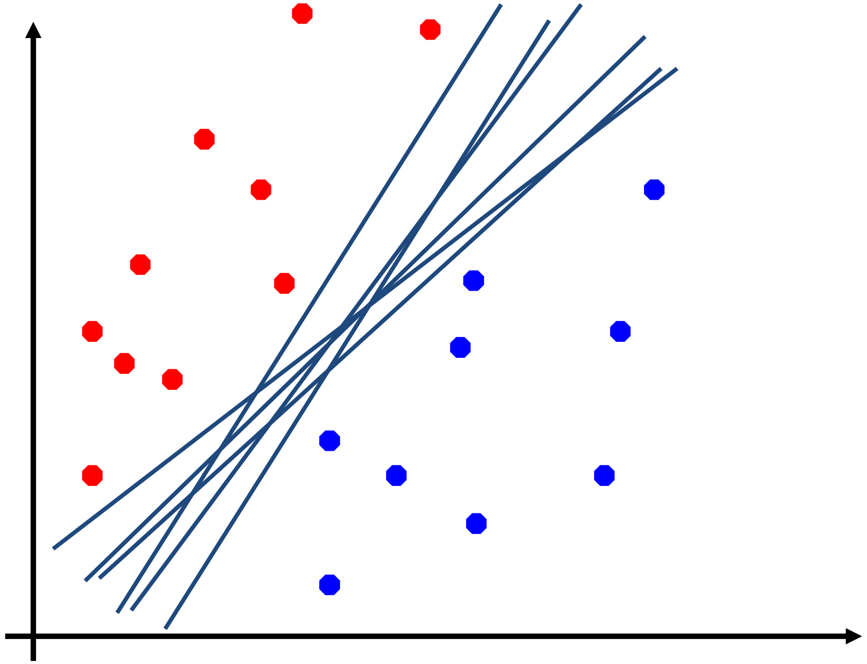


$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$

$$W^T X + b > 0 \quad for\ \textbf{red}\ samples$$
$$W^T X + b < 0 \quad for\ \textbf{blue}\ samples$$

Linear separators

Which of the linear separators is optimal?

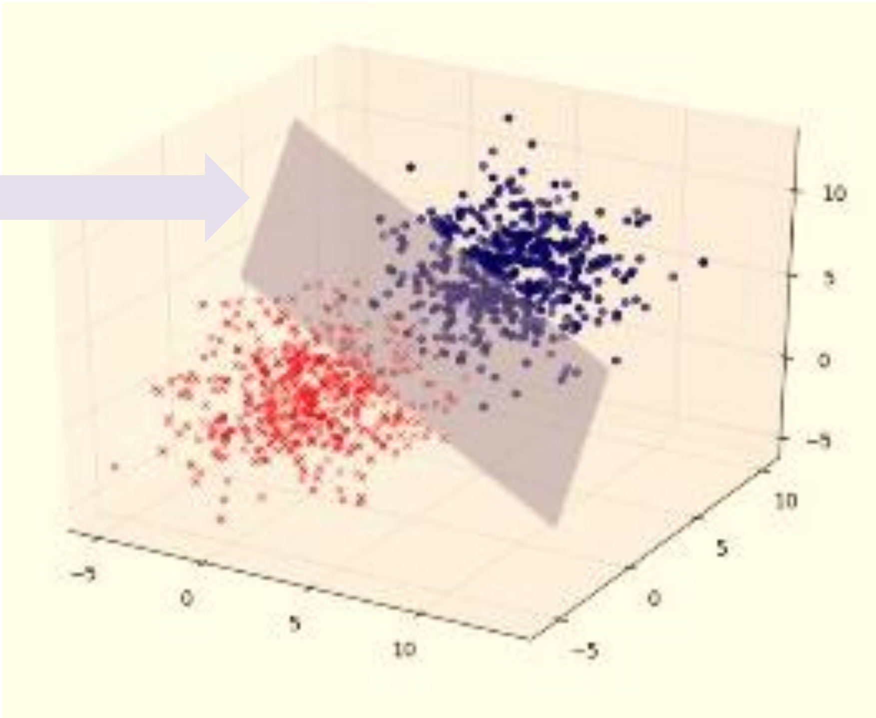


What is a hyperplane in geometry?

