

Machine Learning

Linear Regression
Logistic Regression
Support Vector Machines

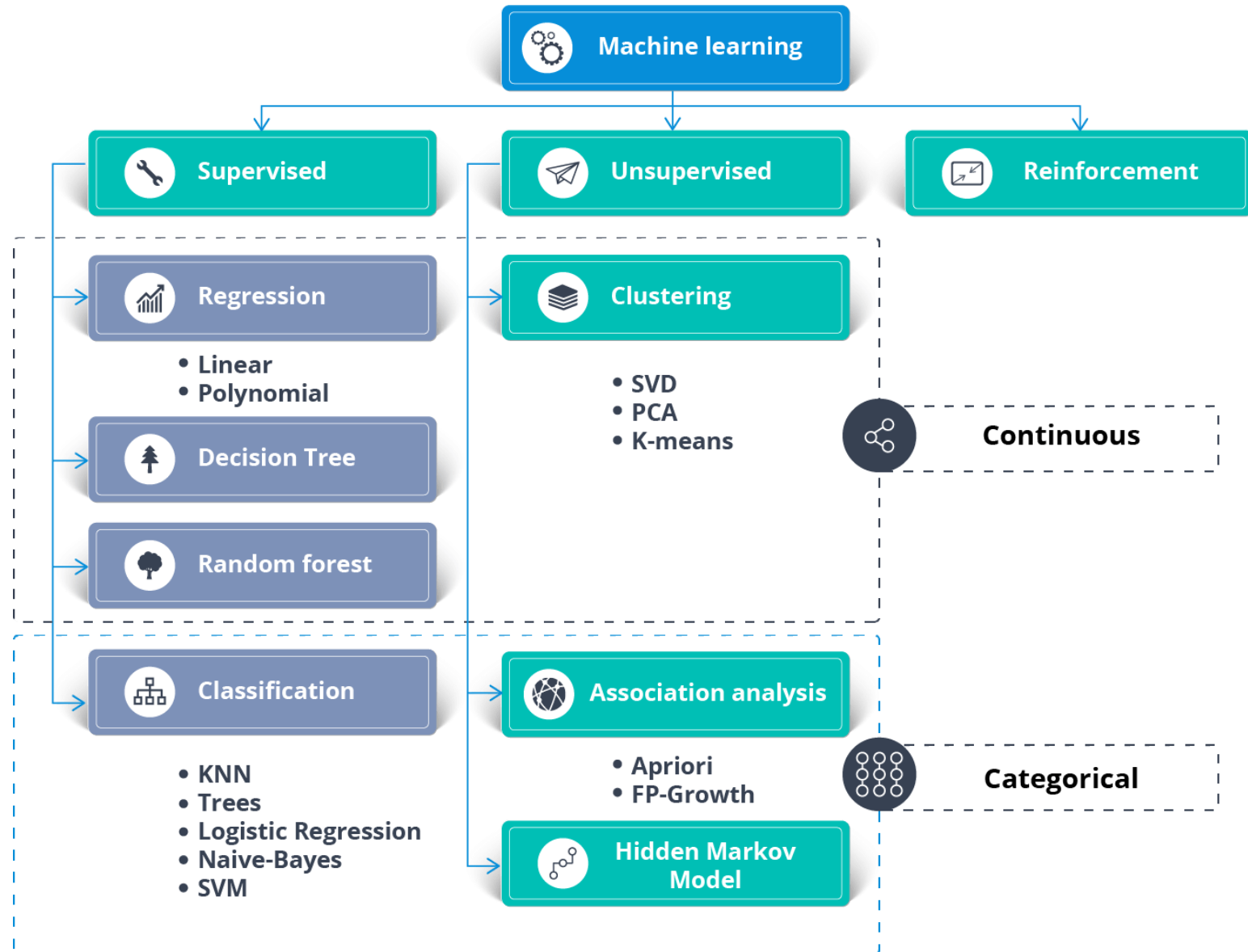
Francesco Pugliese, PhD

neural1977@gmail.com

We will discuss about...

- ✓ What is Linear Regression ?
- ✓ What is Logistic Regression ?
- ✓ What is Support Vector Machines ?

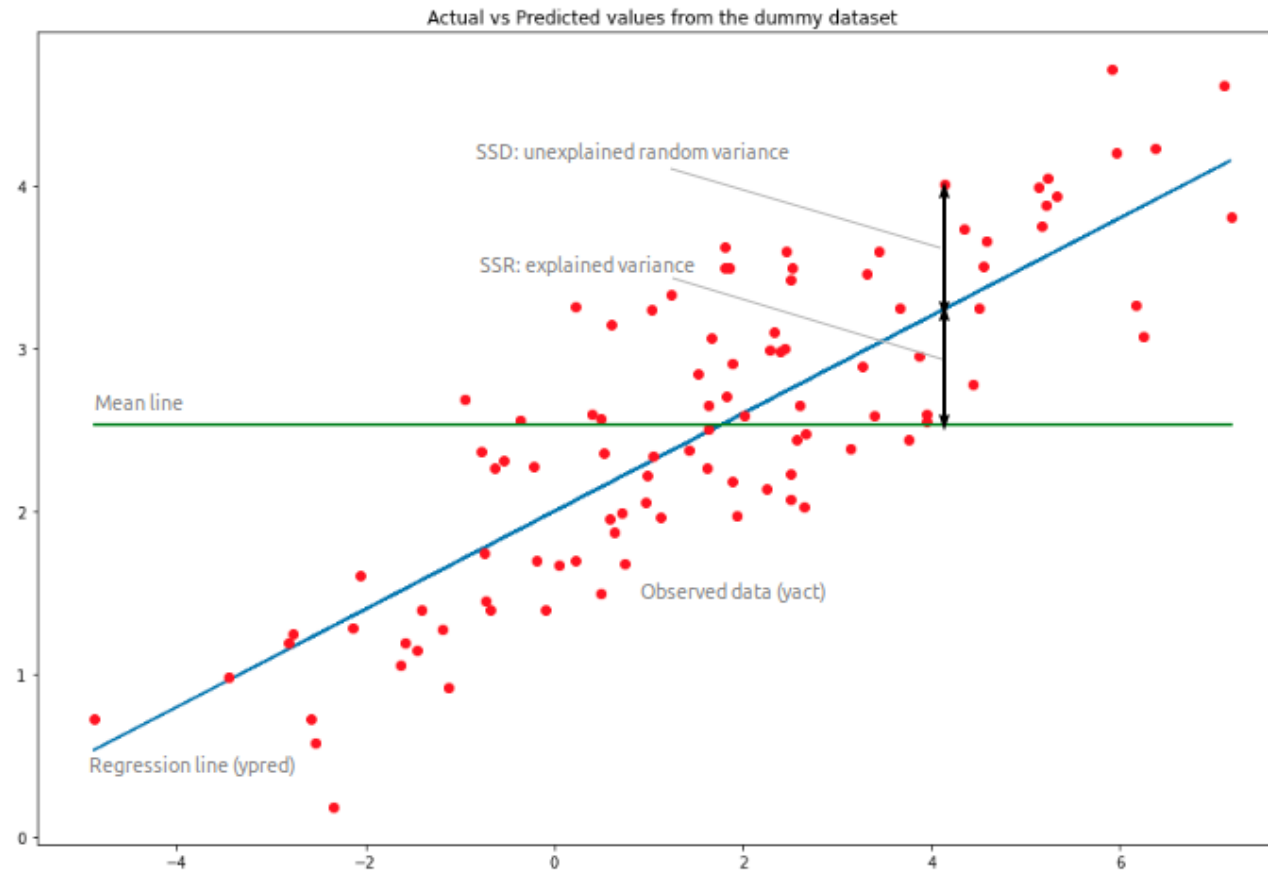
Algorithm Classification



Linear Regression

- Linear regression is a basic and commonly used type of predictive analysis.
- Our questions are:
 - (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable?
 - (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they indicated by the magnitude and sign of the beta estimates—impact the outcome variable?
- These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables.

Linear Regression



Linear Regression

- ✓ It is a machine learning algorithm based on **supervised learning**.
- ✓ Models the relationship between a dependent variable and one or more independent variables.
- ✓ Mainly used when working with **scalar** and **exploratory** variables.
- ✓ Linear Regression finds its application to determine the extent to which there exists a **linear relationship** between a dependent variable (scalar) and one or more independent variables (exploratory).
- ✓ A single independent variable is used to predict the value of a dependent variable.

