CSC 4504 : Langages formels et applications

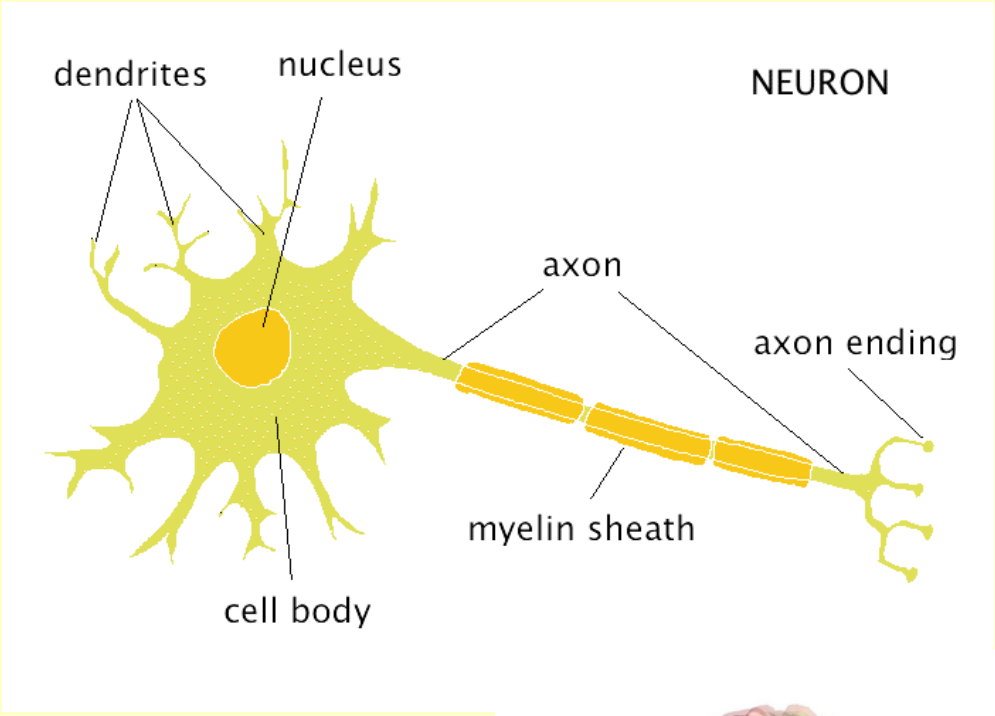
J Paul Gibson, D311

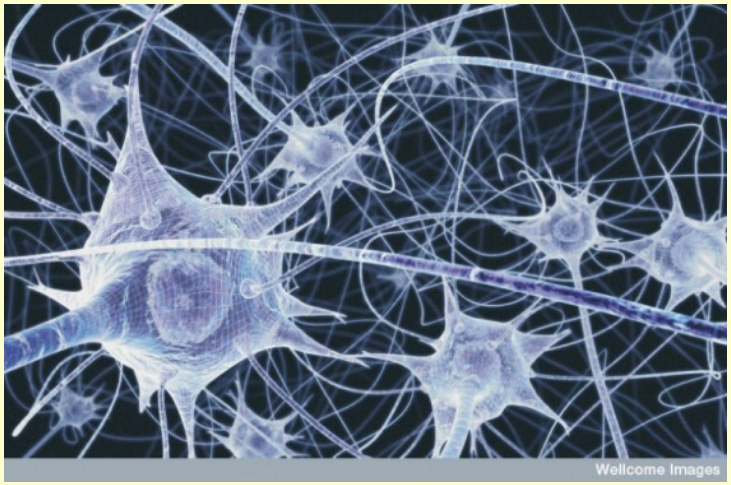
paul.gibson@telecom-sudparis.eu

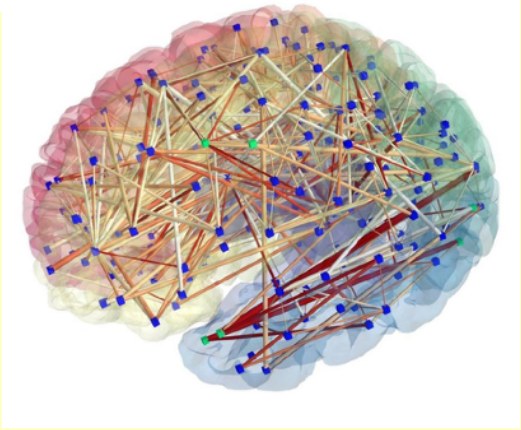
/~gibson/Teaching/CSC4504/Problem5-NeuralNetworks.pdf

Neural Networks

Neural Networks







Neural Networks

The following slides are a summary of material (by DANIEL SHIFFMAN) found at:

http://natureofcode.com/book/chapter-10-neural-networks/

1943 - Warren S. McCulloch (neuroscientist) and Walter Pitts (logician) developed the first conceptual **model** of an artificial neural network

"A logical calculus of the ideas imminent in nervous activity," describes the concept of a **neuron**, a single cell living in a network of cells that computes - it receives inputs, processes those inputs, and generates an output.