With a 3-dimensional space, hyperplanes are the 2-dimensional planes.
With a 2-dimensional space, its hyperplanes are the 1-dimensional lines.

Hyperplanes are a key tool to create SVMs

Hyperplane

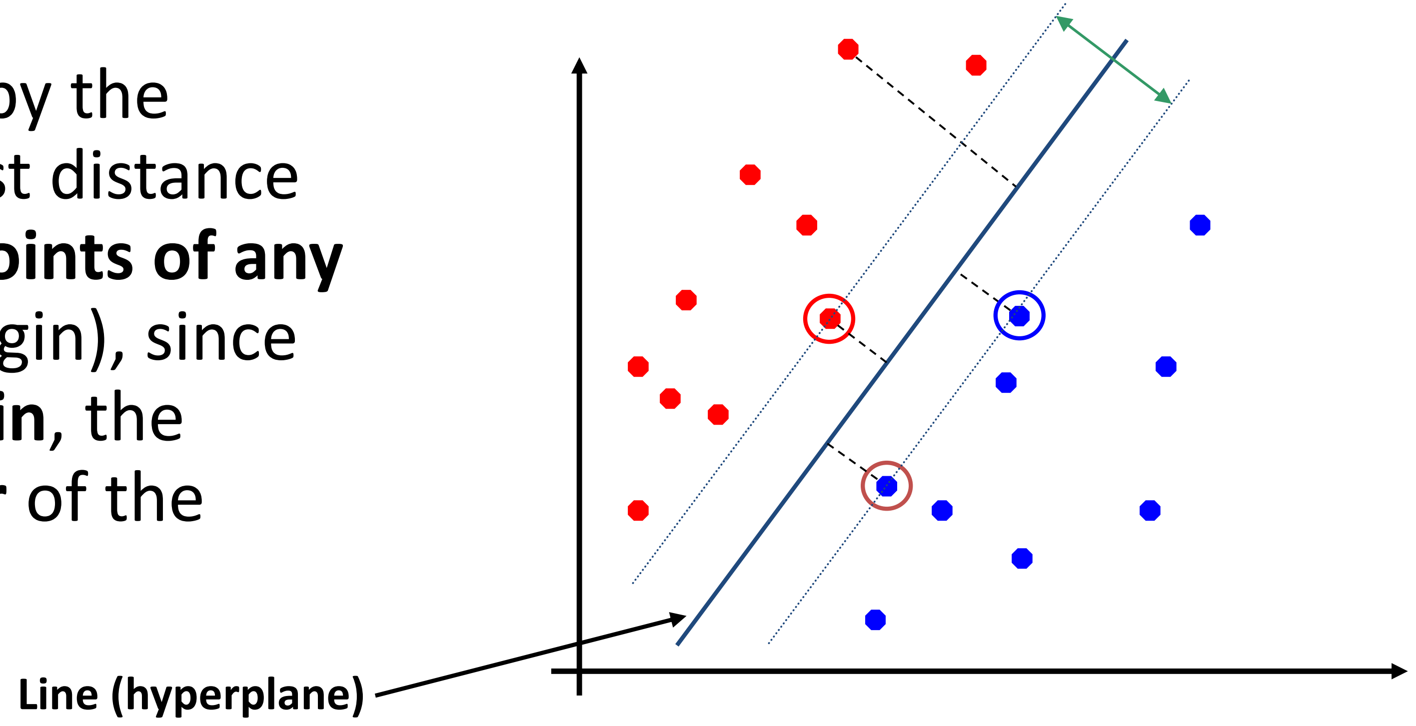


Hyperplane: in 2d, it's called a line, in 3d it's called a plane, more than 3d Hyperplane

Good separation using support vectors

Binary classification can be viewed as the task of separating classes in feature space.