Linear Regression

- The simplest form of the regression equation with one dependent and one independent variable is defined by the formula $y = c + b * x$, which is a line (y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable).
- **Hypothesis function** is, as its meaning states, our **prediction** of how the final curve after the process of regression will look like. In the case of linear regression with one variable it looks like:

$$h_{\Theta}(x) = \Theta_0 + \Theta_1 * x$$

- In the case of Multivariate Linear Regression we have:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d$$

Linear Regression

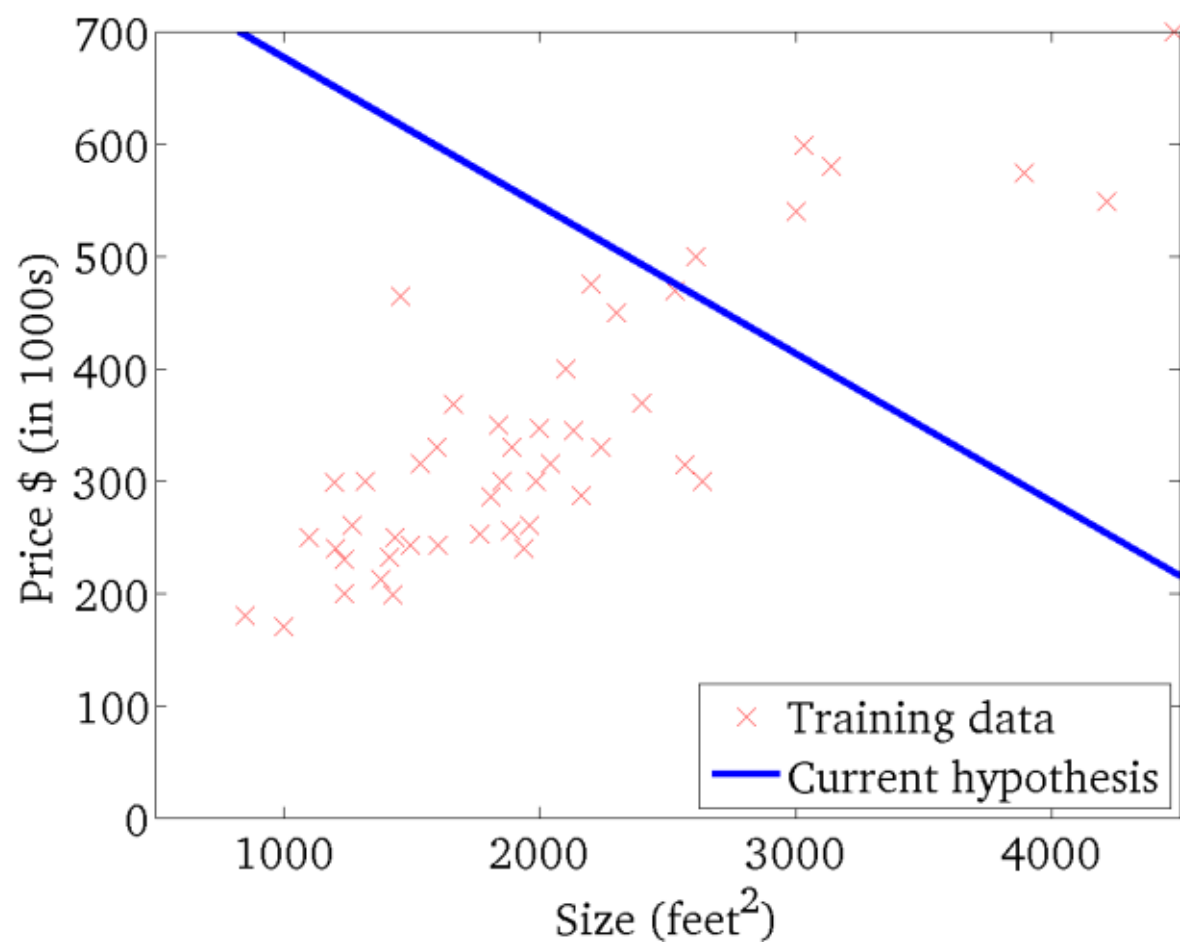
- 'x' is or are the **independent variable** on which the hypothesis depends.
 2. '**Theta**' are the model parameters, 0 is our **bias** variable and 1 is our **weight** variable.
- **Loss Function** is any equation which gives us an idea of how close we are to the required hypothesis. Higher the loss (or cost) function, farther are we from the required curve, and viceversa. Thus, this is the measure we are willing to minimize to implicitly reduce the error. A typical **Mean Squared Error** cost function looks like this:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

Linear Regression: Gradient Descent

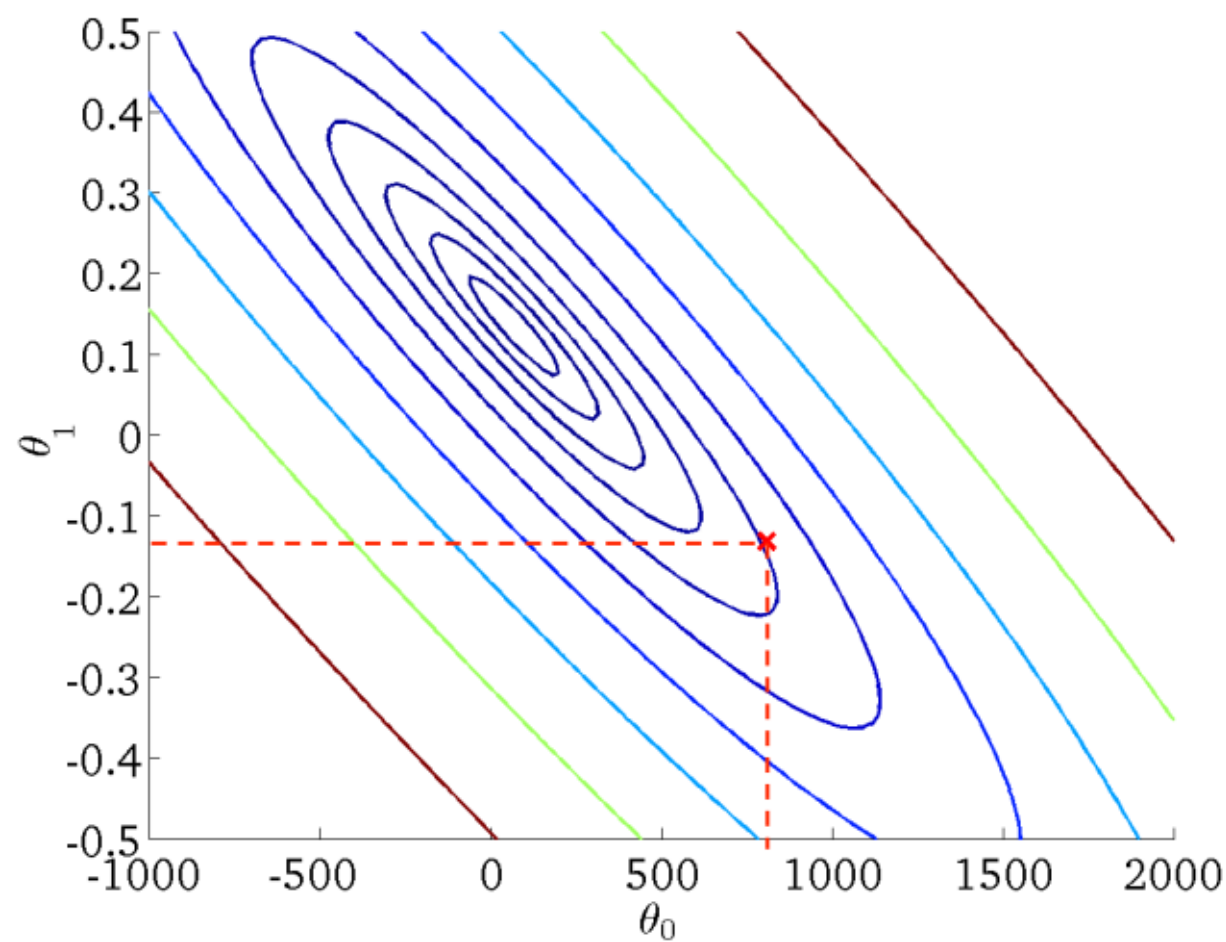
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



