BTS | Barcelona Technology School

Classical Data Analysis

Master in Big Data Solutions 2020-2021

Filipa Peleja

Víctor Pajuelo

filipa.peleja@bts.tech

victor.pajuelo@bts.tech

About me ..

Filipa Peleja

- **Industry**
  - Lead Data Scientist at Levi – Data Science Team
  - Sr Data Scientist at Vodafone – Big Data Analytics Team
  - Ph.D. intern at Yahoo! Labs Barcelona
- **Lecturer/trainer at Neueda, University NovaIMS, CodeOps and Rumos**
  - Applied Data Science
  - Recommender Systems
  - Text Analytics
  - Social Media Analysis
  - Exploratory Data Analysis
  - Statistics
- **Academia**
  - Researcher at NovaLincs
  - Bch, Masters & Ph.D. in Compute Science

LinkedIn contact: https://www.linkedin.com/in/peleja/

Today's class

Contents

1. Support Vector Machine practical intro with code examples
2. The backbone of Support Vector Machine theory
3. Support Vector Machine theory

Today's Objective

- Understand the intuition behind SVMs
- Get to know the main hyperparameters and main ways to use an SVM

Let's git things done!

Let’s see it again

Pull Session 5 notebooks

$ git clone https://github.com/vfp1/bts-cda-2020.git

BTS | Barcelona Technology School

MBDS 2020-2021

Applied Support Vector Machines (SVM)

SVM – Practical applications

Dataset applications of SVM

“All data is the same, machine learning is nothing but a geometry problem”

Lazy Programmer

BTS | Barcelona Technology School

MBDS 2020-2021

Applied Support Vector Machines (SVM)

MNIST

UUID - #S5C1

SVM – Practical application

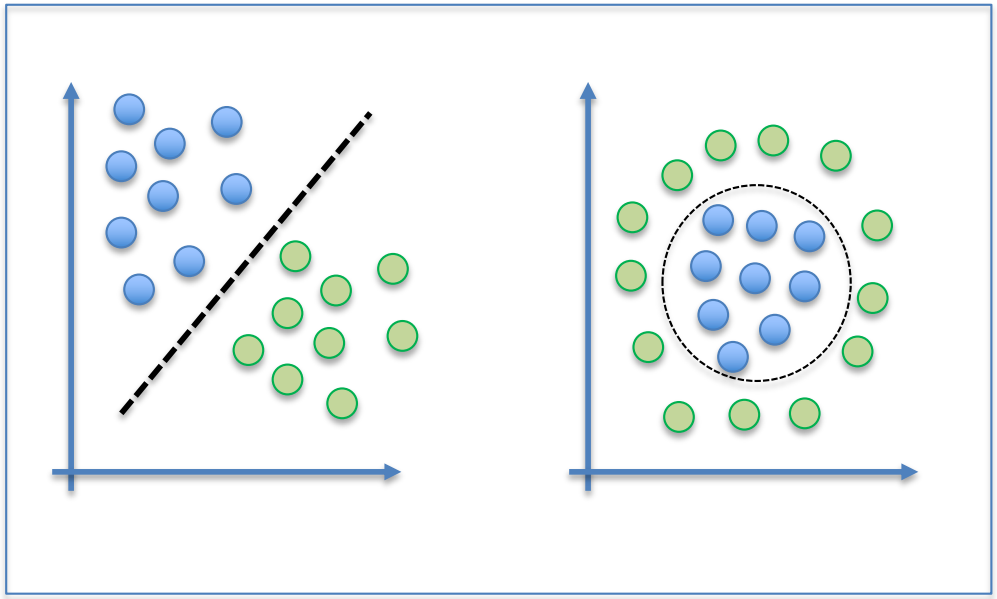
Dataset applications of SVM

- First we will start with a gentle introduction to SVM through scikit-learn
- Then we will dive into the theory and history of SVM
- Most of the code needed will be shown already here

- Some of the examples will be:
  - Medical diagnosis for cancer
  - MNIST dataset classification
  - Regression on the *Concrete Compressive Strenght* dataset

SVM – Practical concerns

- How fast is SVM to make predictions?
- Can it work on big datasets?
- Which hyperparameters do we need to take care of?
- Can it deal with nonlinear data?
  - Sneak peak, SVM's nonlinear capabilities are central to SVM theory



SVM – Image classification

MNIST

- MNIST is a classic machine learning benchmark dataset for image classification
  - When someone creates an algorithm, usually they report it over MNIST dataset, amongst others
- You can download the data from here: https://www.kaggle.com/c/digit-recognizer
- Each digit in MNIST is a grayscale image of 28 x 28
- Each pixel is an integer going from 0 to 255
- Each image is flattened into one dimensional array as we have seen in other lectures

SVM – Image classification

SVM and data scaling

- SVM is **scale sensitive**, meaning that its performance is directly affected by the normalization of the dataset
- The scale of 0 to 255 doesn't work well in some machine learning algorithms, including our SVM
  - Usually we can have scaling such as
    - Min-max scaling, i.e. going from (0..255) to (0..1)
    - Standardize with mean=0, variance=1
  - In our case we can just divide by 255 in order to get our (0..1) scale

- Also, SVM are considerably longer to train than another sklearn algorithms
  - This is due to the fact that SVMs are actually closer to a neural network
  - Prediction is also slower using SVMs

SVM – Image Classification

Notebook time

Go to the class notebook

UUID - #S5C1

BTS | Barcelona Technology School

MBDS 2020-2021

Applied Support Vector Machines (SVM)

SPAM detector

UUID - #S5C2

SVM – Spam detection

Spam dataset

- Collection of 5574 SMS messages tagged as:
  - Ham: legitimate
  - Spam: illegitimate
- You can download the data from here: https://www.kaggle.com/uciml/sms-spam-collection-dataset
  - Alternatively, wait for other download methods in the notebook section
  - Original dataset: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

SVM – Spam detection

Notebook time

Go to the class notebook

UUID - #S5C2

BTS | Barcelona Technology School

MBDS 2020-2021

Applied Support Vector Machines (SVM)

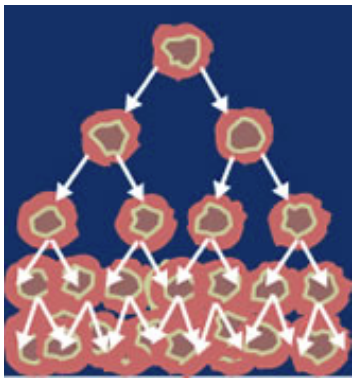
Breast cancer detector

UUID - #S5C3

SVM – Breast cancer detector

Breast cancer dataset

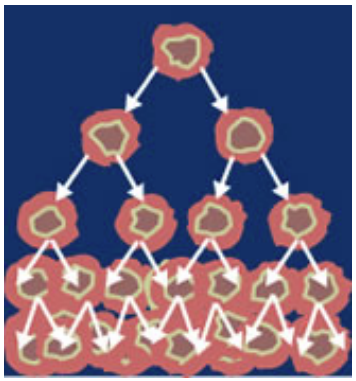
- Collection of 569 instances of breast cancer samples with about 32 attributes



```
Data Set Characteristics:
    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radius, field
        13 is Radius SE, field 23 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign
```

SVM – Breast cancer detector

Breast cancer dataset



- You can download the data from here: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
  - Alternatively, wait for other download methods in the notebook section

SVM – Breast cancer detector

Sneak peak: SVMs kernel

- The kernel is an important component of SVMs, we will have just a sneak peak here
- What the kernel function does is to return the inner product between two points in a suitable feature space
  - This defines a notion of distance and similarity
  - *"The kernel trick consists of observing that many machine learning algorithms can be written exclusively in terms of dot products between examples"* Deep Learning, p.138

- There are 4 main types of kernel functions for SVMs in sklearn:
  - Linear
  - Polynomial
  - RBF (Gaussian)
  - Sigmoid

SVM – Breast cancer detector

Notebook time

Go to the class notebook

UUID - #S5C3

BTS | Barcelona Technology School

MBDS 2020-2021

Applied Support Vector Machines (SVM)

Concrete Compressive Strength Regressor

UUID - #S5C4

SVM – Concrete regressor

Concrete regressor dataset

- Collection of 1030 instances of concrete compressive strength with about 9 variables

Cement (component 1) -- quantitative -- kg in a m3 mixture -- Input Variable
Blast Furnace Slag (component 2) -- quantitative -- kg in a m3 mixture -- Input Variable
Fly Ash (component 3) -- quantitative -- kg in a m3 mixture -- Input Variable
Water (component 4) -- quantitative -- kg in a m3 mixture -- Input Variable
Superplasticizer (component 5) -- quantitative -- kg in a m3 mixture -- Input Variable
Coarse Aggregate (component 6) -- quantitative -- kg in a m3 mixture -- Input Variable
Fine Aggregate (component 7) -- quantitative -- kg in a m3 mixture -- Input Variable
Age -- quantitative -- Day (1~365) -- Input Variable
Concrete compressive strength -- quantitative -- MPa -- Output Variable

SVM – Concrete regressor

Notebook time

Go to the class notebook

UUID - #S5C4

BTS | Barcelona Technology School

MBDS 2020-2021

Applied Support Vector Machines (SVM)

RECAP

SVM – Gentle introduction

Recap

- We look at how to “use” a SVM, but not how it works
  - We will be looking at that during the rest of the class
- Look at the basic steps of a ML script
  - Data loading / Preparation / Train / Evaluation / Prediction
- SVMs are used for a myriad of applications:
  - Image classification
  - Medical diagnosis
  - NLP (Spam detector)
  - Others… (you are free to use those datasets for your own benchmarks)
- We will see how domain knowledge it’s important, but ML is a geometry problem after all

BTS | Barcelona Technology School

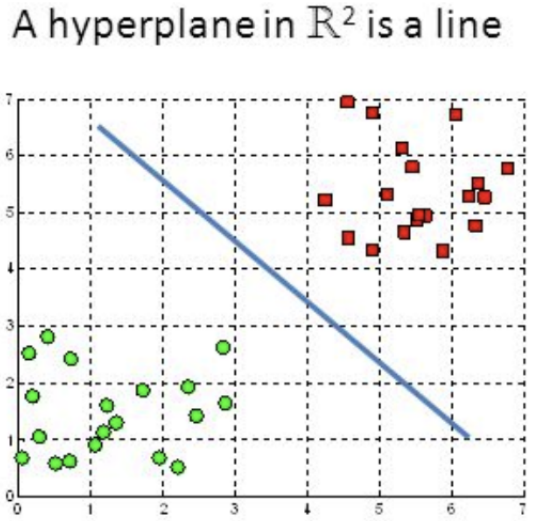
MBDS 2020-2021

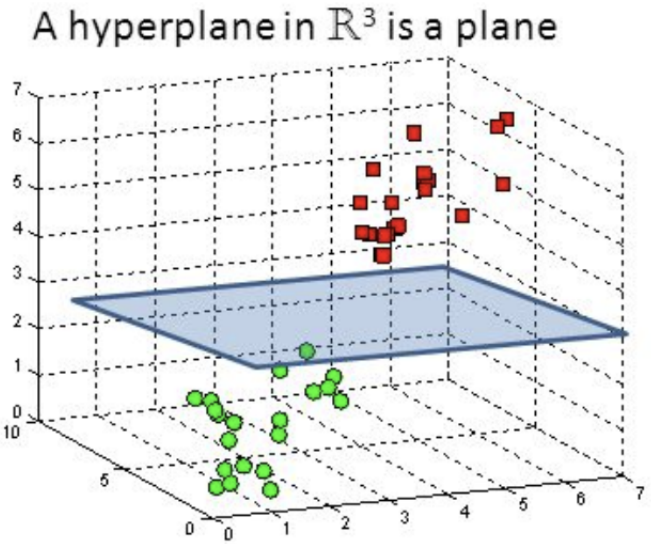
The backbone of Support Vector Machines (SVM)

SVM - Backbone

Linear Classification

- Machine Learning can be understood as geometry and linear algebra problem
- Linear classification denotes those problems that can be separated with a line
  - For **two input (2D)** dimensions we have a **line** for the decision boundary
  - For **three dimensions (3D)** the decision boundary becomes a **plane**
  - For **four or more dimensions (4+D)** the decision boundary becomes a hyperplane

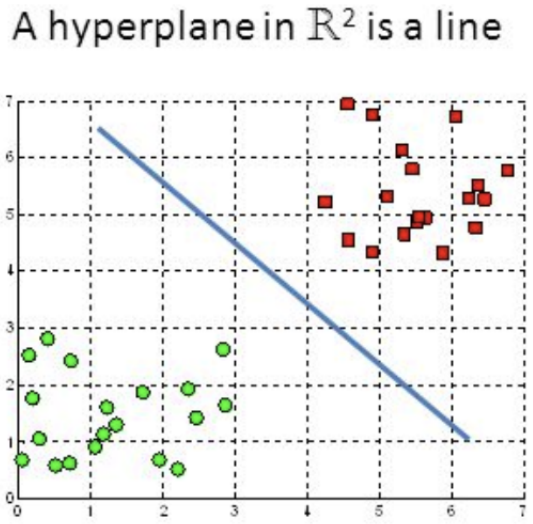


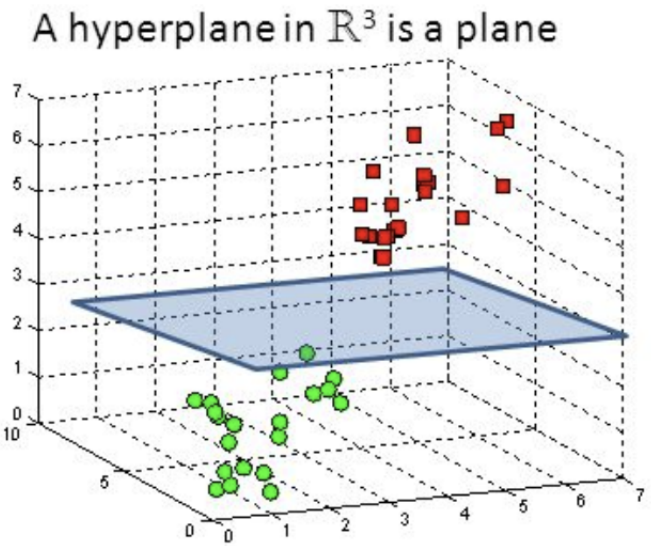


SVM - Backbone

Linear Classification - Geometry

- The geometry equations are dimension invariant, so operating on 2D will be good enough for our intuition
- Geometry is a field of mathematics and we need to deal with their equations
  - What are the equations of a line and its properties?
  - What are the equations of a plane and its properties?
  - What are the equations of a hyperplane and its properties?

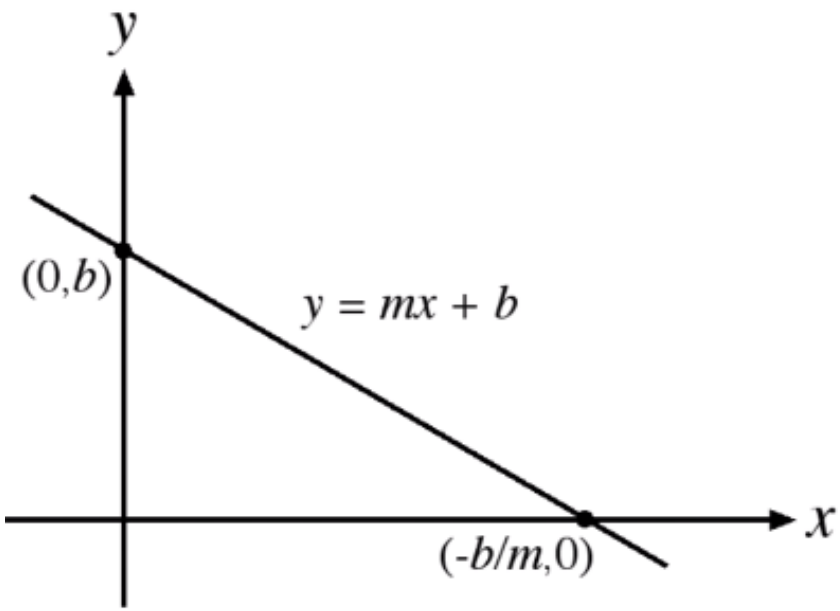




SVM - Backbone

Linear Classification - Line

- The line equation states the following:
  - “m” is the slope
  - “b” is the intercept
  - The line crosses y=0 at x=-b/m
  - The line crosses x=0 at y=b



SVM - Backbone

Linear Classification – Line notation

- Within machine learning, “y” is usually used for targets or labels. So using it for the vertical axis might lead to confusions.
  - Therefore, we are going to use “x_1” for the horizontal axis and “x_2” for the vertical axis
  - It also becomes useful to add in more dimensions such as “x_3, x_4, x_5, … , x_n”
  - The equation then becomes (you will see this notation for machine learning and SVMs across the literature):

$$x_2 = mx_1 + b$$

SVM - Backbone

Linear Classification – Line / Plane / Hyperplane equations

- The equations are dimension invariant
- “w” and “x” are vectors, “b” is a scalar

Another equation of a line can be written as:

$$ax_1 + bx_2 + c = 0$$

Which applied to the traditional equation:

$$x_2 = mx_1 + b$$

$$x_2 = \left(-\frac{a}{b}\right) \times x_1 + \left(-\frac{c}{b}\right)$$

And then turned into **dimension invariant** by

$$w^T x + b = 0$$

$$if\ Dim = 2, w = (w_1, w_2), \qquad x = (x_1, x_2)$$
$$if\ Dim = 3, w = (w_1, w_2, w_3), \qquad x = (x_1, x_2, x_3)$$

SVM - Backbone

Linear Classification – Line / Plane / Hyperplane equations

- If we expand the **dot product** we get the following:

$$w^T x + b = w_1\, x_1 + w_2\, x_2 + w_3\, x_3 + b = 0$$

- Let's put in some numbers to understand it better:

$$w = (1,1),\ b = -1$$

$$if\ x = (0,1) \rightarrow w^T x + b = 1 * 0 + 1 * 1 - 1 = 0$$

$$if\ x = (1,0) \rightarrow w^T x + b = 1 * 1 + 1 * 0 - 1 = 0$$

- Why do we get zero in both points?