A good separation is attained by the **hyperplane** that has the largest distance to the **nearest training data points of any class** (so-called functional margin), since in general the **larger the margin**, the **lower the generalisation error** of the classifier



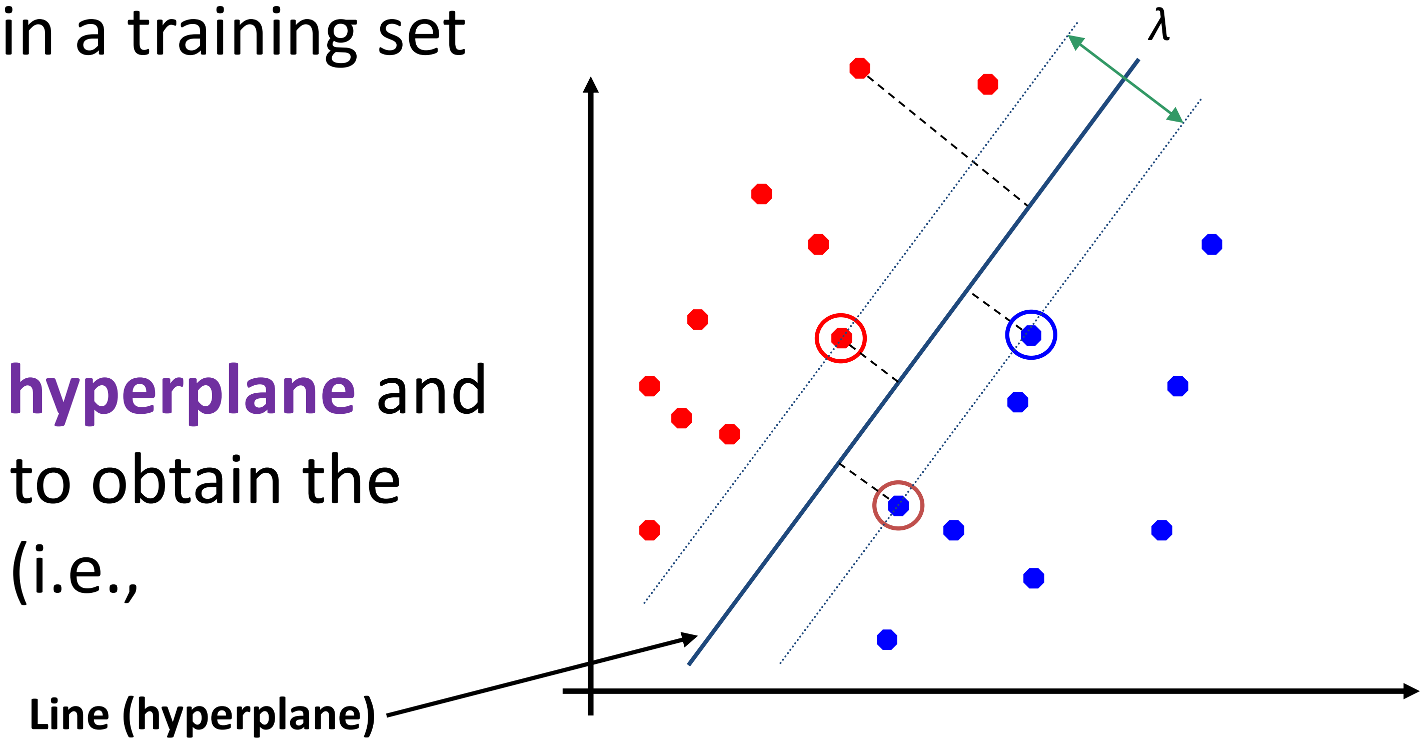
The support vectors are indicated by the circles around them.

Maximum margin classification

Margin λ of the separator is the distance between support vectors.

This **maximum-margin separator** is determined by a subset of the data points in a training set ("support vectors").