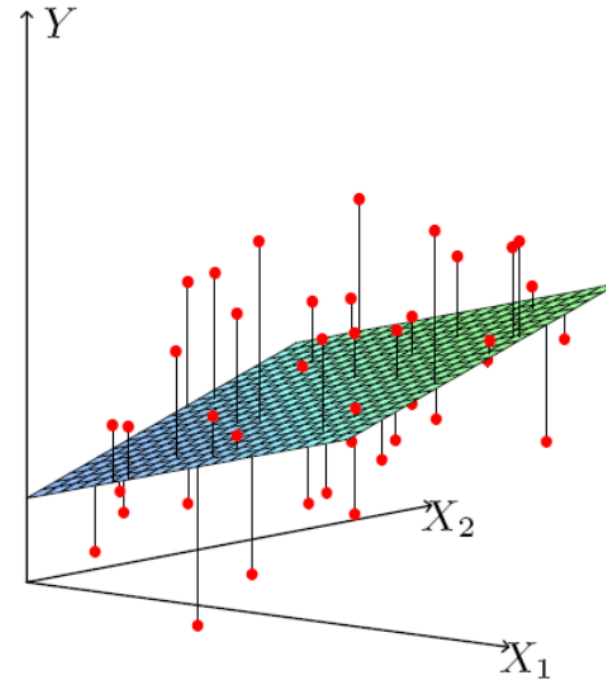
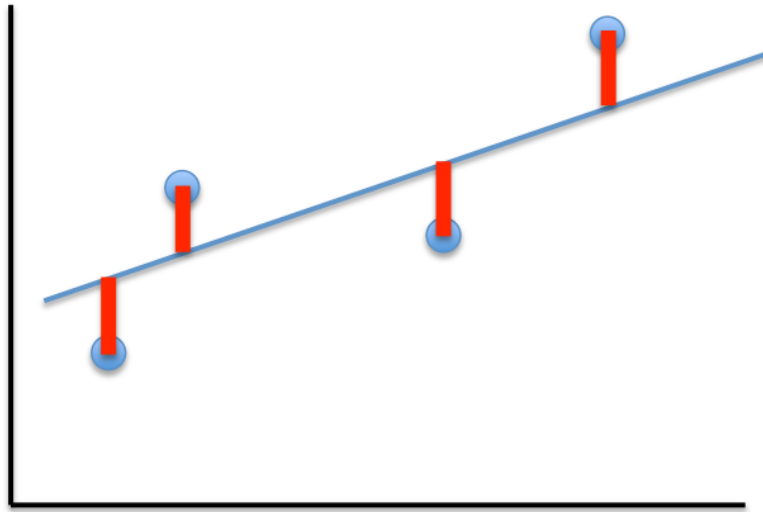
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

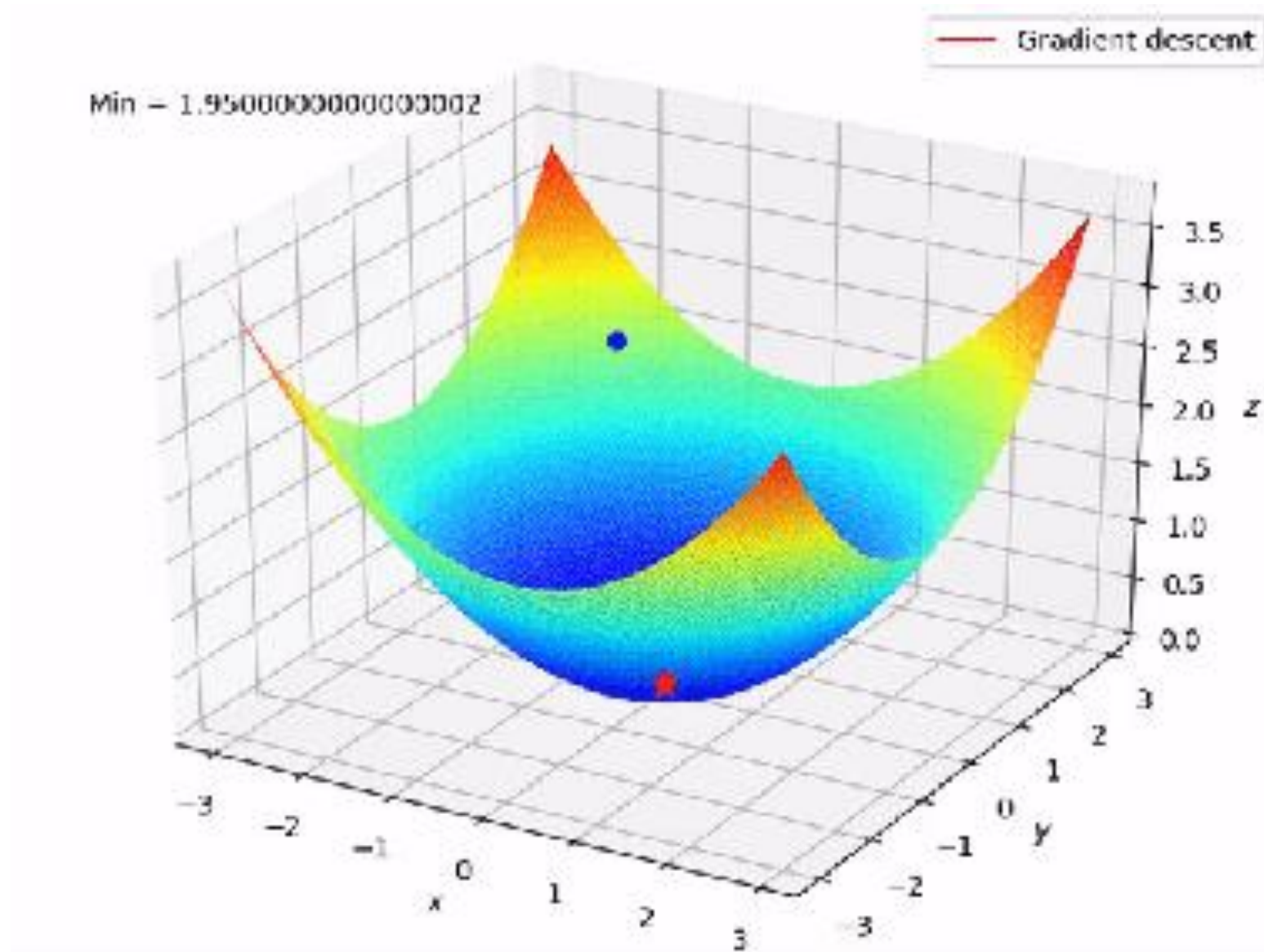


Linear Regression: Gradient Descent

- The constant $\frac{1}{2m}$ is introduced so as to induce a basic normalization to the cost function value based on the number of entries and also to make it look pretty (2 and the 1/2 get cancelled when we differentiate J).



Linear Regression: Gradient Descent



Linear Regression: Gradient Descent

- **Gradient Descent** is a method by which we shall minimize the loss(Cost function). It is an optimization function that changes the values of theta 0 and theta 1 based on the slope of the cost function curve at that point. The changes in theta 0 and theta 1 represent changes on our hypothesis so as to get a better fit to the given data. We accomplish the task by applying this formula:

$$\Theta_i := \Theta_i - \alpha \frac{\partial}{\partial \Theta_i} J(\Theta_0, \Theta_1)$$

- Where alpha is the **Learning Rate**

Linear Regression: Gradient Descent

For Linear Regression:

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}\end{aligned}$$

Linear Regression: Steps of Gradient Descent

- 1) Initialize the parameters Theta
- 2) Repeat until convergence:

$$\Theta_i := \Theta_i - \alpha \frac{\partial}{\partial \Theta_i} J(\Theta_0, \Theta_1)$$

- 3) Convergence is achieved when the L2 norm:

$$\|\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}\|_2 < \epsilon$$

Applications of Linear Regression

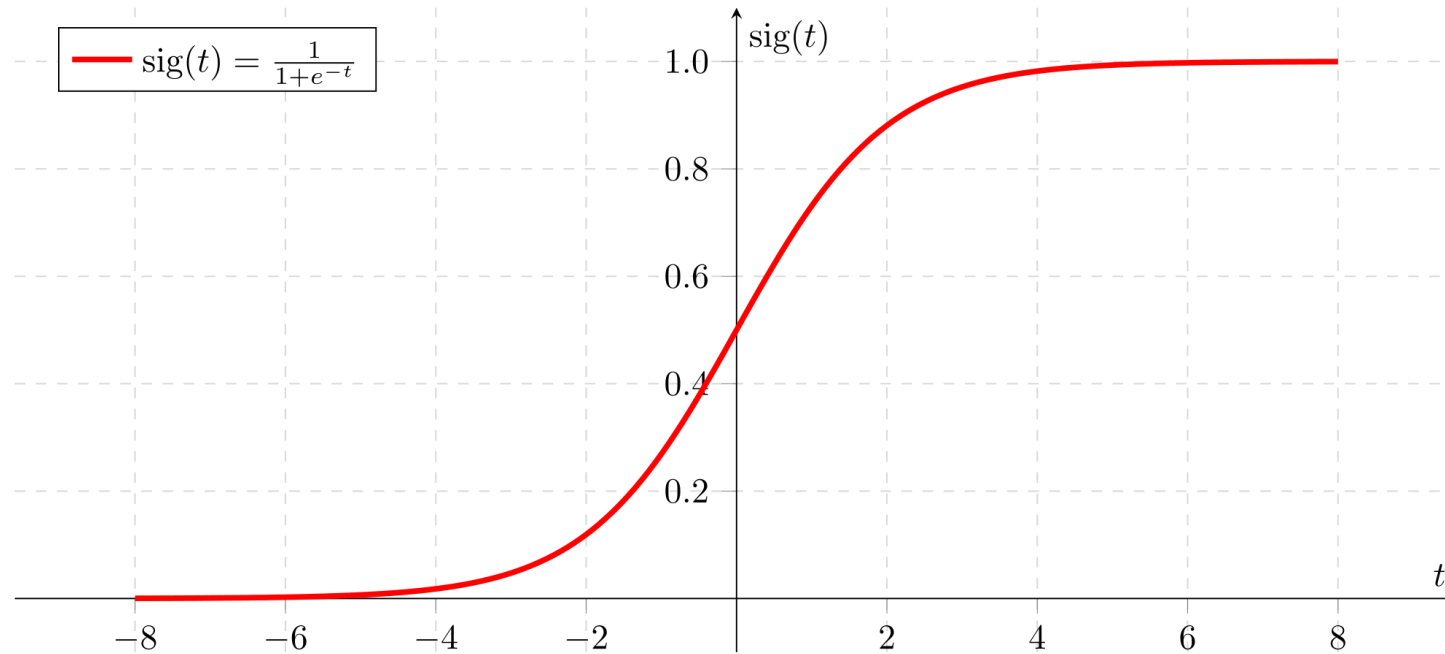
- ✓ Risk Management in financial services or insurance domain
- ✓ Predictive Analytics
- ✓ Econometric
- ✓ Epidemiology
- ✓ Weather data analysis
- ✓ Customer survey results analysis