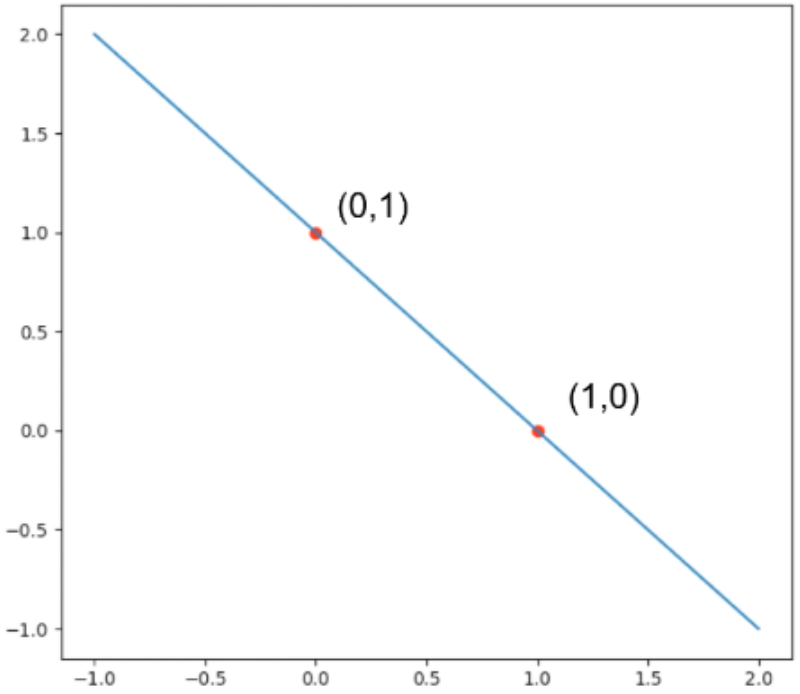
SVM - Backbone

Linear Classification – Line / Plane / Hyperplane equations

- The reason is that all points that fall on the line are according to the equation:

$$w^T x + b = w_1\, x_1 + w_2\, x_2 + w_3\, x_3 + b = 0$$

- We can potentially use this for classification as we shall see soon



SVM - Backbone

Linear Classification – Line / Plane / Hyperplane equations

- What about points that do not fall on the line?

$$x = (1,2)$$

$$w^T x + b = 1 * 1 + 1 * 2 - 1 = 2$$

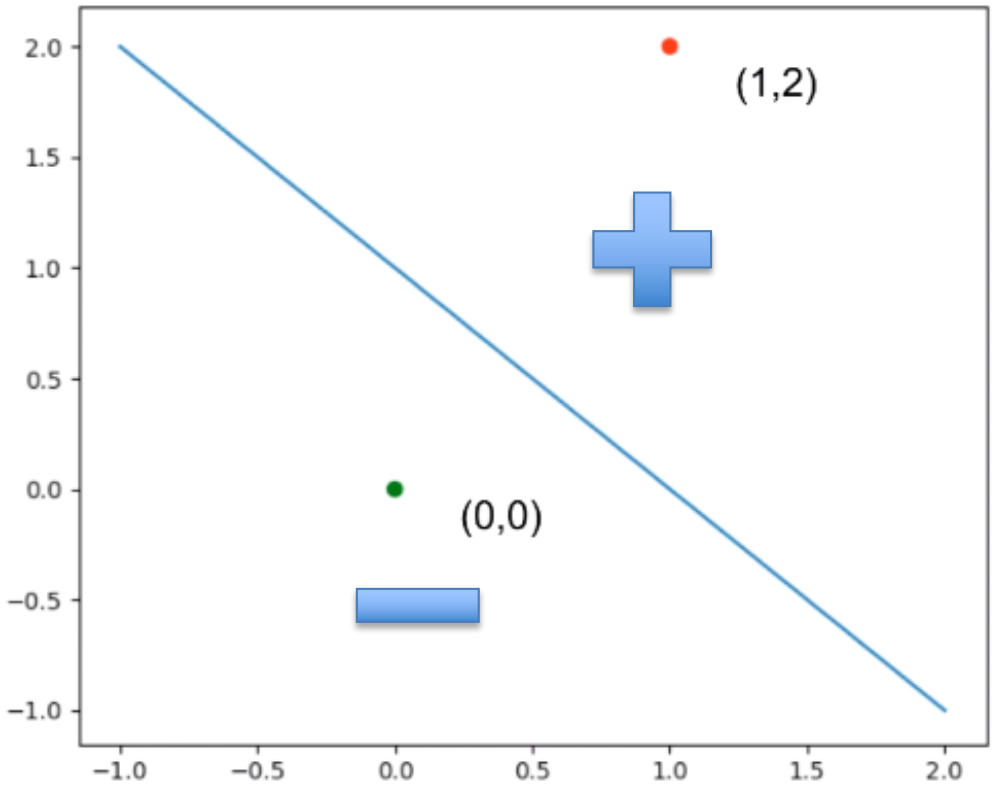
$$x = (0,0)$$

$$w^T x + b = 1 * 0 + 1 * 0 - 1 = \quad -1$$

SVM - Backbone

Linear Classification – Line / Plane / Hyperplane equations

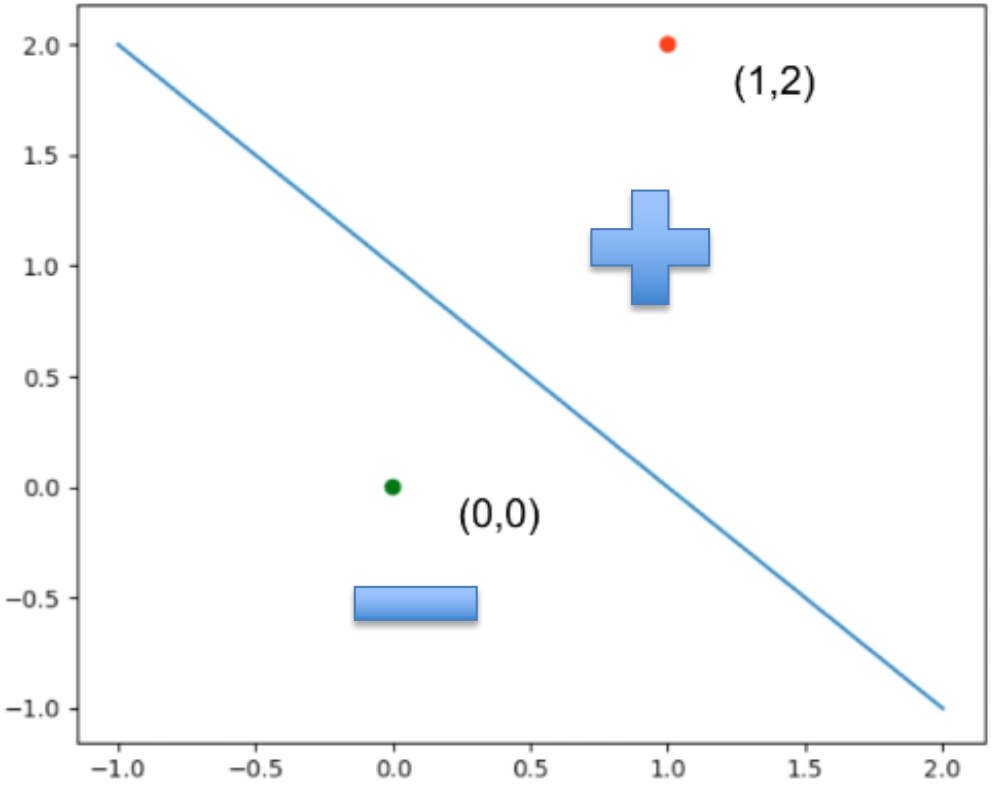
- The points on the right side yield:
  - $w^T x + b > 0$
- The points at the line yield
  - $w^T x + b = 0$
- The points at the left side of the line yield
  - $w^T x + b < 0$



SVM - Backbone

Linear Classification – Line / Plane / Hyperplane equations

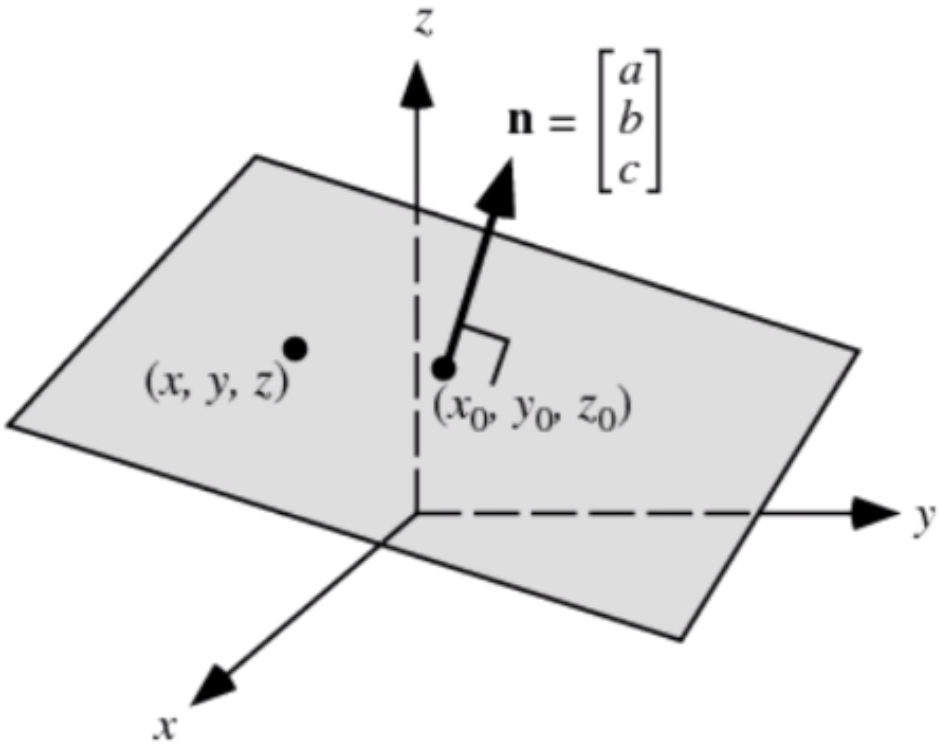
- This is the basic building blocks for machine learning
- Given some instance “x”, where are we located?
- We can plug in the $w^T + b = 0$ and check it out
- The predictions of Logistic Regressors are (0..1) but for SVMs are (-1 .. +1)

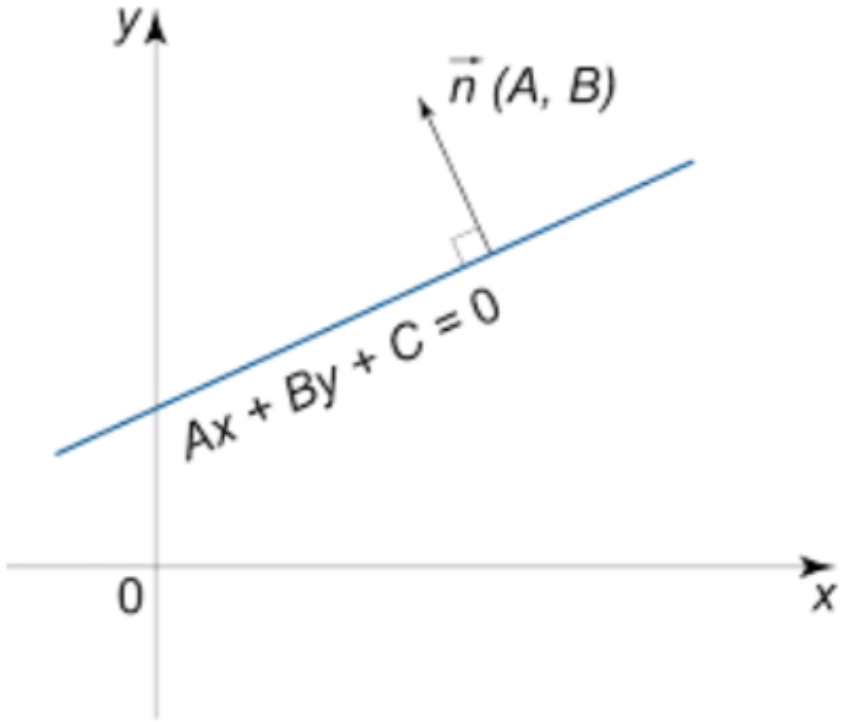


SVM - Backbone

Logistic regression

- The **vector of weights** (w) is a normal vector, meaning that it is **perpendicular** to the **line** in a 2D space and perpendicular to the **plane** in a 3D space

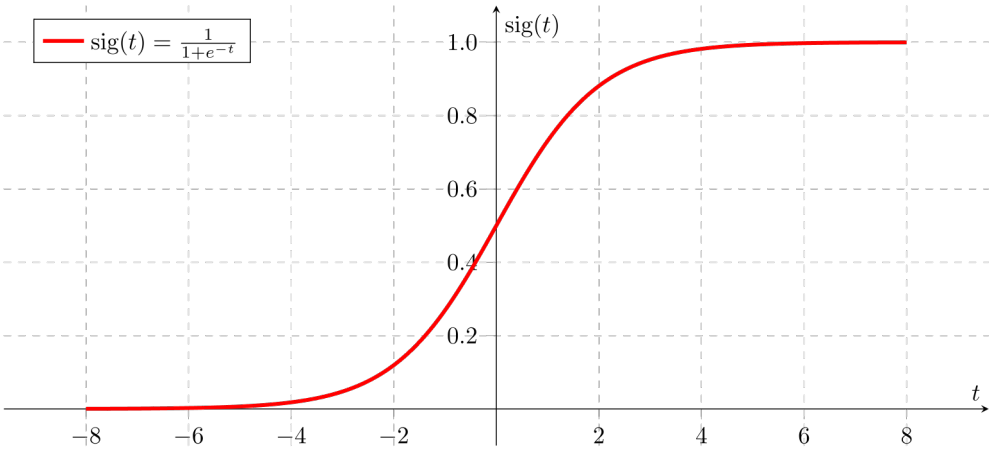




SVM - Backbone

Logistic regression

- In logistic regression we pass the decision function to a sigmoid (logistic function)
  - There is a gradient for classification (0, 1)
  - If the input is positive, we will get from 0.5 to 1
  - If the input is negative, we will get 0 to 0.5
  - If the input is zero, we will get 0.4



$$\sigma(w^T x + b),\ \sigma(z) = 1/(1 + e^{-z})$$

SVM - Backbone

Logistic regression

- Logistic regression has a probabilistic output:
  - The probability of “y=1” given “x” is the sigmoid of the decision function
  - The probability of “y=0” given “x” is “1-decision rule” since the logistic regression outputs are always between 0 and 1
  - Thus:
    - If output is greater than 0.5 predict 1
    - If output is smaller than 0.5 predict 0

$$p(y = 1 \mid x) = \sigma(w^T x + b)$$

$$p(y = 0 \mid x) = 1 - p(y = 1 \mid x)$$

SVM - Backbone

Linear vs. Logistic

	Linear Classifier	Logistic Regression
Formula	$w^T x + b$	$\sigma(w^T x + b)$
Predict 1	Output > 0	Output > 0.5
Predict 0	Output < 0	Output < 0.5
On the line	Output = 0	Output = 0.5

SVM - Backbone

Loss and Regularization

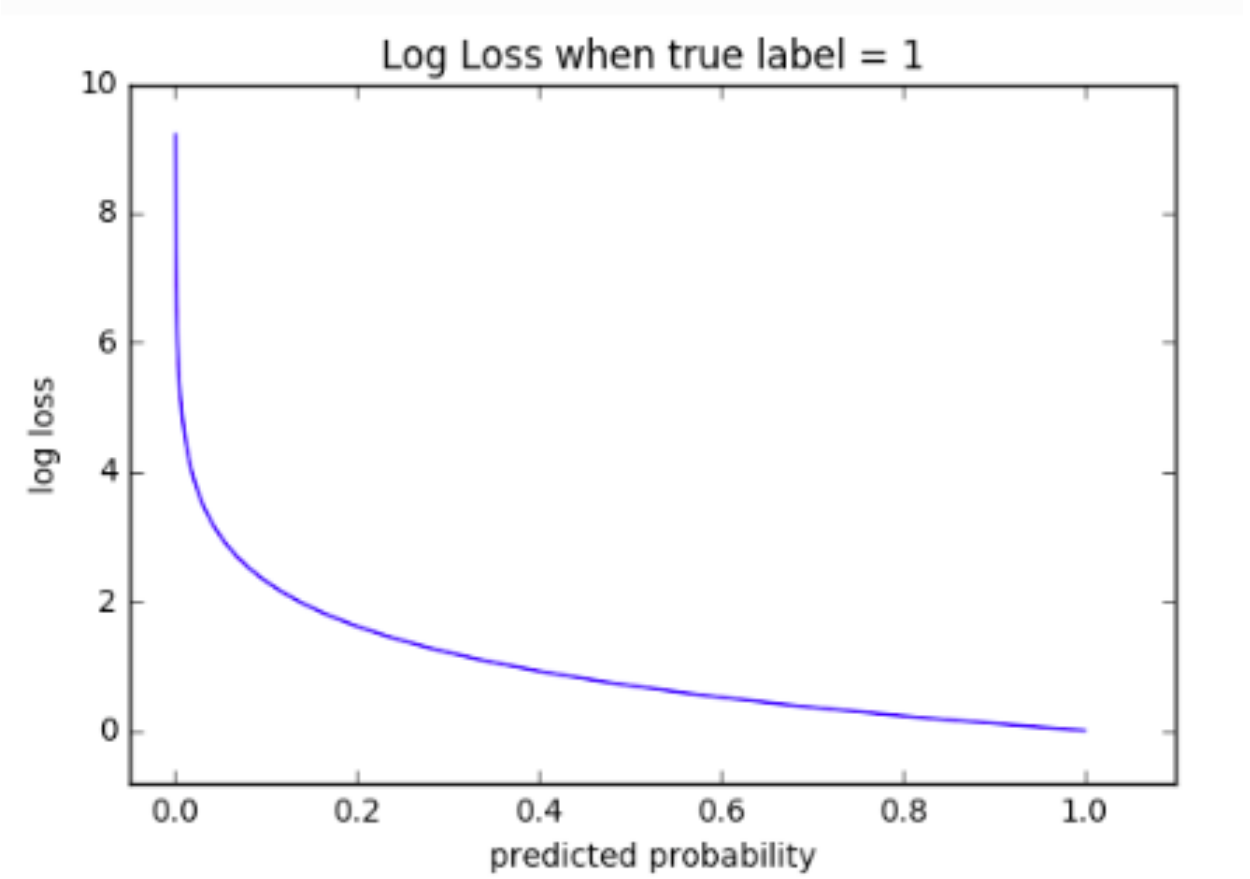
- Why do we need to look at Logistic Regression?
  - The probabilistic nature of LR allows us to train the model (find weights and biases)
  - The basic form of the formula to find “w” and “b” in Logistic Regression is the Loss function
  - Loss = Error Penalty + Regularization
    - Recognizing this basic form is crucial to be able to interpret them when studying SVM
- Logistic Regression uses Cross Entropy Loss:
  - Encourages the y (label/target) to be close to $\hat{y}$ (prediction)
    - If the $y = 1$ and $\hat{y} = 1$, loss is 0
    - If the $y = 1$ and $\hat{y} = 0$, loss is infinity
    - If I am exactly right my loss is 0, if I am exactly wrong my loss is infinity. Everything in between is a finite loss.

$$L = -\sum_{i=1}^{N} \{y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\}$$

SVM - Backbone

Loss and Regularization

- The CELoss or Log loss, measures the performance of a model with probability values between 0 and 1
- CELoss increases as predicted probability diverges from label
- A predicting probability of 0.1 when the label is 1, would be really bad and result in a high loss value