In SVM, we aim to find a **right hyperplane** and then **maximize the margin** (λ) to obtain the parameters of the hyperplane (i.e., optimization problem)

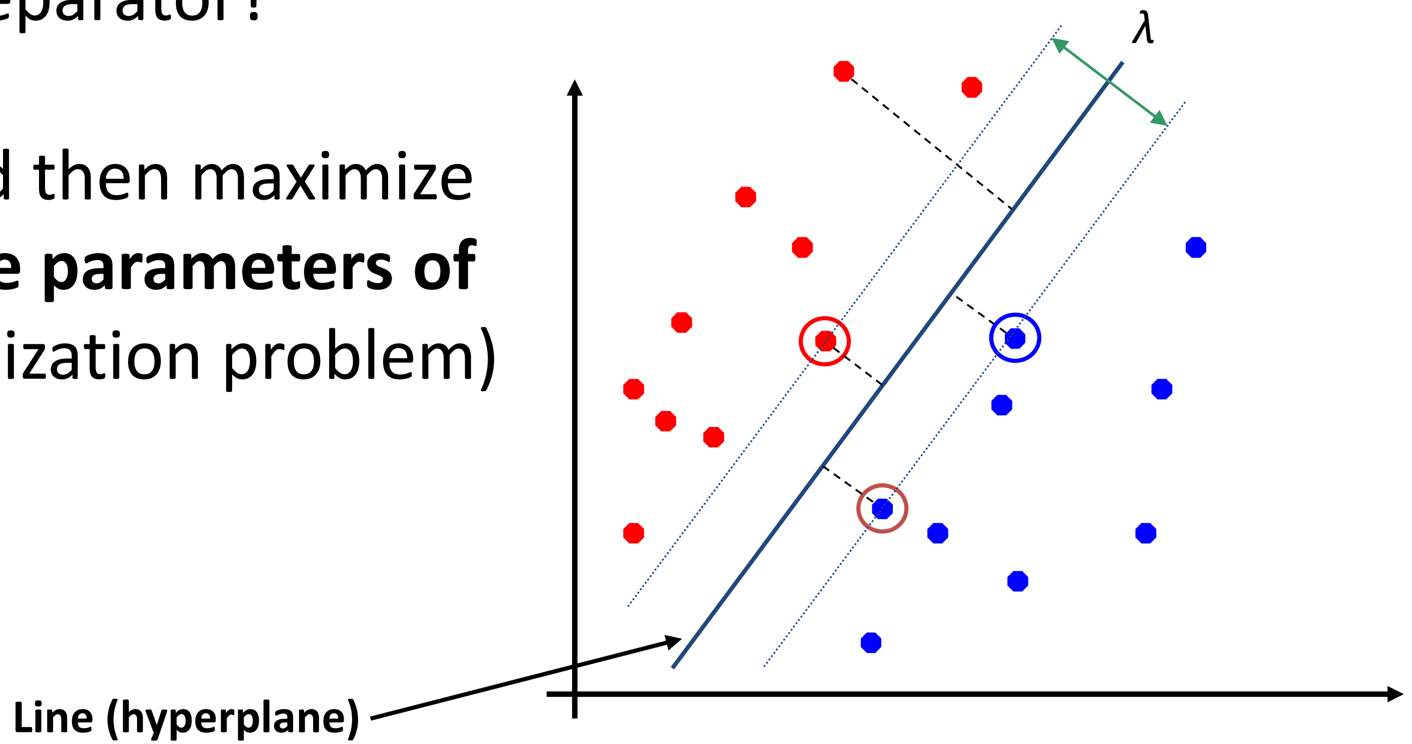


The support vectors are indicated by the circles around them.

Two key points when designing an SVM

1) Assess the level of your dataset complexity. Do you need a **linear** or **non-linear/Kernel** hyperplane function as a separator?

2) Find a right hyperplane and then maximize the margin (λ) to obtain **the parameters of the hyperplane** (i.e., optimization problem)



The support vectors are indicated by the circles around them.