Their work is not meant to accurately describe how the biological brain works. Rather, an (artificial) **neural network** was designed as a computational model based on the brain to solve certain kinds of problems.

These problems are usually easy-for-a-human, difficult-for-a-machine: e.g., pattern recognition.... but things are changing

Neural Networks

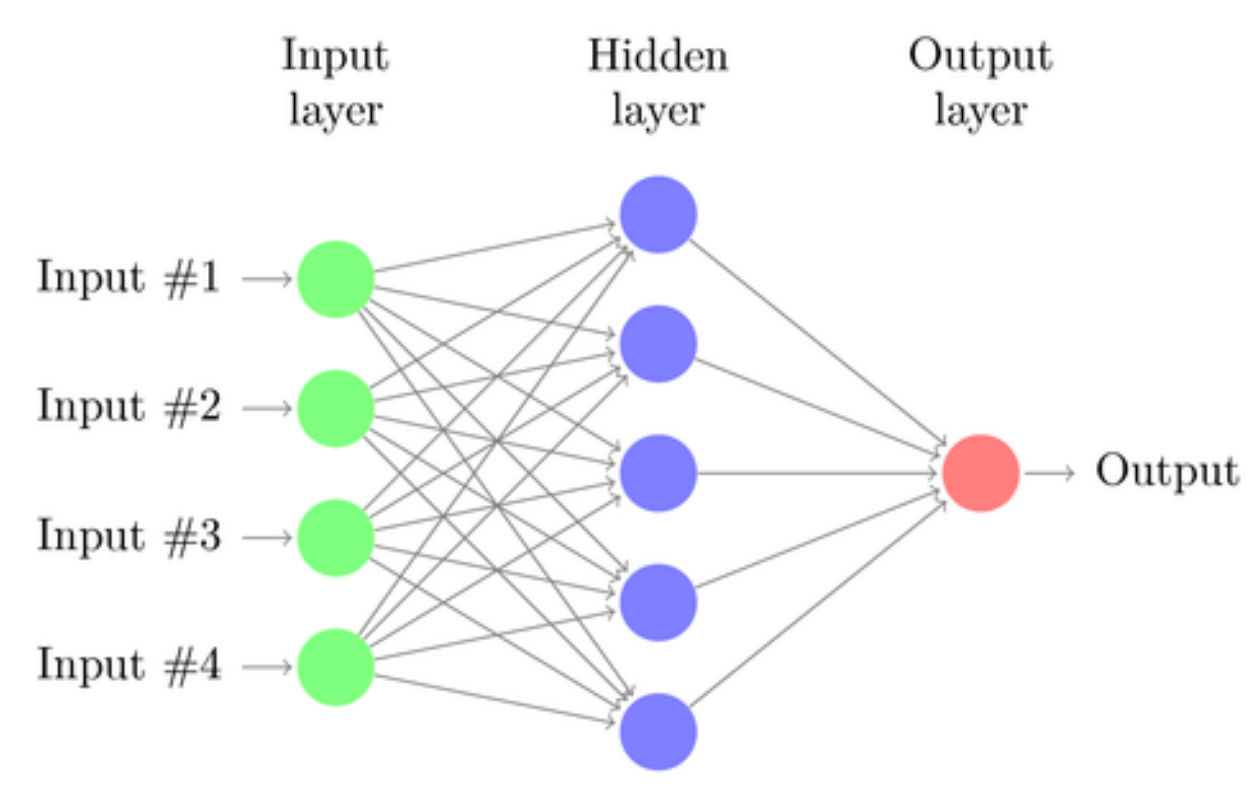
Learning strategies:

- ***Supervised Learning*** —Essentially, a strategy that involves a teacher that is smarter than the network itself. For example, let's take the facial recognition example. The teacher shows the network a bunch of faces, and the teacher already knows the name associated with each face. The network makes its guesses, then the teacher provides the network with the answers. The network can then compare its answers to the known "correct" ones and make adjustments according to its errors.

- ***Unsupervised Learning*** —Required when there isn't an example data set with known answers. Imagine searching for a hidden pattern in a data set. An application of this is clustering, i.e. dividing a set of elements into groups according to some unknown pattern.

