Universidade do Minho
Departamento de Informática
Synthetic Intelligence Lab

# Aviso

No dia 03/11/2022, devido à realização de um teste com um n.º elevado de alunos a aula T das 9:00, muda de sala:

| UO | Data | Sala atual | Horário | Curso | UC | Docente | Nova sala atribuída |
|---|---|---|---|---|---|---|---|
| **DI** | 03/11/2022 | Edifício 1 - 0.04 | 9h00-10h00 | 1º MEI + 4º MEINF | Dados e Aprendizagem Automática-T2 | Victor Alves | Edifício 2 - 0.07 |

Universidade do Minho
Departamento de Informática

Synthetic Intelligence Lab

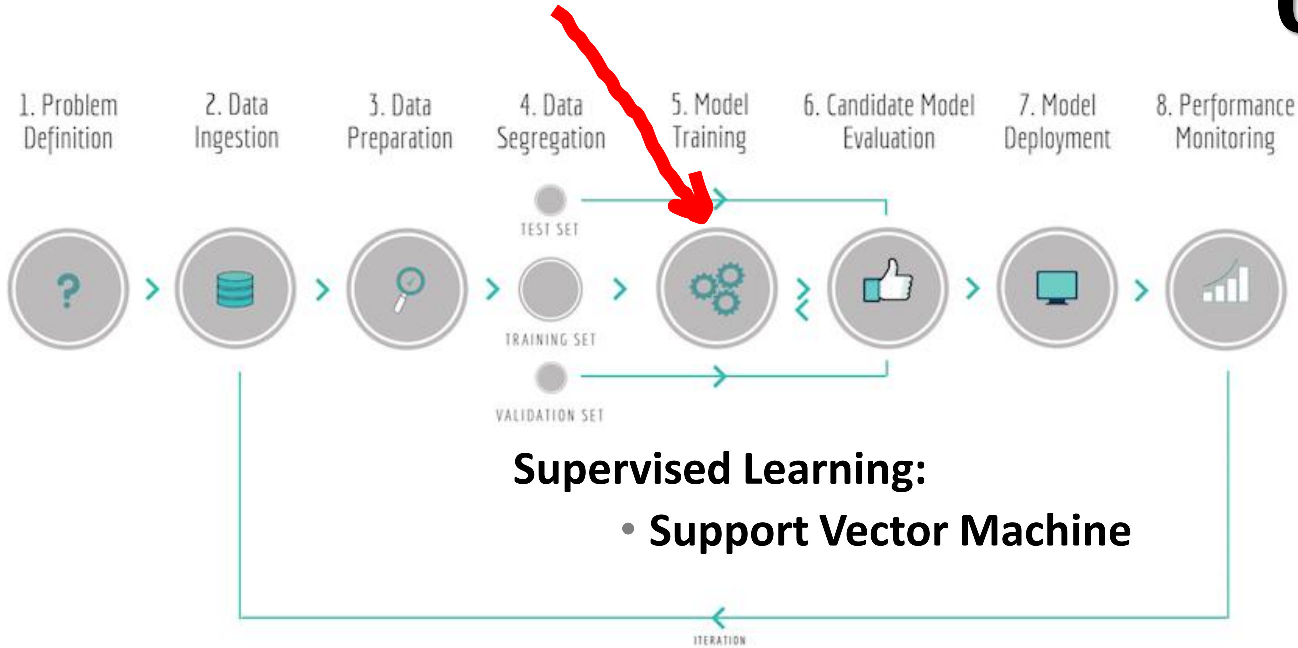
# DADOS e APRENDIZAGEM AUTOMÁTICA

## *Support Vector Machine*

MESTRADO (integrado) EM ENGENHARIA INFORMÁTICA

Universidade do Minho
Departamento de Informática
ISLab
Synthetic Intelligence Lab