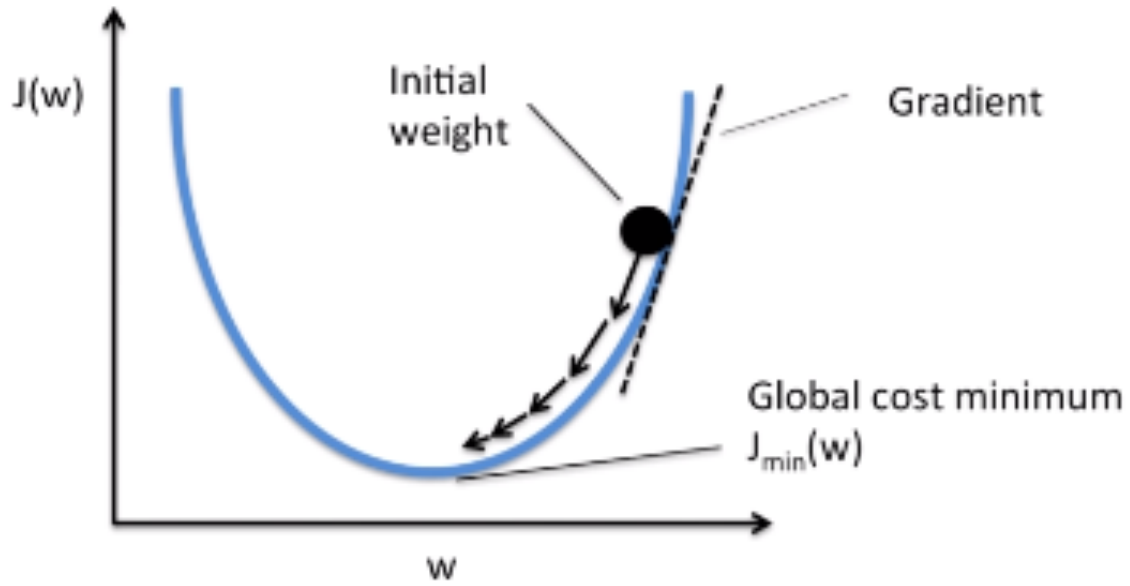


$$L = -\sum_{i=1}^{N} \{y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\}$$

SVM - Backbone

Loss and Regularization

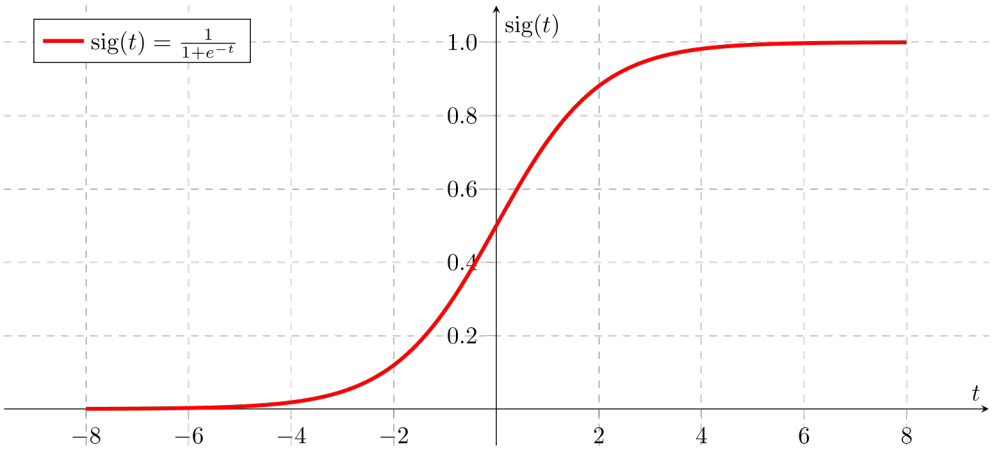
- How do we minimize the loss?
  - Using gradient descent to update “w” and “b”
    - This is called training or fitting
  - The space of CELoss is convex, meaning that we always will find the global minima



SVM - Backbone

Loss and Regularization

- But wait, **what is wrong with the Log Loss?**
  - We stated that Logistic Regression is between 0 and 1
  - And the output of sigmoid is only 1 and 0 when input is +inf/-inf (asymptotes)
  - Given that x inputs are finite, the magnitude of "w" needs to be infinite!!! Are there any problems with infinite weight magnitudes?
    - Computation
    - Model instability…
- Logistic Regression Loss encourages the parameters to go to infinity. **How do we control that?**



$$\sigma(w^T x + b),\ \sigma(z) = 1/(1 + e^{-z})$$

SVM - Backbone

Loss and Regularization

- Logistic Regression Loss encourages the parameters to go to infinity. **How do we control that?**
  - Through regularization, penalizing large weights
    - We had a regularization parameter which is a constant (lambda) times the magnitude of the weights squared
- The final loss is then the sum of two terms:
  - Error Penalty: if our prediction is wrong our error increases
  - Regularization Penalty: the larger the weight, the larger the penalty
    - Remember, larger losses are worse

$$L_{reg} = L + \lambda \|w\|^2$$

SVM – Backbone

Prediction confidence and hyperplane

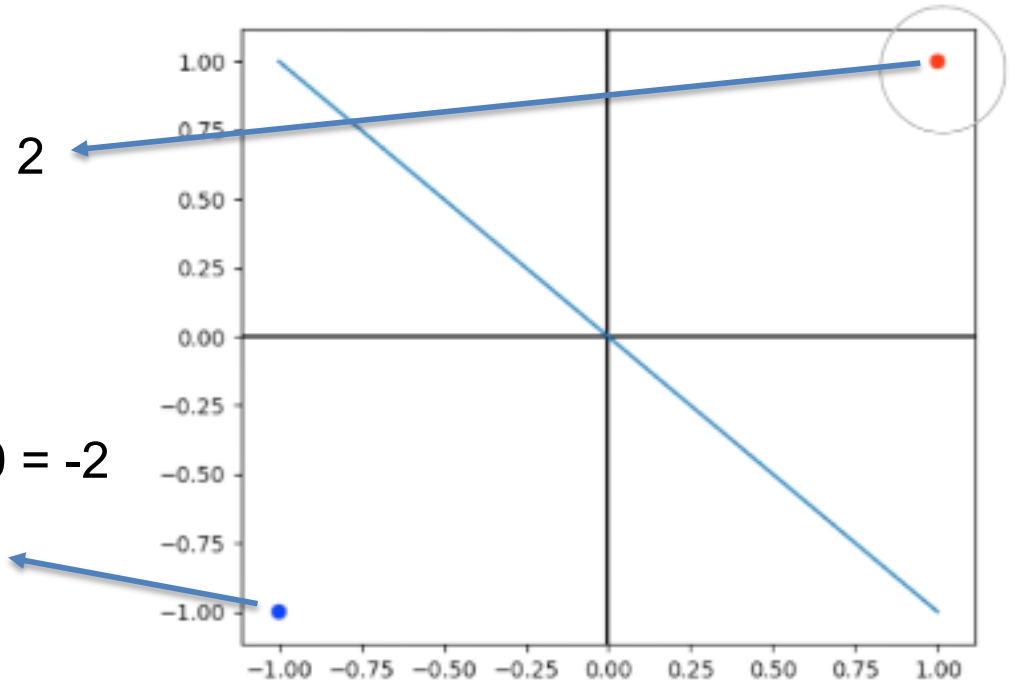
If we assume that w = (1,1) and b = 0:

For the (1,1) point

- logit = $w^T x + b = 1*1 + 1*1+0 = 2$
- $p(y = 1 | x) = \sigma(2) =$ **0.88**

For the (-1,-1) point

- logit = $w^T x + b = 1*-1 + 1*-1 + 0 = -2$
- $p(y = 1 | x) = \sigma(-2) = 0.12$
- $p(y = 0 | x) = 1- \sigma(-2) =$ **0.88**



SVM – Backbone

Prediction confidence and hyperplane

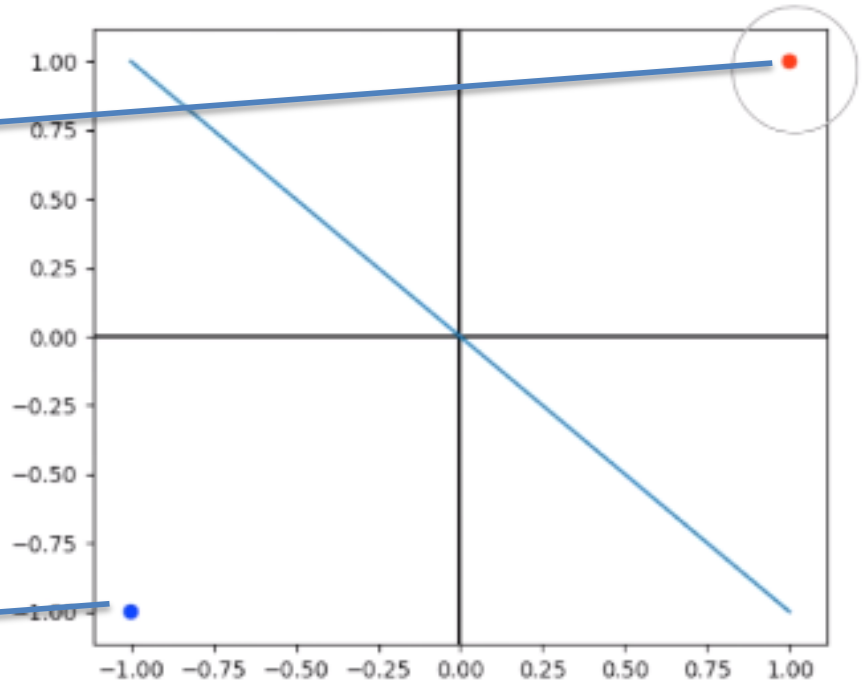
If we assume that w = (1000,1000) and b = 0:

For the (1,1) point

- logit = w^T x + b = 1000*1 + 1000*1 + 0 = 2000
- p(y = 1 | x) = σ(2000) = **0.99**

For the (-1,-1) point

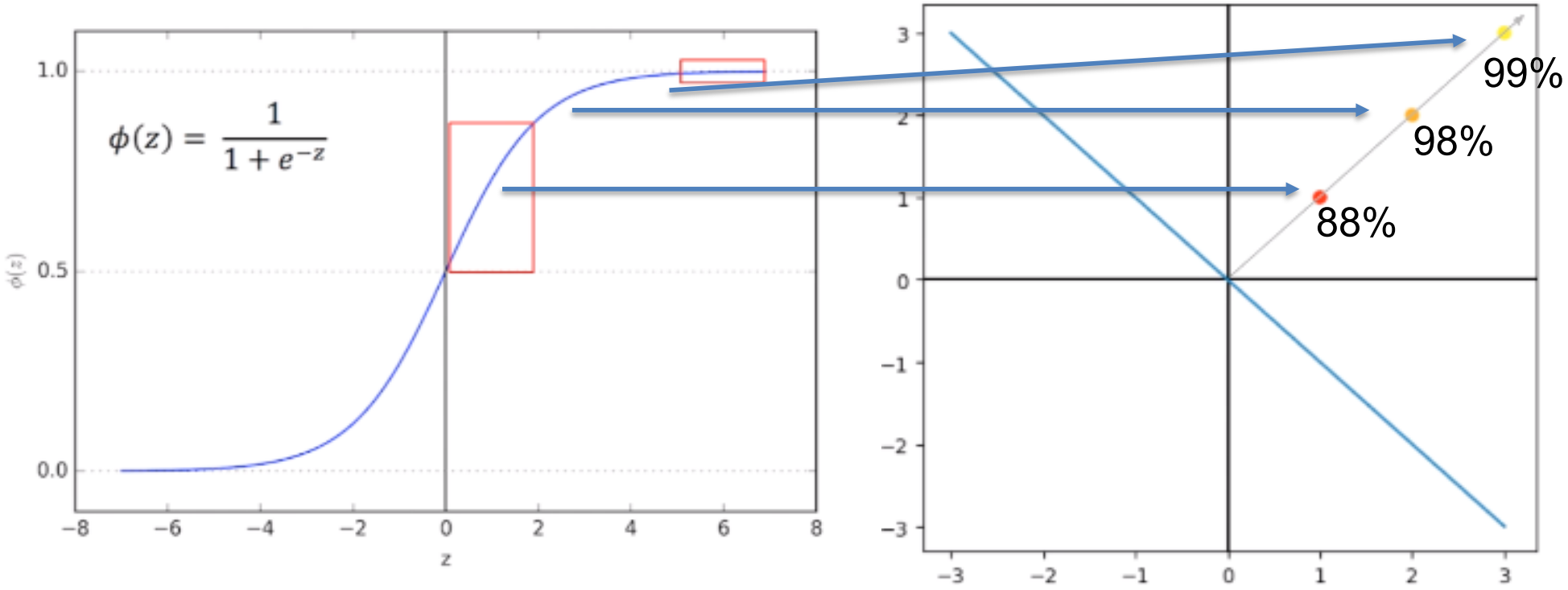
- logit = w^T x + b = 1000*-1 + 1000*-1 + 0 = -2000
- p(y = 1 | x) = σ(-2000) = 0.01
- p(y = 0 | x) = 1- σ(-2000) = **0.99**



This is way Logistic Loss is going to try to pump up weights to infinity. But we know already that we can use regularization for that

SVM – Backbone

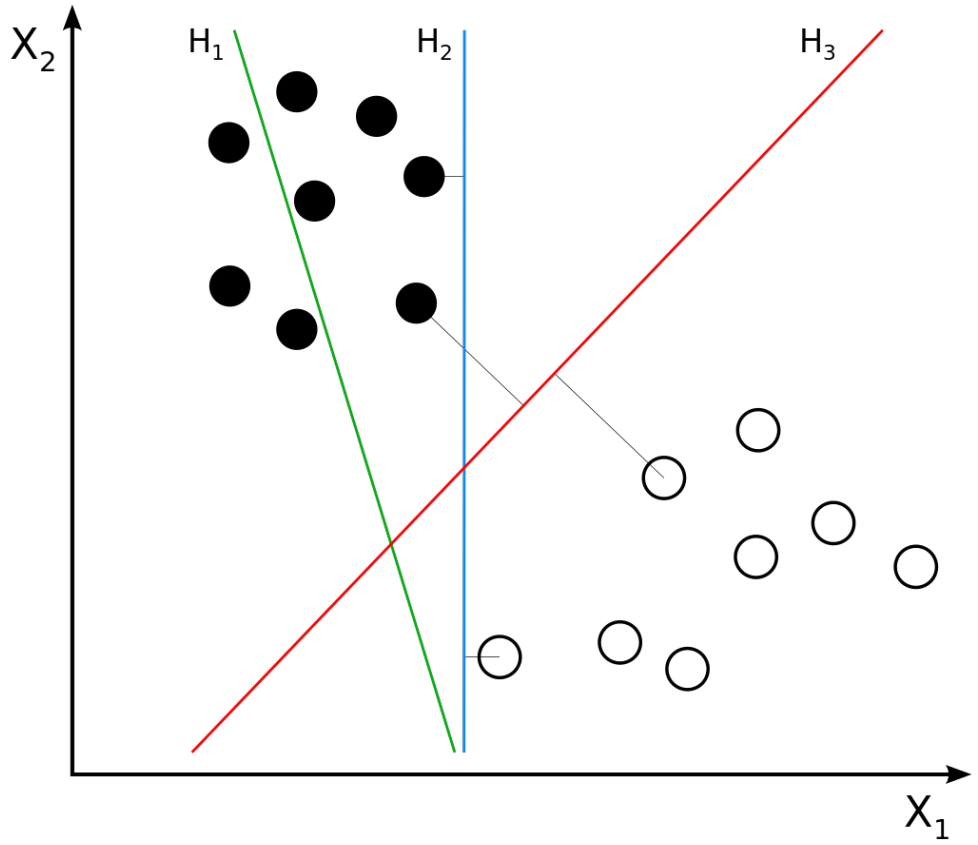
Prediction confidence and hyperplane



The further away from the decision line, the more confidence.

SVM – Backbone

Prediction confidence and hyperplane



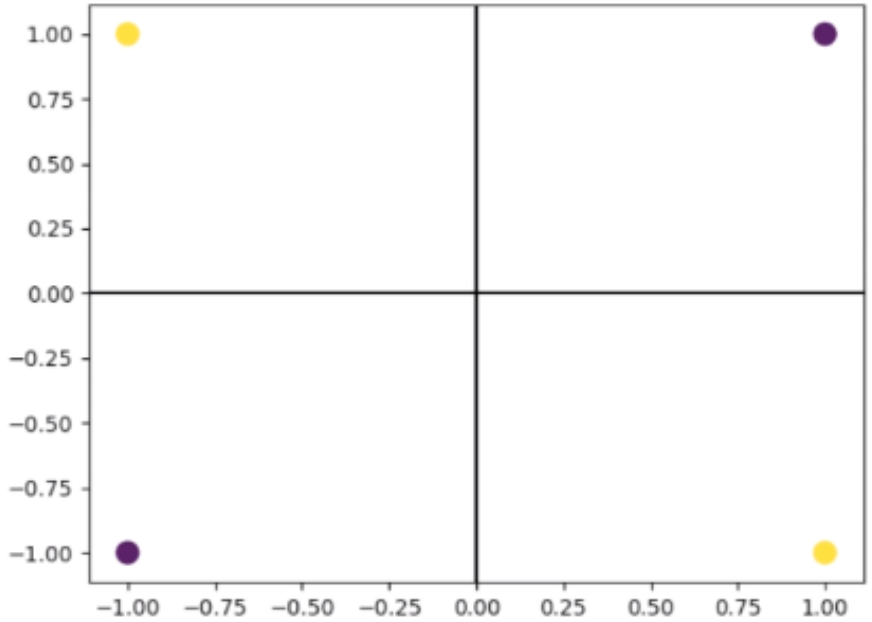
Knowing that the further away we go from the decision line the better, which line will yield the best prediction confidences?

