

Classical Data Analysis

Master in Big Data Solutions 2019-2020



Matteo Manca

matteo.manca@bts.tech

Today's Objective

- Review concepts from previous sessions
- Artificial Neural Networks
 - Theory
 - Artificial Neural Networks with Python

Contents

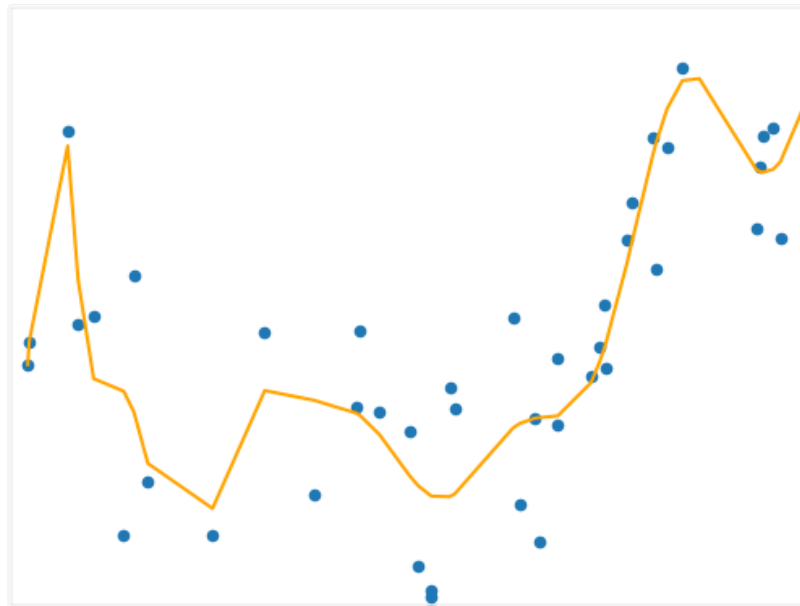
- Introduction to data analysis
- Linear Regression
- Logistic Regression
- Regression analysis (Polynomial, Ridge, Lasso, etc.)
- **Neural Networks**
- Support Vector Machines (SVM)
- Decision Trees
- Ensemble Methods
- Other Classifier (KNN, Naïve Bayes)
- K-Means
- PCA
- Hierarchical Clustering
- Recommender Systems

Previous session: Polynomial Regression

Polynomial regression

Limitation of linear Regression

- **Linear regression** requires the **relation** between the dependent variable and the independent variable to be **linear**.

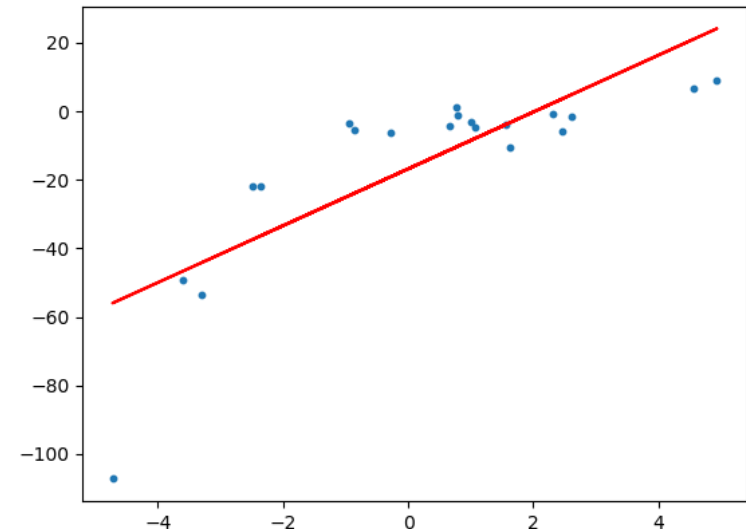
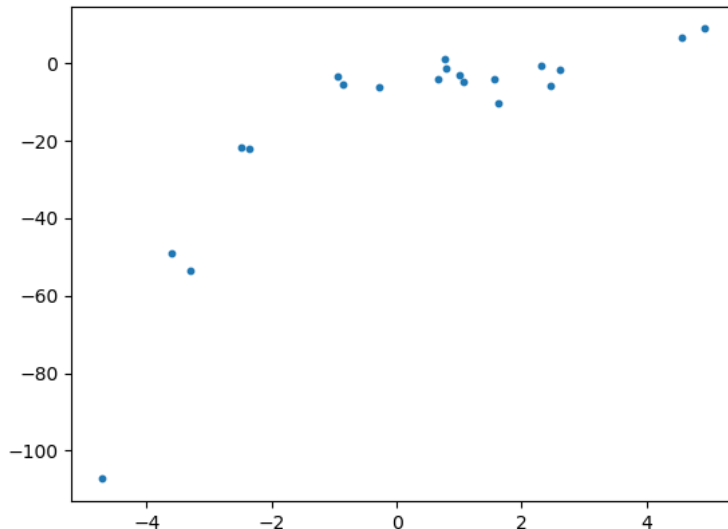


Can linear models be used to fit non-linear data?

Polynomial regression

Why Polynomial Regression?

Linear regression model

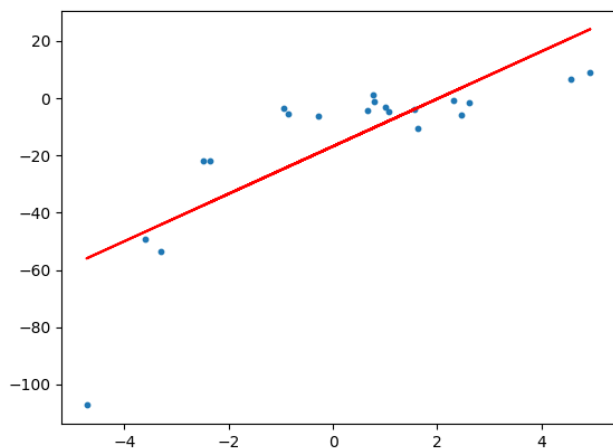


The model is unable to capture the patterns in the data. This is an example of **under-fitting**.

Polynomial regression

Why Polynomial Regression?

Linear regression model



How to overcome **under-fitting**?



Increase the complexity of the model.
We need to add more features to the data

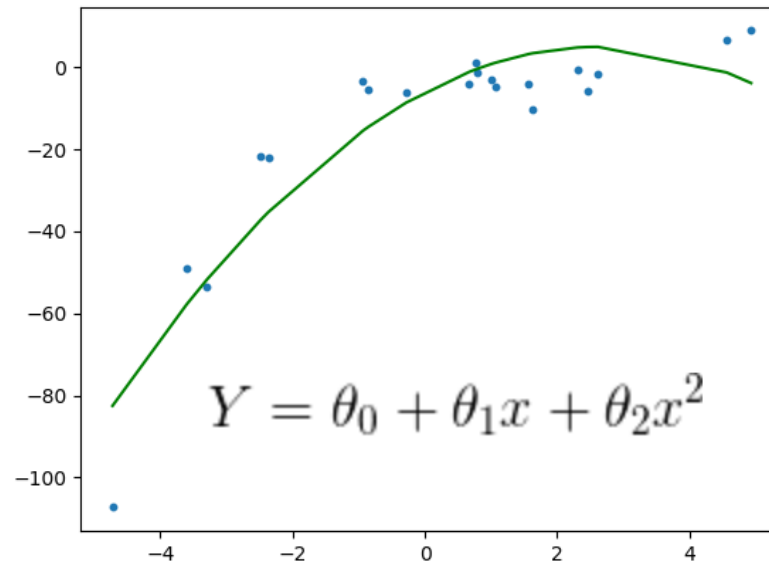
$$Y = \theta_0 + \theta_1 x \quad \longrightarrow \quad Y = \theta_0 + \theta_1 x + \theta_2 x^2$$

This is still considered to be a **linear model** as the coefficients/weights associated with the features are still linear. x^2 is only a new feature.

Polynomial regression

Why Polynomial Regression?

