**Summer Training Report**

**A Project Report submitted in partial fulfilment of**

**the requirements for the award of**

**Bachelor of Engineering**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**:

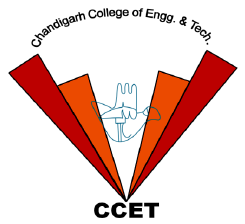
**Priyadarshini**

**(Roll no.: LCO17373)**

**Training Under**

Andrew Ng (Deeplearning.ai)

Coursera



**CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY** **(DEGREE WING)**

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University,

Sector-26, Chandigarh. PIN-160019

## Weekly Activity Reports

**Week 1: Introduction of Deep Learning** - Able to explain the major trends driving the rise of deep learning, and understand where and how it is applied today.

**1.1 Neural network and deep learning**

I think that AI is the new electricity. Starting about 100 years ago, the electrification of our society transformed every major industry, every ranging from transportation, manufacturing, to healthcare, to communications and many more.

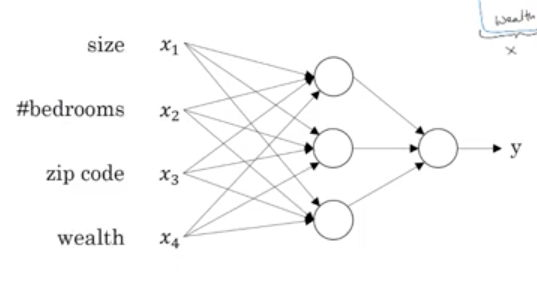
Deep Learning, refers to training Neural Networks, sometimes very large Neural Networks.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance. So, that's a basic neural network. In turns out that as you build out your own neural networks, you probably find them to be most useful, most powerful in supervised learning incentives, meaning that you're trying to take an input x and map it to some output y, like we just saw in the housing price prediction example. In the next video let's go over some more examples of supervised learning and some examples of where you might find your networks to be incredibly helpful for your applications as well.

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculate the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives.[107] The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network.

## Neural Network

Looks like this  
  
Multidimensional input goes to the neurons in first layer. Output of first layer neurons goes to second layer, and so on. Housing problem is structured, ads clicked or not is structured. Audio, image, and are unstructured. Deep learning is taking off now because there is lot of data to train on, and computing power to perform this training.

**1.2 Supervised Learning with Neural Networks**

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances.

**Steps**

In order to solve a given problem of supervised learning, one has to perform the following steps:

* Determine the type of training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set. In the case of handwriting analysis, for example, this might be a single handwritten character, an entire handwritten word, or an entire line of handwriting.
* Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
* Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.
* Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.
* Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.
* Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

**1.3 Applications**

1 Automatic speech recognition

2 Image recognition

3 Visual art processing

4 Natural language processing

5 Drug discovery and toxicology

6 Customer relationship management

7 Recommendation systems

8 Bioinformatics

9 Medical Image Analysis

10 Mobile advertising

11 Image restoration

12 Financial fraud detection

13 Military

**1.4 Structured Data and Unstructured Data**

Structured data is most often categorized as quantitative data, a and it's the type of data most of us are used to working with. Think of data that fits neatly within fixed fields and columns in relational databases and spreadsheets.

Structured data is highly organized and easily understood by machine language. Those working within relational databases can input, search, and manipulate structured data relatively quickly. This is the most attractive feature of structured data.

Unstructured data is most often categorized as qualitative data, and it cannot be processed and analyzed using conventional tools and methods.

Examples of structured data include names, dates, addresses, credit card numbers, stock information, geolocation, and more. Unstructured data is difficult to deconstruct because it has no pre-defined model, meaning it cannot be organized in relational databases. Instead, non-relational, or NoSQL databases, are best fit for managing unstructured data.

Examples of unstructured data include text, video, audio, mobile activity, social media activity, satellite imagery, surveillance imagery – the list goes on and on.

**Week 2: Neural Networks Basic -** Learnt to set up a machine learning problem with a neural network mindset. Learned to use vectorization to speed up your models.