Logistic Regression

- The **Logistic Regression** algorithm determines what class a new input should fall into, that's why Y is a discrete value.
- Classification problems
 - Email -> spam/not spam?
 - Online transactions -> fraudulent?
 - Tumor -> Malignant/benign
- Variable in these problems is Y is either 0 or 1
 - 0** = negative class (absence of something)
 - 1** = positive class (presence of something)

Let's start with **binary class problems**. We can always implement a multiclass classification problem, which is an extension of the binary classification

Logistic Regression



Logistic Regression predicts the probability of occurrence of a binary event utilizing a sigmoid function.

When to use Logistic Regression ?

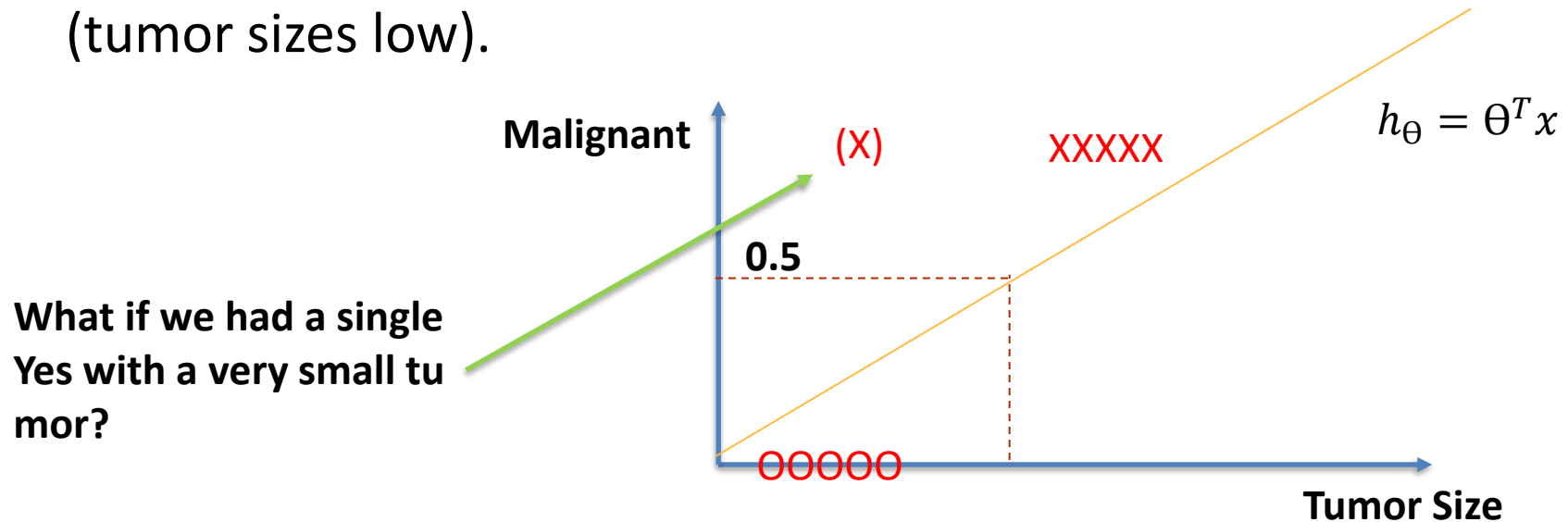
- ✓ It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable.
- ✓ The sigmoid function, also called the logistic function, gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \dots(1)$$

- ✓ If the curve goes to positive infinity, y predicted will become 1
- ✓ If the curve goes to negative infinity, y predicted will become 0
- ✓ If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it like 0 or NO
- ✓ If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that patient will suffer from cancer.

Logistic Regression

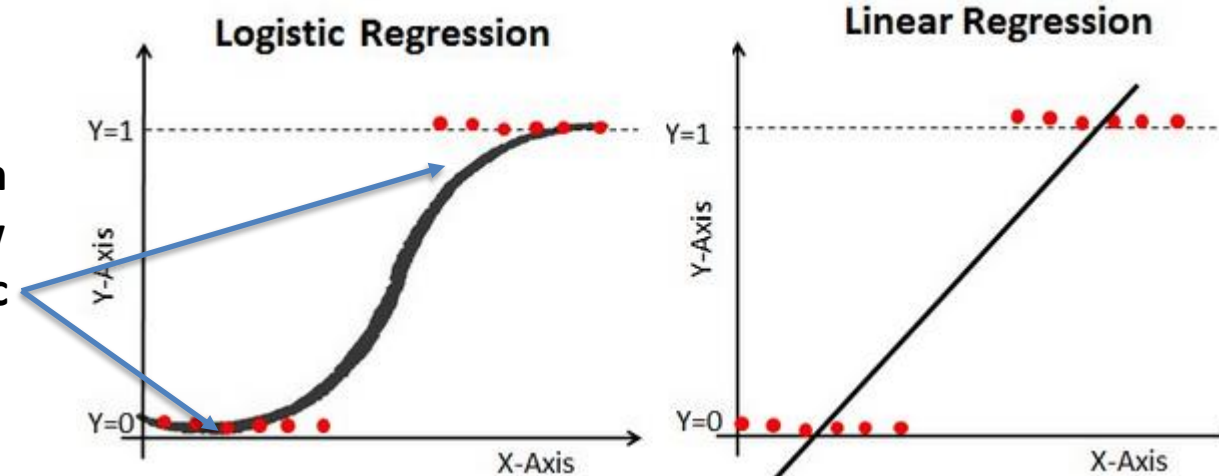
- Let's consider the problem "Tumor Size Versus Malignancy (No or Yes). We could adopt the linear regression which has the following hypothesis function: $h_{\theta} = \theta^T x$
- Then we can threshold the classifier output, namely anything over some values is yes (tumor sizes high) otherwise it is no (tumor sizes low).



Logistic Regression

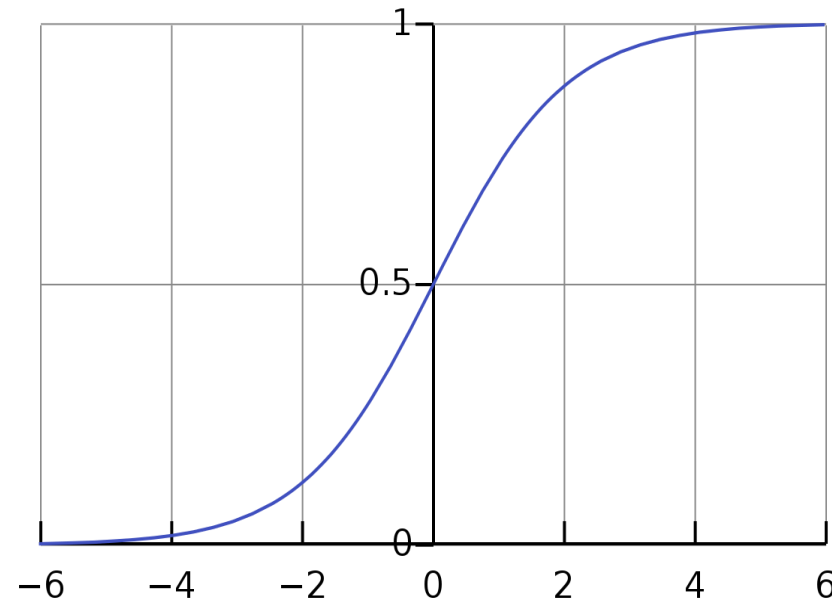
- We can see previously above this does a reasonable job of stratifying the data points into one of two classes but in some cases we might have some yes classified as nos !!!.
- By using a linear regression with threshold We might have small tumors classified as NO!! Moreover, with linear regression, output values can be larger than 1 or less than 0

Instead, we need to generate a value that is always either 0 or 1. This is called a «Classification problem».