Linear regression model 2nd degree polynomial

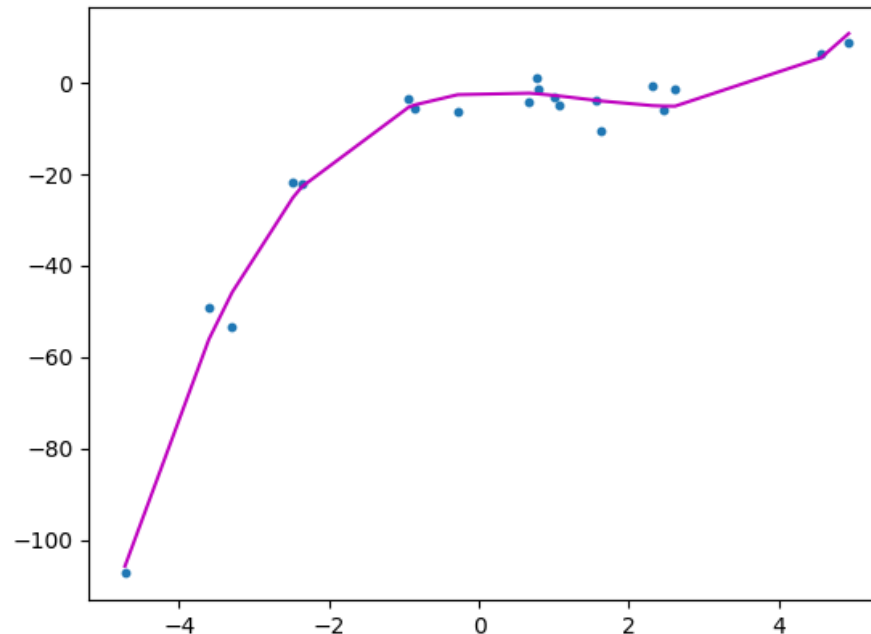


It is clear from the plot that the quadratic curve (2nd degree polynomial) is able to fit the data better than the linear line.

Polynomial regression

Why Polynomial Regression?

Linear regression model polynomial of degree 3

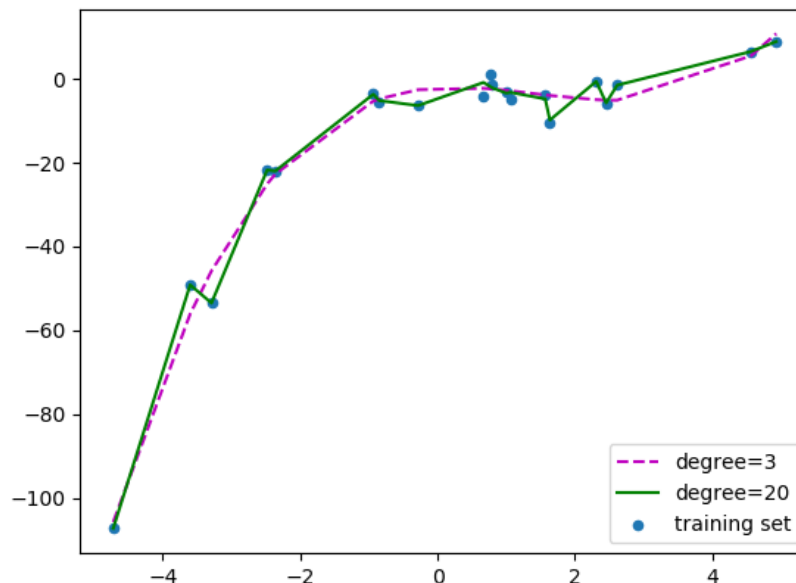


It passes through more data points than the quadratic and the linear plots.

Polynomial regression

Why Polynomial Regression?

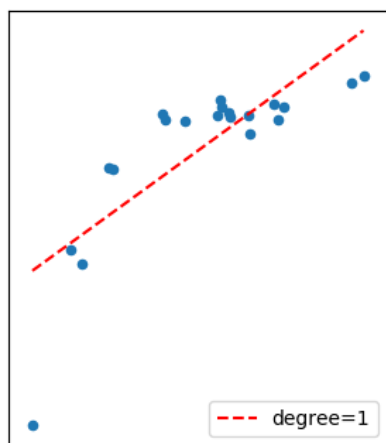
Linear regression model polynomial of degree 20



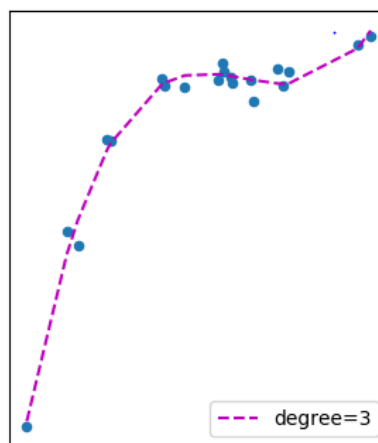
It passes through more data points but here we have a problem: **Overfitting**, i.e., the model is also capturing the noise in the data and even if it works well on the training set it will fail to generalize (the performance in the test set will be poor).

Bias vs Variance trade-off

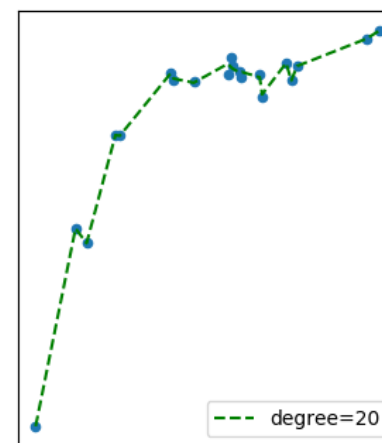
- **Bias:** error due to the model's simplistic assumptions in fitting the data. A **high bias** means that the model is unable to capture the patterns in the data and this results in **underfitting**.
- **Variance:** error due to the complex model trying to fit the data. **High variance** results in **over-fitting** the data. Variance refers to how much the model is dependent on the training data.