SVM – Backbone

Polynomial feature expansions

- As we know, the **XOR problem** is a non-linear problem
- One way to solve this classification problem is by “adding dimensionality” or “features”
- Here we could do “x_1x_2”, the multiplication or interaction effect, as it measures the effect of two features interacting with each other. It is easy to see how it works here:
  - **Purple: 1*1 = (-1)*(-1) = 1**
  - **Yellow: (-1)*1 = 1*(-1) = -1**

XOR problem or “exclusive or”



Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

SVM – Backbone

Polynomial feature expansions

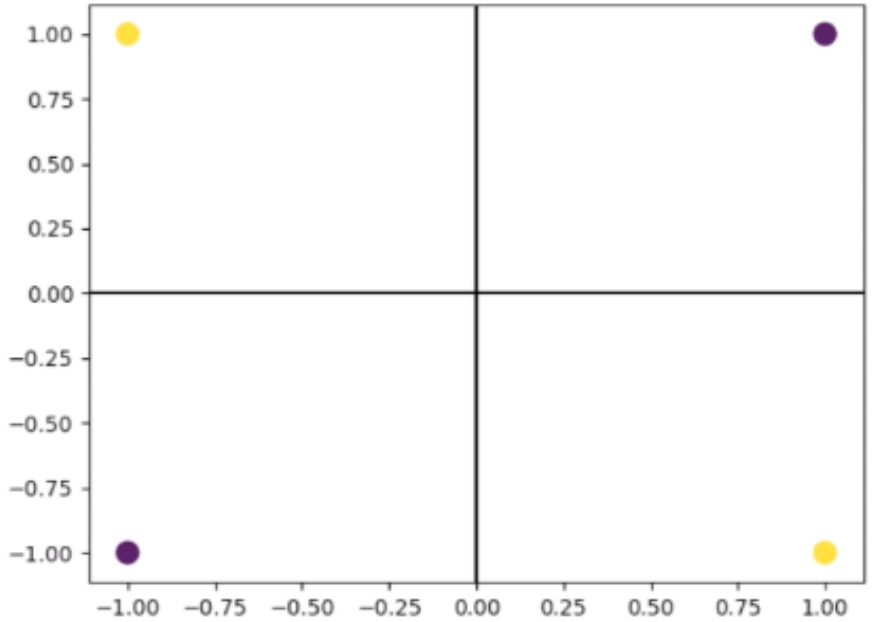
- If $x_1x_2 > 0$ we predict purple
- If $x_1x_2 < 0$ we predict yellow

- This can be turned into the linear equation we already know:
  - If $x_3 = x_1x_2$ so that $x=(x_1x_2x_3)$

$$w^T x + b = 0 \text{ , where:}$$
$$w = (0, 0, 1)$$
$$b = 0$$

- The first two weights are zero as we only consider the third weight, the feature multiplication
- So we solved a nonlinear problem with a linear equation **by adding some handcrafted features**
- **Where is the problem here?**

XOR problem or "exclusive or"



Input 1	Input 2	Output
0	0	0
0	1	1
1	1	0
1	0	1

SVM – Backbone

Polynomial feature expansions

- Usually features are **not so engineered** by domain knowledge, i.e. we do not decide on a multiplication per se
- Usually we will proceed by doing a **polynomial feature expansion** (of degree 2 in this case). So we will add automatically all those features and train the model:
  - $x_1, x_2, x_1^2, x_1^2, x_1 x_2$

- **The problem:**
  - For 3 inputs, we have 9 features… $x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3$
  - For 4 inputs, we have 14 features…

    $x_1, x_2, x_3, x_4, x_1^2, x_2^2, x_3^2, x_4^2, x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_3x_4$
  - And if we want higher degree polynomials, we need to include the both the lower and higher degree…

    Degree-3 terms: $x_1^3, x_2^3, x_1^2x_2, x_1x_2^2$

    Degree-2 terms: x_1^2, x_2^2, x_1x_2

This is called exploding dimensionality

SVM – Backbone

Polynomial feature expansions

- SVMs know how to deal with **exploding dimensionality**
- We have already gotten a sneak peak in the code **UUID - #S5C4**
- In the following chapters we will see how SVMs can even handle infinite dimensions at a considerable computational rate using the **kernel trick**

kernel : *string, optional (default='rbf')*

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

BTS | Barcelona Technology School

MBDS 2020-2021

The backbone of Support Vector Machines (SVM)

RECAP

SVM – Backbone

Recap

- **Line equation classifier**: $w^T x + b = 0$ where predictions go from -1 to 1
- **Normal vector**: the weight vector is perpendicular to the hyperplane
- **Logistic regression**
  - **Probability**: this allows us to express confidence in prediction, also allows training to find “w” and “b”
  - **Regularization**: avoiding weights to go to infinity. The loss is aimed both at the error penalty and the large weights penalty
  - **Prediction confidence**: data points close to the line have probabilities of 50%, points far are either close to 0% or 100%. We look at the idea of finding a line, plane or hyperplane, such that is as far away as possible from the training examples.
  - **Using linear to solve non-linear problems through polynomial feature expansions**: those expansions are useful in the real world, but their features can grow very fast. SVMs tackle this issue efficiently (we will see how)

BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

SVM – Theory

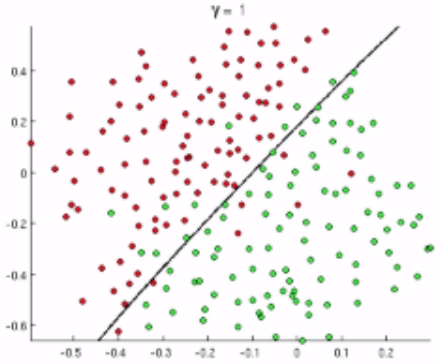
Guideline to this section

- In the previous section we looked at the relevant parts for SVM
  - **Equation for the hyperplane**, and how the weight vector happens to be perpendicular to the hyperplane
  - The **unconstrained magnitude of the weight (|w|)** and how do we use regularization to constrain it
  - The ways of having more **prediction confidence**:
    - Apply bigger weights (not efficient)
    - Having the points as far away as possible from the line/hyperplane

SVM – Theory

Guideline to this section

- We will look at how SVMs deal with those issues
- We will explore the **Linear SVM** which is good to develop an intuition
- Understand SVM as a ***maximum margin classifier***
- Once we understand the Linear SVM, we will look at how some points might be misclassified
  - We will solve it by ***soft margin svm***, which acts as a regularizer and gives us an objective that looks a lot like a regularized Logistic Regression
- To end with, we will discuss the **loss function for SVM** and how it relates to Cross Entropy Loss



The theory of Support Vector Machines (SVM)

What is actually an SVM?

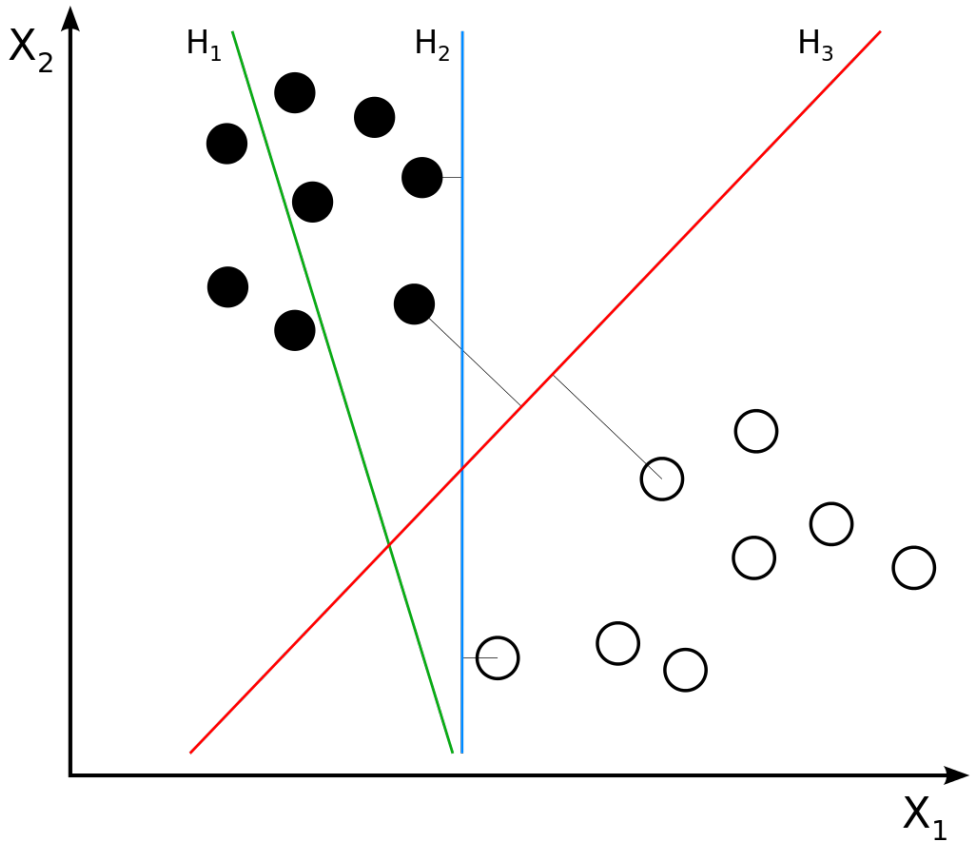
SVM – Theory

What is a SVM?

- Powerful and **versatile Machine Learning model**, capable of performing:
  - Linear / Non-linear classification
  - Regression
  - Outlier detection
- SVMs are particularly **well suited** for classification of **complex small/medium sized** datasets
- SVMs are relatively “new”:
  - The **main idea** appeared in 1963.
  - In 1992, the non-linearity of SVMs was introduced through applying the **kernel trick** to maximum-margin hyperplanes.
  - The **soft margin classification** which defines modern SVMs was published in 1995.
- SVMs at the time **rivaled with Neural Nets** and in some tasks are still a powerful model to use
- An amazing interview with Vapnik, the SVM co-creator> https://www.youtube.com/watch?v=STFcvzoxVw4

SVM – Theory

What is a SVM?



- **H1** -- does not separate the classes
- **H2** -- separates the classes but by a very small margin and it will not generalize well to new data
- **H3** -- separates the classes with a maximal margin possible, i.e. the longest possible perpendicular distance between the closest samples

*Our **OBJECTIVE** is to find the line that separates the two clouds by the maximum distance possible, i.e. to **maximize the margin***

BTS | Barcelona Technology School

MBDS 2020-2021

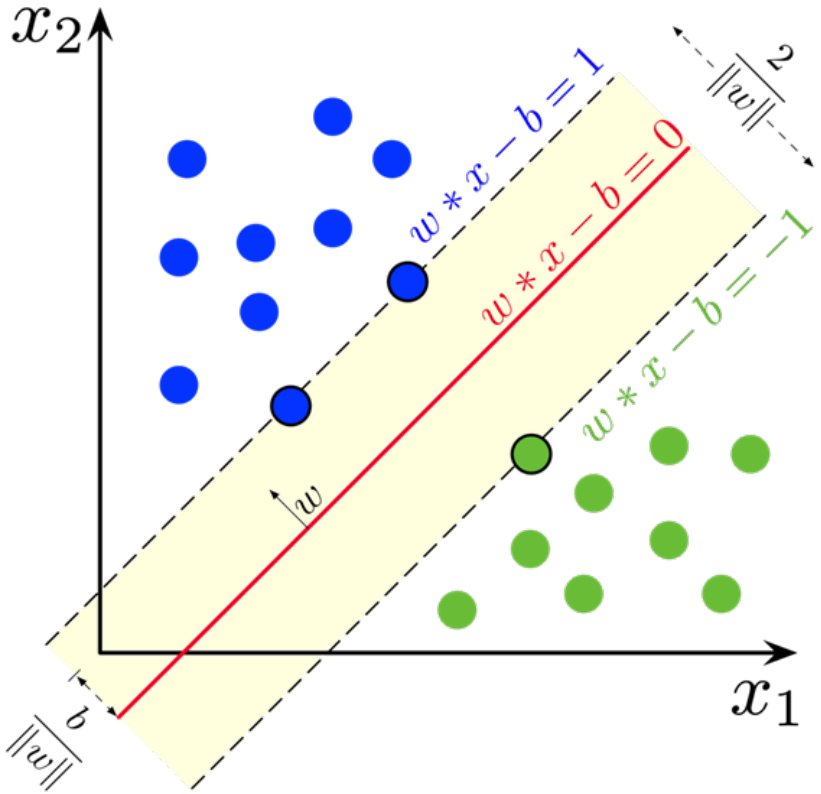
The theory of Support Vector Machines (SVM)

The Linear SVM

Hard margin classification

SVM – Theory

What is a SVM? A peak into the Linear SVM



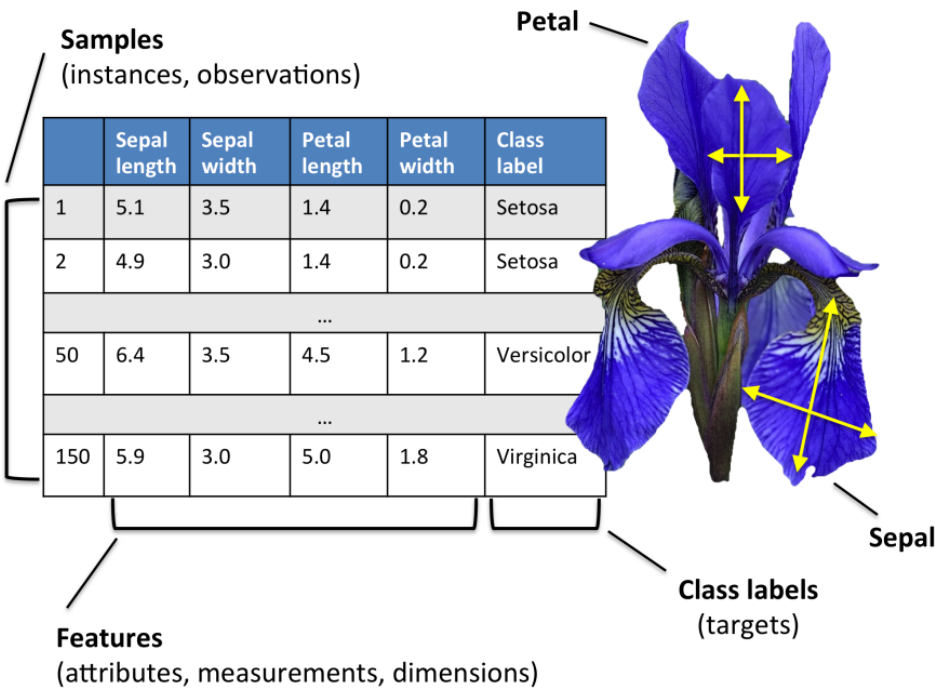
- The red line represents the hyperplane line of contact with the feature space
- The green and blue “lines” are the support vectors, the longest distance possible between the closest variables
- Dividing the bias with the norm of the vector to the hyperplane determines the offset of the hyperplane from the red line along the normal vector

- $\omega * x - b$ is the decision function
- x_1 and x_2 are the variables
- $\vec{w}$ is the normal vector to the decision hyperplane
- $\|w\|$ is the norm of a vector is the magnitude of the vector quantity or the way to measure the size of a vector

SVM – Theory

LINEAR SVM – Iris Dataset

- The simpler version of an SVM is the Linear SVM
- We will explore the Linear SVM through the Iris Dataset, which comes packaged in sklearn:
  - The dataset consists of 3 different types of irises (Setosa, Versicolour and Virginica) petal and sepal length, stored in a 150x4 numpy.ndarray
  - The rows are the samples
  - The columns are:
    - Sepal Length
    - Sepal Width
    - Petal Length
    - Petal Width

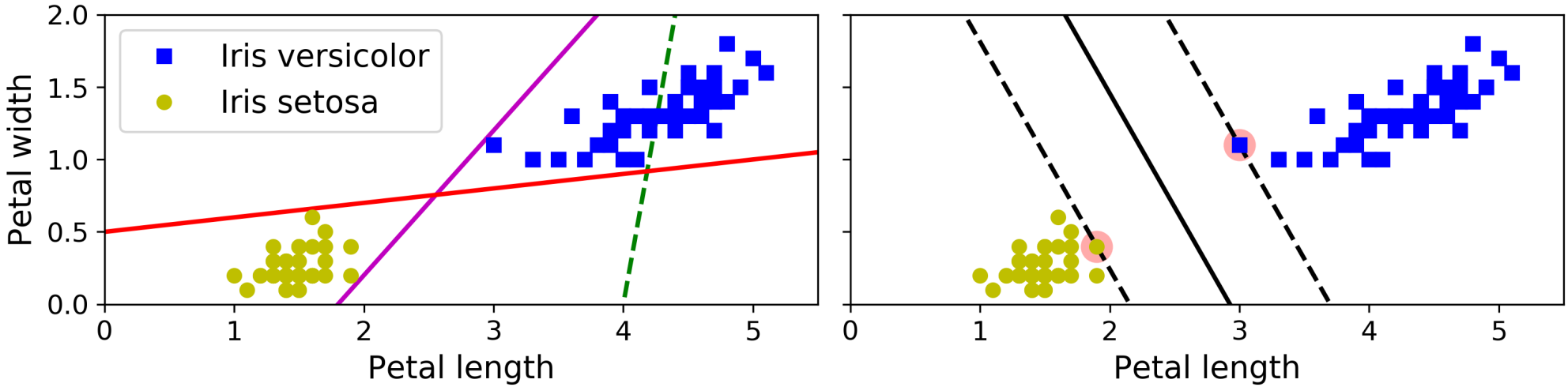


	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

SVM – Theory

Linear SVM

- The two classes **Iris versicolor** and **Iris setosa** can be clearly separated with a straight line – they are **linearly separable**
- **Left plot** shows decision boundaries for three different classifiers:
  - The green does not even separate the classes
  - The other two separate the classes but the decision boundaries come so close to the instances that it might not perform well over new samples

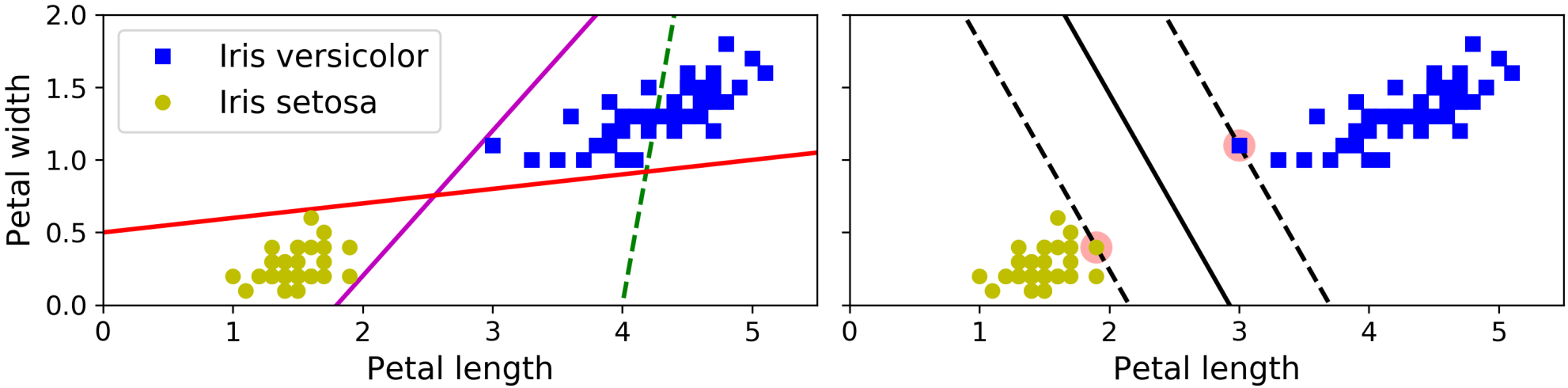


*Code to reproduce the graphs can be seen at **UUID - #S5C5***

SVM – Theory

Linear SVM

- The **right plot** shows the SVM classifier, represented by the solid line which is the *decision boundary*
  - This line does not only separate the two classes but stays as far away from the closest training instances as possible
- SVMs can be thought of as a classifier fitting the widest possible street (parallel dashed lines between the classes)
  - This is called ***large margin classification***
  - It maximizes the "size of the street"