- ***Reinforcement Learning*** —A strategy built on observation. Think of a little mouse running through a maze. If it turns left, it gets a piece of cheese; if it turns right, it receives a little shock. (Don't worry, this is just a pretend mouse.) Presumably, the mouse will learn over time to turn left. Its neural network makes a decision with an outcome (turn left or right) and observes its environment (yum or ouch). If the observation is negative, the network can adjust its weights in order to make a different decision the next time. Reinforcement learning is common in *robotics* and *autonomous vehicles*.

Neural Networks



A neural network is a “connectionist” computational system. Information is processed collectively, in parallel throughout a network of nodes (the nodes, in this case, being neurons)

Neural Networks

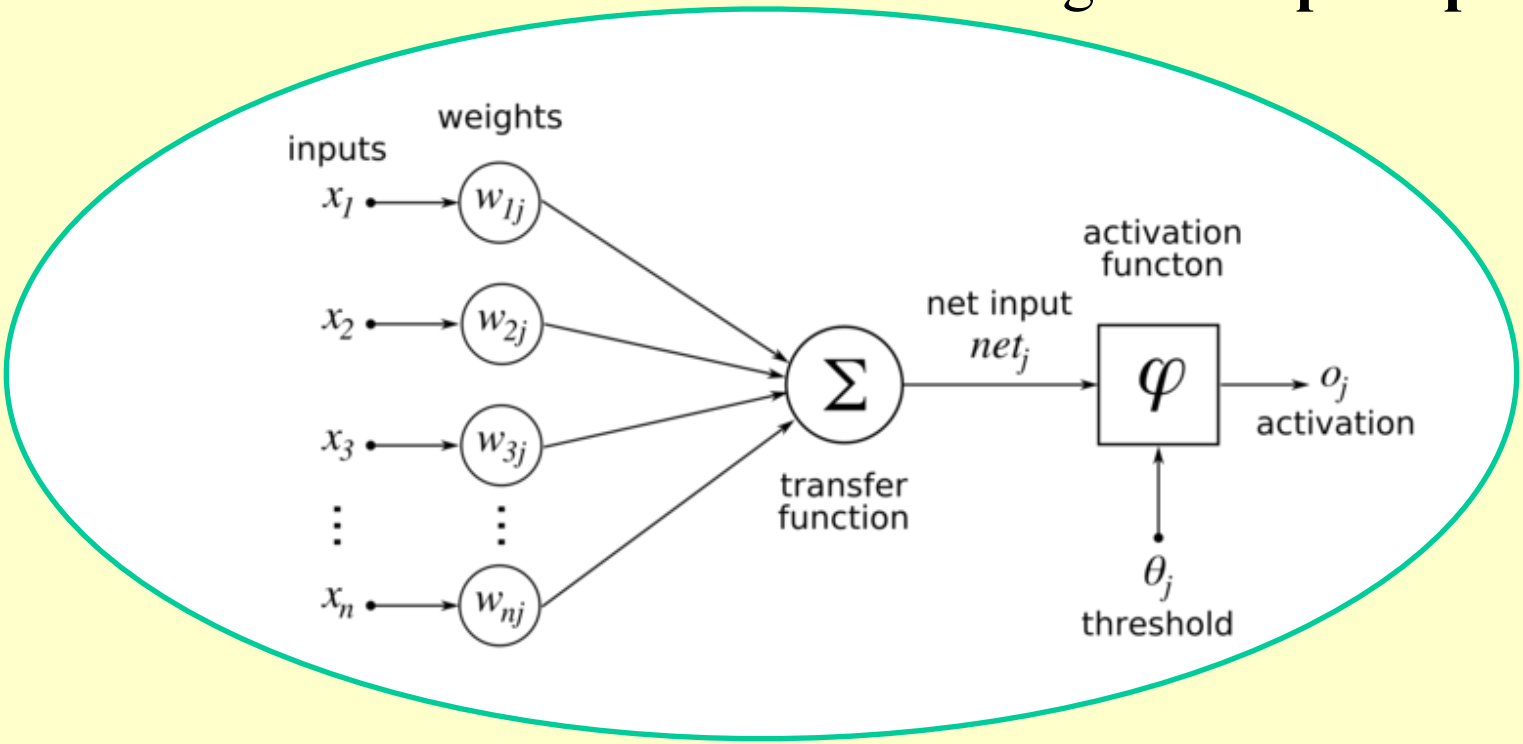
The Perceptron

Invented in 1957 by Frank Rosenblatt at the Cornell Aeronautical Laboratory, a perceptron is the simplest neural network possible: a computational model of a single neuron. A perceptron consists of one or more inputs, a processor, and a single output.