Underfit
High Bias
Low Variance

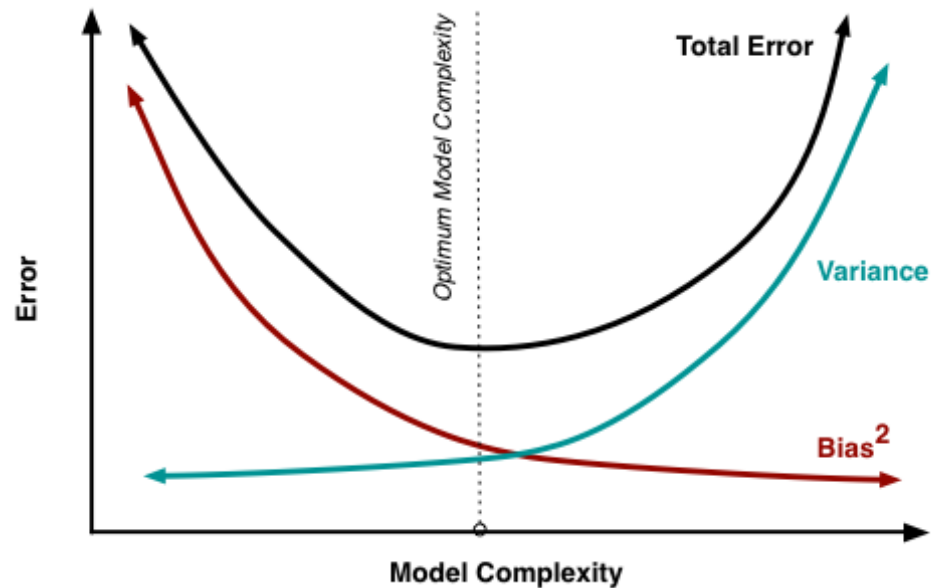


Correct Fit
Low Bias
Low Variance



Overfit
Low Bias
High Variance

Bias vs Variance trade-off



Regularization: Ridge, Lasso and Elastic Net

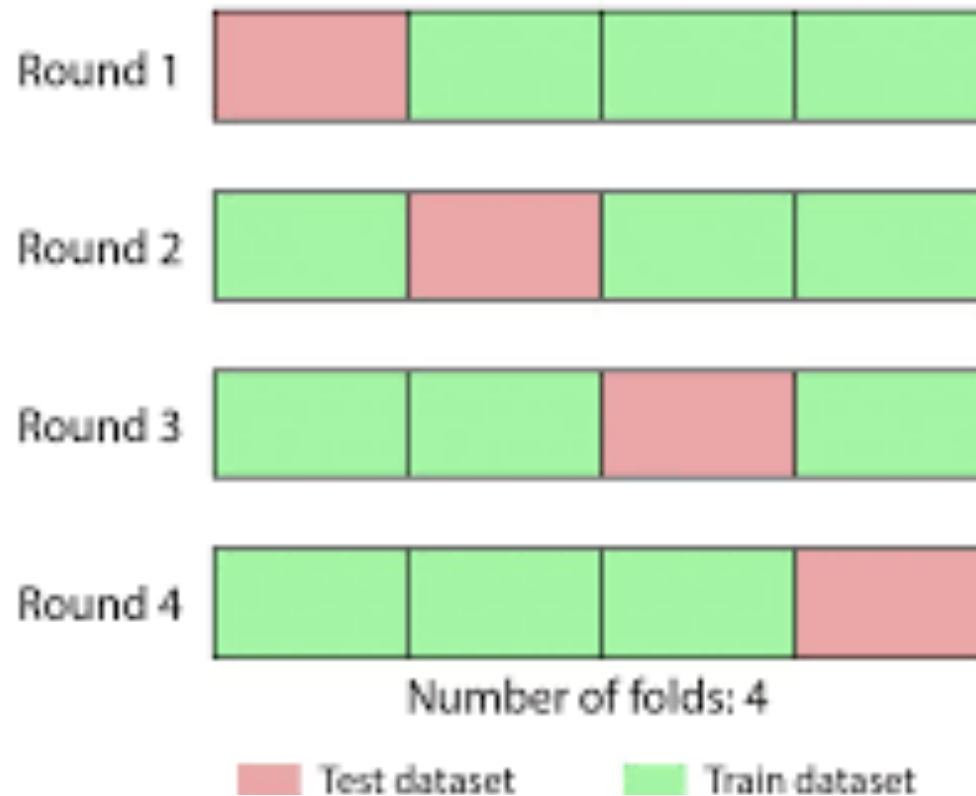
Regularization

How to address overfitting

- In classical linear regression if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This might lead to **overfitting** in small datasets.
- Reduce Number of feature (Manually or Model selection algorithm)
- **Regularization**
- Collect More data
- **Cross Validation**
- Ensambling (Combine prediction from multiple separate models)

Regularization

Cross Validation



Regularization

How to address overfitting

- In classical linear regression if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This might lead to **overfitting** in small datasets.
- Regularization techniques consist of modifying the cost function, adding a penalty term, to restrict the values of our coefficients.
- Regularization techniques:
 - **Lasso**, also called L1-Norm
 - **Ridge**, also called L2-Norm
 - **Elastic Net**: is a combination of Lasso and Ridge

Regularization

LASSO

- Adds an additional term to the cost function, adding the sum of the coefficient values (the L-1 norm) multiplied by a constant lambda.

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

- Lambda set the coefficients of the bad predictors mentioned above 0 (**feature selection**).
- If lambda=0, we effectively have no regularization.
- Large lambda leads coefficients to 0.

Regularization

RIDGE

- Sums the **squares of coefficient** values (the L-2 norm) and multiplies it by some constant lambda.

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{\operatorname{argmin}} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

- will decrease the values of coefficients, but is unable to force a coefficient to exactly 0 (**No feature selection**).
- If lambda=0, we effectively have no regularization
- It will also select groups of colinear features (grouping effect)

Analysis of both Lasso and Ridge regression has shown that neither technique is consistently better than the other; try both methods to determine which to use!

Regularization

Elastic Net

- Includes both L-1 and L-2 norm regularization terms.

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}} (\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1).$$

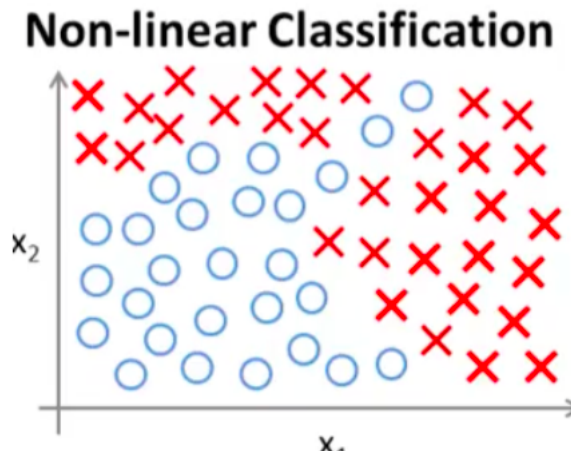
- It seems to have the predictive power better than Lasso, while still performing feature selection. We therefore get the best of both worlds, performing feature selection of Lasso with the feature-group selection of Ridge.

Artificial Neural Networks

Logistic Regression

Why do we need another learning algorithm?

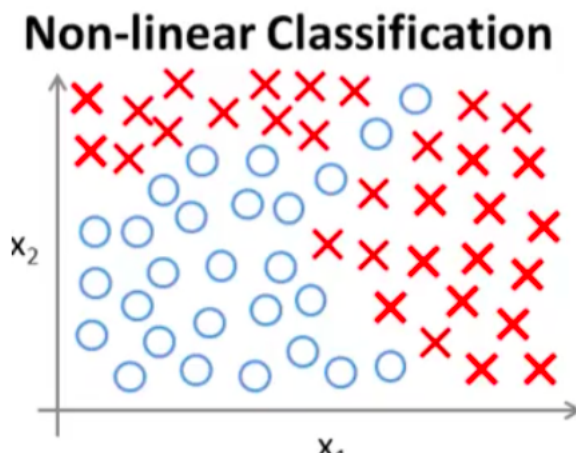
- Linear Regression
 - Logistic regression
-
- Example of machine learning problem where we need complex non-linear hypotheses



Logistic Regression

Why do we need another learning algorithm?

Example of machine learning problem where we need complex non-linear hypotheses



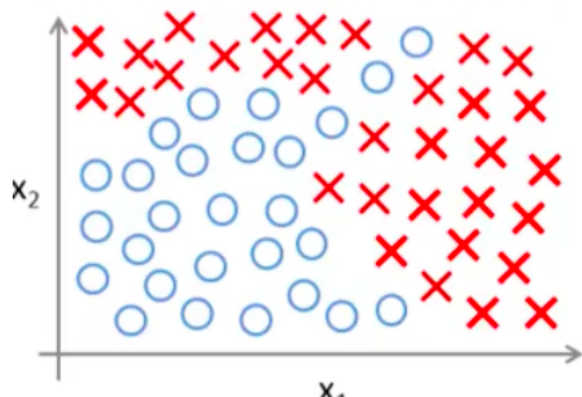
To apply logistic regression to this problem we have to use a lot of non linear features like the following sigmoid function

$$g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1 x_2 + \Theta_4 x_1^2 x_2 + \dots)$$

Logistic Regression

Why do we need another learning algorithm?

Non-linear Classification



To apply logistic regression to this problem we have to use a lot of non linear features

$$g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1 x_2 + \Theta_4 x_1^2 x_2 + \dots)$$

Logistic regression method tends to work well when we have only few features, because in that case we can add all polynomial terms.

In many machine learning problems we will have a lot more feature than just two.

Logistic Regression

Why do we need another learning algorithm?

To apply logistic regression to this problem we have to use a lot of non linear features

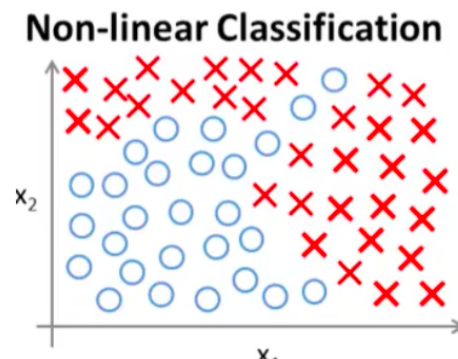
$$g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1 x_2 + \Theta_4 x_1^2 x_2 + \dots)$$



Many features → Complex Model



- High Risk of overfitting
- computation would be too much expensive.