Performance of SVM in general

- SVMs work very well in practice.
  - You must choose a linear or kernel function (i.e., hyperplane) and its parameters, but the rest is automatic.
  - The test performance is **very good**.
- SVM can be **computationally expensive** for big datasets
  - The computation of the maximum-margin hyperplane depends on the **square** of the number of training samples.

QUEEN'S UNIVERSITY BELFAST
SCHOOL OF ELECTRONICS, ELECTRICAL ENGNIEERING AND COMPUTER SCIENCE

An optimal SVM classifier in R using e1071 package

1) TUNING:

```
Tuning_model <- tune(svm, Trainingset450k17, label_vector,
scale = F, tolerance = 0.00001, type = "C-classification",
kernel = "radial", probability = T
ranges = list(cost= seq(0.0, 1.0, 0.2), gamma = seq(0, 15, 1)),
tunecontrol= tune.control(sampling = “cross”, cross=10), seed=i)
```

The darkest shades of blue indicating the best (see the two plots).

Narrowing in on the darkest blue range and performing further tuning.

```
Plot(Tuning_model, xlime=range(0:15), ylime=range(0:1))
Plot(Tuning_model, xlime=range(0.2:0.25), ylime=range(8:12))
```

2) TRAINING:

```
Radial_model <- svm(Trainingset450k17, label_vector,
scale = F, tolerance = 0.00001, type = "C-classification",
kernel = "radial",
cost = optimum_cost, gamma = optimum_gamma,
probability = T, seed = i)
```

3) TESTING (PREDICTION):

```
Radial_model <- predict(object= Radial_model, newdata = seq.test.BEM.97, probability=T)
```

Three key steps

1) Tuning

Choose a hyperplane; try linear or nonlinear (polynomial or RBF kernels) and find it’s parameters

2) Training

Train the classifier based on the identified

parameters of the hyperplane

3) Testing

Test the trained classifier by giving it some new samples (without subgroups)

Find the parameters of a non-linear function (kernel function)

TUNING:

Tuning_model <- tune(svm, **Trainingset450k17**, label_vector,
scale = F, tolerance = 0.00001, type = "C-classification",
kernel = "radial", probability = T
ranges = list(**cost**= seq(0.0, 1.0, 0.2), **gamma** = seq(0, 15, 1)),
tunecontrol= tune.control(**sampling = "cross"**, cross=**10**), seed=i)

Input training dataset: **Trainingset450k17**

Label_vector: a vector of all sample **class labels (subgroup labels)**

1) Tuning

Choose a hyperplane and find it's parameters: **radial basis function** with two parameters which are **cost** and **gamma**

Using a **grid search**
and **10-fold cross validation technique**

Run multiple times the ***tune()*** to find the best (optimum) parameters

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

Default for the tolerance of termination criterion: 0.001)

Tuning the model; grid search and 10-fold cross validation

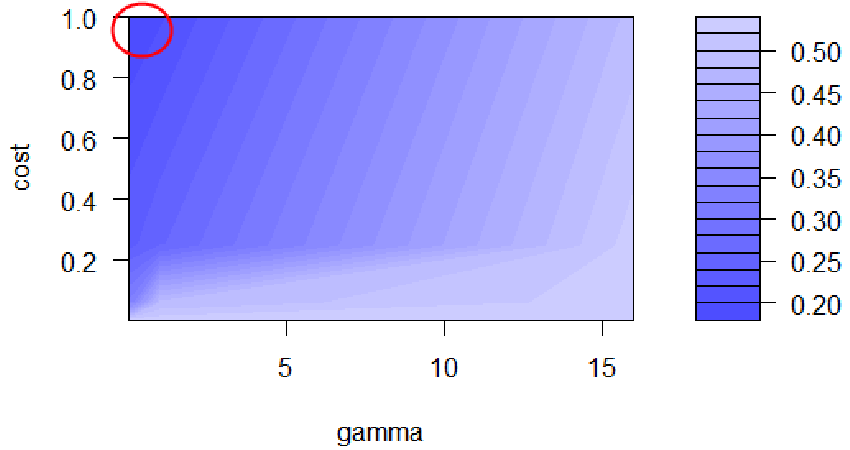
TUNING:

```
Tuning_model <- tune(svm, Trainingset450k17, label_vector,
scale = F, tolerance = 0.00001, type = "C-classification",
kernel = "radial", probability = T
ranges = list(cost= seq(0.0, 1.0, 0.2), gamma = seq(0, 15, 1)),
tunecontrol= tune.control(sampling = “cross”, cross=10), seed=123456)

Plot(Tuning_model, xlime=range(0:15), ylime=range(0:1))
```

The darkest shades of blue indicating the best (see the plot).