We combine perceptrons in a multi-layered network

Neural Networks

Weights in a **perceptron**



A neural network is not just a complex system, but a complex ***adaptive*** system, meaning it can change its internal structure based on the information flowing through it. Typically, this is achieved through the adjusting of *weights*. This adaption can be thought of as **learning**

A **perceptron** follows the "**feed-forward**" model, meaning inputs are sent into the neuron, are processed, and result in an output.

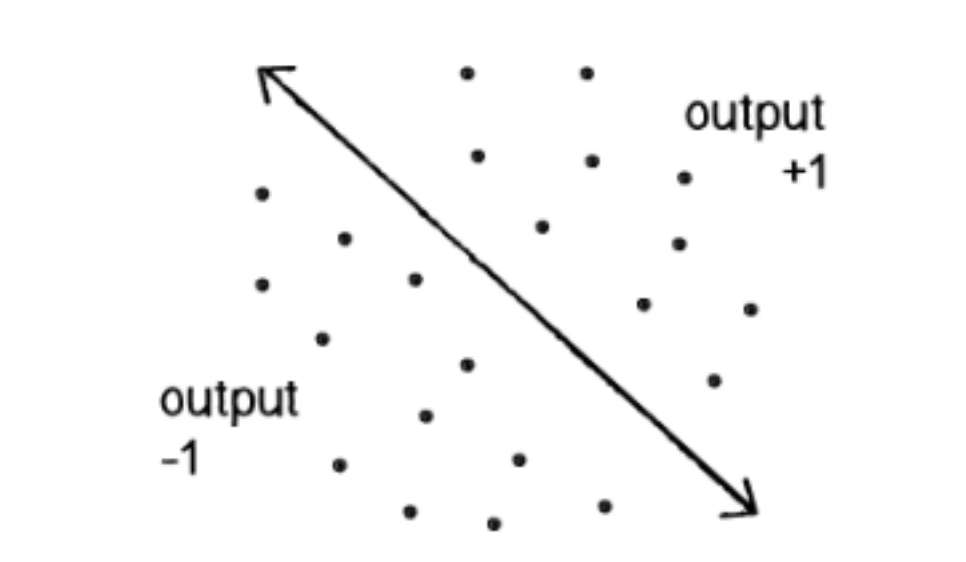
Neural Networks

The Perceptron Algorithm:

1. For every input, multiply that input by its weight.
2. Sum all of the weighted inputs.
3. Compute the output of the perceptron based on that sum passed through an activation function (e.g., the sign of the sum).

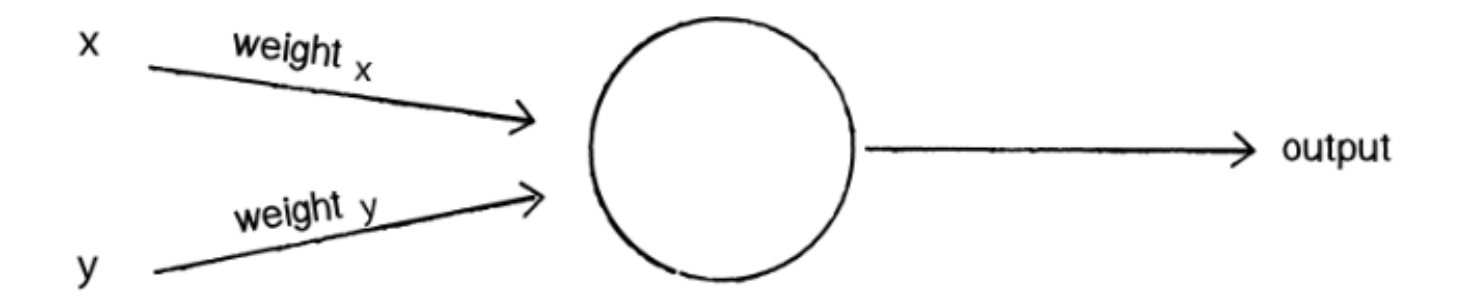
Simple Pattern Recognition Using a Perceptron

Now that we understand the computational process of a perceptron, we can look at an example of one used for pattern recognition:

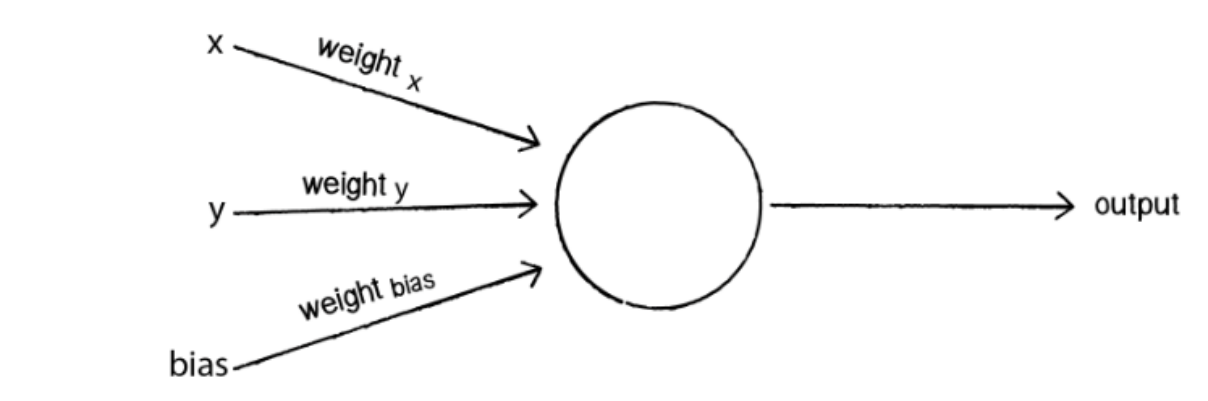


Can we train a perceptron to decide if a given point is above or below a specified line?

Neural Networks

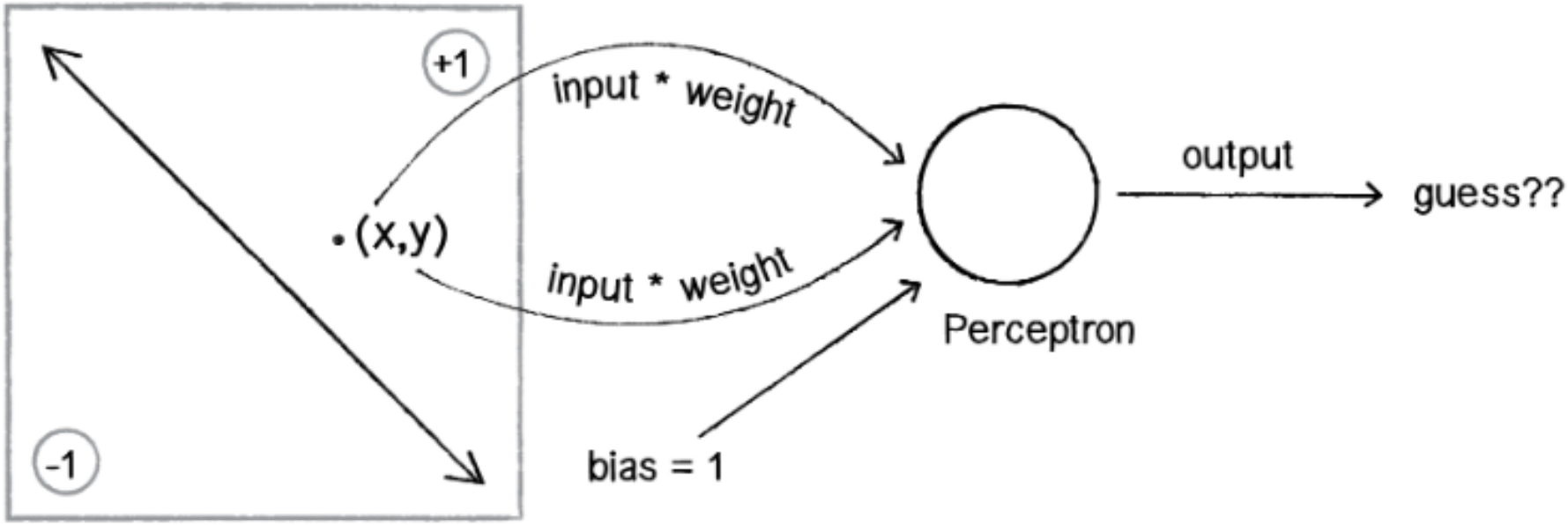


Initial design is flawed because the point 0,0 will always give a 0 output (for any weights)



A bias input always has the value of 1 and is also weighted

Neural Networks



Did the perceptron get it right? At this point, the perceptron has no better than a 50/50 chance of arriving at the right answer. Remember, when we created it, we gave each weight a random value. A neural network isn't magic. It's not going to be able to guess anything correctly unless we teach it how to!

Neural Networks

Supervised learning

With this method, the network is provided with inputs for which there is a known answer. This way the network can find out if it has made a correct guess. If it's incorrect, the network can learn from its mistake and adjust its weights. The process is as follows:

1. Provide the perceptron with (randomly generated bounded)inputs for which there is a known answer.
2. Ask the perceptron to guess an answer.
3. Compute the error. (Did it get the answer right or wrong?)
4. **Adjust all the weights according to the error**.
5. Return to Step 1 and repeat!