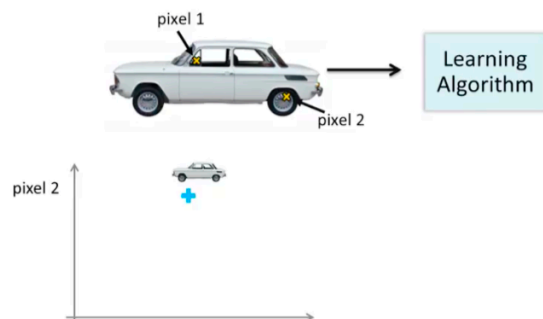
Logistic Regression

Example of ML problem with many examples: Computer Vision

- We want to train a classifier able to examine an image and tell us if the image is a car or not



An image is a matrix of pixels

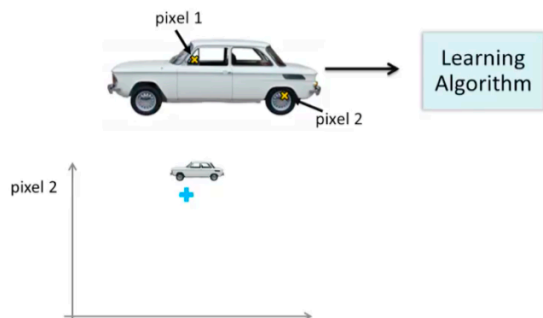


Imagine to plot a object of the dataset using a couple of pixel locations (pixel 1 and pixel 2)

Logistic Regression

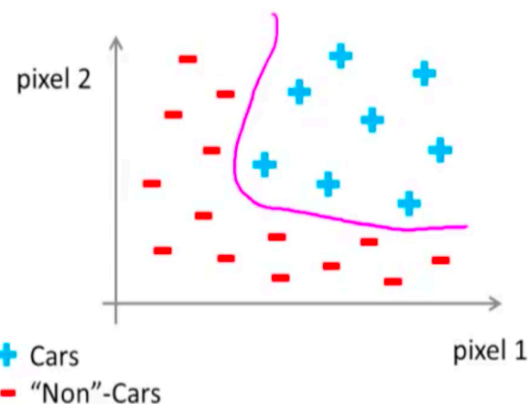
Example of ML problem with many examples: Computer Vision

- We want to train a classifier able to examine an image and tell us if the image is a car or not



Imagine to plot a object of the dataset using a couple of pixel locations (pixel 1 and pixel 2)

Plot all objects

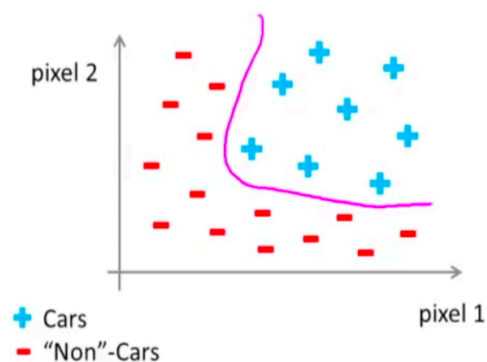


if we use a 50×50 pixel we have a vector of 2500 pixel intensities. If we try to learn a non-linear hypothesis by including all the quadratic terms, we obtain about 3 million features and it would be too large.

Logistic Regression

Example of ML problem with many examples: Computer Vision

Plot all objects

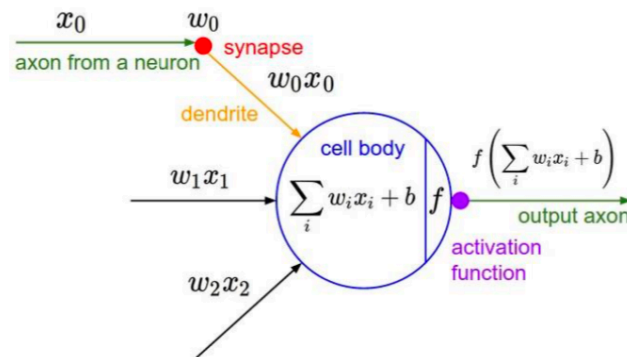
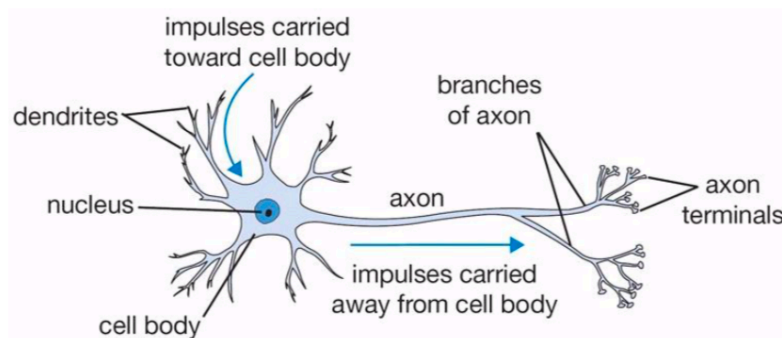


if we use a 50×50 pixel we have a vector of 2500 pixel intensities. If we try to learn a non-linear hypothesis by including all the quadratic terms, we obtain about 3 million features and it would be too large.

Logistic regression with the addition of quadratic (or cubic) features is not a good idea to learn a complex non-linear hypothesis when the number of features is large because we end up with too many features.

Artificial Neural Network

- Neural Network is a learning algorithm inspired on how human brain works and it is very useful to approach different machine learning problems.

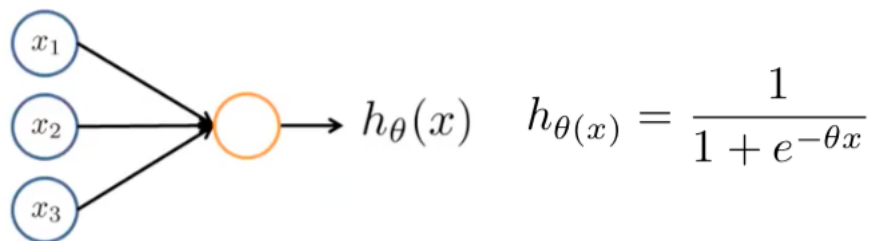


- Neurons pass information through the synapses
- Synapses between neurons adapt
- Clusters of neurons learn to perform computations.

Artificial Neural Network

Model Representation : Logistic Unit

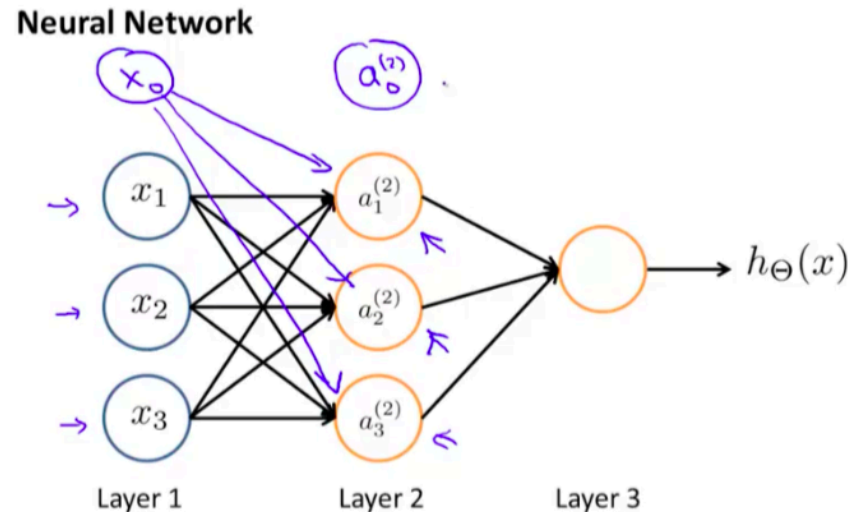
- We model a neuron as a simple logistic unit



- In neural networks the sigmoid function (or logistic function) is also called **Activation function** and the parameters θ are called weights of the model.