**2.1 Binary Classification**

**Binary classification** is the task of [classifying](https://en.wikipedia.org/wiki/Statistical_classification) the elements of a given [set](https://en.wikipedia.org/wiki/Set_(mathematics)) into two groups (predicting which group each one belongs to) on the basis of a [classification rule](https://en.wikipedia.org/wiki/Classification_rule). Contexts requiring a decision as to whether or not an item has some [qualitative property](https://en.wikipedia.org/wiki/Qualitative_property), some specified characteristic, or some typical binary classification include:

* [Medical testing](https://en.wikipedia.org/wiki/Medical_test) to determine if a patient has certain disease or not – the classification property is the presence of the disease.
* A "pass or fail" [test method](https://en.wikipedia.org/wiki/Test_method) or [quality control](https://en.wikipedia.org/wiki/Quality_control) in factories, i.e. deciding if a specification has or has not been met – a [go/no go](https://en.wikipedia.org/wiki/Go/no_go) classification.
* [Information retrieval](https://en.wikipedia.org/wiki/Information_retrieval), namely deciding whether a page or an article should be in the [result set](https://en.wikipedia.org/wiki/Result_set) of a search or not – the classification property is the relevance of the article, or the usefulness to the user.

Binary classification is [dichotomization](https://en.wikipedia.org/wiki/Dichotomization) applied to practical purposes, and in many practical binary classification problems, the two 2 groups are not symmetric – rather than overall accuracy, the relative proportion of different [types of errors](https://en.wikipedia.org/wiki/Type_I_and_type_II_errors) is of interest. For example, in medical testing, a [false positive](https://en.wikipedia.org/wiki/False_positives_and_false_negatives#False_positive_error) (detecting a disease when it is not present) is considered differently from a [false negative](https://en.wikipedia.org/wiki/False_positives_and_false_negatives#False_negative_error) (not detecting a disease when it is present).

**2.2 Logistic regression**

In[statistics](https://en.wikipedia.org/wiki/Statistics), the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Logistic regression is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that in its basic form uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), although many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), **logistic regression** (or **logit regression**) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (a form of [binary regression](https://en.wikipedia.org/wiki/Binary_regression)).

Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labeled "0" and "1". In the logistic model, the [log-odds](https://en.wikipedia.org/wiki/Log-odds) (the [logarithm](https://en.wikipedia.org/wiki/Logarithm) of the [odds](https://en.wikipedia.org/wiki/Odds)) for the value labeled "1" is a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of one or more [independent variables](https://en.wikipedia.org/wiki/Independent_variable) ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a [continuous variable](https://en.wikipedia.org/wiki/Continuous_variable) (any real value). The corresponding [probability](https://en.wikipedia.org/wiki/Probability) of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The [unit of measurement](https://en.wikipedia.org/wiki/Unit_of_measurement) for the log-odds scale is called a [*logit*](https://en.wikipedia.org/wiki/Logit), from ***log****istic un****it***, hence the alternative names. Analogous models with a different [sigmoid function](https://en.wikipedia.org/wiki/Sigmoid_function) instead of the logistic function can also be used, such as the [probit model](https://en.wikipedia.org/wiki/Probit_model" \o "Probit model); the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a *constant rate*, with each independent variable having its own parameter; for a binary dependent variable this generalizes the [odds ratio](https://en.wikipedia.org/wiki/Odds_ratio).

**Gradient descent**

To minimize our cost, we use Gradient Descent just like before in Linear Regression. There are other more sophisticated optimization algorithms out there such as conjugate gradient like BFGS, but you don’t have to worry about these. Machine learning libraries like Scikit-learn hide their implementations so you can focus on more interesting things!

**s′(z)=s(z)(1−s(z))**

Which leads to an equally beautiful and convenient cost function derivative:

**C′=x(s(z)−y)**

### **2.3 Python and Vectorization**

Vectorization is used to speed up the Python code without using loop. Using such a function can help in minimizing the running time of code efficiently. Various operations are being performed over vector such as dot product of vectors which is also known as scalar product as it produces single output, outer products which results in square matrix of dimension equal to length X length of the vectors, Element wise multiplication which products the element of same indexes and dimension of the matrix remain unchanged.

## 2.4 Python Basics with numpy (Programming Assignment)

Welcome to your first (Optional) programming exercise of the deep learning specialization. In this assignment you will:

- Learn how to use numpy.

- Implement some basic core deep learning functions such as the SoftMax, sigmoid, dsigmoid, etc...