Logistic Regression: Hypothesis function

- Sigmoid function looks like a curve that crosses 0.5 at the origin, then flattens out asymptotically at 0 and 1.
- Given this, we need to fit θ to our data
- When our hypothesis $h_{\theta}(x)$ outputs a number, we treat that value as the estimated probability that $y=1$ on input x



Logistic Regression: Hypothesis function

- We can write this using the following notation:

$$h_{\theta} = P(y = 1 | x ; \theta)$$

Which means probability that $y = 1$, given x , parameterized by θ .

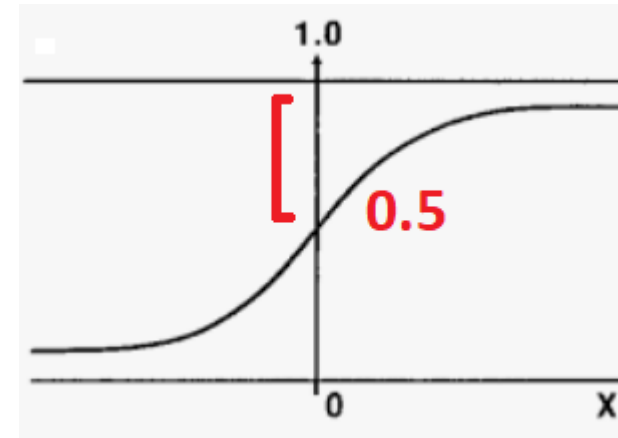
- Since this is a «classification task» we want that y must be 1 or 0 . So we must have:
- $P(y = 1 | x ; \theta) + P(y = 0 | x ; \theta) = 1$
- $P(y = 0 | x ; \theta) = 1 - P(y = 1 | x ; \theta)$

Logistic Regression: Decision Boundary

- One way of using the sigmoid function is:
 - 1) When the probability of y being 1 is greater than 0.5 then we can predict $y = 1$
 - 2) Else we predict $y = 0$

- So the hypothesis predicts $y = 1$ when $\theta^T x \geq 0$

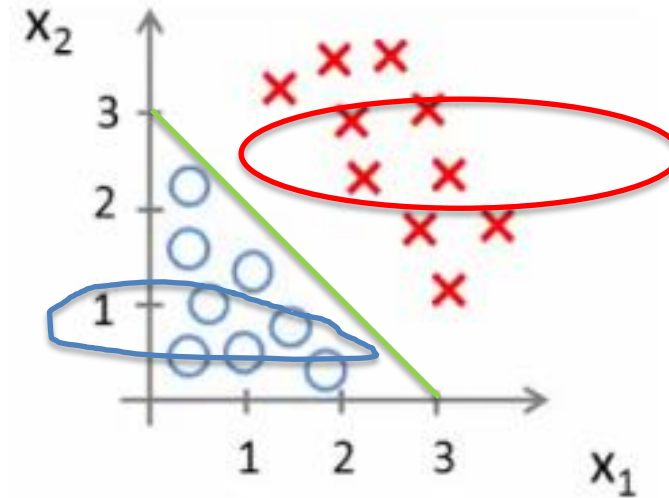
- When $\theta^T x < 0$ then the hypothesis predicts $y = 0$.



Logistic Regression: Decision Boundary

- $h_{\theta} = G(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
- For $\theta = [-3, 1, 1]$ we have:
$$-3 + x_1 + x_2 \geq 0$$

So if $(x_1 + x_2 \geq 3)$ we predict $y = 1$

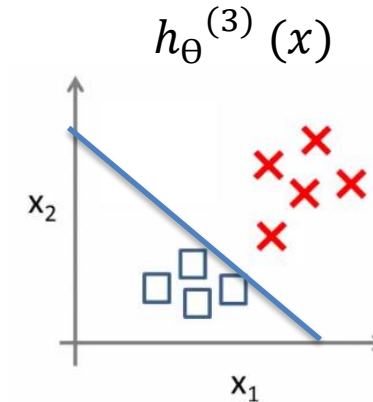
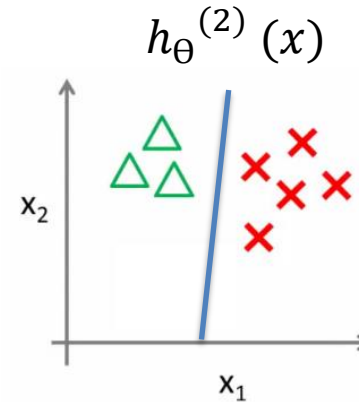
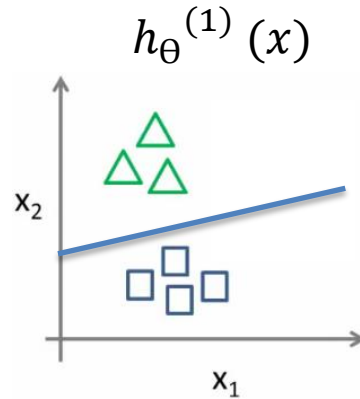


$x_1 + x_2 \geq 3$ is our graphical plot of the **decision boundary**.

- Concretely, the straight line is the set of points where $h_{\theta}(x) = 0.5$ exactly. The decision boundary is a property of the hypothesis function.

Logistic Regression: Multiclass Classification Problem

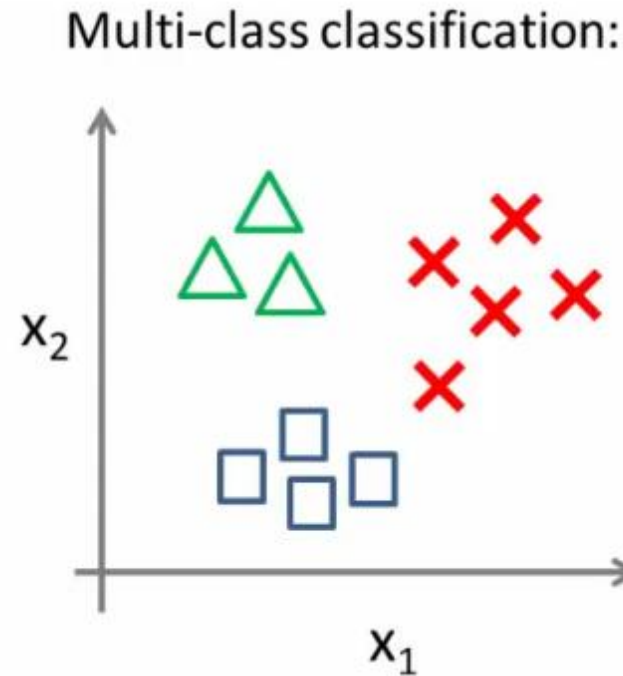
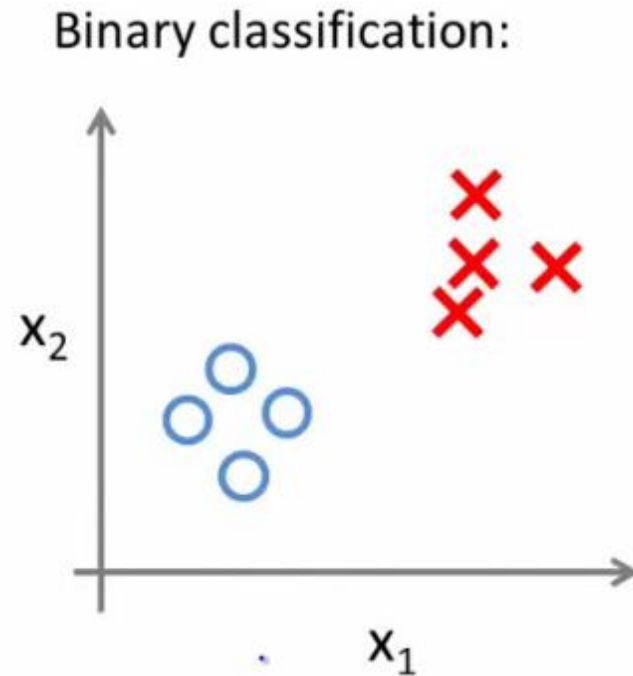
- We split the training set into three separate binary classification problems.



- Train a logistic regression classifier $h_{\theta}^{(i)}(x)$ for each class i in order to predict the probability that $y = 1$
- On a new input, x to make a prediction, pick the class i that maximizes the probability that

Logistic Regression: Multiclass Classification Problem

- We adopt a **one vs. all classification** that makes a binary classification work for a multi-class classification.