Artificial Neural Network

Model Representation : Logistic Unit

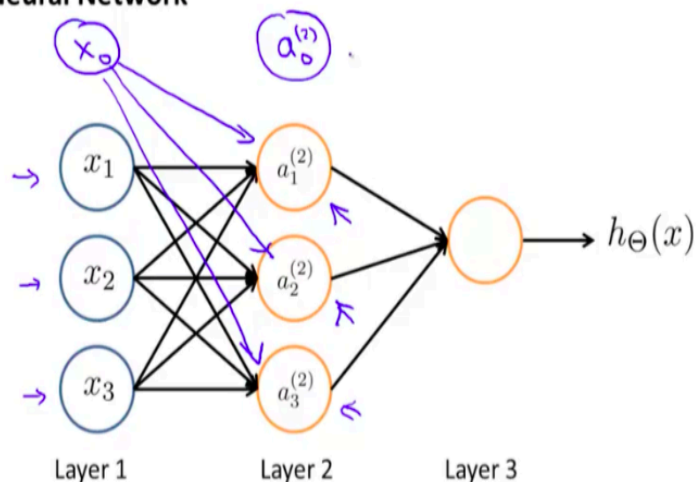


- a neural network is a proof of different neurons models that interact each other

Artificial Neural Network

Model Representation :Logistic Unit

Neural Network

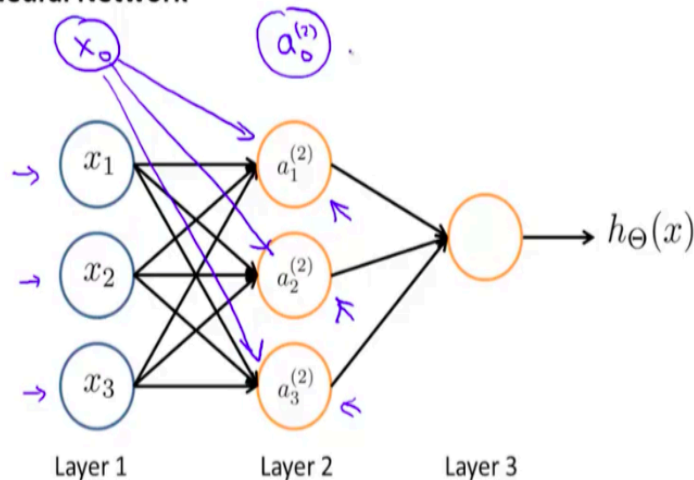


- 3 neurons $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$ more the bias unit $a_0^{(2)}$;
- input layer (layer 1): in this layer we input the features x_1 , x_2 and x_3 (of our training set);
- hidden layer (layer 2): it contains values that we don't observe in the training set;
- output layer (layer 3): it has a neuron / units (it can have more units) that output the final value computed by the hypothesis.

Artificial Neural Network

Model Representation :Logistic Unit

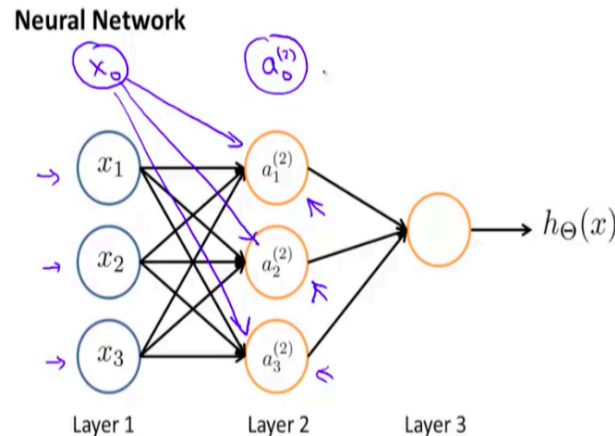
Neural Network



- $a_i^{(j)}$: Activation of unit i in layer j (i.e., the value that is computed by and that is the output of a specify unit);
- $\Theta^{(j)}$: matrix of weights that controls the function mapping from layer j to layer $j + 1$.

Artificial Neural Network

Model Representation: Computation



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

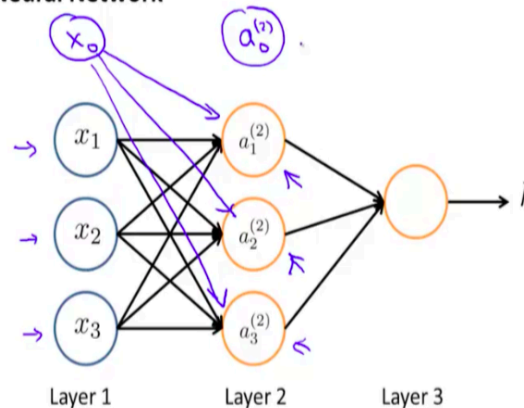
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} x_3^{(2)})$$

Artificial Neural Network

Model Representation: Computation

Neural Network



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} x_3^{(2)})$$

If a network has S_j units in layer j , S_{j+1} units in layer $j + 1$, then $\Theta^{(j)}$ is a $S_{j+1} \times S_j + 1$ matrix:

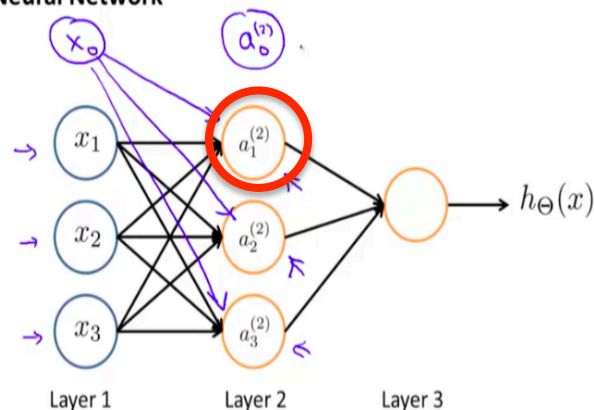
$$\Theta^{(j)} \in \mathbb{R}^{S_{j+1} \times S_j + 1}$$

$\Theta^{(1)}$ is 3×4 matrix, so $\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$.

Artificial Neural Network

Model Representation: Computation

Neural Network



$$\Theta^{(j)} \in \mathbb{R}^{S_{j+1} \times S_j + 1}$$

- $J=1$
- $S_{j+1} = S_2 = 3$
- $S_j = S_1 = 4$ (we are considering also x_0)
- $\Theta^{(1)}$ is 3×4 matrix, so $\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

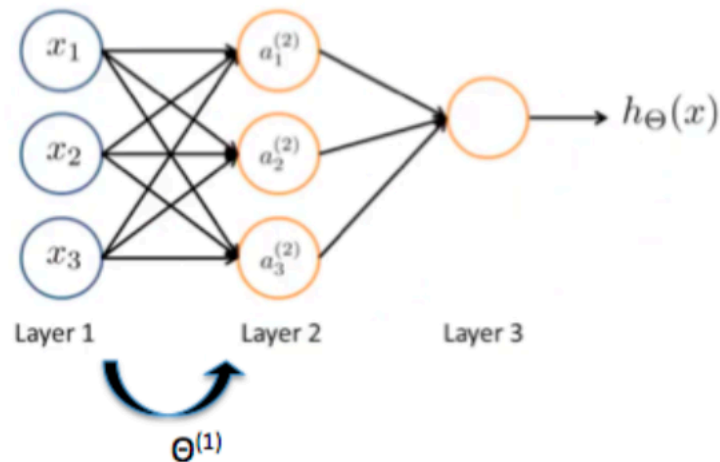
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} x_3^{(2)})$$

To compute the activation function of a neuron $a_1^{(2)}$ we need a 1×4 vector. Since in the layer 2 there are 3 neurons we need 3 1×4 vectors, i.e., a 3×4 matrix

Artificial Neural Network

Model Representation: Computation



$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

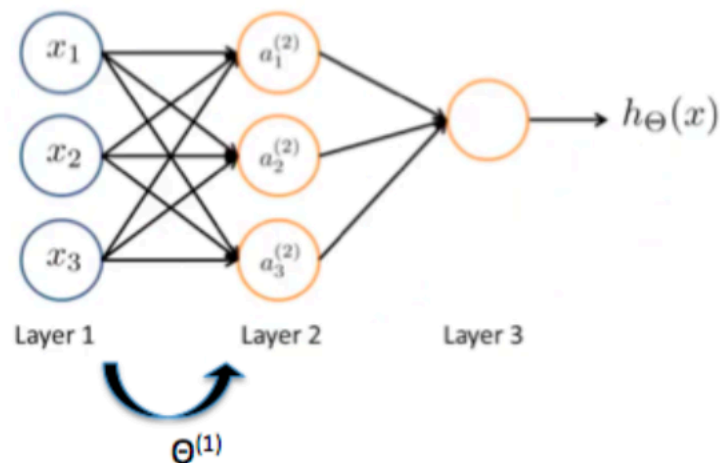
$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} x_3^{(2)})$$

Hypothesis function

The computation of $h_{\Theta}(x)$ is called **forward propagation** because it starts with the activation of the input units, it propagates that to the hidden layer computing the activation function of this layer and the propagation follows with the activation of the output layer.