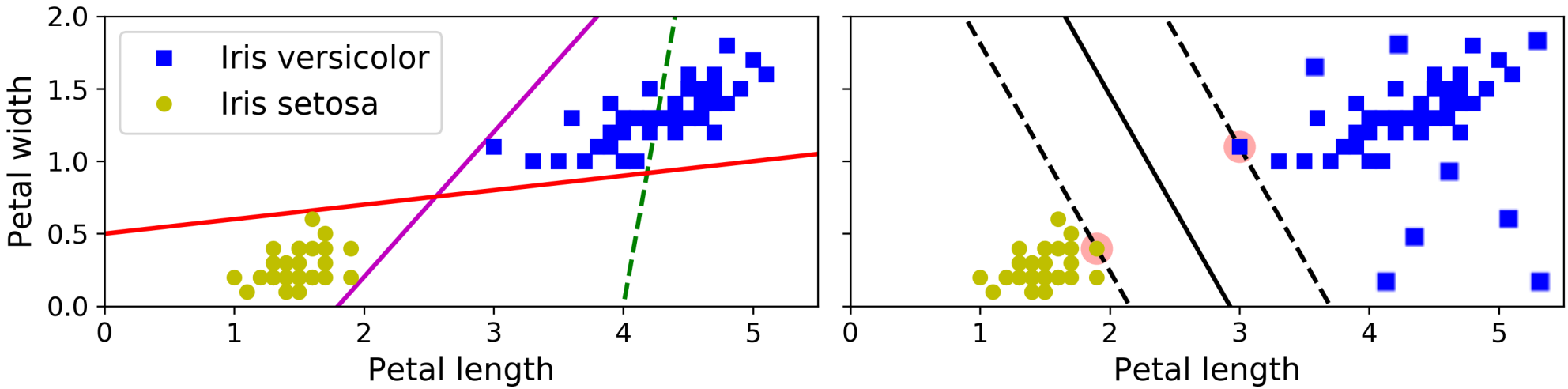


*Code to reproduce the graphs can be seen at **UUID - #S5C5***

SVM – Theory

Linear SVM - Adding instances

- Notice that adding more instances/samples “off the street” **does not affect at all** the decision boundary
- The boundary is fully supported by the samples located on the edge of the street:
  - Those instance are called **support vectors** (circled in red)

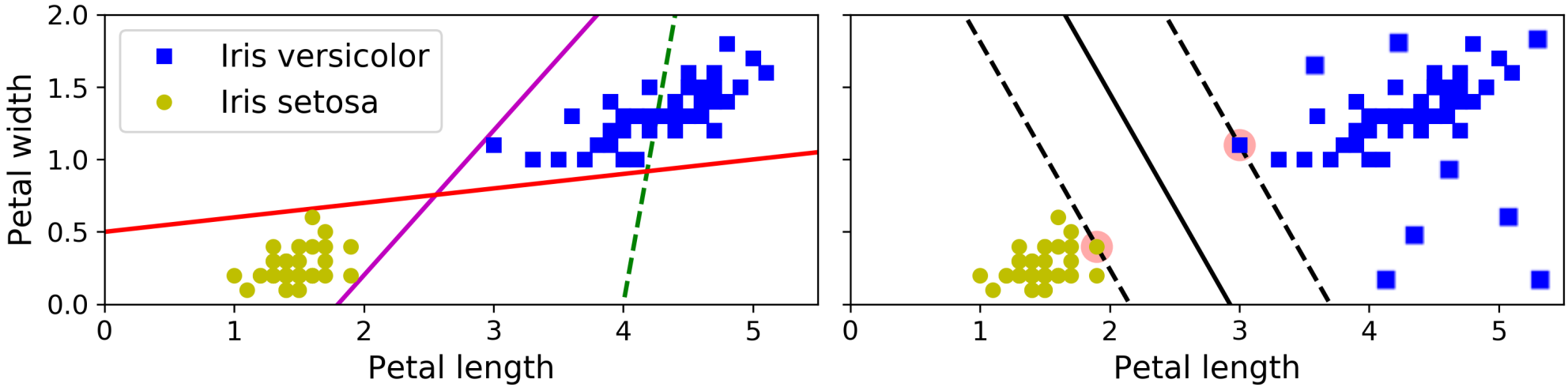


*Code to reproduce the graphs can be seen at **UUID - #S5C5***

SVM – Theory

Linear SVM - Adding instances

- If we want to increase the training size of our dataset, the separability in SVMs will be defined by the “edge cases”. It is important to bear in mind before going crazy on adding more and more samples.

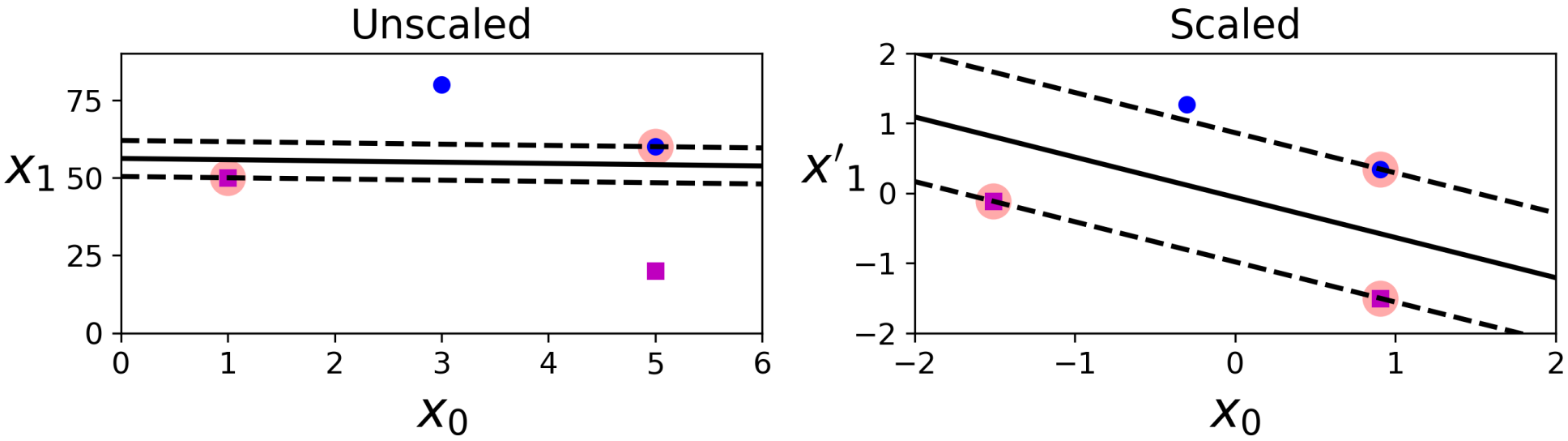


*Code to reproduce the graphs can be seen at **UUID - #S5C5***

SVM – Theory

Linear SVM – Sensitivity to feature scales

- SVMs are **extremely sensitive** to feature scales
- The **left plot is unscaled** so the vertical axis is much larger than the horizontal one
  - The widest possible street is almost horizontal
- The **right plot is scaled** using sklearn StandardScaler
  - The decision boundary is much more clear



*Code to reproduce the graphs can be seen at **UUID - #S5C6***

SVM – Theory

Linear SVM – Sensitivity to feature scales

- Take care! SVMs are **extremely sensitive** to feature scales. Always use scaling on your data before using SVMs

BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

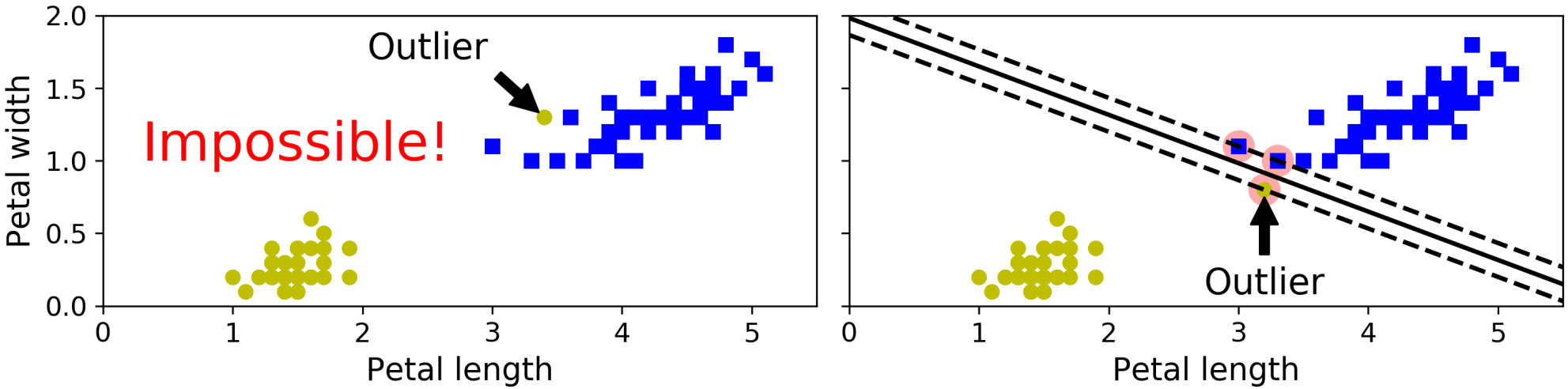
The Linear SVM

Soft margin classification

SVM – Theory

Soft Margin Classification

- **Hard margin classification**: all instances must be off the street and on the right side for each class
- **Two issues** with this:
  - **1)** This only works if the data is **linearly separable**
  - **2)** It is **sensitive to outliers**
- The figure below shows the Iris dataset with one outlier:
  - In the left case, it is **impossible to find a hard margin** (dataset is not completely linearly separable)
  - In the right case, the **decision boundary is too narrow** and it will not generalize well

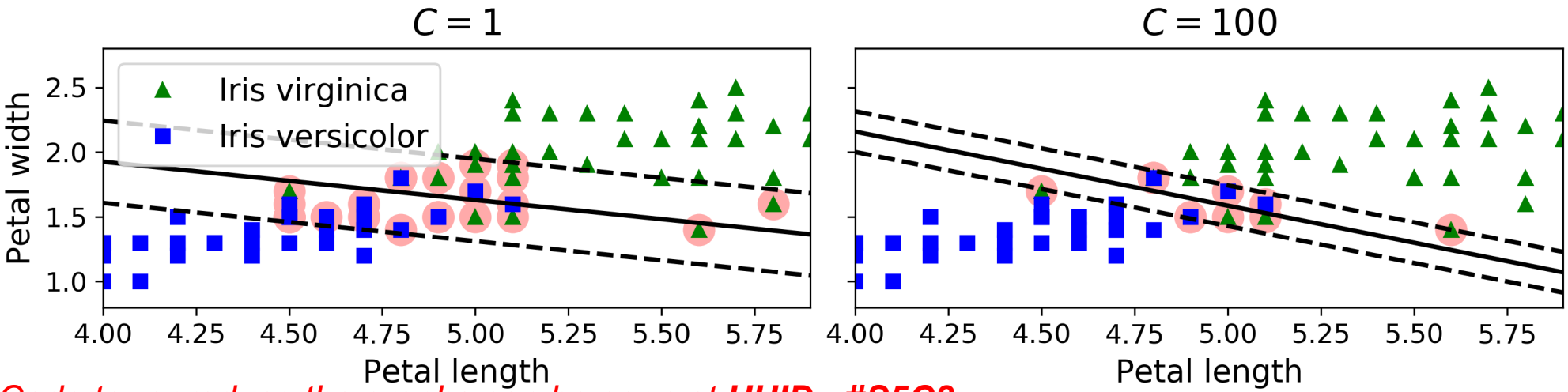


*Code to reproduce the graphs can be seen at **UUID - #S5C7***

SVM – Theory

Soft Margin Classification – Hyperparameter C

- We need to use a **more flexible model**. The objective is to find a good balance between:
  - Keeping the street as large as possible
  - Limiting **margin violations:**
    - Samples in the middle of the street or the other side
    - A margin should be considered a wall
- More flexible and regularized models are called **soft margin classification**

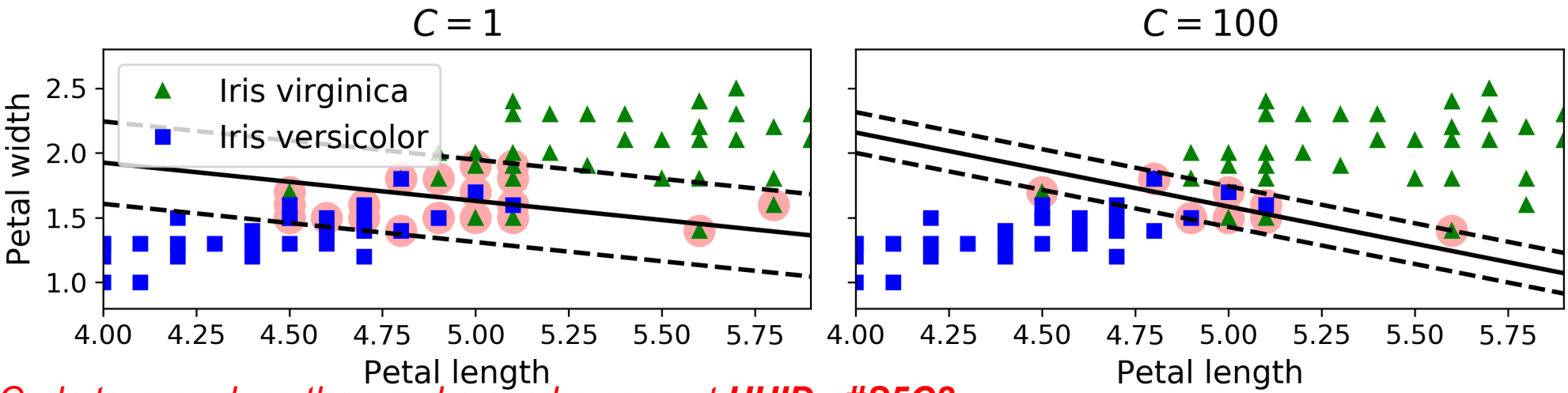


*Code to reproduce the graphs can be seen at **UUID - #S5C8***

SVM – Theory

Soft Margin Classification – Hyperparameter C

- We can regularize the margins of SVM by **tweaking the hyperparameter C**:
  - Low values of C yield more flexible models (left one)
  - High values of C yield more restrictive models (right one)
- Both have margin violations, usually it is better to have as few as possible
  - The one on the **left** has **more margin violations** but will **generalize better** in front of new data
  - The one on the **right** has **less margin violations** but will **generalize worst** when adding new data

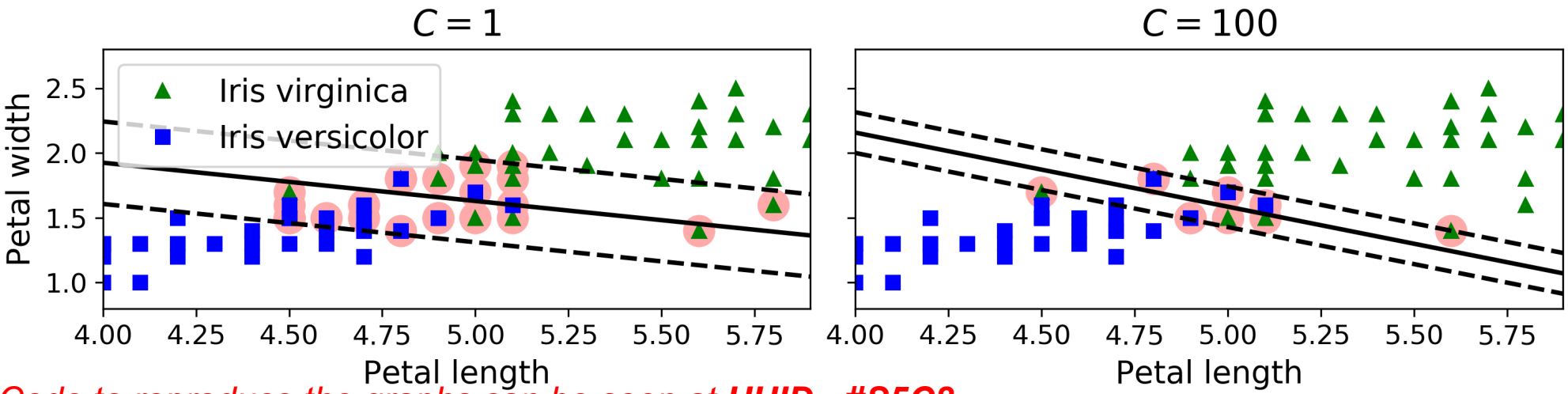


*Code to reproduce the graphs can be seen at **UUID - #S5C8***

SVM – Theory

Soft Margin Classification – Hyperparameter C

- If our SVM is overfitting, we can try to regularize it by reducing the C hyperparameter

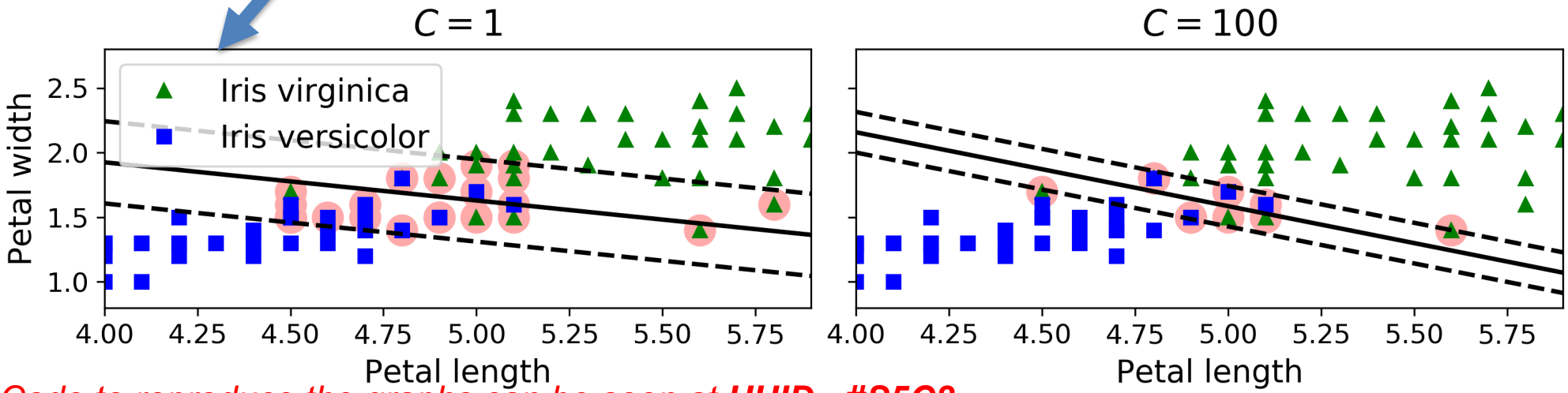


*Code to reproduce the graphs can be seen at **UUID - #S5C8***

SVM – Theory

Soft Margin Classification – Hyperparameter C

```python
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC

iris = datasets.load_iris()
X = iris["data"][:, (2, 3)]  # petal length, petal width
y = (iris["target"] == 2).astype(np.float64)  # Iris virginica

svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("linear_svc", LinearSVC(C=1, loss="hinge", random_state=42)),
    ])

svm_clf.fit(X, y)
```



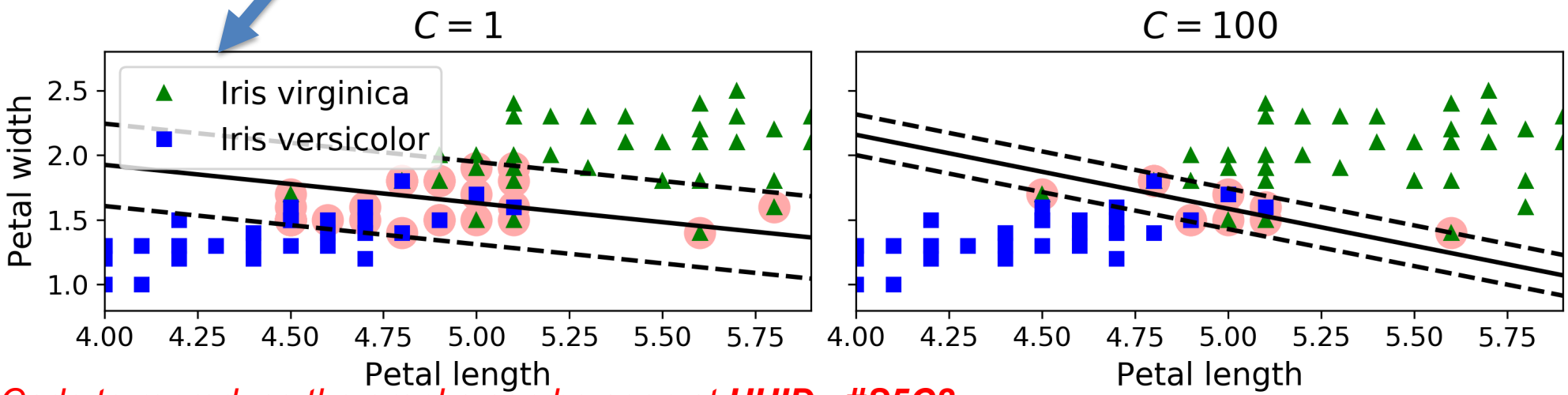
*Code to reproduce the graphs can be seen at **UUID - #S5C8***

SVM – Theory

Soft Margin Classification – Hyperparameter C

```
[ ]  1 svm_clf.predict([[5.5, 1.7]])

     array([1.])
```

- Beware! Unlike Logistic Regression classifiers, SVM classifiers do not output probabilities for each class.