Performance of SVM model – error rate



interval

0 1

cost= seq(0.0, 1.0, 0.2)

interval

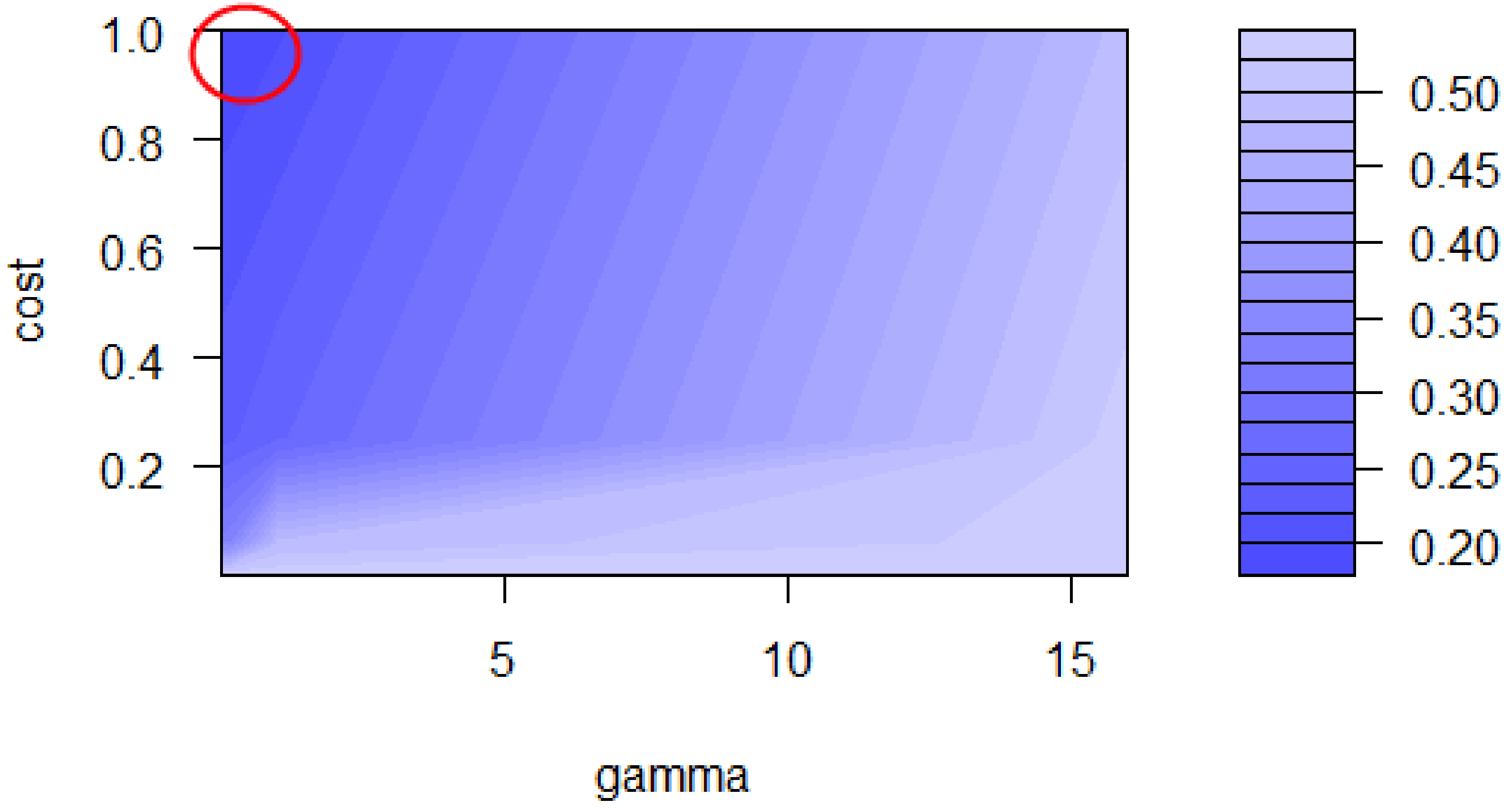
0 15

gamma = seq(0, 15, 1)

Further tuning

The darkest shades of blue indicating the lowest error.

Performance of SVM model – error rate



Narrowing in on the darkest blue range and performing further tuning.

Tuning the model; grid search and 10-fold cross validation

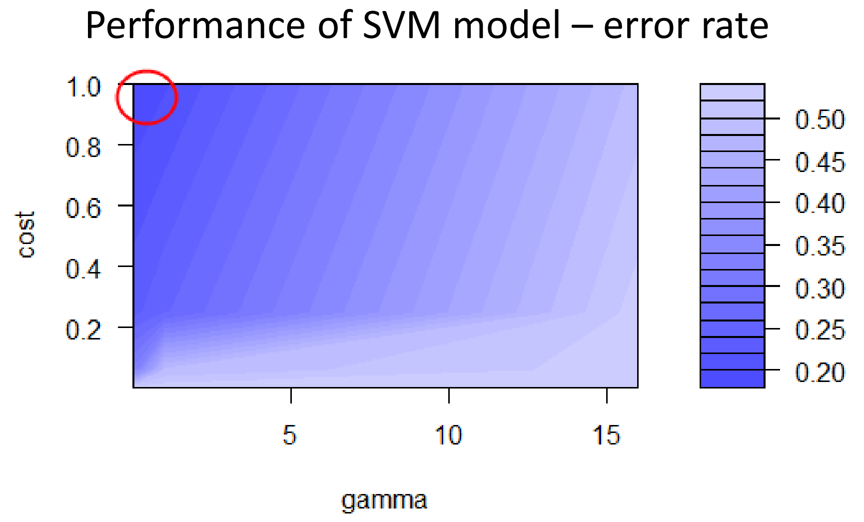
TUNING:

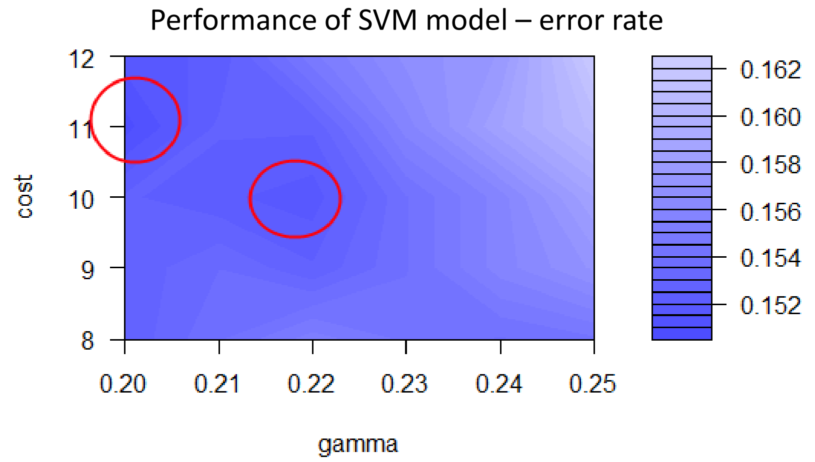
```
Tuning_model <- tune(svm, Trainingset450k17, label_vector,
scale = F, tolerance = 0.00001, type = "C-classification",
kernel = "radial", probability = T
ranges = list(cost= seq(8, 12, 1), gamma = seq(0.20, 0.25, 0.01)),
tunecontrol= tune.control(sampling = “cross”, cross=10), seed=123456)

Plot(Tuning_model, xlime=range(0:15), ylime=range(0:1))

Plot(Tuning_model, xlime=range(0.2:0.25), ylime=range(8:12))
```