Artificial Neural Network

Model Representation



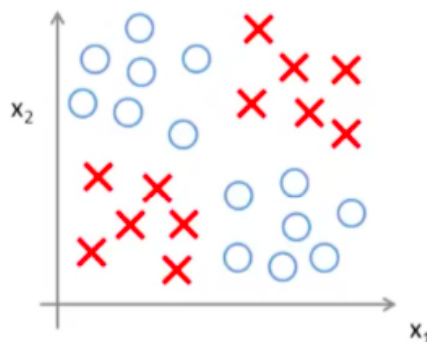
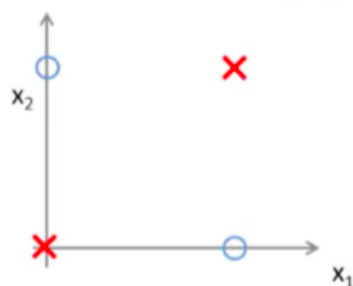
Architecture of a Neural Network: how the different neurons of a network are connected each others.

Artificial Neural Network

Example and Intuition

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).



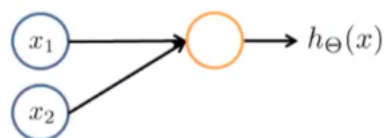
✗ Positive ($y=1$)
○ Negative ($y=0$)

- Input features x_1 and x_2 that are binary values
- On the left we have a simplified version of a more complex (on the right) learning problem.
- We would like to learn a non-linear function (decision boundary) to separate positive and negative examples.

Artificial Neural Network

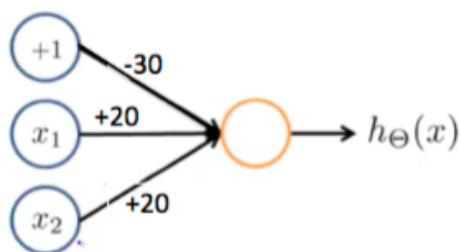
Example and Intuition

Let's start building a neural network that fits the **AND logic function** at first and then some other logic functions



- $x_1, x_2 \in \{0, 1\}$
- $y = x_1 \wedge x_2$

Let's add the bias unit and assign some values to the weights (parameters) of the network



- $\Theta_{10}^{(1)} = -30$
- $\Theta_{11}^{(1)} = +20$
- $\Theta_{12}^{(1)} = +20$

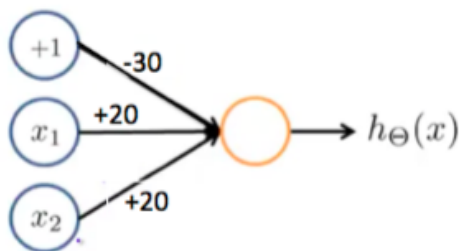
AND Hypothesis function

$$h_{\Theta}(x) = g(-30 + 20x_1 + 20x_2)$$

Artificial Neural Network

Example and Intuition

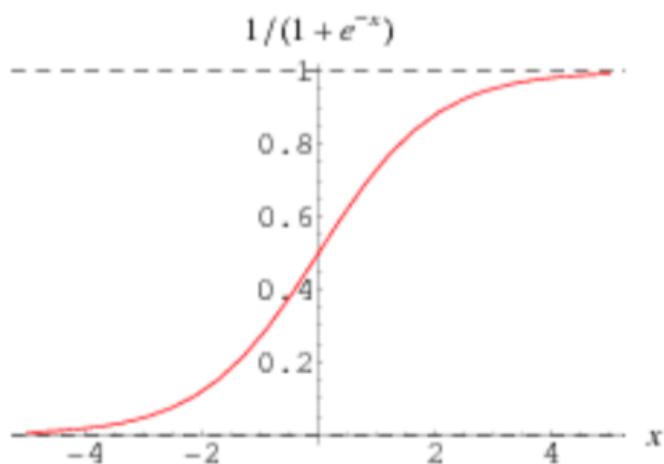
AND logic function



- $\Theta_{10}^{(1)} = -30$
- $\Theta_{11}^{(1)} = +20$
- $\Theta_{12}^{(1)} = +20$

AND Hypothesis function

$$h_{\Theta}(x) = g(-30 + 20x_1 + 20x_2)$$

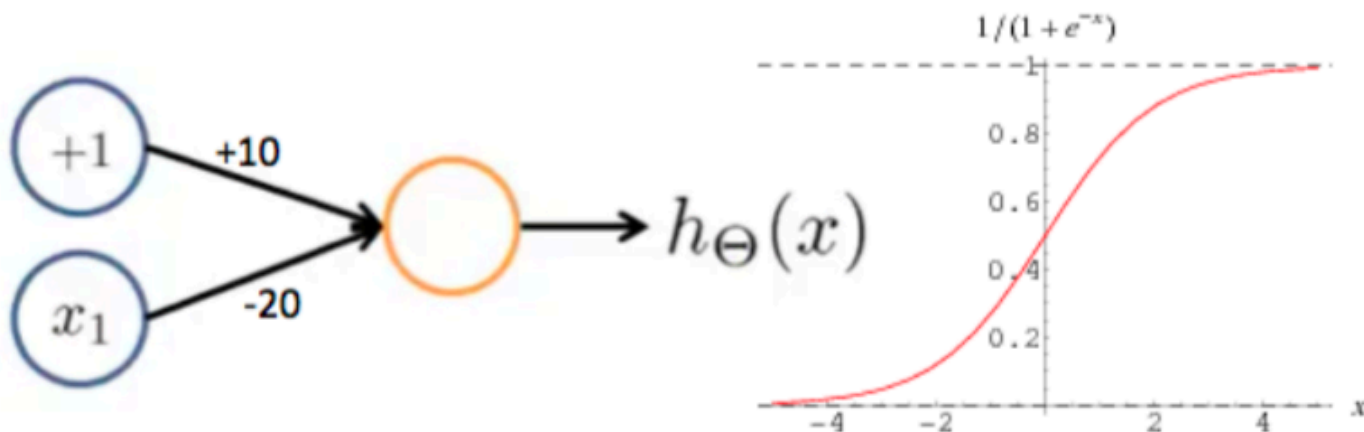


x_1	x_2	$h_{\Theta}(x) = g(-30 + 20x_1 + 20x_2)$
0	0	$g(-30) \approx 0$
0	1	$g(-30 + 20) \approx 0$
1	0	$g(-30 + 20) \approx 0$
1	1	$g(-30 + 20 + 20) \approx 1$

Artificial Neural Network

Example and Intuition

NOT logic function

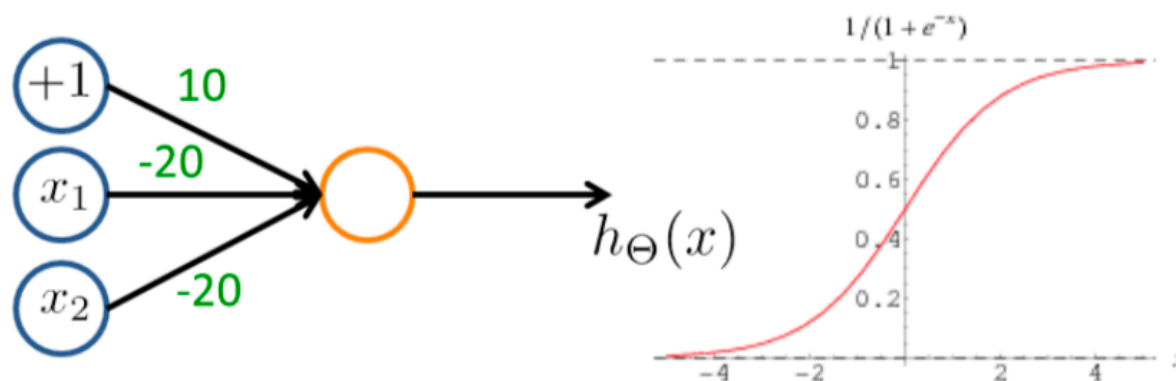


x_1	$h_{\Theta}(x) = g(10 - 20x_1)$
0	$g(10) \approx 1$
1	$g(+10 - 20) \approx 0$