- Learn how to handle data by normalizing inputs and reshaping images.

- Recognize the importance of vectorization.

- Understand how python broadcasting works.

## 2.5 Logistic Regression with a Neural Network mindset (Programming Assignment)

Welcome to the first (required) programming exercise of the deep learning specialization. In this notebook you will build your first image recognition algorithm. You will build a cat classifier that recognizes cats with 70% accuracy!



As you keep learning new techniques you will increase it to 80+ % accuracy on **cat vs. non-cat**datasets. By completing this assignment, you will:

- Work with logistic regression in a way that builds intuition relevant to neural networks.

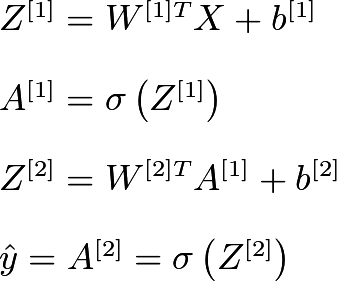
- Learn how to minimize the cost function.

- Understand how derivatives of the cost are used to update parameters.

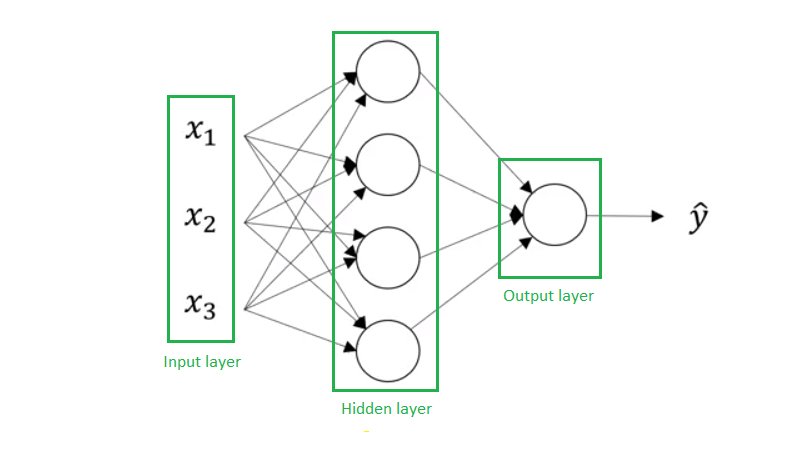
**Week 3: Shallow Neural Networks** - Learnt to build a neural network with one hidden layer, using forward propagation and backpropagation.

**3.1 The Shallow Neural Network**

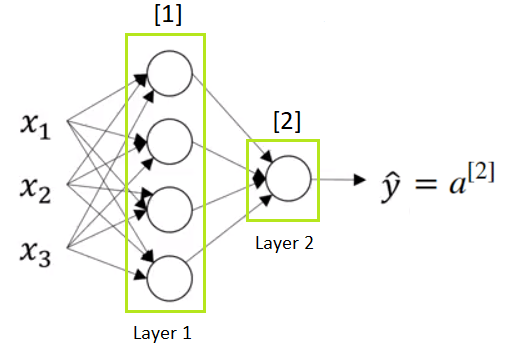
A neural network is built using various hidden layers. Now that we know the computations that occur in a particular layer, let us understand how the whole neural network computes the output for a given input X. These can also be called the forward-propagation equations.



* The first equation calculates the intermediate output Z[1] of the first hidden layer.
* The second equation calculates the final output A[1] of the first hidden layer.
* The third equation calculates the intermediate output Z[2] of the output layer.
* The fourth equation calculates the final output A[2] of the output layer which is also the final output of the whole neural network.