The darkest shades of blue indicating the best (see the two plots).
Narrowing in on the darkest blue range and performing further tuning.





interval

8 12

cost= seq(8, 12, 1)

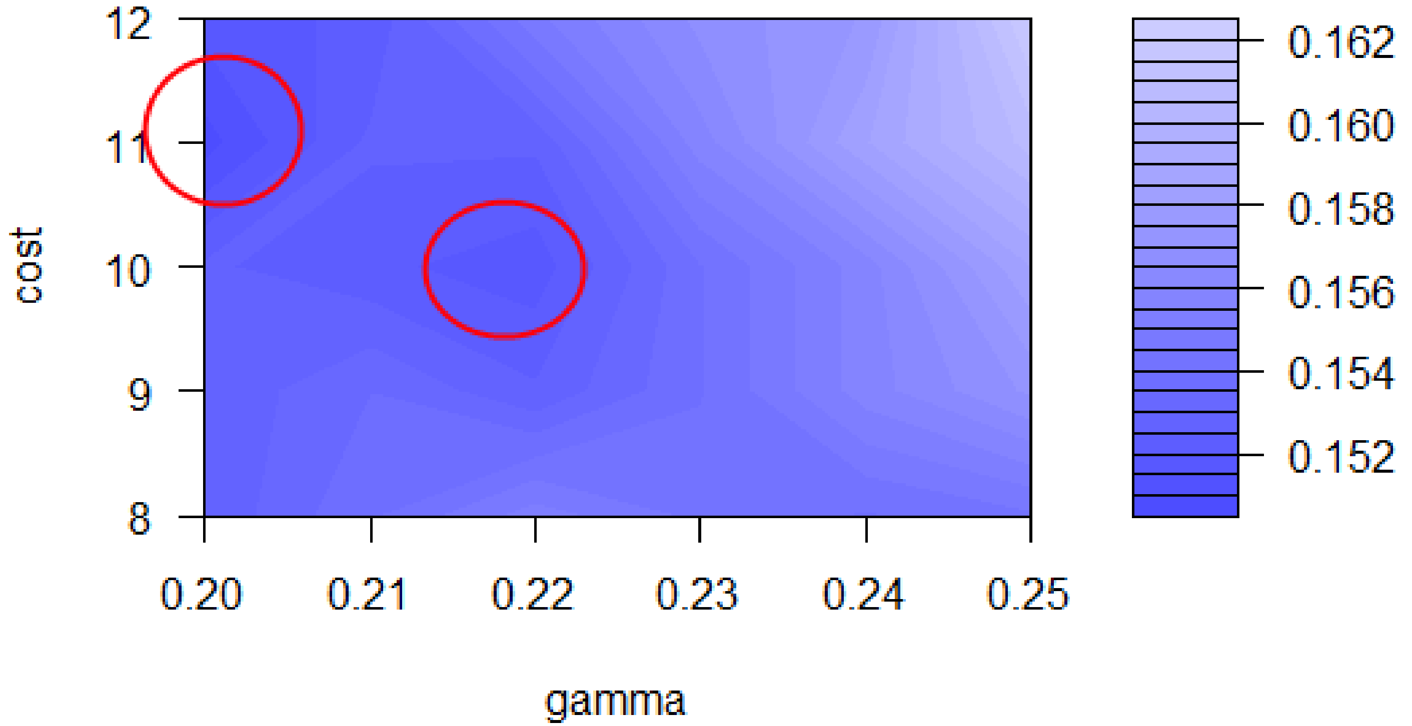
0.20 0.25

interval

gamma = seq(0.20, 0.25, 0.01)

Final parameters of the kernel function

Performance of SVM model – error rate



interval

8 12

cost= seq(8, 12, 1)

interval

0.20 0.25

gamma = seq(0.20, 0.25, 0.01)

Any Questions?