Artificial Neural Network

Example and Intuition

(NOT) AND (NOT) logic function

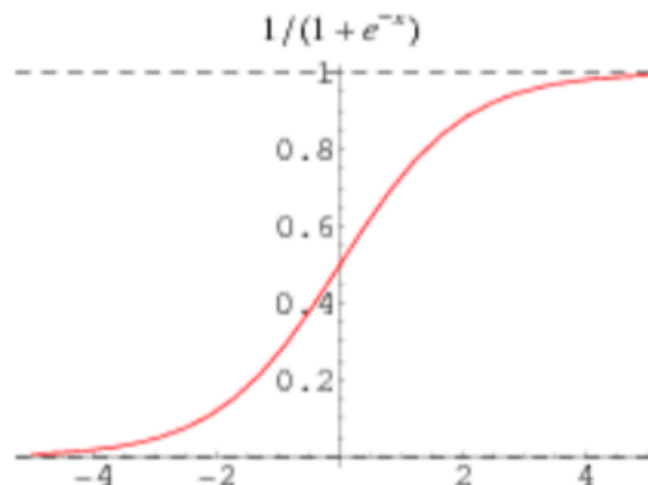
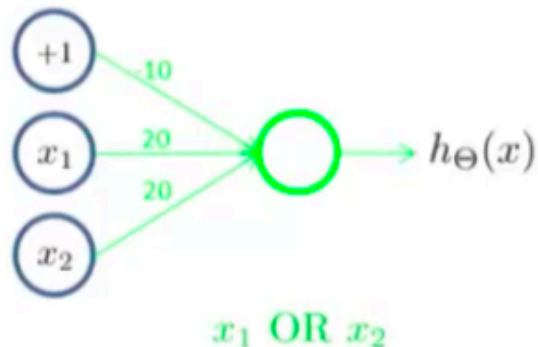


x_1	x_2	$h_{\Theta}(x) = g(10 - 20x_1 - 20x_2)$
0	0	$g(10) \approx 1$
0	1	$g(+10 - 20) \approx 0$
1	0	$g(+10 - 20) \approx 0$
1	1	$g(+10 - 20 - 20) \approx 0$

Artificial Neural Network

Example and Intuition

OR logic function



x_1	x_2	$h_{\Theta}(x) = g(-10 + 20x_1 + 20x_2)$
0	0	$g(-10) \approx 0$
0	1	$g(-10 + 20) \approx 1$
1	0	$g(-10 + 20) \approx 1$
1	1	$g(-10 + 20 + 20) \approx 1$

Artificial Neural Network

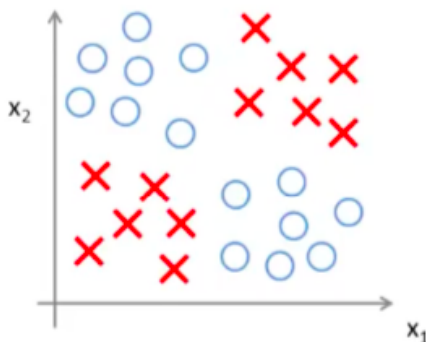
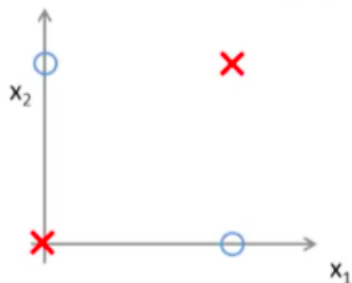
Example and Intuition

XNOR logic function

we want to put together the neural networks we have seen previously in order to compute $x_1 \text{ XNOR } x_2$.

Non-linear classification example: XOR/XNOR

x_1, x_2 are binary (0 or 1).

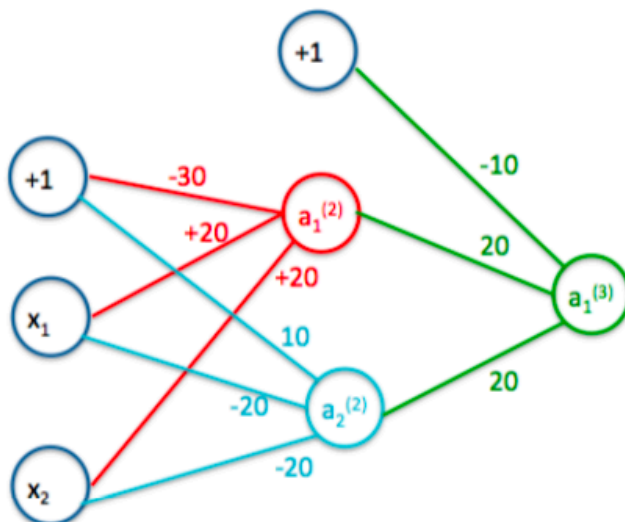
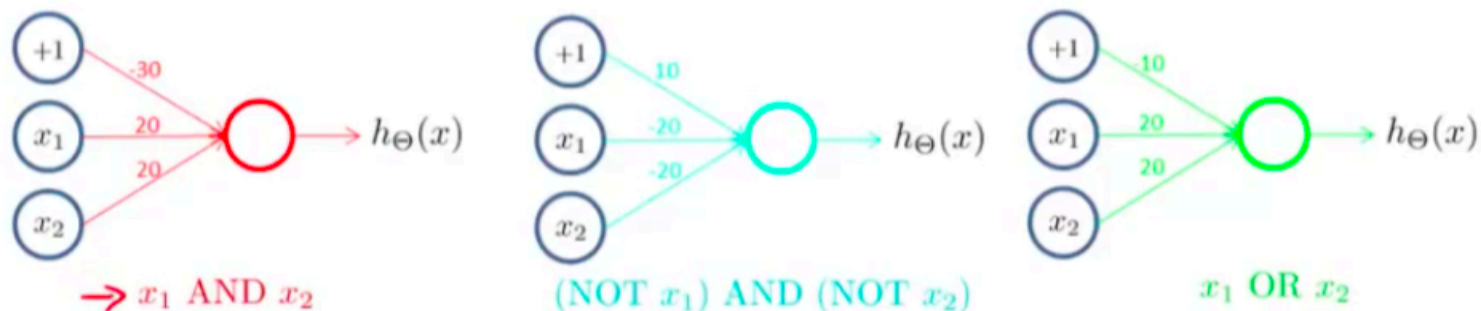


- ✗ Positive ($y=1$)
- Negative ($y=0$)

Artificial Neural Network

XNOR logic function

we want to put together the neural networks we have seen previously in order to compute x_1 XNOR x_2 .



x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Artificial Neural Network

Multiclass Classification

- We want to recognize four categories of objects (given an image): pedestrian, car, motorcycle, truck.
- Network with four output units

Multiple output units: One-vs-all.



Pedestrian



Car



Motorcycle



Truck



$$h_{\Theta}(x) \in \mathbb{R}^4$$

Want $h_{\Theta}(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_{\Theta}(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.
 when pedestrian when car when motorcycle

Artificial Neural Network

Neural Networks: learning Cost function

The cost function for a neural network can be defined as a generalization of the logistic regression cost function:

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k) \right] \\ + \frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} (\Theta_{ij}^{(l)})^2$$

- L is the total number of layer in the considered network
- S_l is the number of units in layer l.