If the perceptron guesses the correct answer, then the guess equals the desired output and the error is 0. If the correct answer is -1 and we've guessed +1, then the error is -2. If the correct answer is +1 and we've guessed -1, then the error is +2.

Desired	Guess	Error
-1	-1	0
-1	+1	-2
+1	-1	+2
+1	+1	0

The error is the determining factor in how the perceptron's weights should be adjusted. For any given weight, what we are looking to calculate is the change in weight, often called *Δweight* (or "delta" weight, delta being the Greek letter Δ).

```
NEW WEIGHT = WEIGHT + ΔWEIGHT
```

Δweight is calculated as the error multiplied by the input.

```
ΔWEIGHT = ERROR * INPUT
```

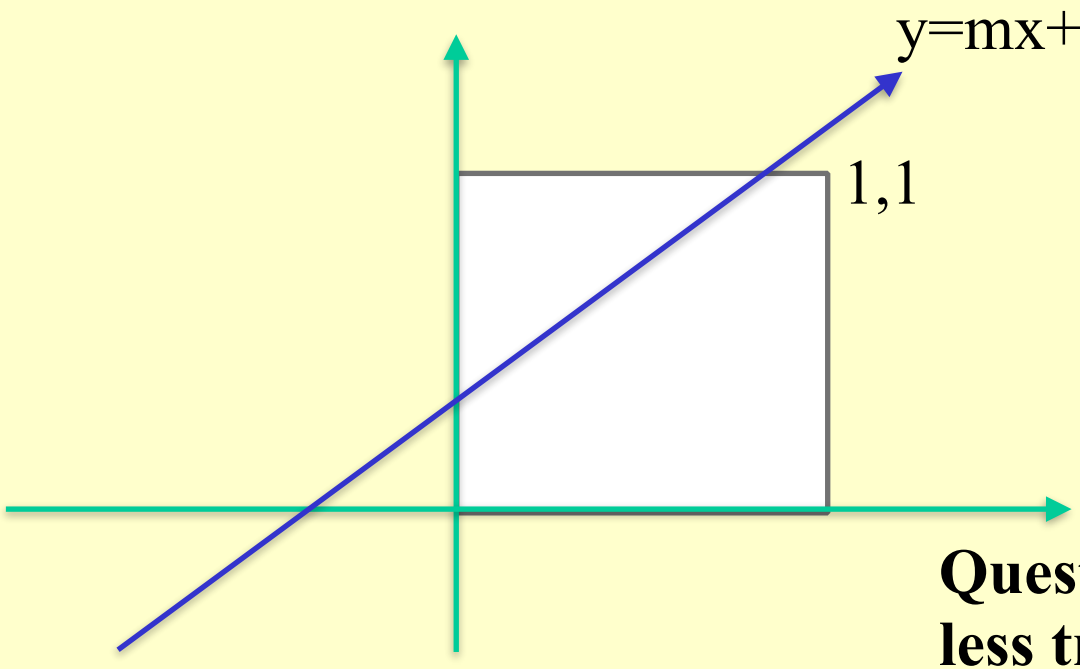
Therefore:

```
NEW WEIGHT = WEIGHT + ERROR * INPUT
```

Neural Networks

Problem 1: above or below the line

Implement the simple perceptron (as at the web site, 10.4 Coding the Perceptron- http://natureofcode.com/book/chapter-10-neural-networks/) in the language of your choice; and test that it learns as required (for a range of different values of m and c)