Logistic Regression

- ✓ Logistic Regression is used when the dependent variable is **binary**.
- ✓ It is a go-to method for binary classification problems in statistics.
- ✓ First, it is quintessential to understand **when** to use linear regression and when to use logistic regression.
- ✓ **Linear vs Logistic regression –**
 - ✓ Linear regression is used when the dependent variable is **continuous** and the nature of the regression line is linear.
 - ✓ Logistic regression is used when the dependent variable is **binary** in nature.

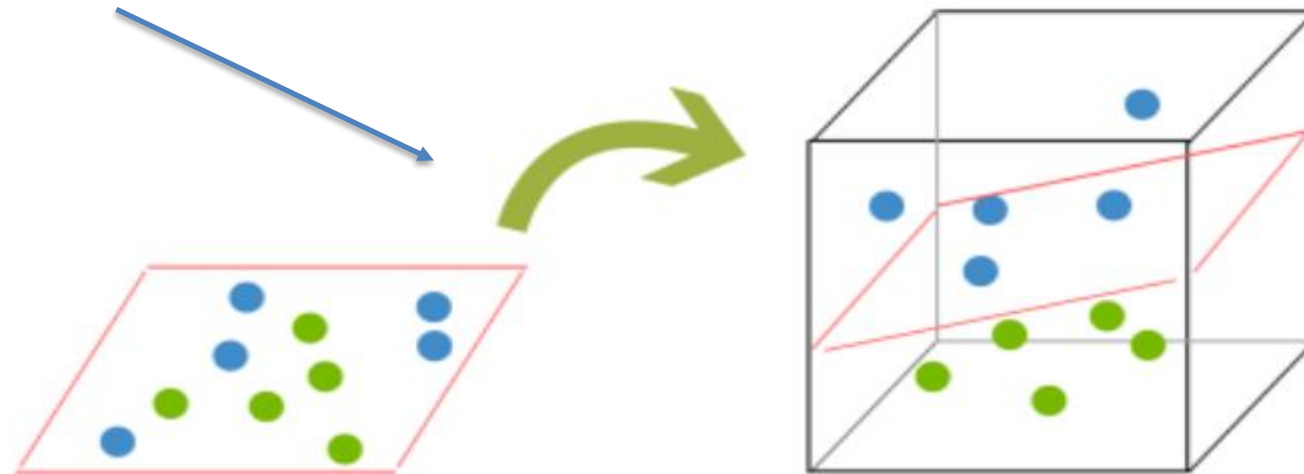
Applications of Logistic Regression

- ✓ Cancer Detection
- ✓ Trauma and Injury Severity Score
- ✓ Image Segmentation and Categorization
- ✓ Geographic Image Processing
- ✓ Handwriting recognition
- ✓ Prediction whether a person is depressed based on bag of words from the corpus

SUPPORT VECTOR MACHINES (SVMs)

- A **Support Vector Machine (SVM)** is a supervised machine learning algorithm that can be employed for both classification and regression purposes.

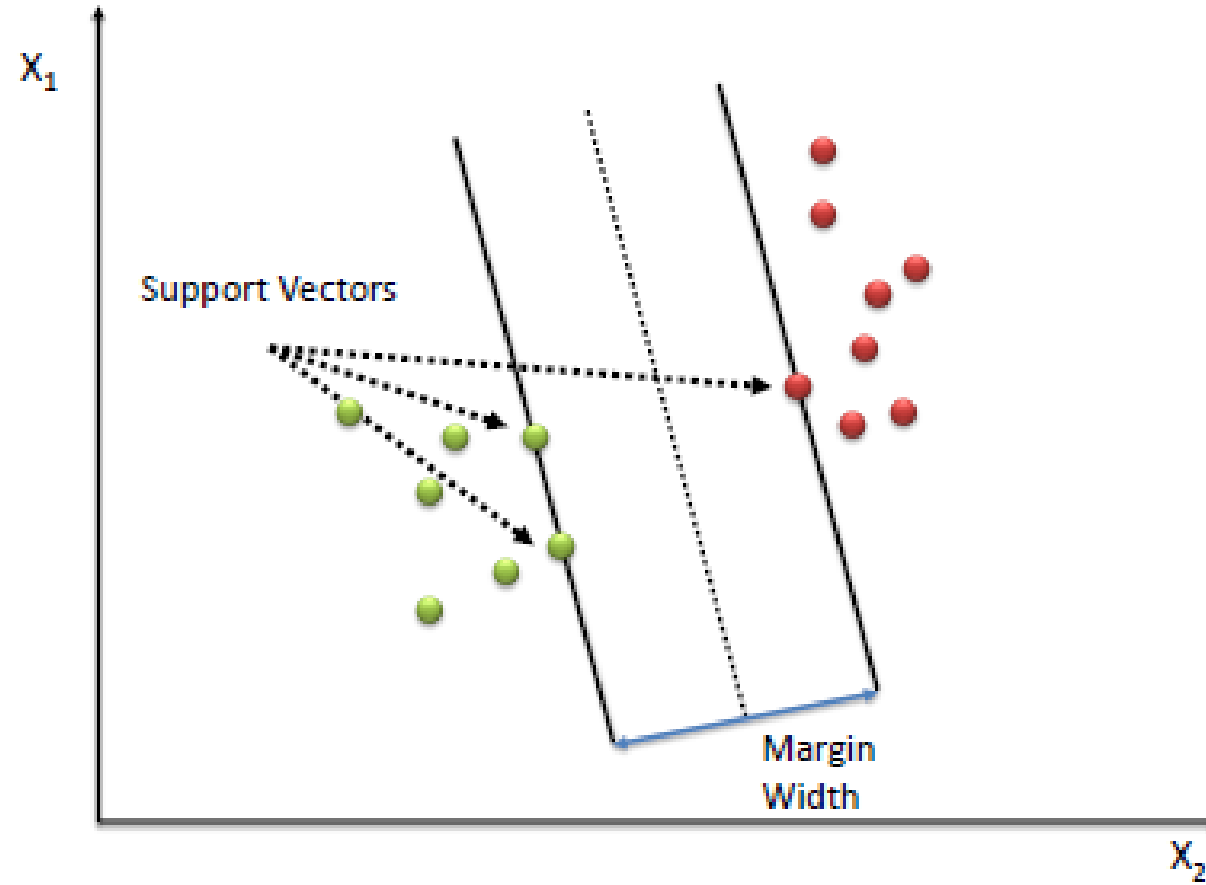
THE PROBLEM OF DIVIDING THE DATASET INTO TWO OR MORE CLASSES



Support Vector Machines

- ✓ Create a line or a hyperplane which separates the data into multiple classes.
- ✓ Is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.
- ✓ Transforms your data base on it, finds an optimal boundary between the possible outputs.
- ✓ Performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors that define the hyperplane are called the **support vectors**.
- ✓ **The SVM Algorithm:**
 - ✓ Define an optimal hyperplane with a maximized margin
 - ✓ Map data to a high dimensional space where it is easier to classify with linear decision surfaces
 - ✓ Reformulate problem so that data is mapped implicitly into this space

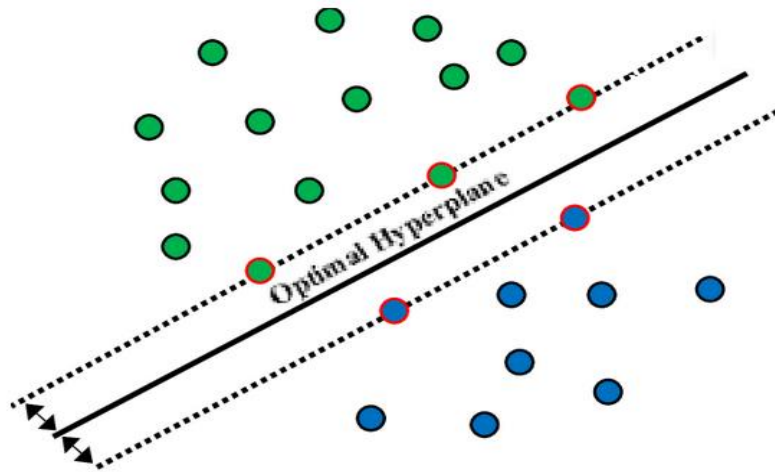
Support vector machines(SVM)



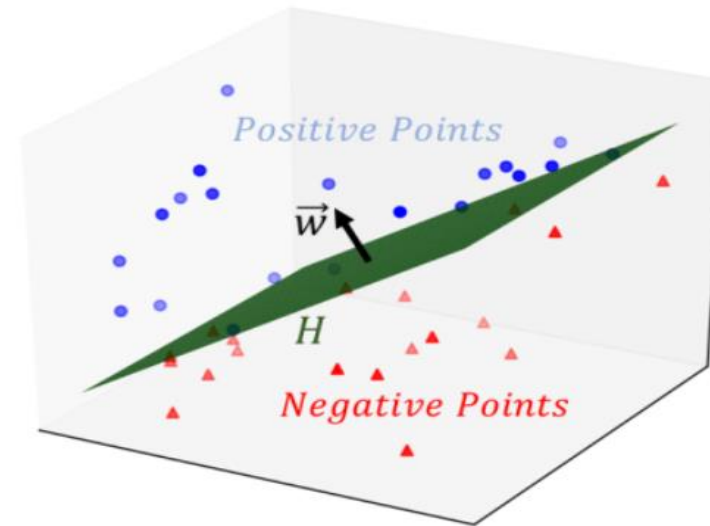
SUPPORT VECTOR MACHINES (SVMs)

- **SVMs** are a machine learning model that is based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.