# Contents

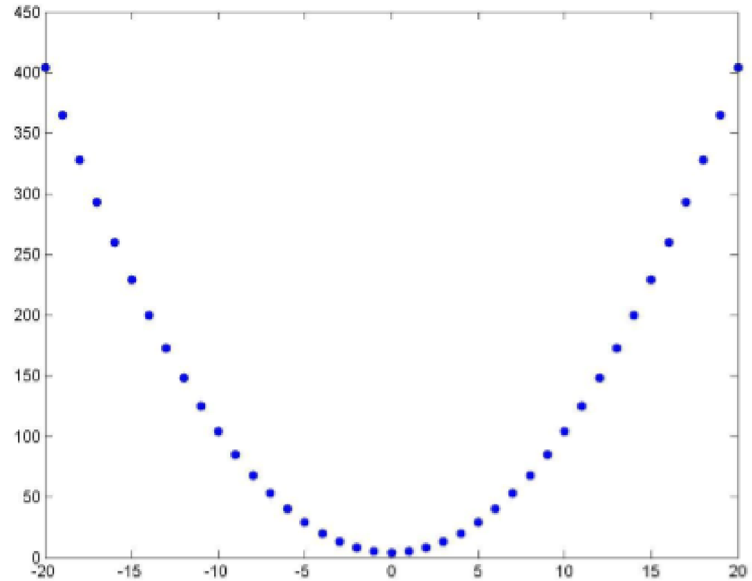



**Supervised Learning:**

- **Support Vector Machine**

# Problem

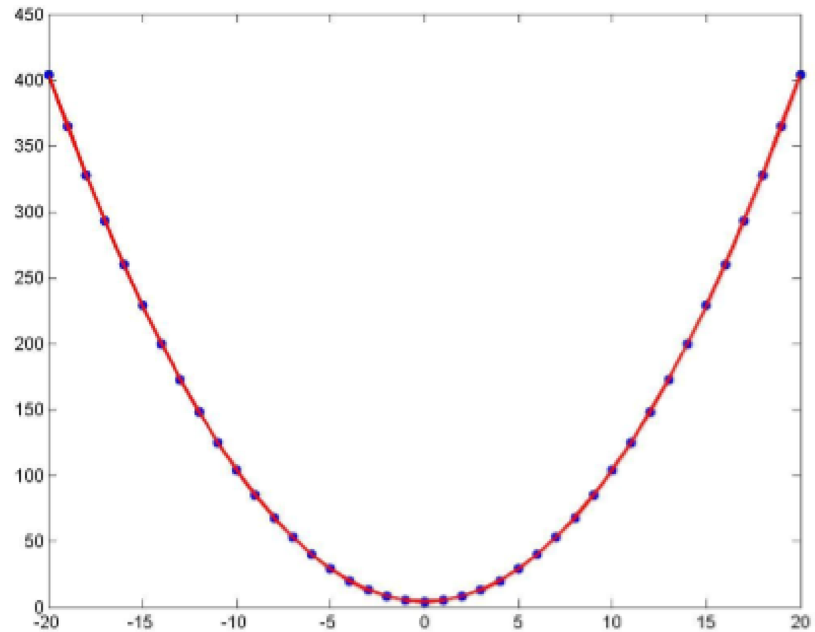
- Predict y given x




- Try to fit a function to describe this

# Problem

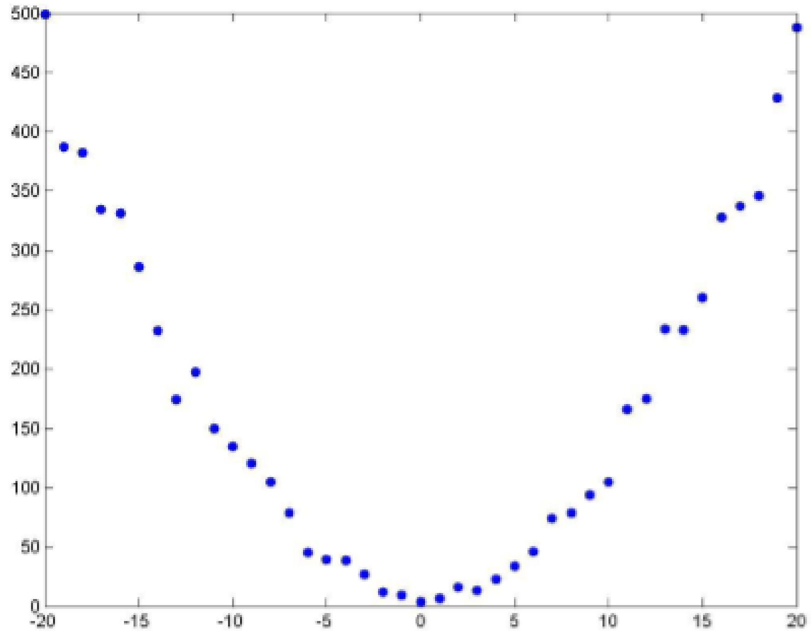
- Easy!




- For a new point we will be correct

# Problem

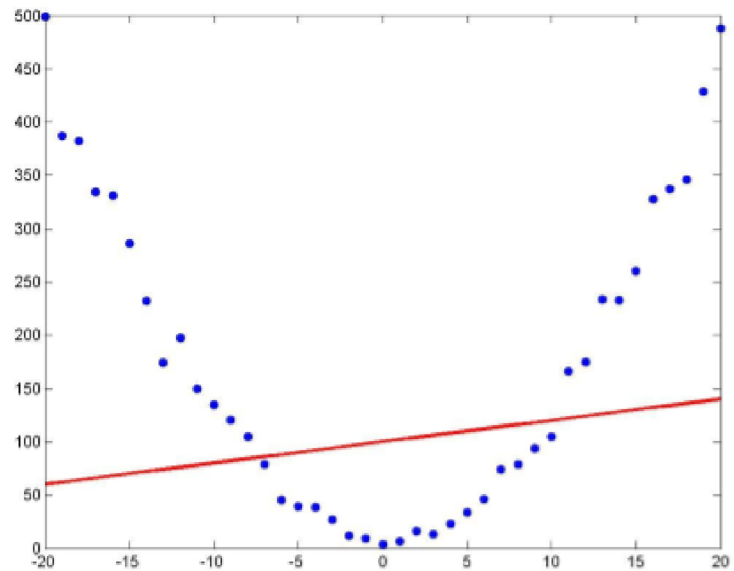
- What if we add some noise?




- In real environment (data) we cannot "see" the function

# Problem

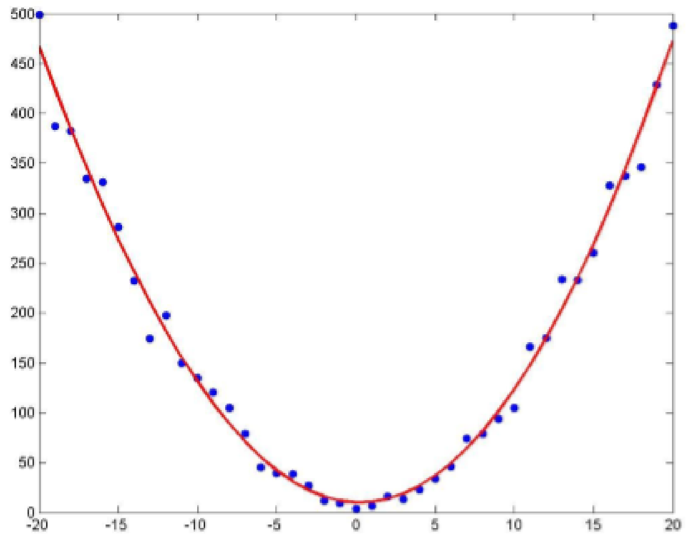
- We could assume the relationship to be linear




- How wrong are we?
- How do we know which parameters are the best?

# Problem

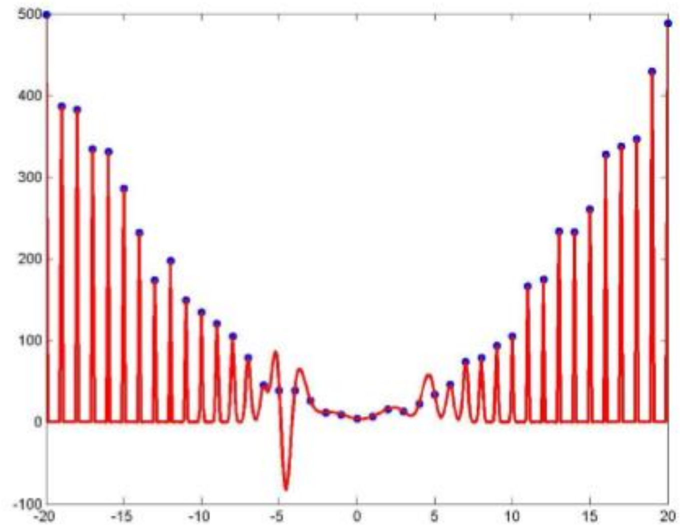
- Linear is still bad
- Increase parameters, and we can consider a quadratic function




- This is better
- But still has some loss
- Increase parameters!

# Problem

- **Zero error** on training set!
- What about a test example?
- Too specific = **overfitting**
- Empirical risk minimization: overfits if you follow it blindly



# Problem

- How to avoid it?
- Need to **balance training error and capacity** of a function:
  - Too complex a function will overfit;
  - Not enough complexity will not generalize well either.

# Support Vector Machine

- Support Vector Machine is a **supervised Machine Learning algorithm** that can be used for both classification (mostly) or regression problems.

- Usually a learning algorithm tries to **learn the most common characteristics** (what differentiates one class from another) of a class and the classification is based on those representative characteristics learnt (classification is based on differences between classes). The SVM works in the other way around. It finds the **most similar examples between classes**. Those will be the support vectors.

- The main idea is to plot each data item as a point in an n-dimensional space (n is the number of features), performing classification by **finding the hyper-plane that differentiates the classes**.

# Support Vector Machine

- Works well for **classifying higher-dimensional data** (datasets with lots of features)
- Finds higher-dimensional support vectors which “divides” the data
- **Applies kernels to represent data in higher-dimensional spaces** to find hyperplanes that might not be apparent in lower dimensions