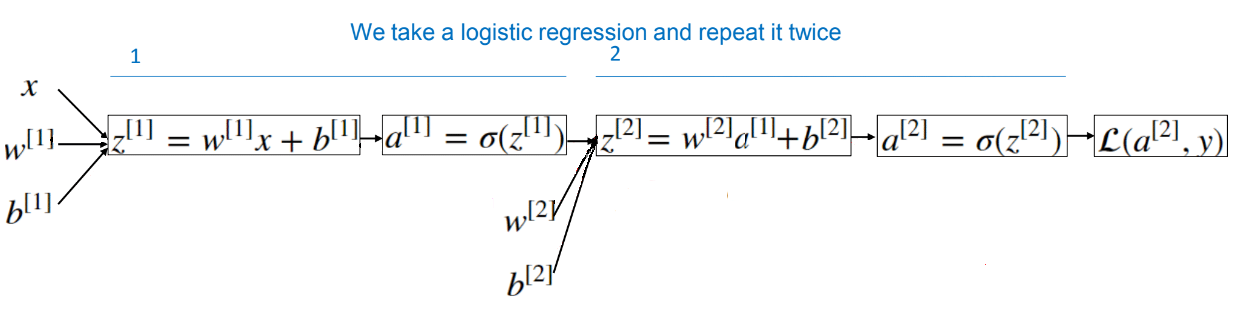


An example of a neural network is shown in the picture below. We can see we can form a neural network is created by stacking together several node units. One stack of nodes we will call a layer.



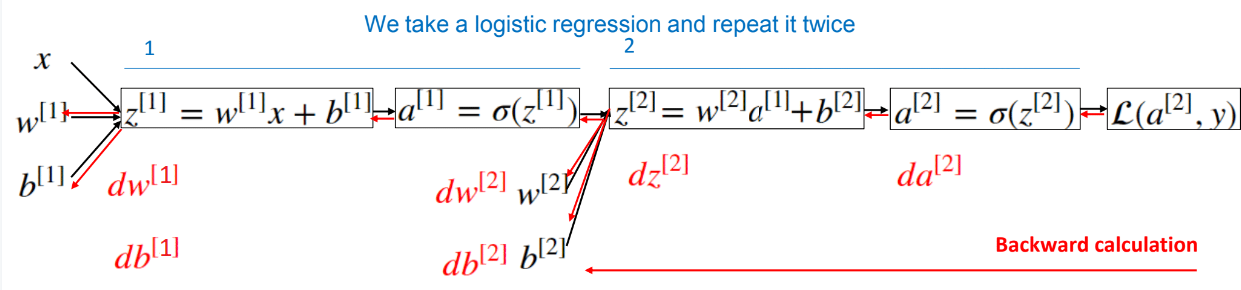
The first stack of nodes we will call Layer 1, and the second we will call Layer 2. We have two types of calculations in every node in the Layer 1, as well as in the Layer 2 ( which consists of just one node).  We will use a superscript square bracket with a number of particular layer to refer to an activation function or a node that belongs to that layer. So, a superscript [1] refers to the quantities associated with the first stack of nodes, called Layer 1. The same is with a superscript [2] which refers to the second layer. Remember also that x(i) refers to an individual training example.

The computation graph that corresponds to this Neural Network looks like this:



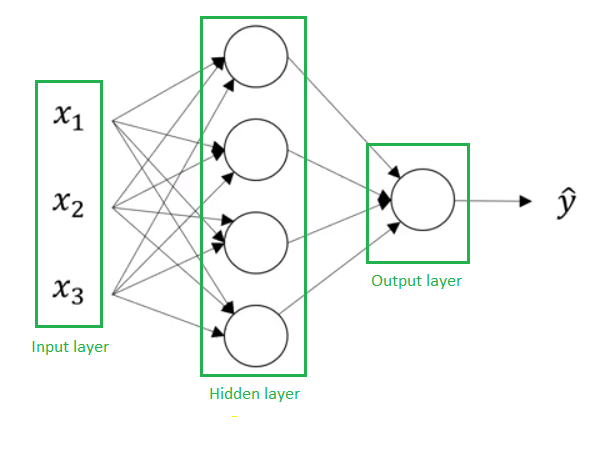
So, after computing z [1], similarly to the logistic regression, there is a computation of a [1] and that’s sigmoid of z [1]. Next, we compute z [2] using another linear equation and then compute a [2] which is  the final output of the neural network. Let’s remind ourselves once more that a [2] =y^. The key intuition to take away is that, whereas for Logistic Regression we had z followed by a calculation, and in this Neural Network we just do it multiple times.

In the same way, in a Neural Network we’ll end up doing a backward calculation that looks like this:



## 3.2 Neural Networks Representation

We will now represent a single layer Neural Network. It is a Neural network with one input layer, one hidden layer and the output layer, which is a single node layer, and it is responsible for generating the predicted value y^.