2D Case

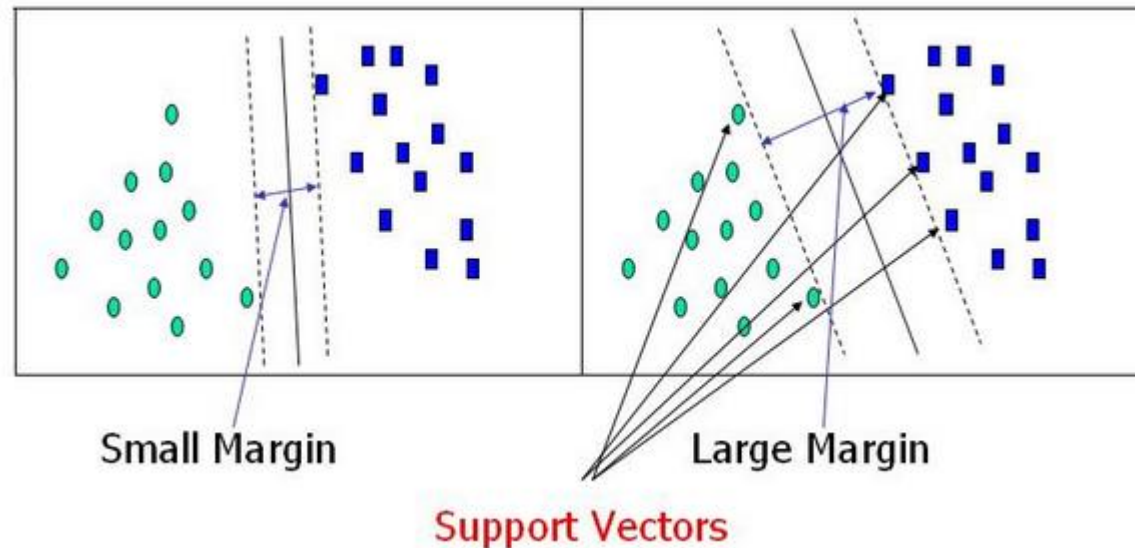


3D Case



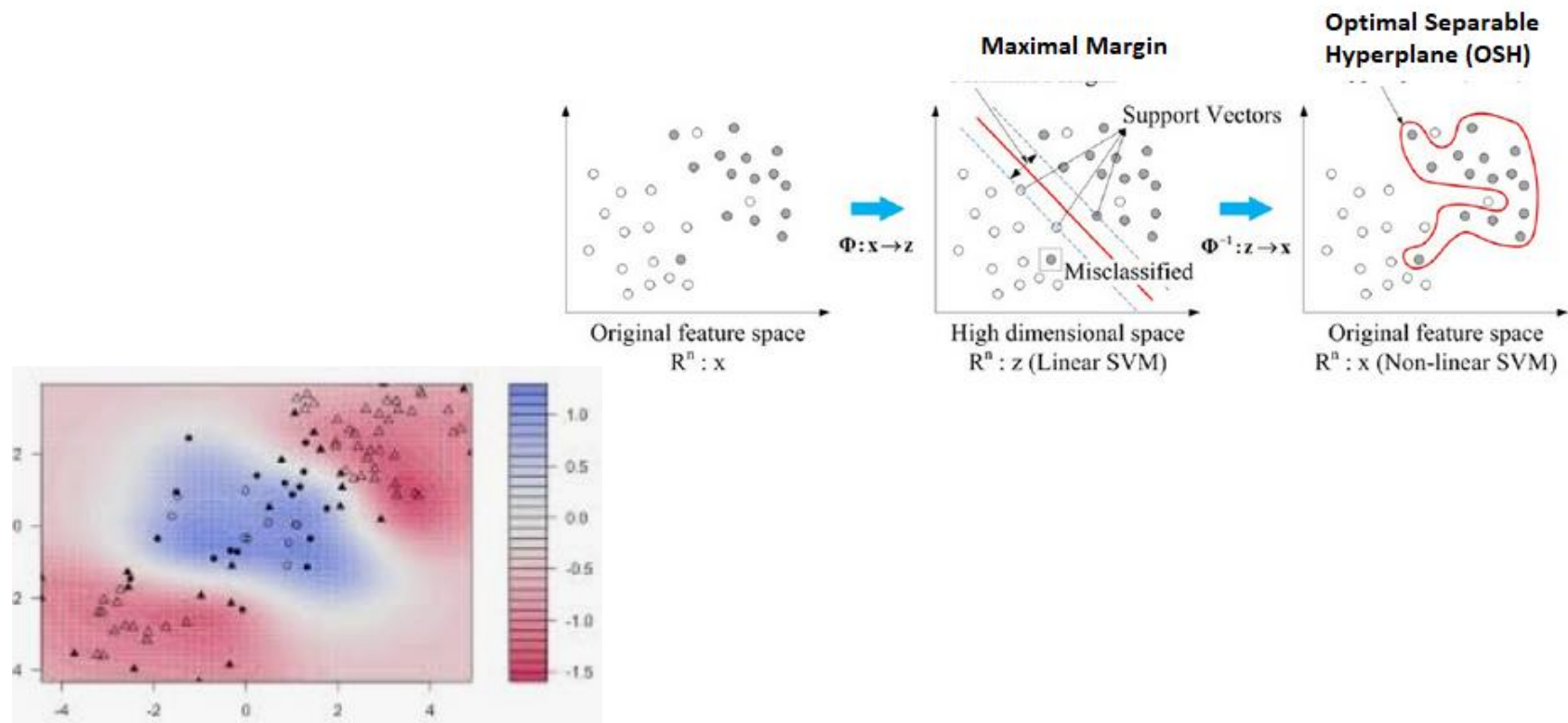
SUPPORT VECTOR MACHINES (SVMs)

- **Support Vectors** are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set..



SUPPORT VECTOR MACHINES (SVMs)

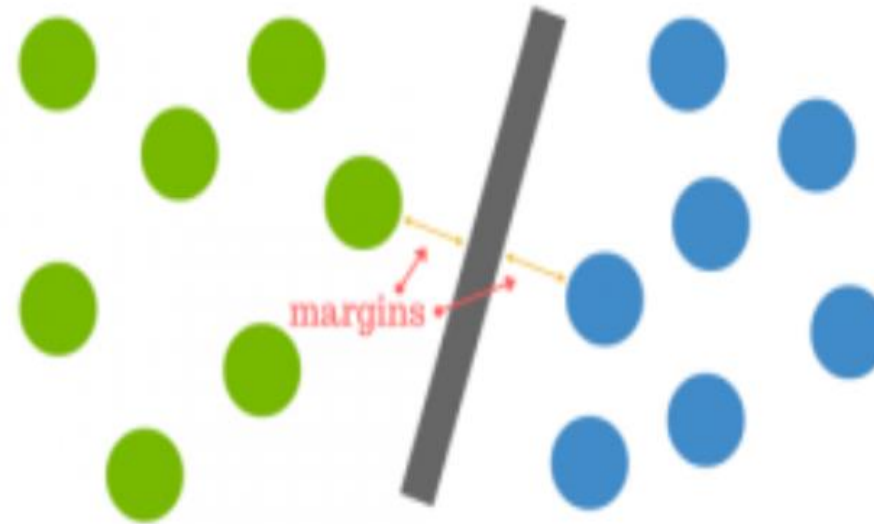
- Compared to both logistic regression and neural networks, a SVM sometimes gives a cleaner way of learning non-linear functions



SUPPORT VECTOR MACHINES (SVMs)

- Compared to both logistic regression and neural networks, a SVM sometimes gives a cleaner way of learning non-linear functions

The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.