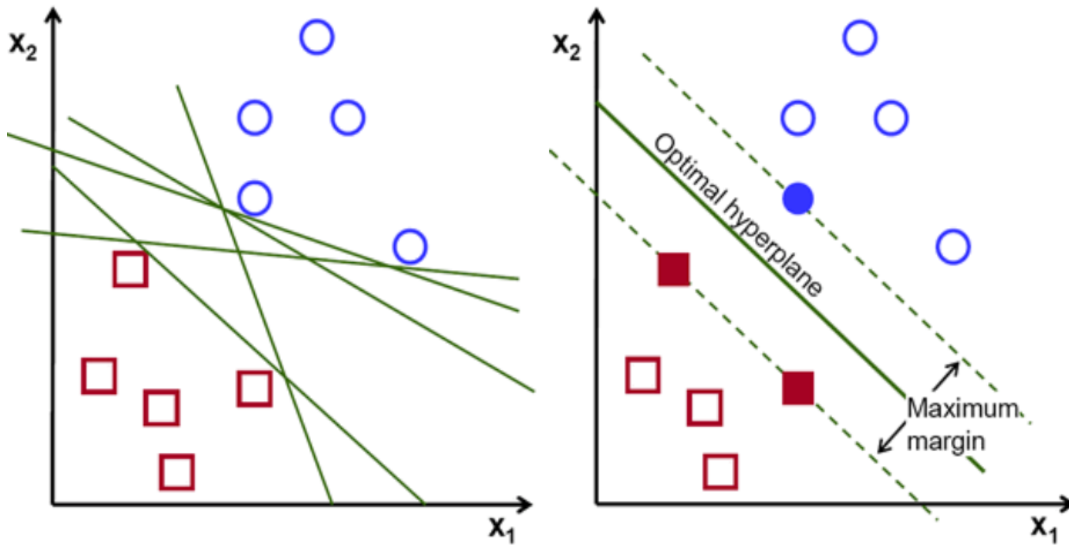



Image source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Universidade do Minho
Departamento de Informática
ISLab
Synthetic Intelligence Lab

# Support Vector Machine

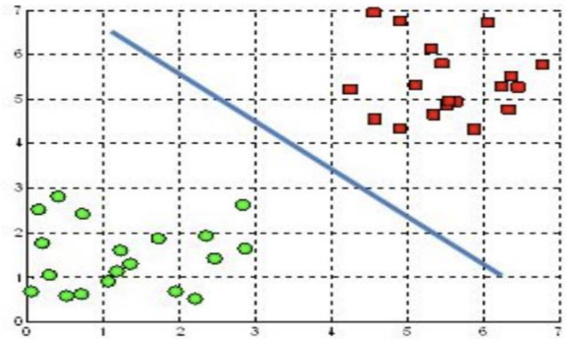
- Hyperplanes are decision boundaries that help classify the data points;

- Data points falling on either side of the hyperplane can be attributed to different classes;

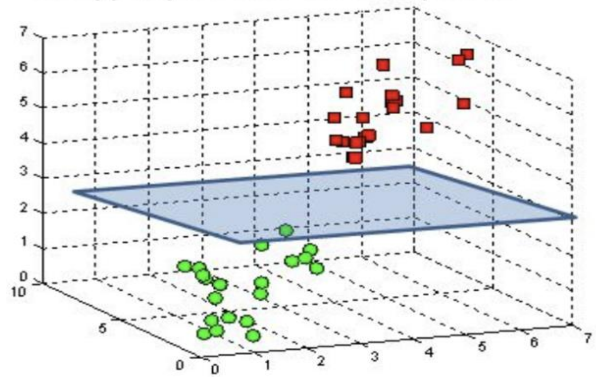
- The dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane;

- Difficult to imagine when the number of features exceeds 3.