Artificial Neural Network

Neural Networks: learning Cost function

$$J(\Theta) = -\frac{1}{m} \left[\sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_{\Theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_{\Theta}(x^{(i)}))_k) \right] \\ + \frac{\lambda}{2m} \sum_{l=1}^L \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} (\Theta_{ij}^{(l)})^2$$

- As in the case of logistic regression we need to **minimize the cost function** in order to find the parameters (weights) that are the best decision boundary for our data.
- We use **backpropagation algorithm and gradient descent**.

Artificial Neural Network

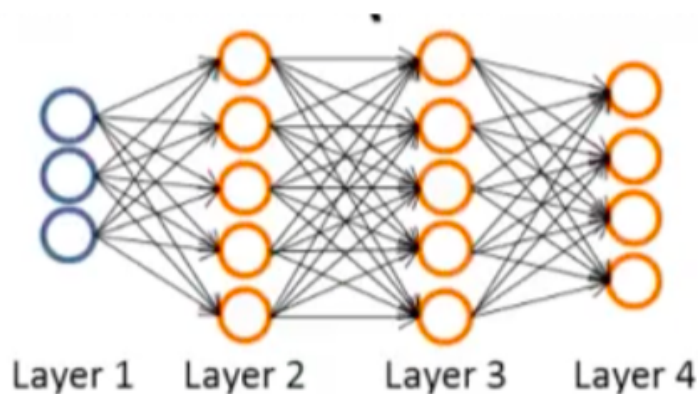
Neural Networks: learning Cost function

- $\delta^{(l)}_j$: **Error** of node j in layer l (it can be defined as the difference between the hypothesis output $h_{\theta}(x)$ and the real output value y)
- The backpropagation algorithm is based on the intuition that for each node j of the neural network, the value $\delta^{(l)}_j$, that represents the error of the node j in layer l is computed.

Artificial Neural Network

Neural Networks: learning Cost function

- $\delta^{(l)}_j$: **Error** of node j in layer



$$\delta^{(4)} = a^{(4)} - y$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} \cdot * g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} \cdot * g'(z^{(2)})$$

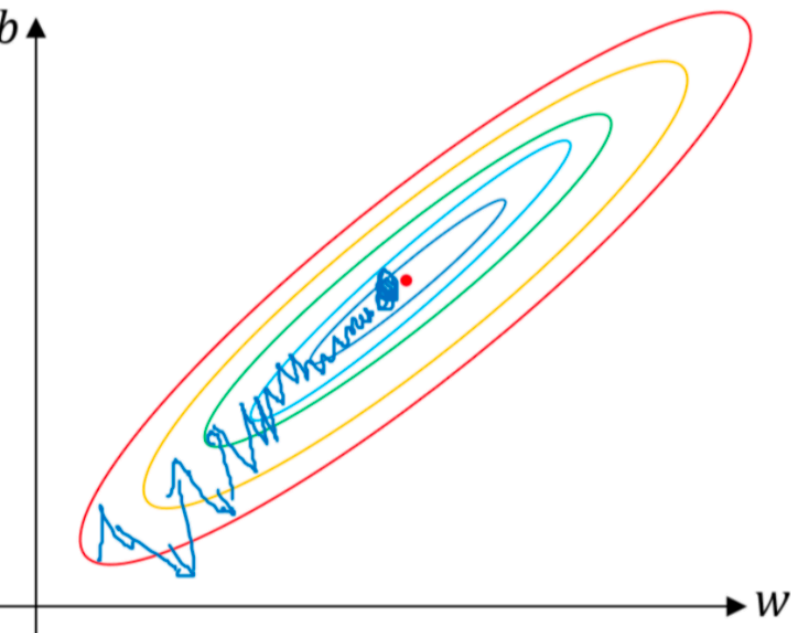
The error of the output units is straightforward to compute (by comparing to the desired output). The error of the weights in the hidden units is derived based on their contribution to the output.

Artificial Neural Network

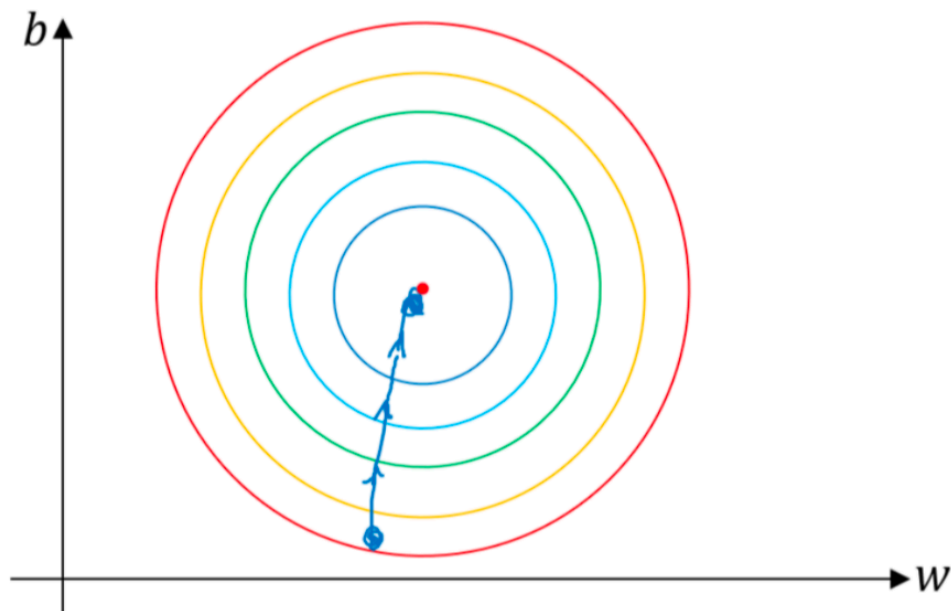
Gradient Descent

If the data is not normalized the algorithm will take longer to converge to the minima.

Unnormalized



Normalized



Artificial Neural Network

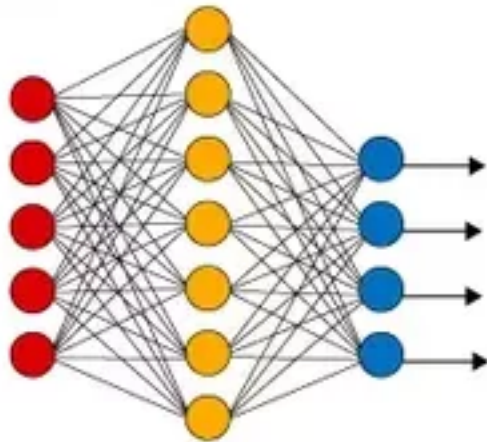
Activation Functions

- So far we are using sigmoid, but in some cases other functions can be a lot better.
- **Tanh** activation function (usually works better than sigmoid activation function for hidden units)
- **Relu**: generally used in the hidden layers (much faster when compared to sigmoid or tanh)

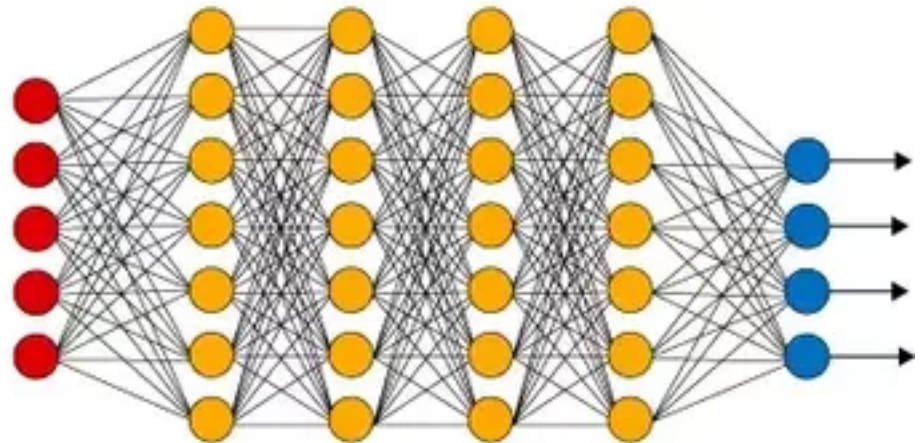
Artificial Neural Network

Neural Networks VS Deep Learning

Simple Neural Network



Deep Learning Neural Network



● Input Layer

● Hidden Layer

● Output Layer

Resources

- 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.