SVMs Loss Function

- Alternative view of Logistic Regression. If we look at the loss (cost) function:

$$-(y \log h_{\theta}(x) + (1 - y) \log(1 - h_{\theta}(x)))$$

each example contributes a term like the one below to the overall cost function

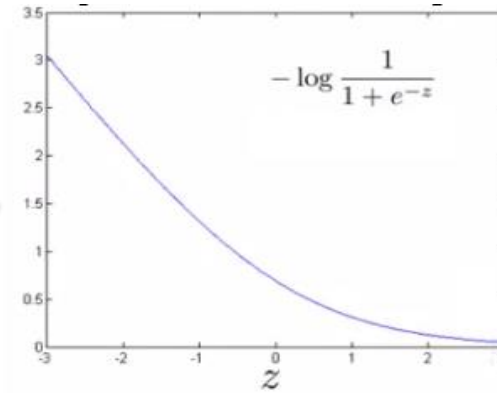
- For the overall cost function, we sum over all the training examples using the above function, and have a $1/m$ term. So we get:

$$-(y \log \frac{1}{1 + e^{-\theta^T x}} + (1 - y) \log(1 - \frac{1}{1 + e^{-\theta^T x}}))$$

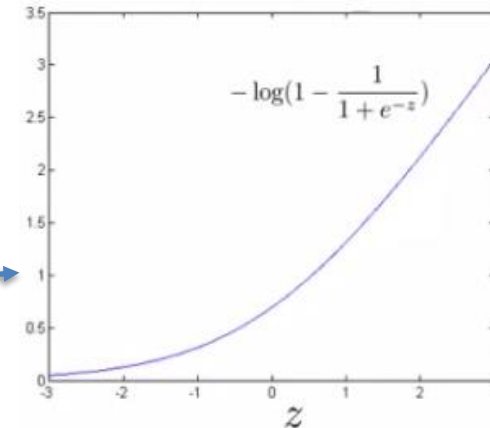
SVMs Loss Function

- So each training example contributes that term to the cost function for logistic regression.

- If $y = 1$ then only the first term in the objective matters. If we plot the functions vs. z we get the chart beside. If z is big, the cost is low and this is good!



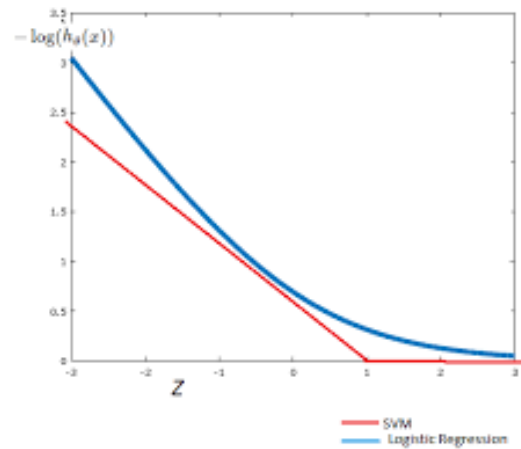
- But if z is 0 or negative the cost contribution is high.



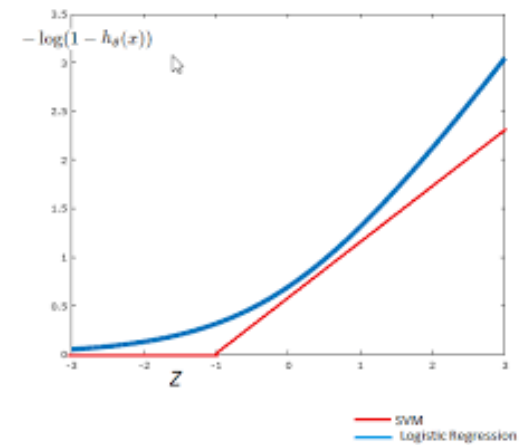
SVM Loss Function

- Instead of a curved line we create two straight lines (red) which act as an approximation to the logistic regression $y = 1$ and $y = 0$ functions
- So this is the new $y=1$ cost function that gives the SVM a computational advantage and an easier optimization problem. We call this function **cost1(z)**. Equivalently we call the $y=0$ function as **cost0(z)**.

Cost1



Cost0




SVM Loss Function

- In **Logistic Regression** the Loss Function was:

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \left(-\log h_{\theta}(x^{(i)}) \right) + (1 - y^{(i)}) \left(-\log(1 - h_{\theta}(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

- With **Support Vector Machines** the Loss Function now is:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$


- There is the regularizer term still, but something must change.

SVM Loss Function

1. Get rid of the $1/m$ terms. This is just a slightly different convention
 By removing $1/m$ we should get the same optimal values for $1/m$ is a constant, so should get same optimization e.g. say you have a minimization problem which minimizes to $u = 5$. If your cost function * by a constant, you still generates the minimal value. That minimal value is different, but that's irrelevant.

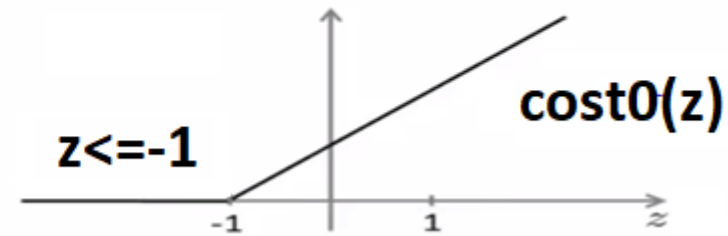
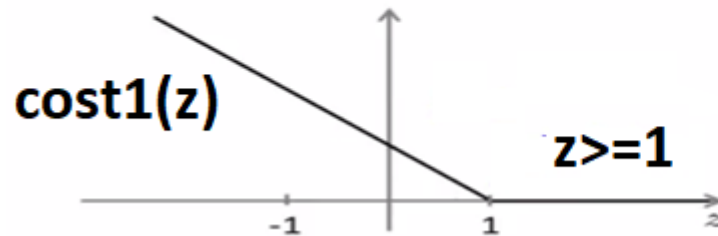
2. For logistic regression we had two terms; Training data set term (i.e. that we sum over m) = A. Regularization term (i.e. that we sum over n) = B. So we could describe it as $A + \lambda B$. Need some way to deal with the trade-off between regularization and data set terms. Set different values for λ to parametrize this trade-off. Instead if parameterize- tion this as $A + \lambda B$. For SVMs the convention is to use a different parameter called C. So do $CA + B$. If C were equal to $1/\lambda$ then the two functions ($CA + B$ and $A + \lambda B$) would give the same value

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

SVMs Hypothesis Function

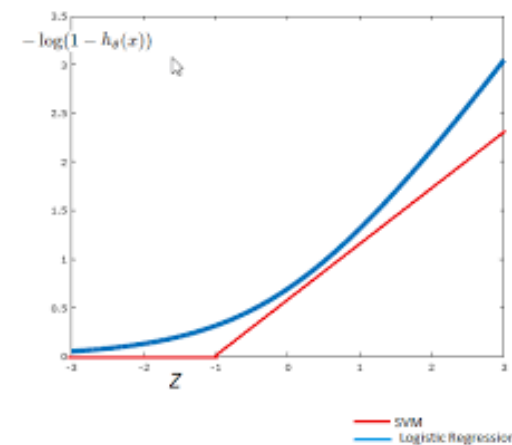
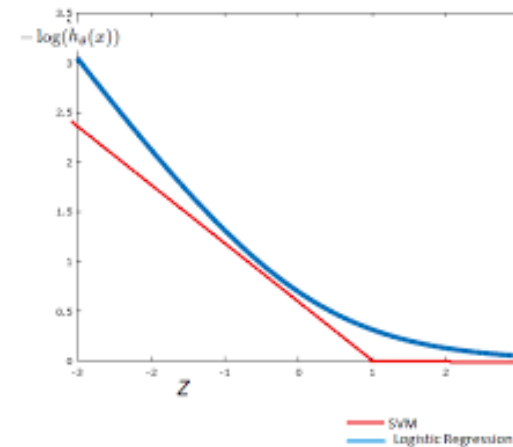
- Unlike logistic, $h_{\theta}(x)$ doesn't give us a probability, so the model is not stochastic. Instead we get a direct prediction of 1 or 0.
- So the $h_{\theta}(x) = 1$ when $\theta^T x \geq 0$
- whereas $h_{\theta}(x) = 0$ when $\theta^T x < 0$

$$\min_{\theta} C \sum_{i=1}^m \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$



So this is the new $y=1$ cost function
Gives the SVM a computational
advantage and an easier
optimization problem
We call this function **$\text{cost}_1(z)$**

- If $y = 0$ then only the second term matters. Same as before, if z is small then the cost is low and viceversa.



SVMs: Large Margin Intuition

- Sometimes people refer to SVM as **large margin classifiers**.
 - So the $h_{\theta}(x) = 1$ when $\theta^T x \geq 1$
 - whereas $h_{\theta}(x) = 0$ when $\theta^T x < -1$

Applications of SVM

- ✓ Face detection — classify between face and non-face areas on images
- ✓ Text and hypertext categorization
- ✓ Classification of images
- ✓ Bioinformatics — protein, genes, biological or cancer classification.
- ✓ Handwriting recognition
- ✓ Drug Discovery for Therapy

In recent times, SVM has played a very important role in **cancer detection** and its therapy with its application in classification.

Francesco **Pugliese**