A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane

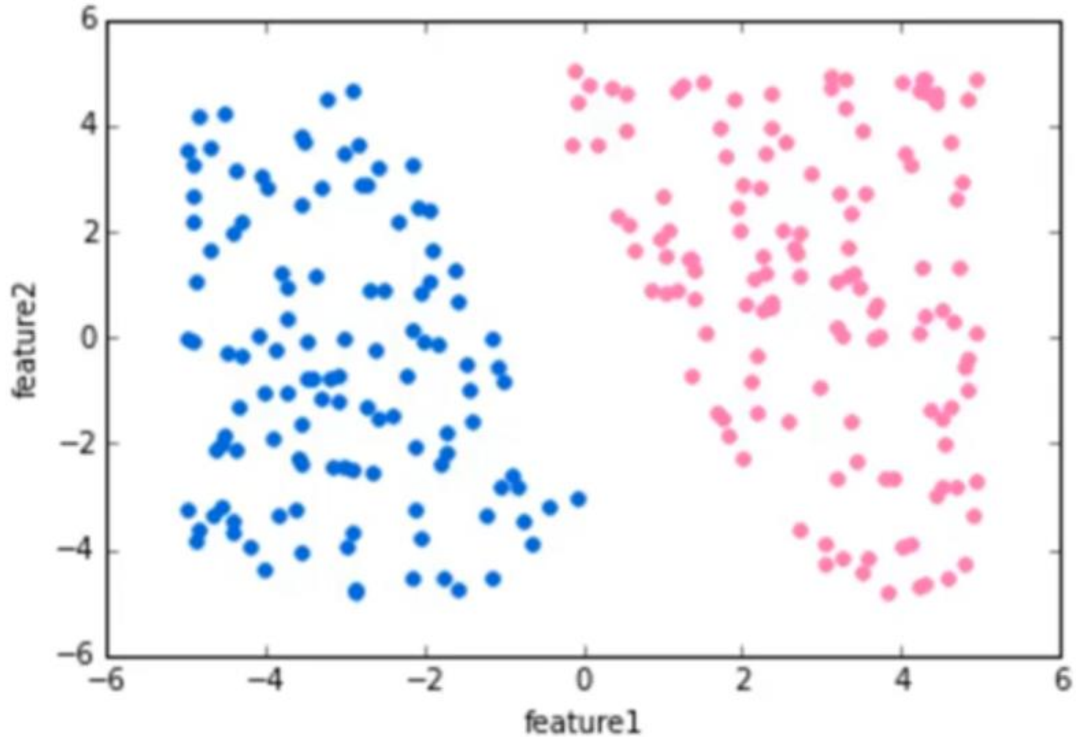


Source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

# Support Vector Machine

**How it works?**

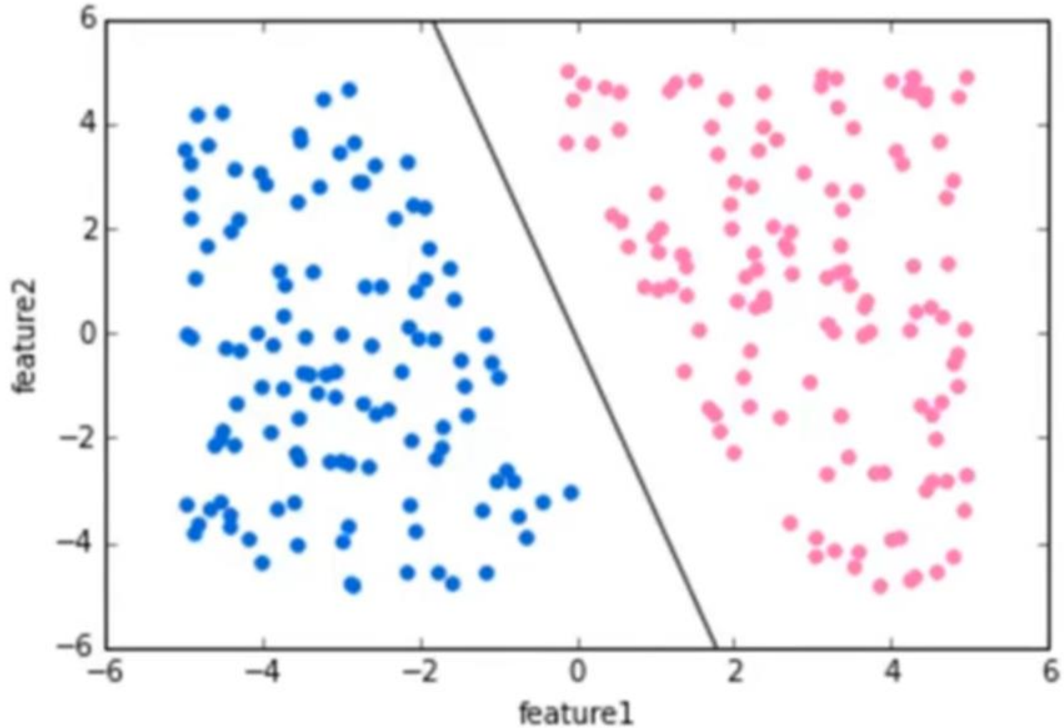
1) Image a labelled training data



# Support Vector Machine

**How it works?**

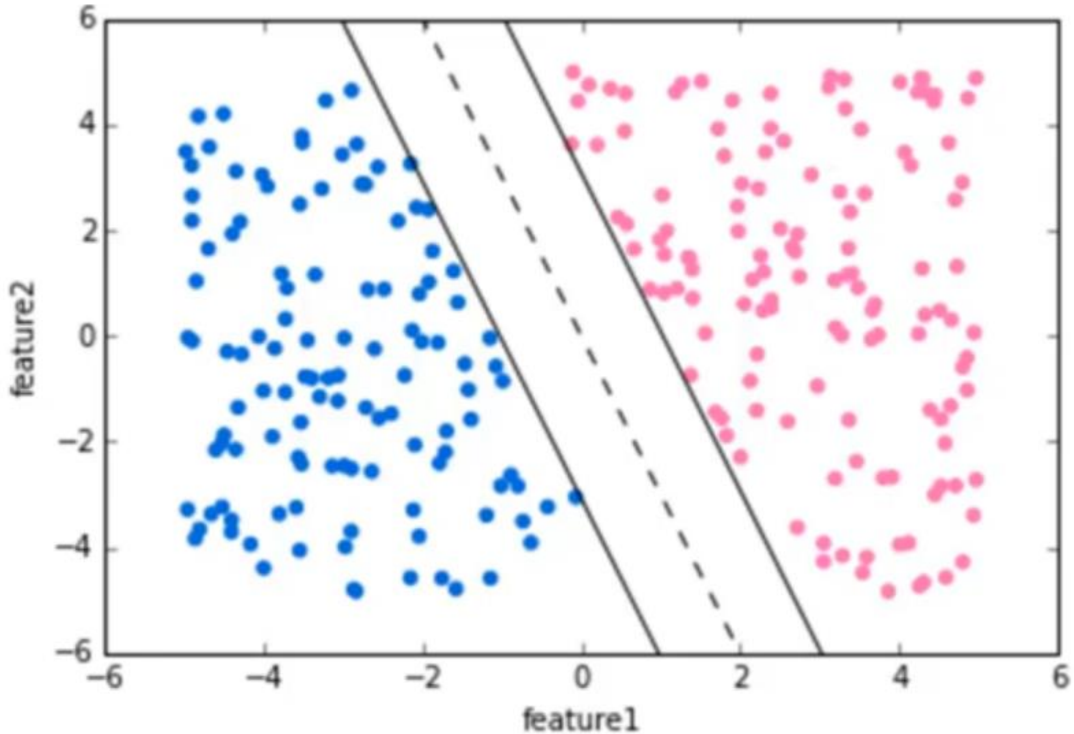
1) Image a labelled training data
2) Draw a separating "hyperplane" between the classes



# Support Vector Machine

**How it works?**

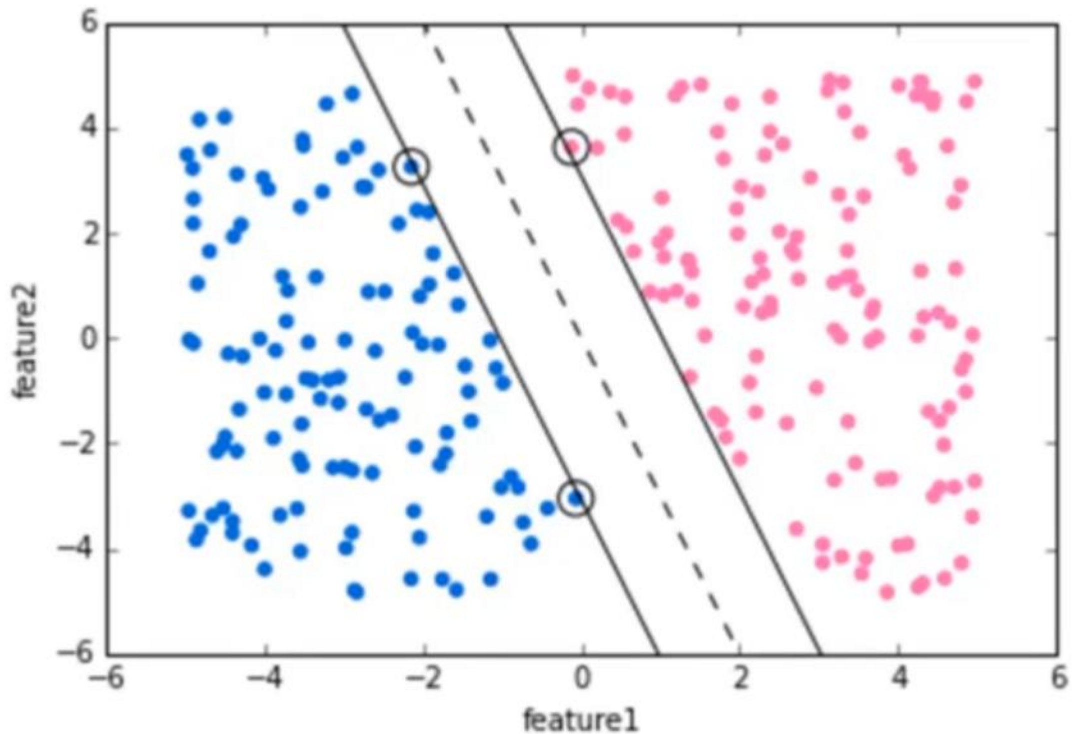
1) Image a labelled training data
2) Draw a separating "hyperplane" between the classes – many options that separate perfectly…
3) Choose a hyperplane that maximizes the margin between classes



# Support Vector Machine

How it works?

1) Image a labelled training data
2) Draw a separating "hyperplane" between the classes – many options that separate perfectly…
3) Choose a hyperplane that maximizes the margin between classes – vector points that the margin lines touch are known as Support Vectors



# Loss Functions

- We are looking to maximize the margin between the data points and the defined hyperplane. For that, SVMs use the **hinge loss**:

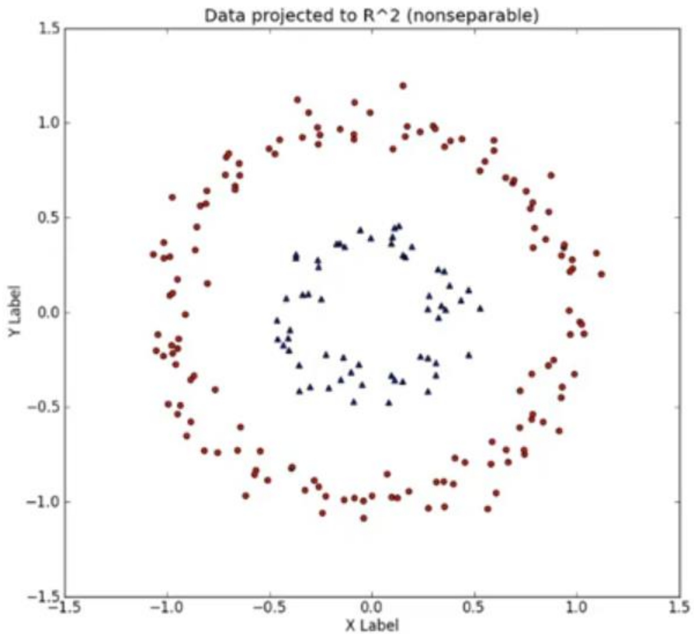
$$\ell(y) = \max(0, 1 - t \cdot y)$$

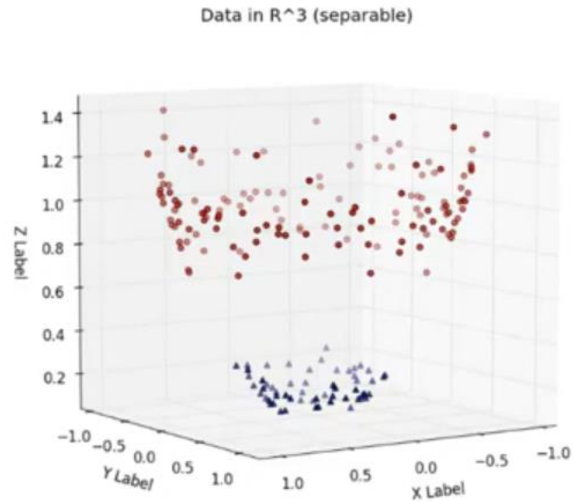
Return 0 if the predicted value and the real value have the same sign.

Otherwise: calculate loss!

# Support Vector Machine

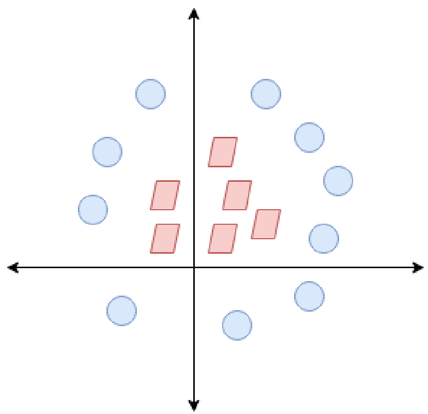
The idea can be expanded to non-linearly separable data through the “kernel trick”

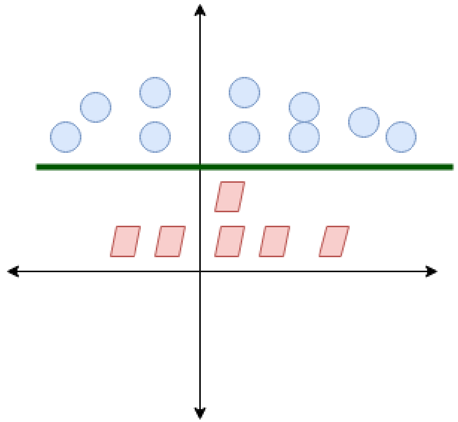


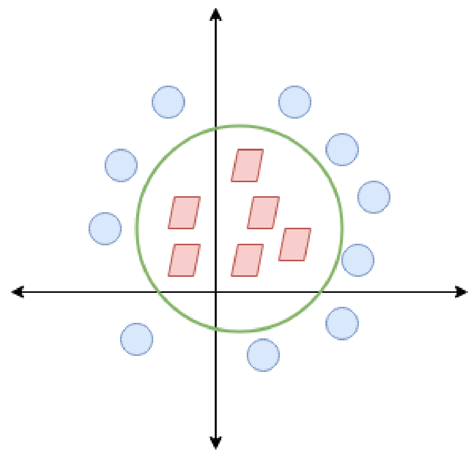



# Support Vector Machine - Kernel

$$z = x^2 + y^2$$

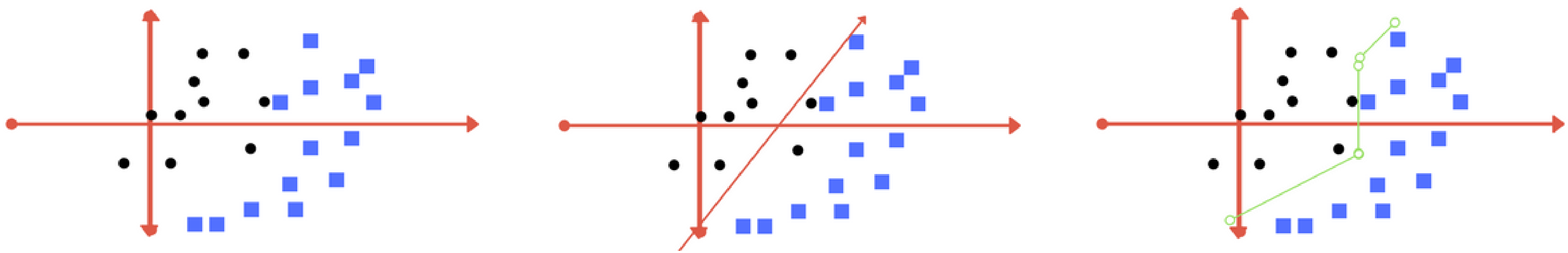




This transformation
is called kernel

# Support Vector Machine - Hyperparameters



Which hyperplane is better ?

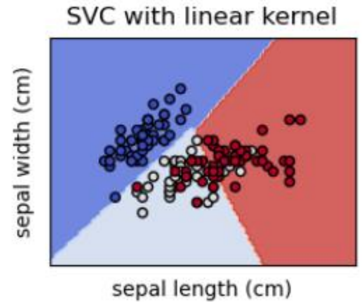
**Tuning parameters: Kernel, Regularization, Gamma and Margin.**

# Support Vector Machine - Kernel

- Different kernels provide different results for a given dataset

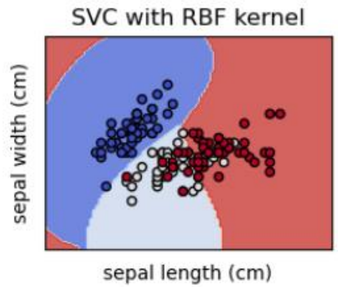
**Linear**: A Linear Kernel

$$K(x, x') = x \cdot x'$$



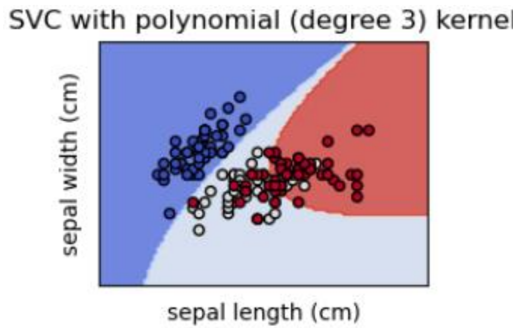

**Gaussian Radial Basis Function** (RBF):

$$K(x, x') = \exp(-\gamma \|x - x'\|^2),$$
$$\text{where } \gamma = \frac{1}{2\sigma^2}$$