Configuration
0<=c<=1
0.5<=m<=2

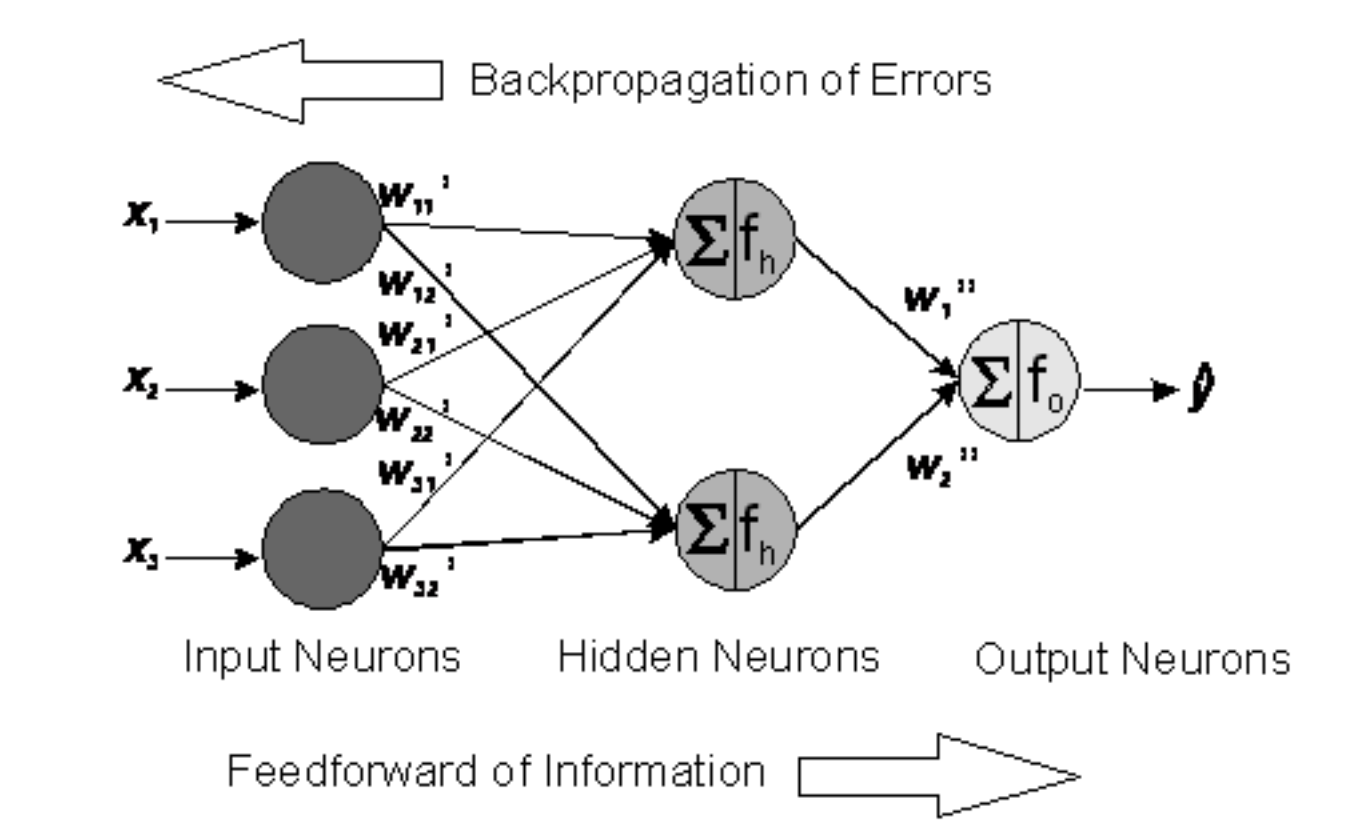
Train: 100 points in grid 0,0 … 1,1

Test: 100 points in grid 0,0 … 1,1

Question: do we need more or less training

Neural Networks

Back propagation



http://www.frank-dieterle.de/phd/2_7_1.html

A multilayer feedforward *backpropagation* neural network for three input variables x_1, x_2, x_3 and one response variable y

Neural Networks

In one line, deep neural networks are artificial neural networks (ANN) with multiple hidden layers of units between the input and output layers.