*Code to reproduce the graphs can be seen at **UUID - #S5C8***

SVM – Theory

Soft Margin Classification – More on pipeline hyperparameters

```
LinearSVC(C=1, class_weight=None, dual=True,
fit_intercept=True, intercept_scaling=1, loss='hinge',
max_iter=1000, multi_class='ovr', penalty='l2',
random_state=42, tol=0.0001, verbose=0)
```

- LinearSVC regularizes the bias term, so the training set needs to be centered by substracting the mean before fitting. Make sure to use **StandardScaler** first
- The loss should be **hinge** ! (more on that soon)
- To increase performance set **dual=False**, unless *there are more features than training instances* (this is due to duality, more on that later)

*Code at **UUID - #S5C8***

BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

The Loss of SVMs

SVM – Theory

Logistic Regression Loss

- We need to look at the loss of Logistic Regression (Cross-Entropy loss)
  - It is important to note that when **y=1**
    - Only the first term matters as 1-1 is 0 and the second term is zeroed
  - It is important to note that when **y=0**
    - Only the second term matters, as the first one is multiplied by zero

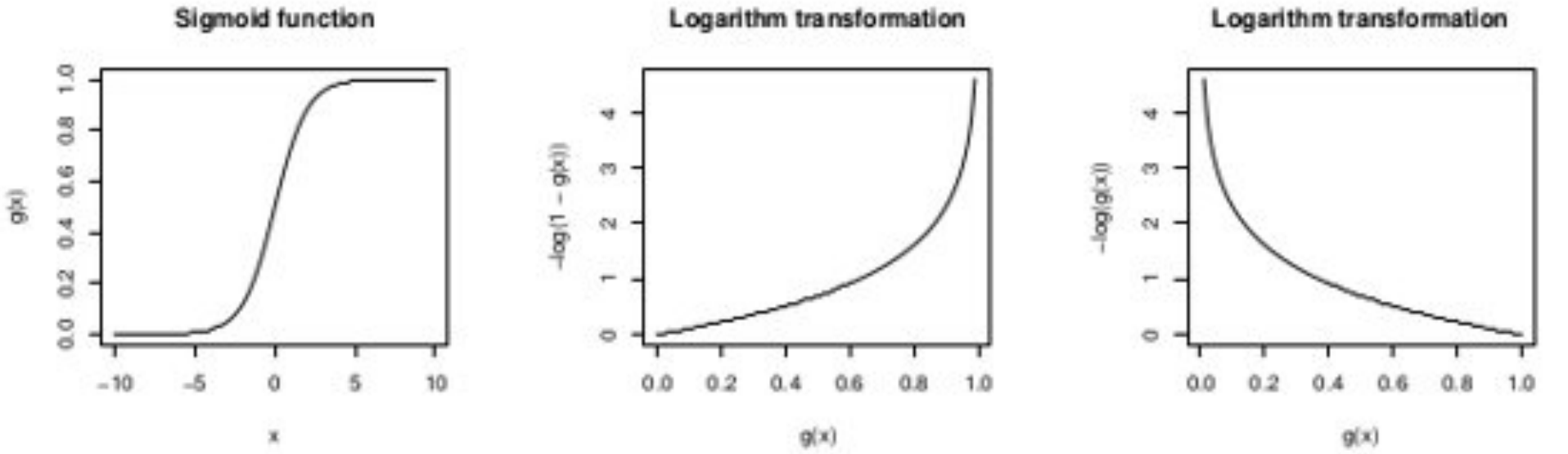
$$L = -\sum_{i=1}^{N} \{y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)\}$$

$$L = -[y \log \sigma(w^T x) + (1 - y) \log(1 - \sigma(w^T x))]$$

(For a single instance)

SVM – Theory

Logistic Regression Loss



(a) *Sigmoid function.* **(b)** *Cost for* $y = 0$. **(c)** *Cost for* $y = 1$.

Figure B.1: *Logarithmic transformation of the sigmoid function.*

$$L = -\left[y \log \sigma(w^T x) + (1 - y) \log(1 - \sigma(w^T x))\right]$$

SVM – Theory

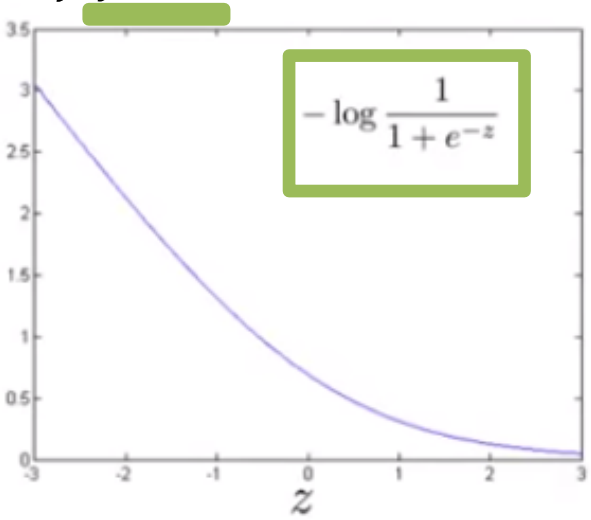
Logistic regression loss

$$L = -\left[y \log \sigma(w^T x) + (1-y) \log(1 - \sigma(w^T x))\right]$$

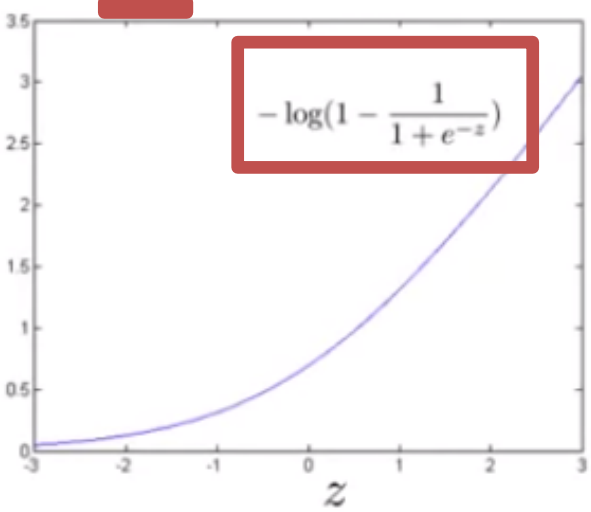
$$L = -y \log \frac{1}{1 + e^{-w^T x}} - (1-y) \log\left(1 - \frac{1}{1 + e^{-w^T x}}\right)$$

$$Given\ z = w^T x$$

$If\ y = 1, we\ want\ z \gg 0$



$If\ y = 0, we\ want\ z \ll 0$



SVM – Theory

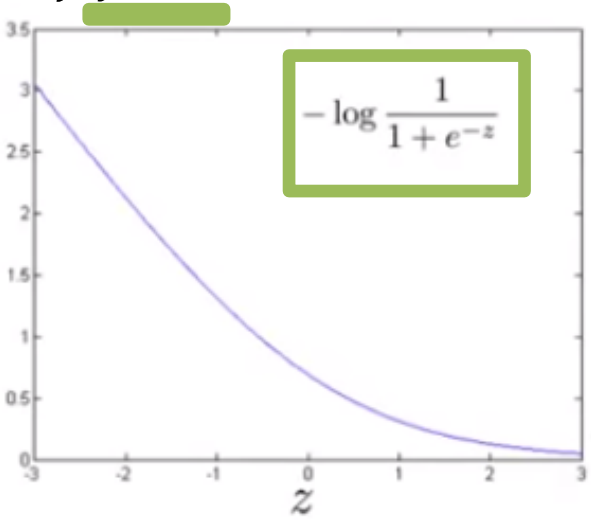
Derivation of hinge loss

$$L = -\left[y \log \sigma(w^T x) + (1-y) \log(1 - \sigma(w^T x))\right]$$

$$L = -y \log \frac{1}{1 + e^{-w^T x}} - (1-y) \log\left(1 - \frac{1}{1 + e^{-w^T x}}\right)$$

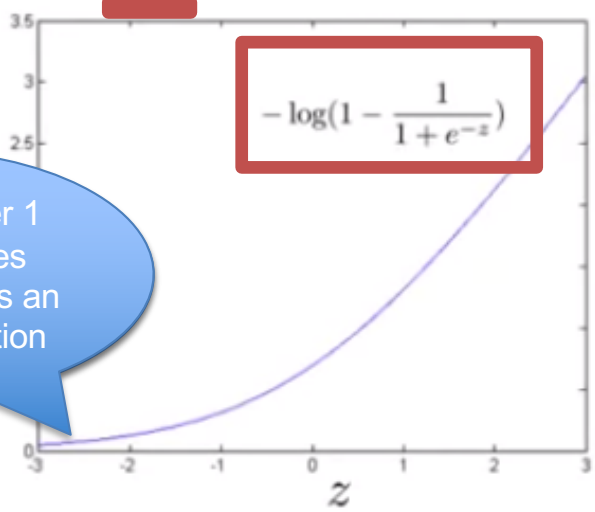
$$Given\ z = w^T x$$

$If\ y = 1, we\ want\ z \gg 0$



When y is either 1 or 0, z becomes very large… as is an asymptotic relation

$If\ y = 0, we\ want\ z \ll 0$



SVM – Theory

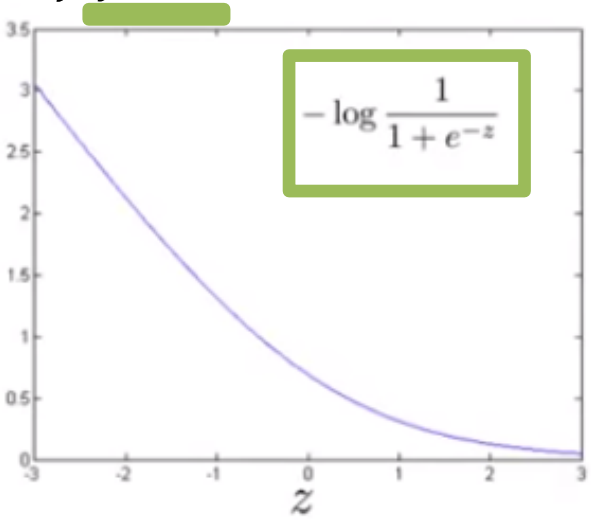
Derivation of hinge loss

$$L = -\left[y \log \sigma(w^T x) + (1-y) \log(1 - \sigma(w^T x))\right]$$

$$L = -y \log \frac{1}{1 + e^{-w^T x}} - (1-y) \log\left(1 - \frac{1}{1 + e^{-w^T x}}\right)$$

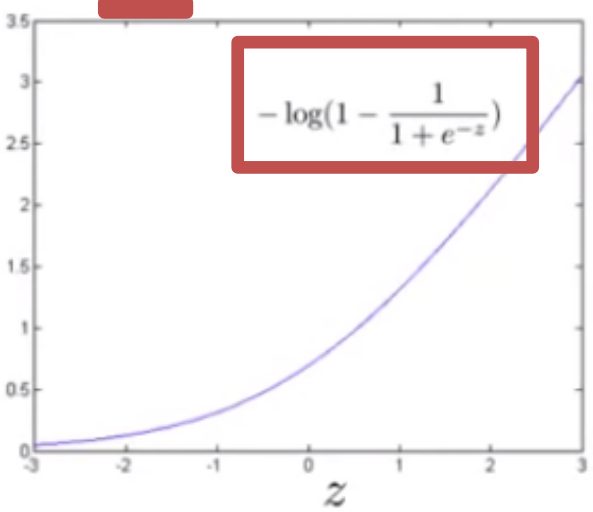
$$Given\ z = w^T x$$

$If\ y = 1, we\ want\ z \gg 0$



What if we got rid of the assymptote?

$If\ y = 0, we\ want\ z \ll 0$



SVM – Theory

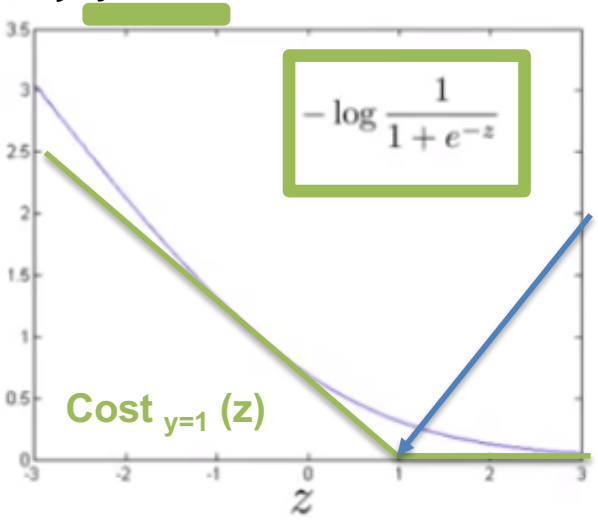
Derivation of hinge loss

$$L = -\left[y \log \sigma(w^T x) + (1-y) \log(1 - \sigma(w^T x))\right]$$

$$L = -y \log \frac{1}{1 + e^{-w^T x}} - (1-y) \log\left(1 - \frac{1}{1 + e^{-w^T x}}\right)$$

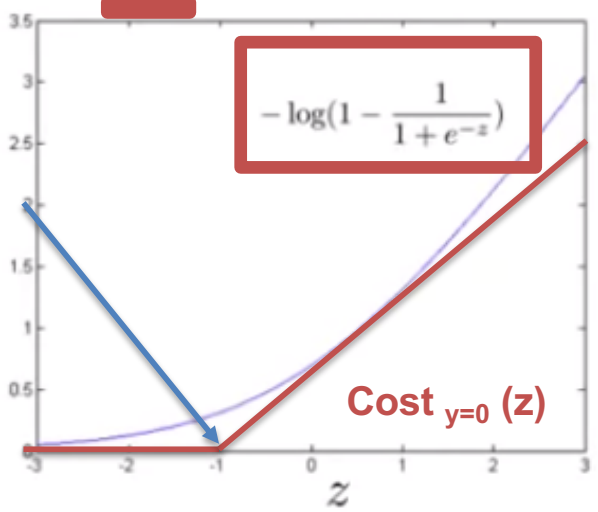
$$Given\ z = w^T x$$

$If\ y = 1, we\ want\ z \gg 0$



Hinge loss!

$If\ y = 0, we\ want\ z \ll 0$



It behaves similarly to the Logistic Regression loss but it brings the advantage of easier computation and easier optimization problem

SVM – Theory

Derivation of final SVM loss

Remember

$$L_{reg} = L + \lambda \|w\|^2$$

Constant to give more weight either to error or penalty

Error penalty

Regularization

LR Loss $(m, number\ of\ samples; n, number\ of\ features)$

$$= \min w \ \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log\frac{1}{1+\ e^{-w^T x}}\right) + \left(1\ - y^{(i)}\right)\left(-\log\left(1\ - \frac{1}{1+\ e^{-w^T x}}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

SVM Loss $(m, number\ of\ samples; n, number\ of\ features)$

$$= \min w \ \ C\sum_{i=1}^{m}\left[y^{(i)}\ cost_{y\to 1}\left(w^T\ x^{(i)}\right) + \left(1\ - y^{(i)}\right) cost_{y\to 1}\left(w^T\ x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n} w_j^2$$

SVM – Theory

Derivation of final SVM loss

Remember

$$L_{reg} = L + \lambda \|w\|^2$$

Constant to give more weight either to error or penalty

Error penalty

Regularization

LR Loss $(m, number\ of\ samples; n, number\ of\ features)$

$$= \min w \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log\frac{1}{1+e^{-w^T x}}\right) + (1-y^{(i)})\left(-\log\left(1-\frac{1}{1+e^{-w^T x}}\right)\right)\right] + \frac{1\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

SVM Loss $(m, number\ of\ samples; n, number\ of\ features)$

$$= \min w\ C\sum_{i=1}^{m}\left[y^{(i)}\ cost_{y\to 1}\left(w^T x^{(i)}\right) + (1-y^{(i)})cost_{y\to 1}\left(w^T x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n} w_j^2$$

SVM – Theory

Derivation of final SVM loss

Remember

$$L_{reg} = L + \lambda \|w\|^2$$

Constant to give more weight either to error or penalty

Error penalty

Regularization

L2, penalizes large weights

$$\textbf{LR Loss}\ (m, number\ of\ samples; n, number\ of\ features)$$

$$= \min w \left[\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \left(-\log \frac{1}{1 + e^{-w^T x}} \right) + (1 - y^{(i)}) \left(-\log \left(1 - \frac{1}{1 + e^{-w^T x}} \right) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right]$$

$$\textbf{SVM Loss}\ (m, number\ of\ samples; n, number\ of\ features)$$

$$= \min w \left[C \sum_{i=1}^{m} \left[y^{(i)}\ cost_{y \to 1} \left(w^T x^{(i)} \right) + (1 - y^{(i)}) cost_{y \to 1} \left(w^T x^{(i)} \right) \right] + \frac{1}{2} \sum_{j=1}^{n} w_j^2 \right]$$

You can use L1, which restricts weights going to 0, but in this case due to dimensionality explosion L2 is better

SVM – Theory

Derivation of final SVM loss

Remember

$$L_{reg} = L + \lambda \|w\|^2$$

Constant to give more weight either to error or penalty

Error penalty

Regularization

L2, penalizes large weights

LR Loss $(m, number\ of\ samples; n, number\ of\ features)$

$$= \min w \left[\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \left(-\log \frac{1}{1 + e^{-w^T x}} \right) + (1 - y^{(i)}) \left(-\log \left(1 - \frac{1}{1 + e^{-w^T x}} \right) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right]$$

SVM Loss $(m, number\ of\ samples; n, number\ of\ features)$

$$= \min w \left[C \sum_{i=1}^{m} \left[y^{(i)}\ cost_{y \to 1} \left(w^T x^{(i)} \right) + (1 - y^{(i)}) cost_{y \to 1} \left(w^T x^{(i)} \right) \right] + \frac{1}{2} \sum_{j=1}^{n} w_j^2 \right]$$