**Polynomial**:

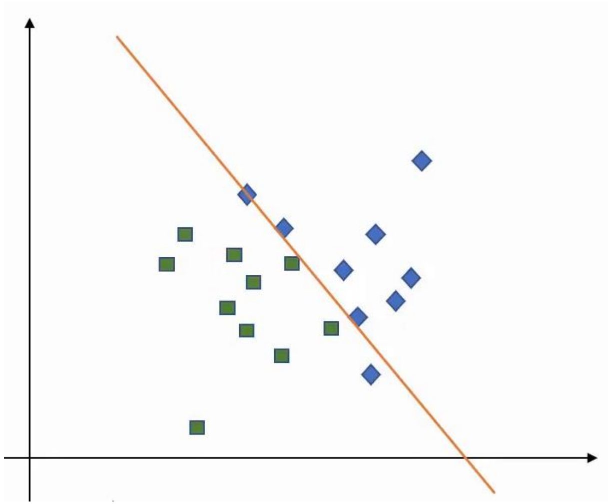
$$K(x, x') = (x^T x' + c)^d$$




**Sigmoidal**:

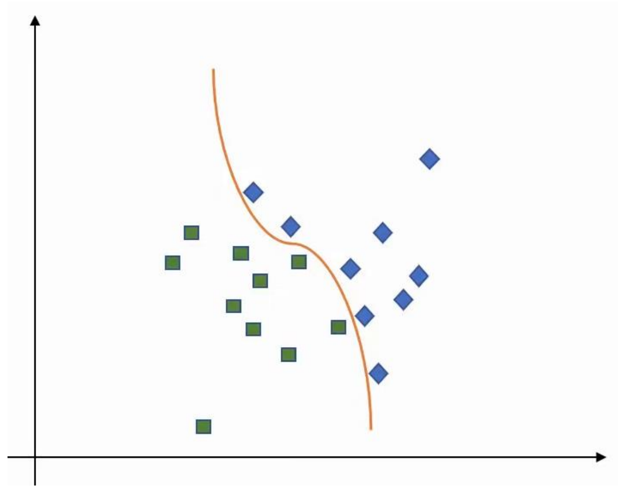
$$K(x, x') = \tanh(kx^T x' - \delta)$$

# Support Vector Machine - Regularization

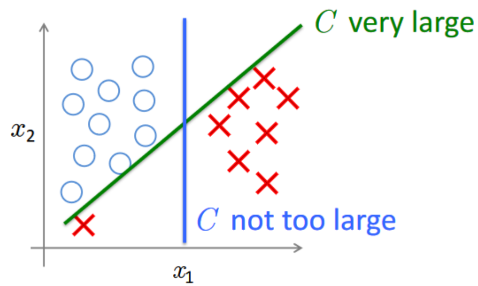


**Low Regularization (C)** - the optimizer to look for a **larger-margin** separating hyperplane, even if that hyperplane misclassifies more points.

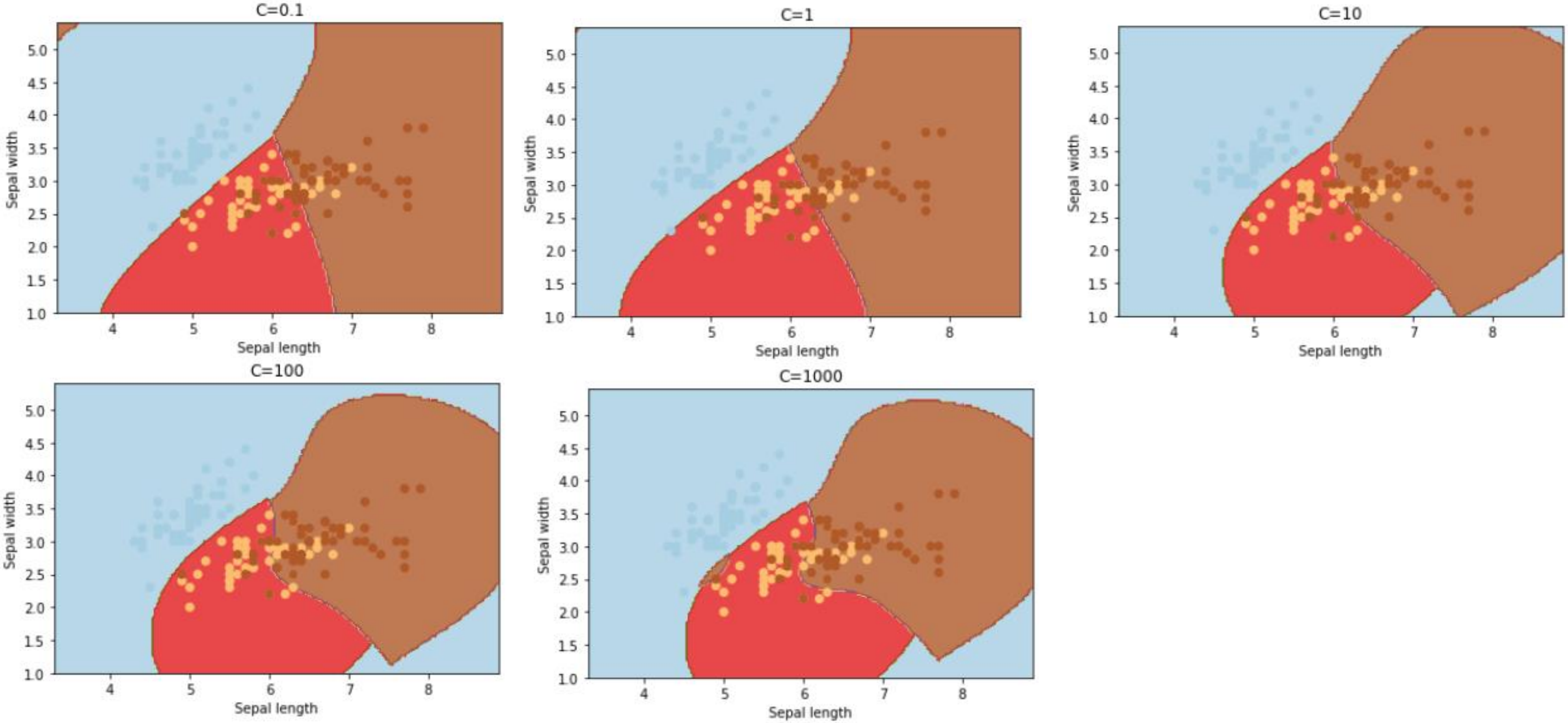
```
model = svm.SVC(kernel="rbf", C = 1)
model = svm.SVC(kernel="rbf", C = 10)
model = svm.SVC(kernel="rbf", C = 100)
model = svm.SVC(kernel="rbf", C = 1000)
model = svm.SVC(kernel="rbf", C = 10000)
```



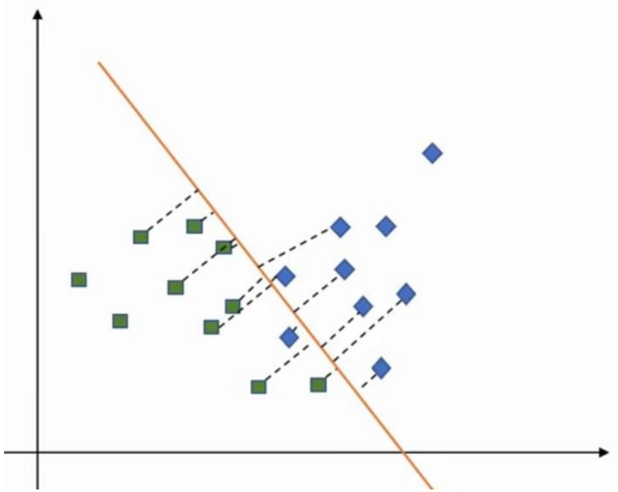
**High Regularization (C)** - the optimization will choose a **smaller-margin** hyperplane if that hyperplane does a better job of getting all the training points classified correctly