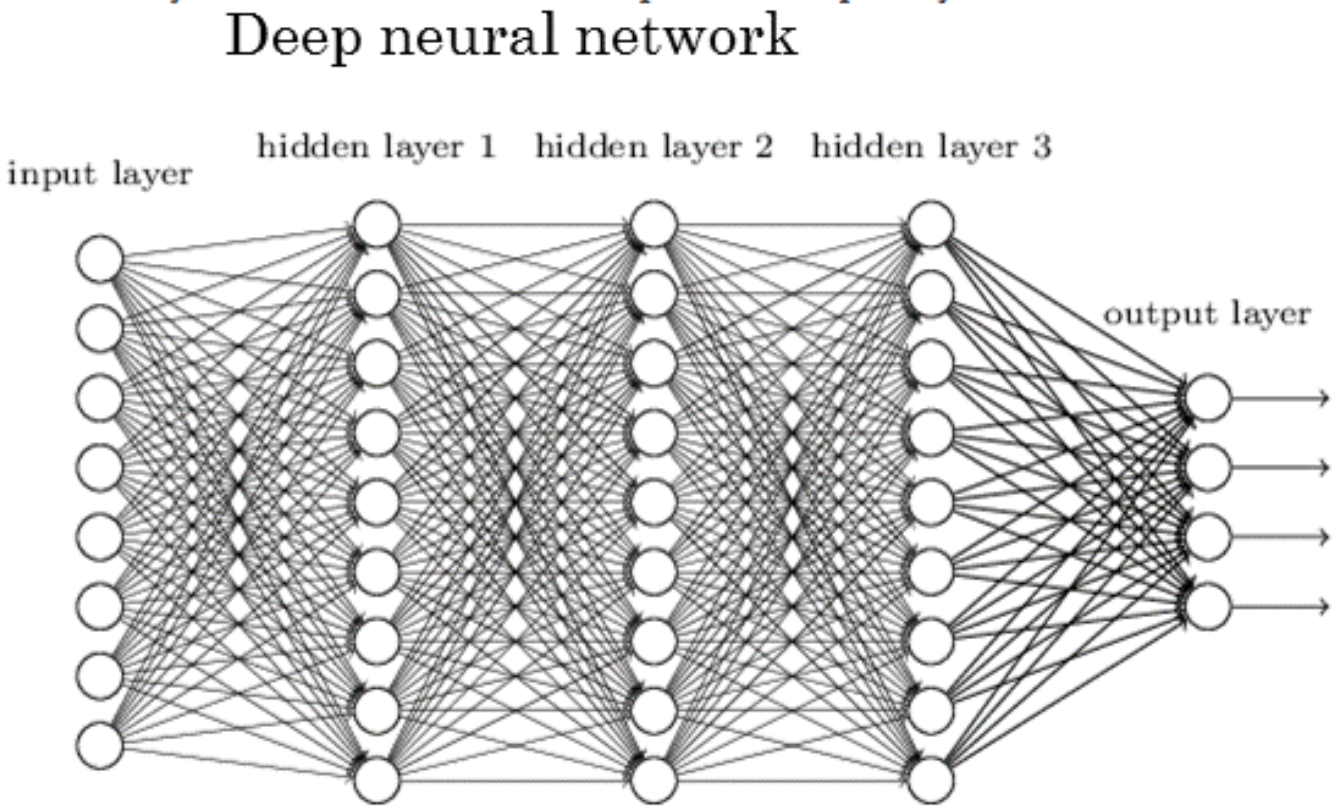


Image Courtesy: Google

Question: what could be the *challenges* with this type of network?

Neural Networks

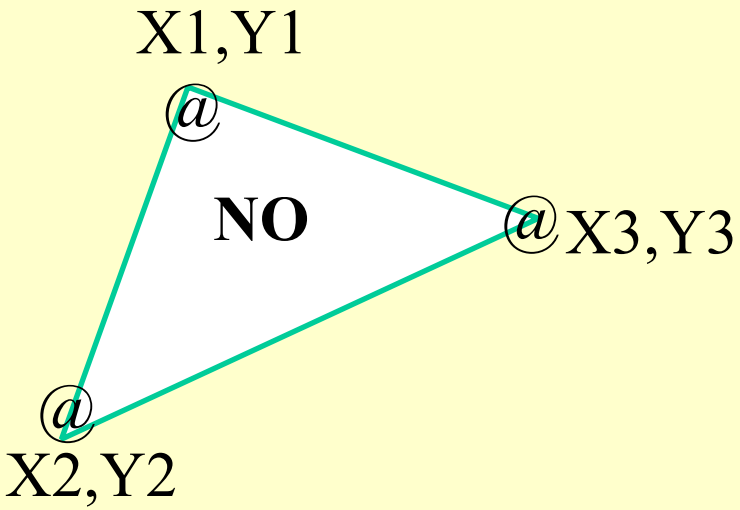
Question: what do you see, if anything ?

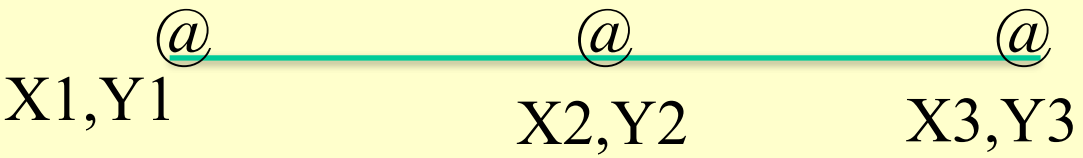
Raise hand when you see something but don't see what you see

Neural Networks

Problem 2:
design and implement a perceptron based NN to learn to recognise whether three points are on a straight line

Hint - you *may* need a different network topology and may need a *back propagation* technique.





NOTE: this is quite challenging

See: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Further Reading

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." Cognitive modeling 5.3 (1988): 1.

Widrow, Bernard, and Michael A. Lehr. "30 years of adaptive neural networks: perceptron, madaline, and backpropagation." Proceedings of the IEEE 78.9 (1990): 1415-1442.

Basheer, I. A., and M. Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application." Journal of microbiological methods 43.1 (2000): 3-31.

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127.