When we minimize this function we have the parameters learned by the SVM

SVM – Theory

Hypothesis for SVM

$$\boldsymbol{SVM\ Loss}\ (m, number\ of\ samples; n, number\ of\ features)$$

$$= \min w\ \ C\sum_{i=1}^{m}\left[y^{(i)}\ cost_{y\to 1}\left(w^T\ x^{(i)}\right) + (1\ - y^{(i)})cost_{y\to 1}\left(w^T\ x^{(i)}\right)\right] + \frac{1}{2}\sum_{j=1}^{n} w_j^2$$

When we minimize this function we have the parameters learned by the SVM

$$h_w(x) = \begin{cases} 1 & if\ w^T x\ \geq 0 \\ 0 & otherwise \end{cases}$$

BTS | Barcelona Technology School

MBDS 2020-2021

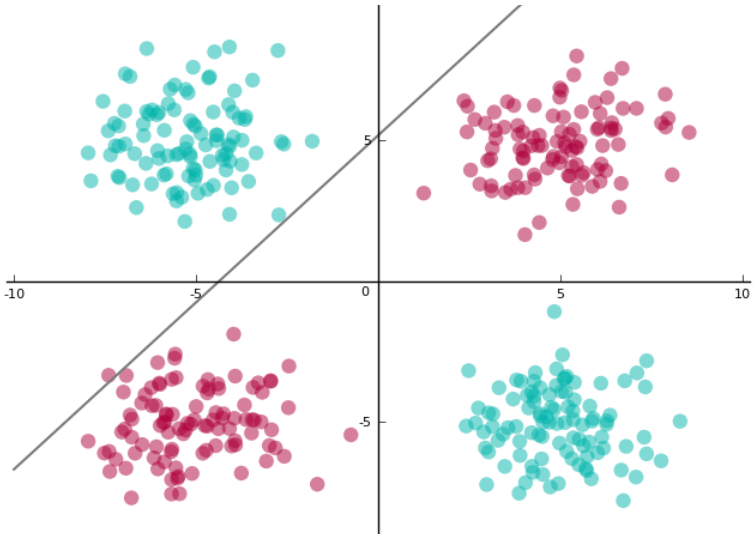
The theory of Support Vector Machines (SVM)

Non-Linear SVMs

SVM – Theory

Non-linear SVMs

- SVM's are able to compute the "best" separating hyper-plane given linearly separable data.
- When there are some mixed examples, we can tune the C parameter so that the model is more flexible.
- SVM's are very efficient because they only use few data points (the support vectors) to find the hyper-plane.
- The hinge loss also makes it efficient at finding the weights
- **But what if the data is not linearly separable?**

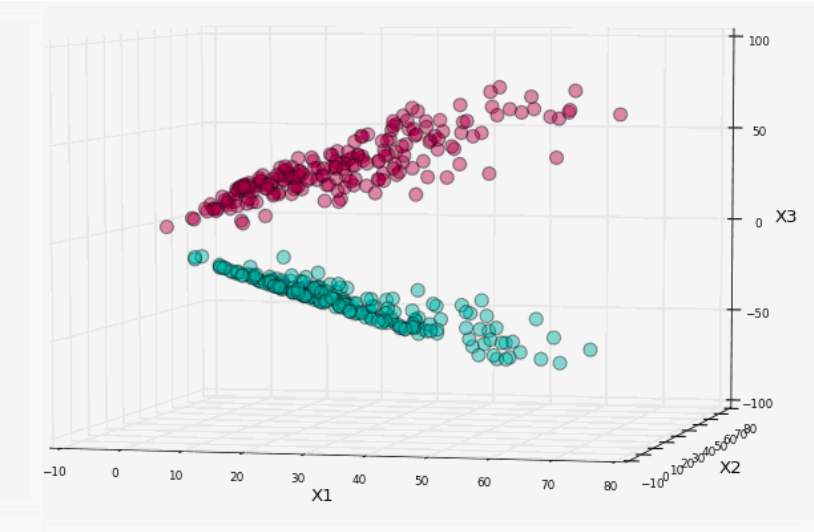


SVM – Theory

Non-linear SVMs

- We can use polynomial feature expansion as we saw earlier

- Adding more dimensions adds more complexity



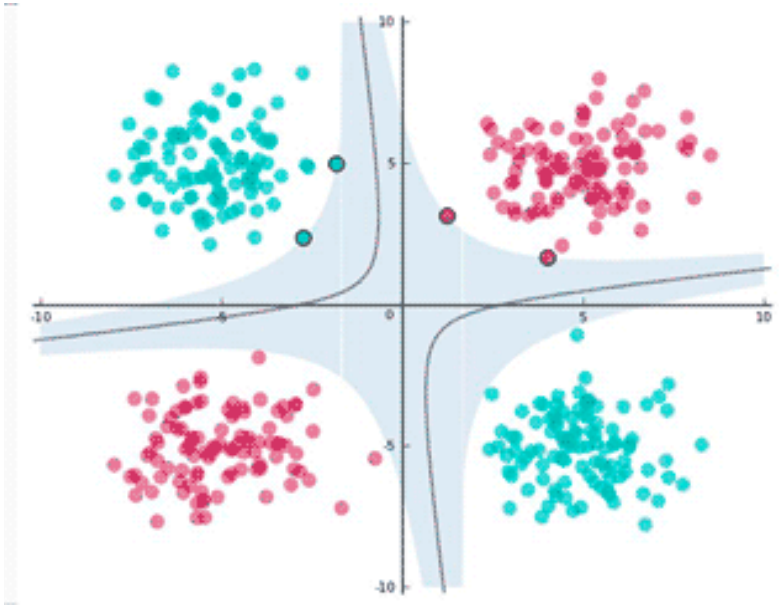
$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$

SVM – Theory

Non-linear SVMs – The kernel trick

- We can use the **kernel trick** to compute the hyperplane in the new dimension

- Notice that when projected, the hyperplane is not a line anymore!

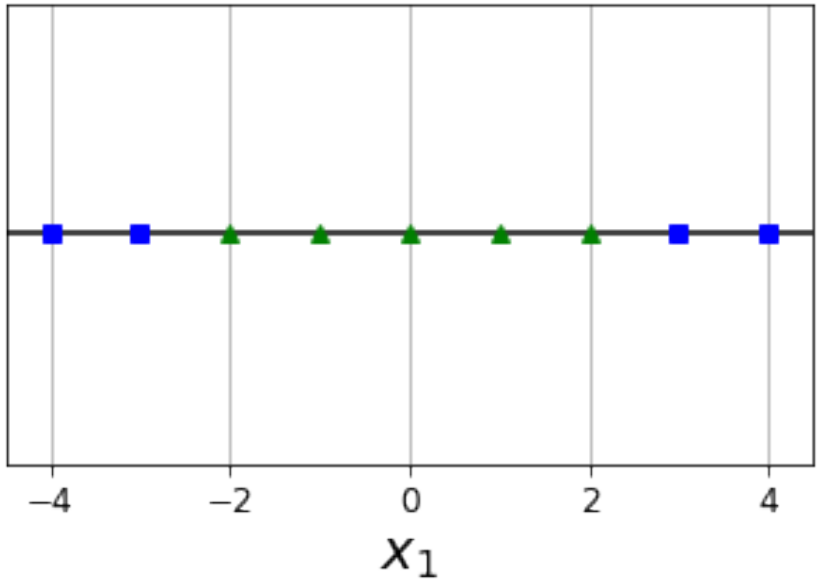


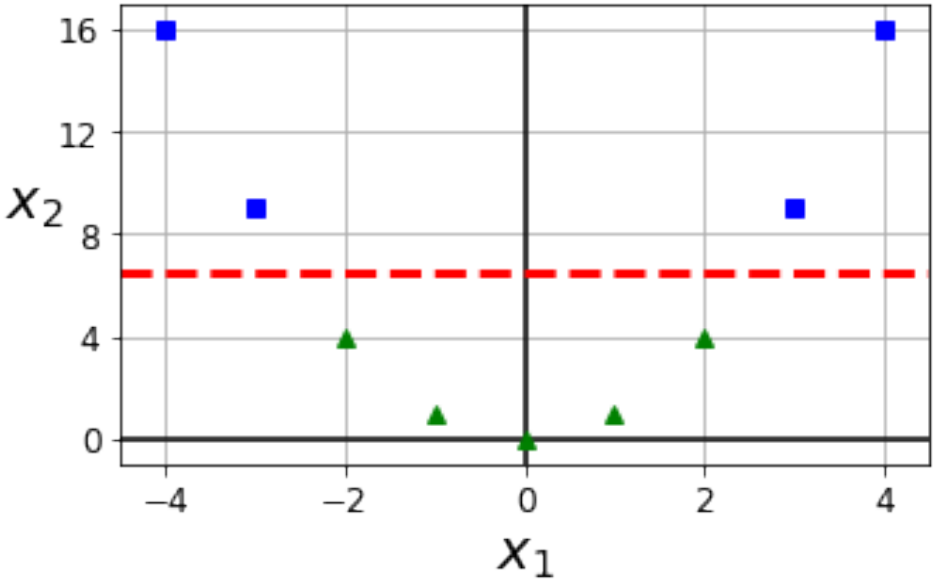
$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$

SVM – Theory

Non-linear SVMs – Polynomial expansion

- As we know one approach to handle nonlinearly separable datasets is to **add more features**
- In some cases this will already result in a linearly separable dataset
- The plot below is not linearly separable but adding another feature $x_2 = (x_1)^2$ resulting in a 2D dataset that can be linearly separated



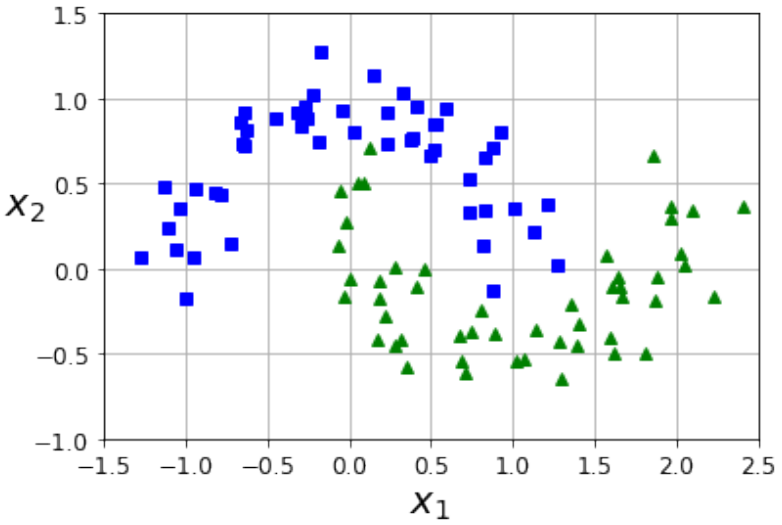


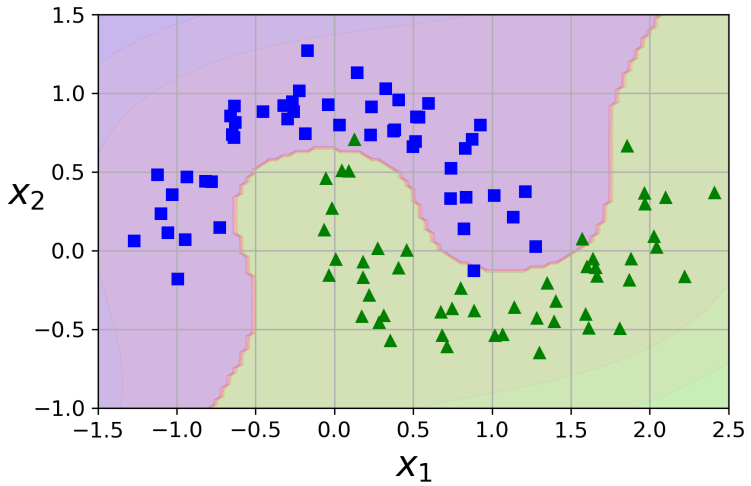
*Code to reproduce the graphs can be seen at **UUID - #S5C9***

SVM – Theory

Non-linear SVMs – Polynomial expansion

- We can add as much polynomial degrees as we want
- In the case below we use the moons dataset, which is a toy dataset for binary classification in which the data points are shaped as two interleaving half circles
- Notice that we are still using a Linear SVM, but with polynomial expansion to the third degree



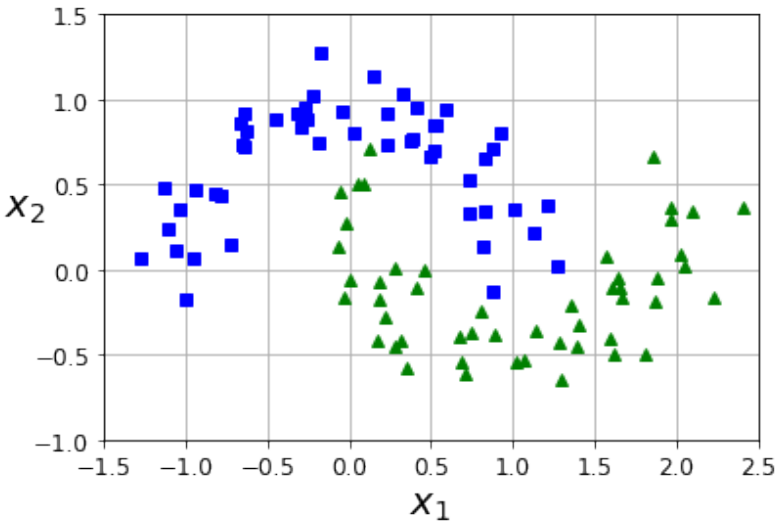


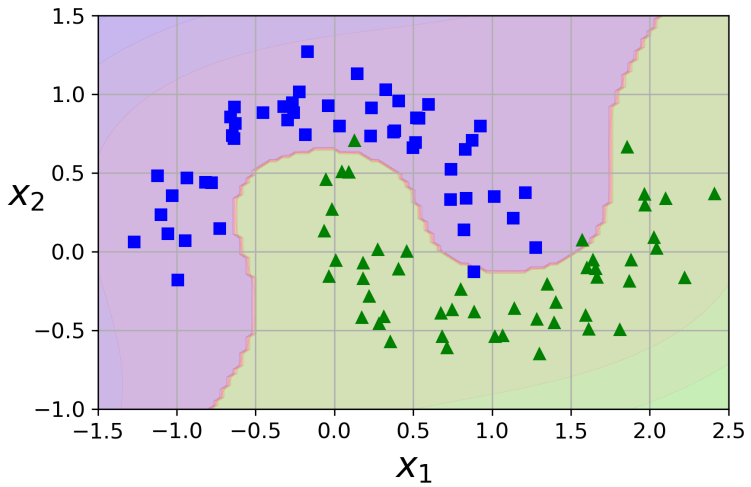
*Code to reproduce the graphs can be seen at **UUID - #S5C9***

SVM – Theory

Non-linear SVMs – Polynomial expansion

```
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

polynomial_svm_clf = Pipeline([
        ("poly_features", PolynomialFeatures(degree=3)),
        ("scaler", StandardScaler()),
        ("svm_clf", LinearSVC(C=10, loss="hinge", random_state=42))
    ])

polynomial_svm_clf.fit(X, y)
```



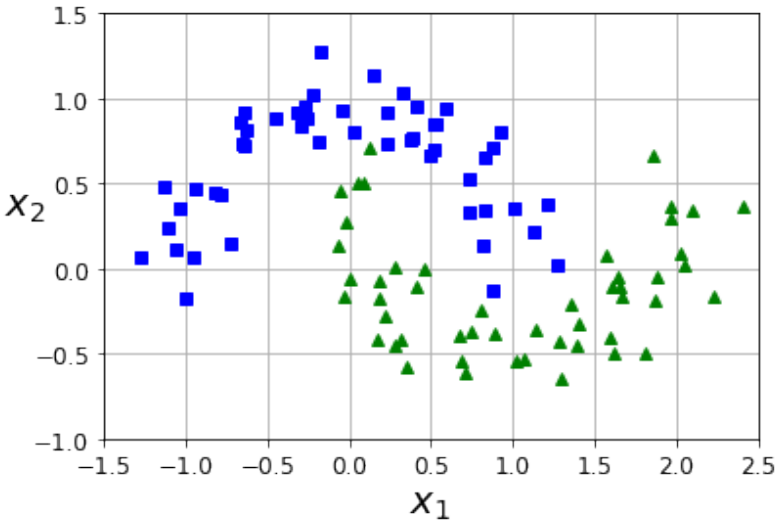


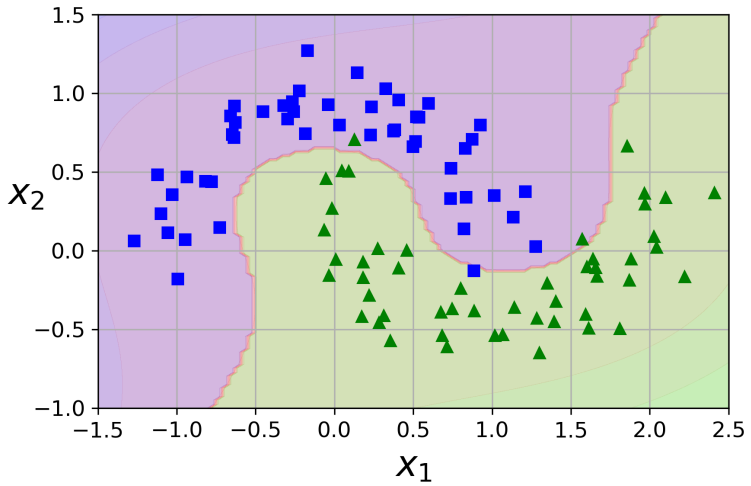
*Code to reproduce the graphs can be seen at **UUID - #S5C9***

SVM – Theory

Non-linear SVMs – Polynomial expansion

- We already know that by adding more features we can deal with nonlinear data. But this can lead to dimensionality explosion





*Code to reproduce the graphs can be seen at **UUID - #S5C9***

BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

Kernel Trick

SVM – Theory

Non-linear SVMs – Kernel trick

- Add polynomial features is simple and can work with many machine learning algorithms. However:
  - At **low degree polynomials** the method cannot deal with very complex datasets
  - With **high polynomial degree** a huge number of features is created, making the model slow and too computationally complex
- We can use the **kernel trick** instead, which consists of observing that many machine learning algorithms can be written exclusively in terms of dot products between examples

SVM – Theory

Non-linear SVMs – Kernel trick

- The kernel function is equal to performing the dot product in the high dimensional space:

Polynomial features

$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$

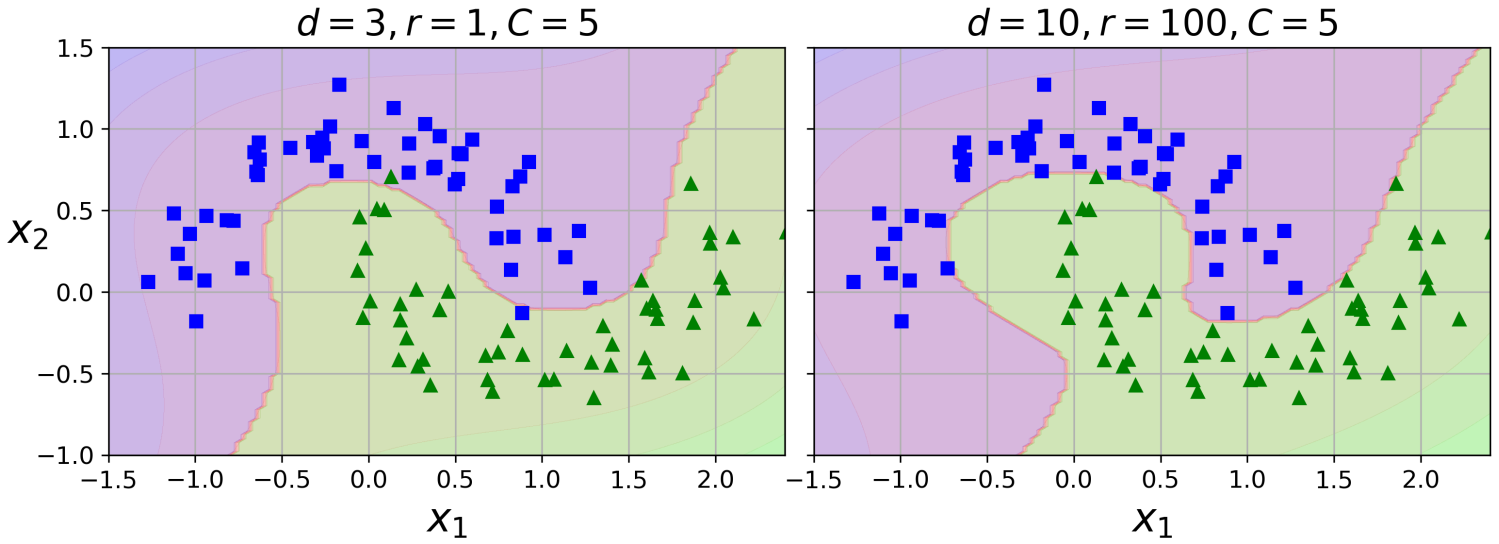
Kernel features (as many features as samples) [m=n]

$$\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip})$$
$$\vec{x_j} = (x_{j1}, x_{j2}, \ldots, x_{jp})$$
$$\vec{x_i} \cdot \vec{x_j} = x_{i1}x_{j1} + x_{i2}x_{j2} + \ldots + x_{ip}x_{jp}$$
$$\begin{aligned} K(\vec{x_i}, \vec{x_j}) &= (\vec{x_i} \cdot \vec{x_j})^2 \\ &= (x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} \\ &= (x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2}) \cdot (x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2}) \end{aligned}$$

SVM – Theory

Non-linear SVMs – Kernel trick

- The kernel trick makes it possible to get the same result as adding many polynomial features but without actually having to add them
- There is no combinatorial explosion of features because you don't actually add any features, the maximum number of features is equal to the number of samples



*Code to reproduce the graphs can be seen at **UUID - #S5C10***

SVM – Theory

Non-linear SVMs – Kernel trick

- Notice that we now use the SVC class, which implements the kernel
- Guidelines:
  - If the model underfits, increase the polynomial degrees
  - If the model overfits, decrease the polynomial degrees
  - The coef0 value controls how much the model is influenced by high-degree polynomials versus low-degree polynomials (represented as “r” in the graphs above)

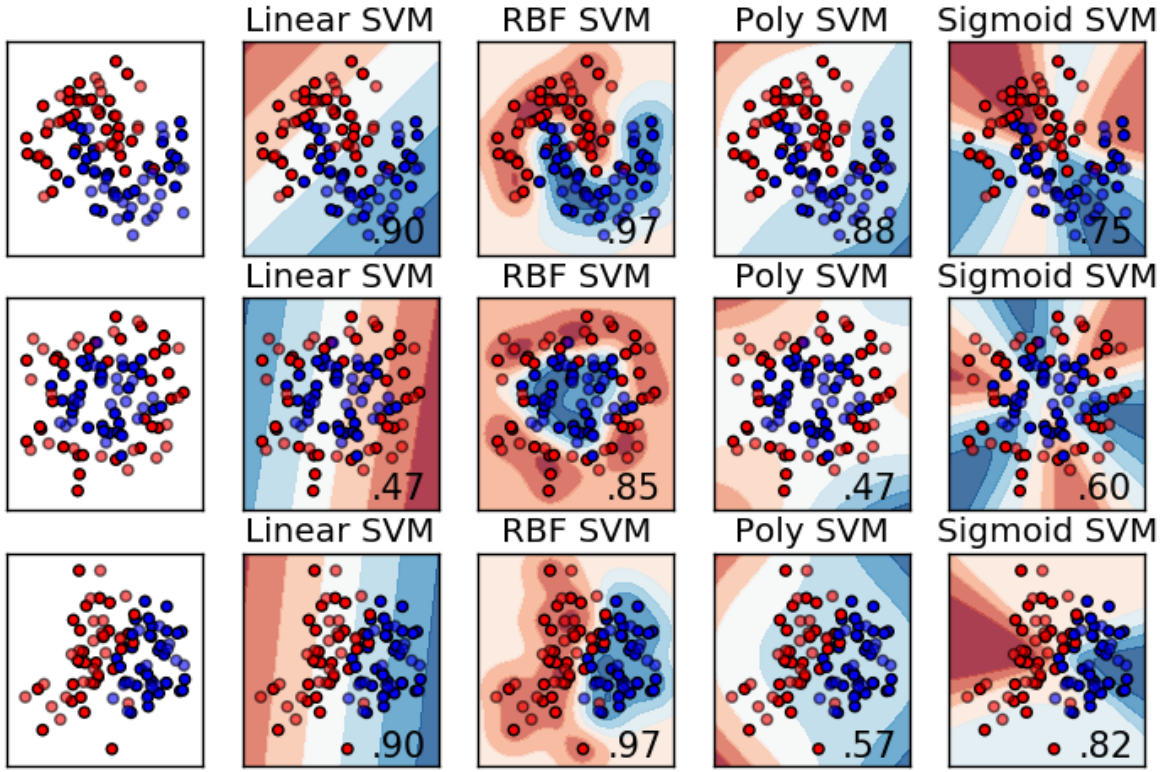
```
from sklearn.svm import SVC

poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

*Code to reproduce the graphs can be seen at **UUID - #S5C10***

SVM – Theory

Non-linear SVMs – Common kernels

- linear : $\langle x, x' \rangle$
- polynomial: (γ⟨x,x′⟩+r)^d
- rbf: exp (−γ‖x−x′‖^2)
- Sigmoid: (tanh (γ⟨x,x′⟩+r)



BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

Duality

SVM – Theory

Non-linear SVMs – Duality

- The SVC class assumes that the number of features==number of samples, thus there is no duality parameter
- In LinearSVC, since we might use polynomial expansion, we can end up with either:
  - Samples < Features
  - Samples > Features
    - Always set dual=False when samples > features

BTS | Barcelona Technology School

MBDS 2020-2021

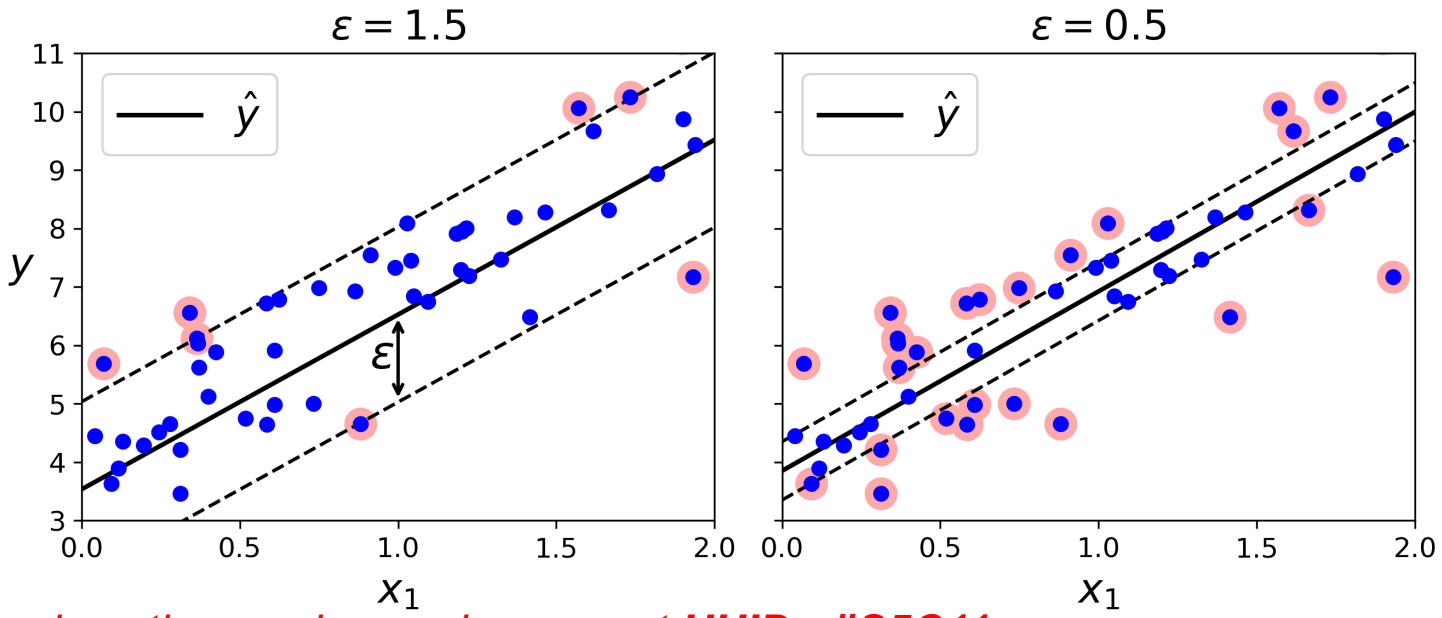
The theory of Support Vector Machines (SVM)

Regression with SVMs

SVM – Theory

Regression with SVMs

- SVM also supports linear and non linear regression
- To use SVMs for regression instead of classification, the trick is to reverse the objective:
  - SVR tries to fit as many instances as possible on the street while limiting margin violations (samples out of the street)
  - The epsilon parameter controls the width of the street
- Adding more samples within the margin does not affect the predictions, so it is said that the model is **epsilon insensitive**

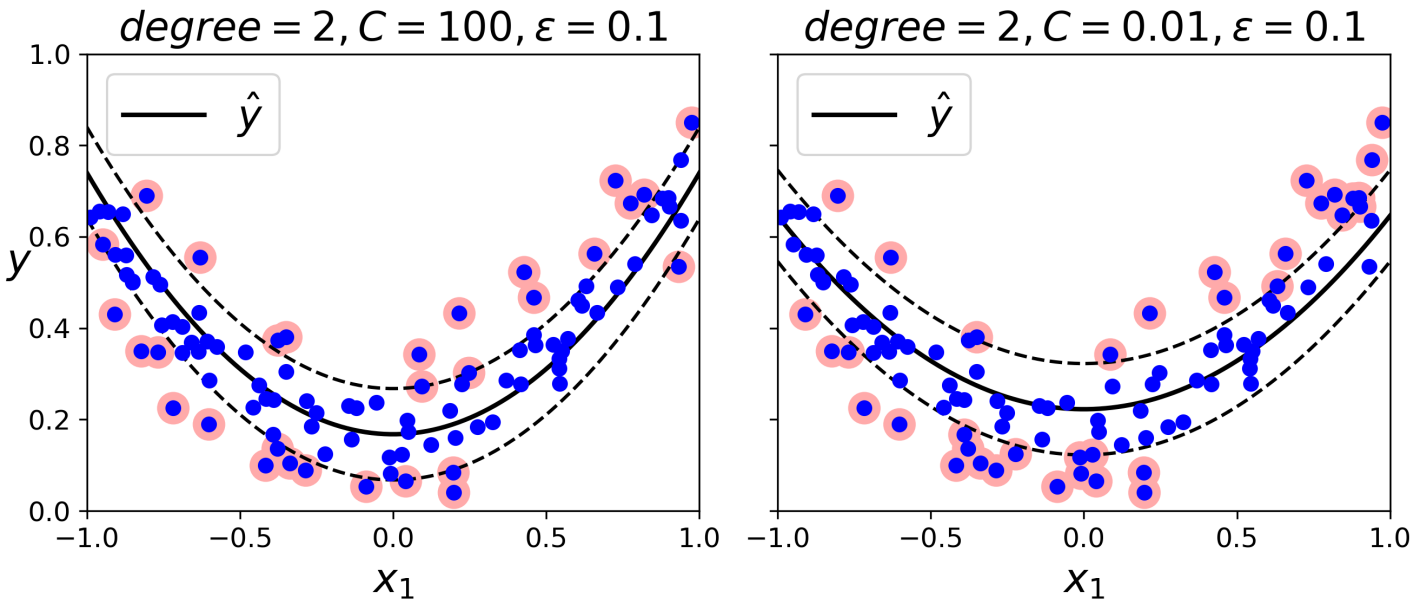


*Code to reproduce the graphs can be seen at **UUID - #S5C11***

SVM – Theory

Regression with SVMs

- We can also use the kernel trick with SVR (SVR is equivalent to SVC, and Linear SVR to Linear SVC)
- We can then use kernels and C regularization to tradeoff error and large weight penalties



*Code to reproduce the graphs can be seen at **UUID - #S5C11***

BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

Multi-class with SVMs

SVM – Theory

Multi-class classification

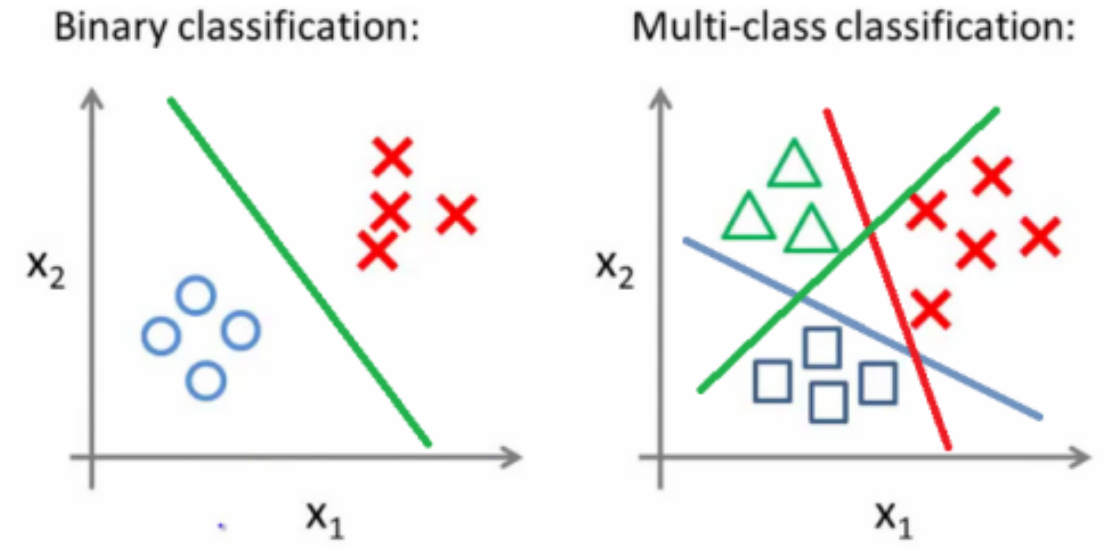
Different approaches

OvA: One-vs-all classification

OvO: One-vs-one

classification (sort ot

Crammer Singer)



BTS | Barcelona Technology School

MBDS 2020-2021

The theory of Support Vector Machines (SVM)

Summary and Hyperparameters

SVM - Parameters

Most important parameters

Parameters	LinearSVC
Penalty	L2 is better
Loss	Hinge is better
Dual	False when samples > features. True when they are the same.
Tolerance (stop criteria)	Default is fine
C	Depends on the problem. Tradeoff between error penalty and weights penalty.
Multi_class	Default is OvR. The crammer_singer is similar to OvA

SVM - Parameters

Most important parameters

Parameters	SVC
C	Depends on the problem. Tradeoff between error penalty and weights penalty.
Kernel	RBF is fine, but many times poly and linear are also fine.
Degree	Degree of polynomials in kernel, ignored by other kernels.
Gamma	The scaling coefficient for kernels. The Auto uses 1/number_features, which is fine
Coef0	Coefficient to give more weights to higher and lower degree polynomials.
Decision_function_shape	One versus one (ovo) or (one versus rest (ovr). Works best with ovo for multiclass, ovr for binary classes.

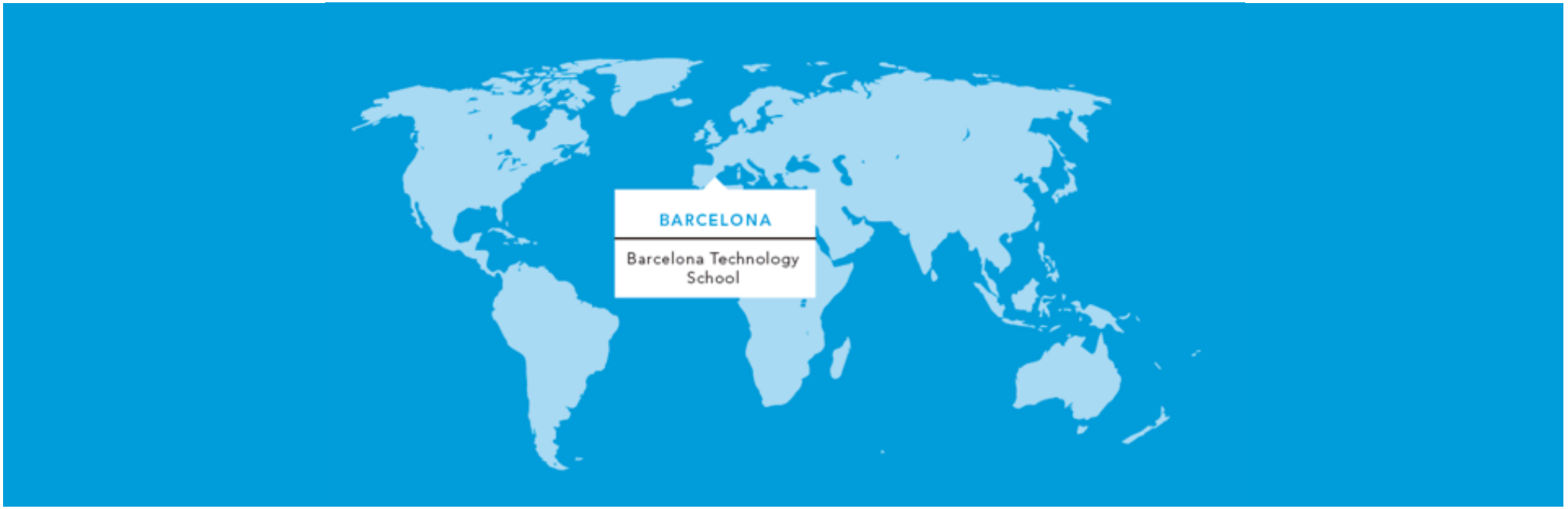
BTS | Barcelona Technology School

MBDS 2020-2021

RECAP

Resources

- Lazy Programmer Tutorials
- Lex Friedman Series (MIT)
- Aurelien Geron, “Hands-on machine learning with scikit-learn, Keras & Tensorflow”
- Tan, Pang-Ning. *Introduction to data mining*. Pearson Education India, 2006.
- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. New York: Springer series in statistics, 2001.
- Andrew Ng lectures: https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN


BARCELONA
Barcelona Technology School