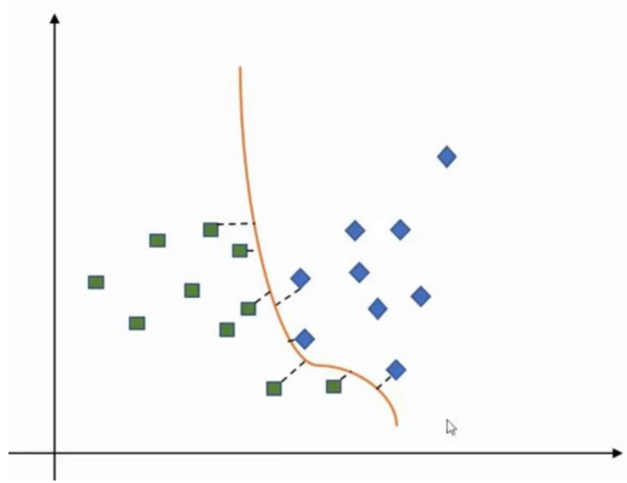
# Kernel = ‘rbf’



# Support Vector Machine - Gamma



**Low Gamma** - **points far away** from plausible hyperplane are considered in calculation for the hyperplane

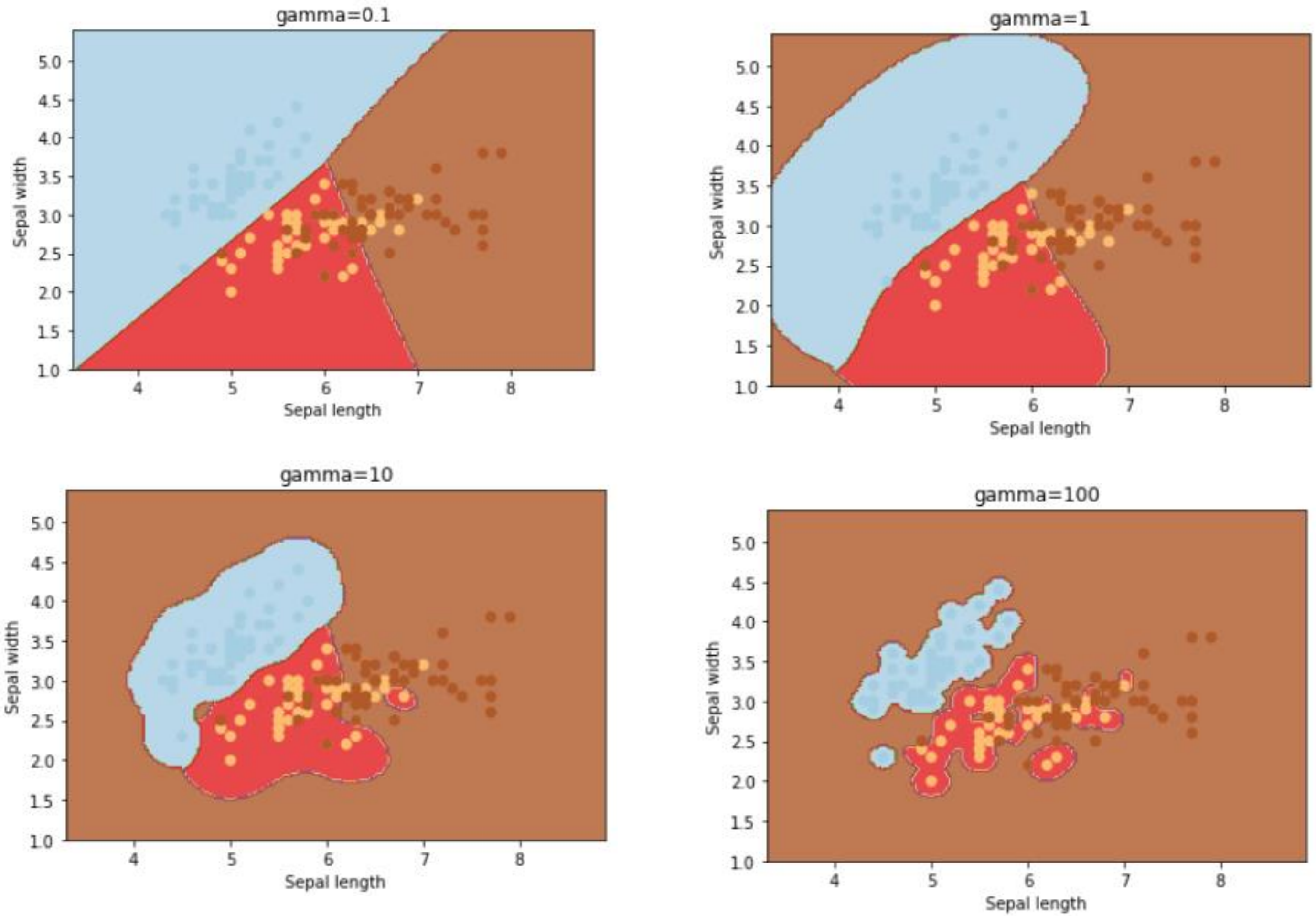


**High Gamma** - **only points close** to plausible line are considered in calculation

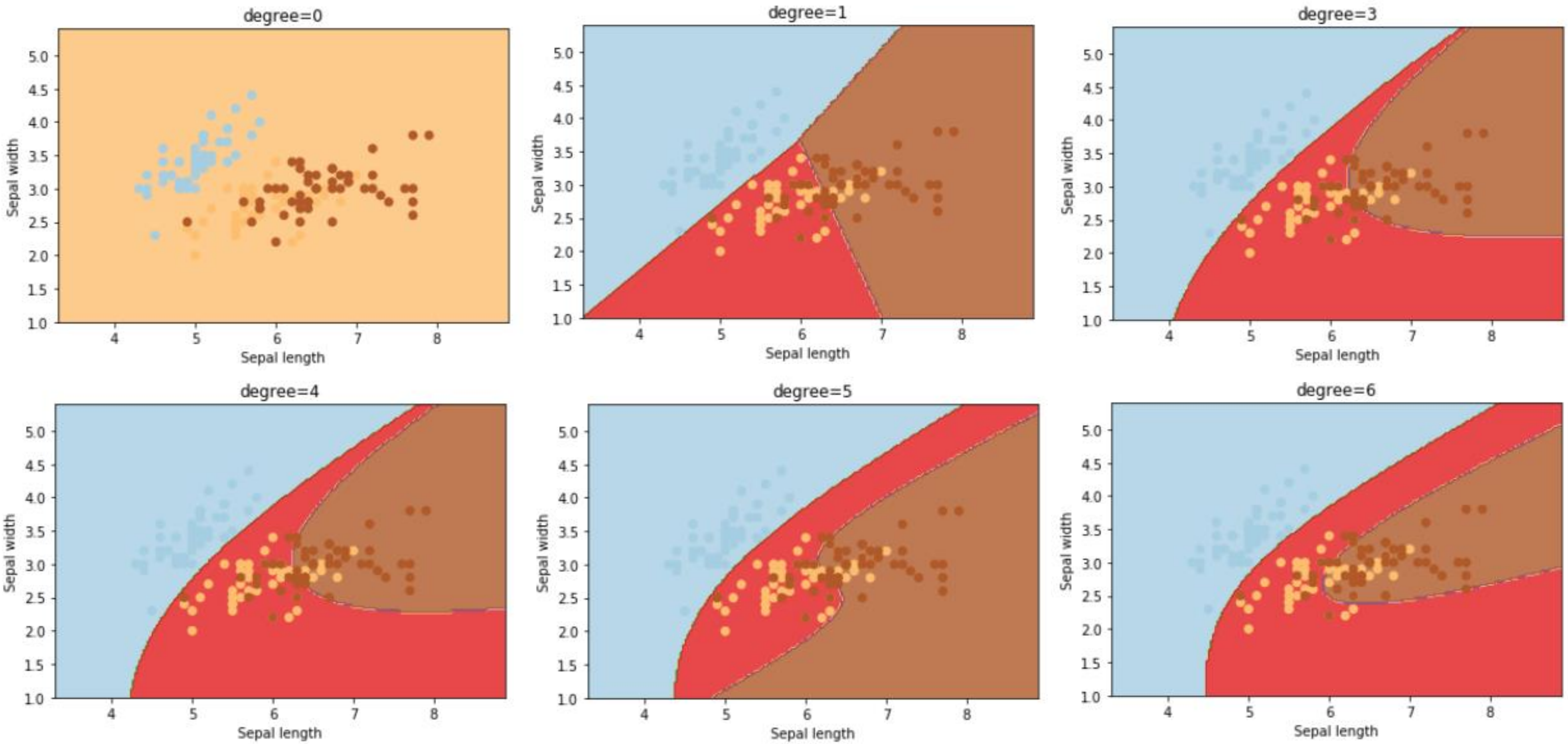
```
model = svm.SVC(kernel="rbf", C=100, gamma=1)
model = svm.SVC(kernel="rbf", C=100, gamma=0.1)
model = svm.SVC(kernel="rbf", C=100, gamma=0.01)
model = svm.SVC(kernel="rbf", C=100, gamma=0.001)
```