We have the following parts of the neural network:

* x1,x2 and x3 are inputs of a Neural Network. These elements are scalars and they are stacked vertically. This also represents an input layer.
* Variables in a hidden layer are not seen in the input set. Thus, it is called a hidden layer.
* The output layer consists of a single neuron only and y^ is the output of the neural network.

In the training set we see what the inputs are and we see what the output should be. But the things in the hidden layer are not seen in the training set, so the name hidden layer just means you don’t see it in the training set. An alternative notation for the values of the input features will be a[0] and the term a also stands for activations. Refers to the values that different layers of the neural network are passing on to the subsequent layers.

The input layer passes on the value x to the hidden layer and we’re going to call that the activations of the input layer a[0]. The next layer, the hidden layer will  in turn generate some set of activations which we will denote as a[1], so in particular, this first unit or this first node will generate the value a[1]1, the second node will generate the value a[1]2 and so on.

a[1] is a  1×4 matrix. a[2] will be a single value scalar and this is the analogous to the output of the sigmoid function in the logistic regression.

When we count layers in a neural network, we do not count an input layer. Therefore, this is a 2-layer neural network. The first hidden layer is associated with parameters w[1] and b[1] . The dimensions of these matrices are:

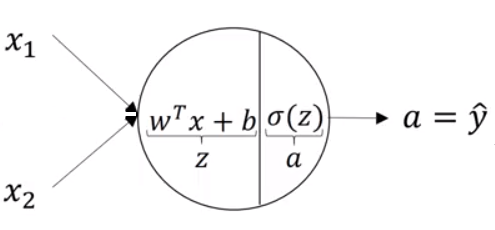
* w[1] is (4,3) matrix
* b[1] is (4,1) matrix

Parameters w[2] and b[2] are associeted with the second layer or actually with the output layer. The dimensions of parameters in the output layer are:

* w[2] is (1,4) matrix
* b[2] is a real number

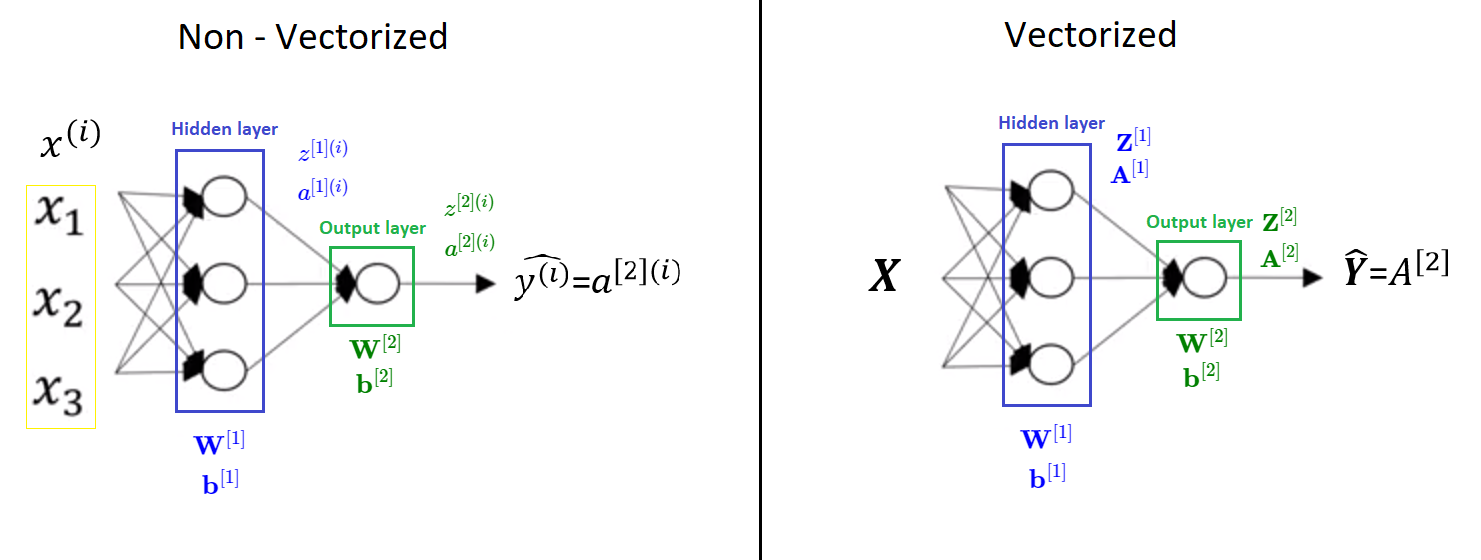
**Computing a Neural Network output**

Computing an output of a Neural Network is like computing an output in Logistic Regression, but repeating it multiple times. We have said that circle in Logistic Regression, or one node in Neural Network, represents two steps of calculations. We have also said that Logistic Regression is the simplest Neural Network. So if we have, for example, two features input vector it looks like this:



**3.3 Vectorizing Across Multiple Training Examples**

In the following picture we can see comparation of vectorized and non-vectorized version.

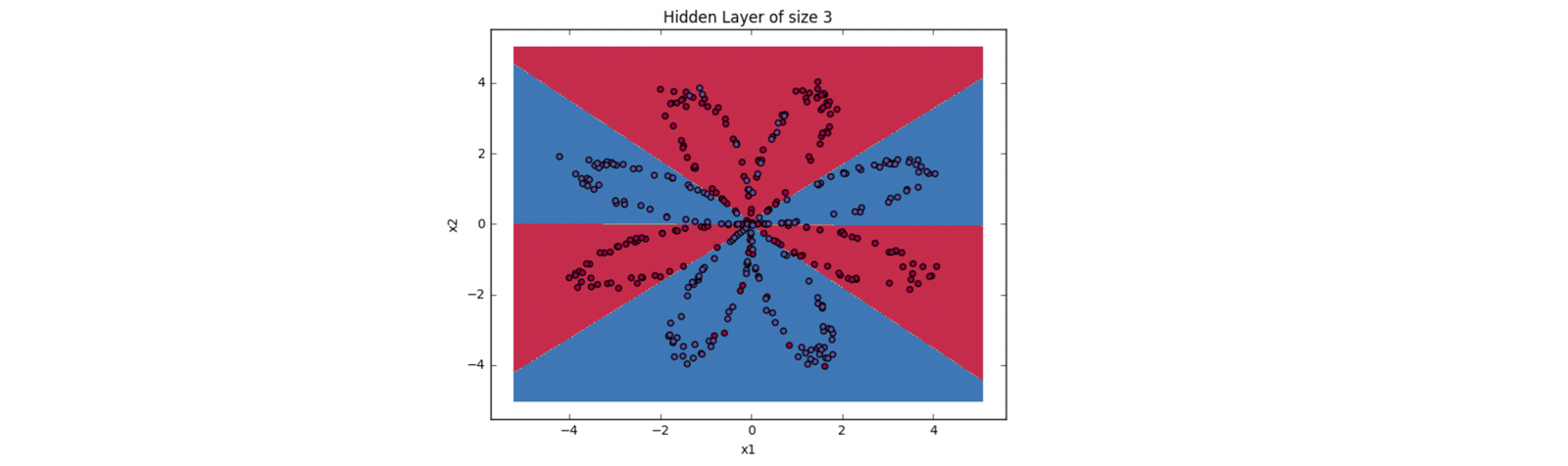


**3.4 Activation function**

The activation function of a node defines the output of that node given an input or set of inputs. A standard integrated circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input. This is similar to the behavior of the linear perceptron in neural networks. However, only nonlinear activation functions allow such networks to compute nontrivial problems using only a small number of nodes, and such activation functions are called nonlinearities.{\displaystyle \phi (v\_{i})=U(v\_{i})\tanh(v\_{i})}

**3.5 Planar data classification with a hidden layer (Programming Assignment)**

Welcome to the second programming exercise of the deep learning specialization. In this notebook you will generate red and blue points to form a flower. You will then fit a neural network to correctly classify the points. You will try different layers and see the results.



By completing this assignment, you will:

- Develop an intuition of back-propagation and see it work on data.

- Recognize that the more hidden layers you have the more complex structure you could capture.

- Build all the helper functions to implement a full model with one hidden layer.

**Week 4: Deep Neural Networks -** Understood the key computations underlying deep learning, used them to build and train deep neural networks, and apply it to computer vision.

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.