# Support Vector Machine - Gamma

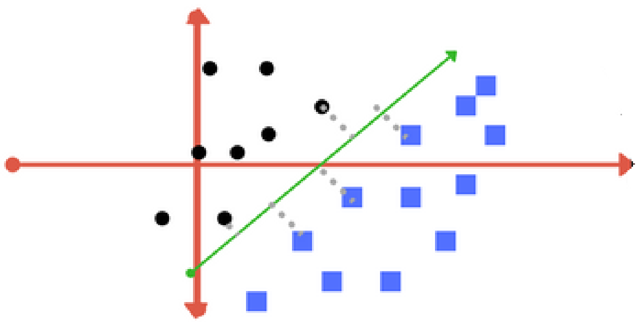
Kernel = 'rbf'



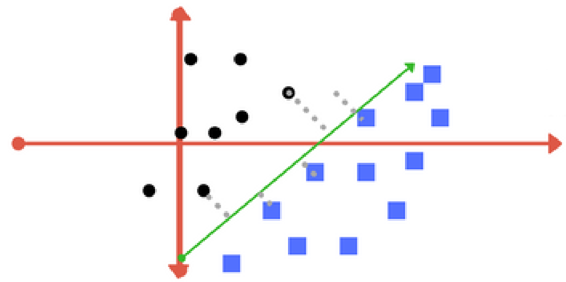
# Kernel = ‘poly’ (degree)



# Support Vector Machine - Margin



**Good margin** – equidistant as far as possible from both sides



**Bad margin**- very close to blue class

The margin should be always **maximized**.

# Multiclass

The **multiclass problem is broken down to multiple binary classification cases**, which is also called **one-vs-one**. In scikit-learn one-vs-one is not default and needs to be selected explicitly. **One-vs-rest** is set as default. It basically divides the data points in class x and rest. Consecutively a certain class is distinguished from all other classes.

The number of classifiers necessary for **one-vs-one** multiclass classification can be retrieved with the following formula (with n being the number of classes): n*(n-1)/2

In the one-vs-one approach, each classifier separates points of two different classes and comprising all one-vs-one classifiers leads to a multiclass classifier.

**Classes**:
- Setosa
- Versicolor
- Virginica

**One-vs-Rest**:
- Setosa vs [Versicolor, Virginica]
- Versicolor vs [Setosa, Virginica]
- Virginica vs [Setosa, Versicolor]

**One-vs-One**:
- Setosa vs Versicolor
- Setosa vs Virginica
- Versicolor vs Virginica
- Virginica vs Versicolor

# Support Vector Machines

**Strengths:**

- Effective on datasets with **multiple features**, like financial or medical data.
- Effective in cases where **number of features is greater than the number of data points**.
- Uses a **subset of training points** in the decision function called support vectors which makes it memory efficient.
- Different **kernel functions** can be specified for the decision function. You can use common kernels, but it's also possible to specify custom kernels.

**Weaknesses**:

- If the number of features is a lot bigger than the number of data points, avoiding over-fitting when **choosing kernel functions and regularization term is crucial**.
- SVMs don't directly provide probability estimates. Those are calculated using an expensive n-fold cross-validation.
- Works **best on small sample sets** because of its high training time.

Universidade do Minho
Departamento de Informática

ISLab
Synthetic Intelligence Lab

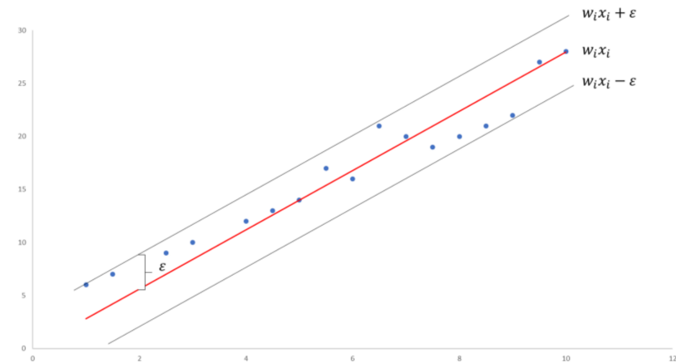
# Support Vector Regression

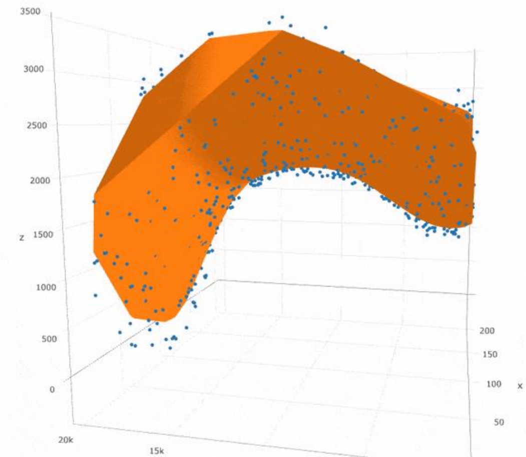
## Implementing Support Vector Regression

- Support Vector Regression is a supervised learning algorithm that is used to predict discrete values.
- Support Vector Regression uses the same principle as the SVMs.
- The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.

- The objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line.
- **Our best fit line is the hyperplane that has a maximum number of points**.

```
from sklearn.svm import SVR
```



# Support Vector Regression

## Strengths:

Although Support Vector Regression is used rarely it carries certain advantages:

- It is **robust to outliers**.
- Decision model can be **easily updated**.
- It has excellent **generalization capability**, with high prediction accuracy.
- Its implementation is easy.

## Weaknesses:

Some of the drawbacks faced by Support Vector Machines while handling regression problems are:

- They are **not suitable for large datasets**.
- In cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform.
- The Decision model **does not perform very well when the data set has more noise** i.e. target classes are overlapping.

# References


- Pisner, Derek A., and David M. Schnyer. "*Support vector machine*." Machine Learning. Academic Press, 2020.

- Cervantes, Jair, et al. "*A comprehensive survey on support vector machine classification: Applications, challenges and trends*." Neurocomputing, 2020.