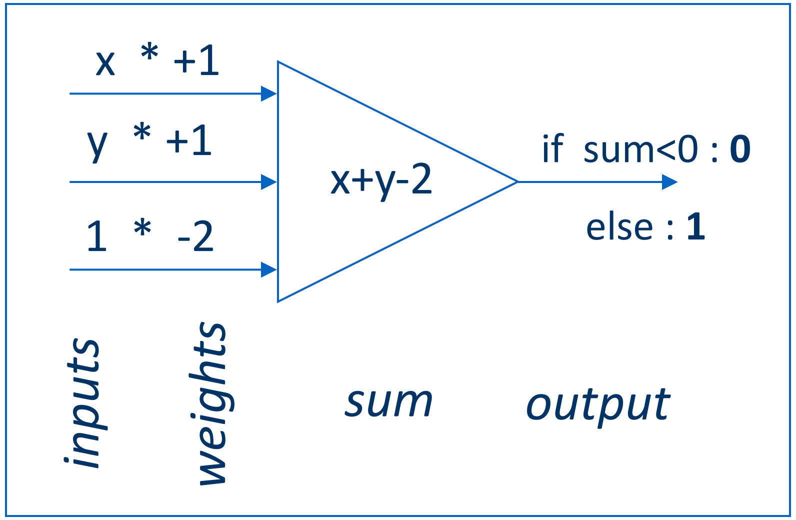
ECEN 758
Data Mining and Analysis

Perceptrons

Prehistory

W.S. McCulloch & W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 5, 115-137.

- This seminal paper pointed out that simple artificial "neurons" could be made to perform basic logical operations such as AND, OR and NOT.



Truth Table for Logical AND

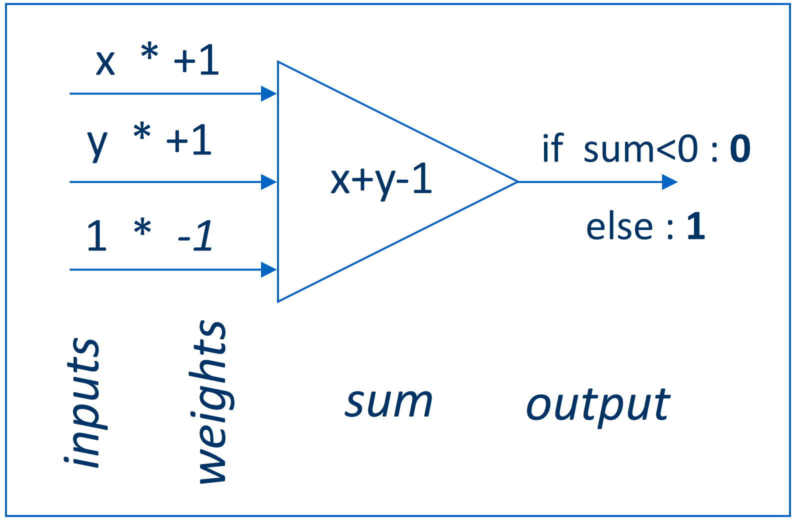
x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

inputs *output*

Nervous Systems as Logical Circuits

Groups of these “neuronal” logic gates could carry out *any* computation, even though each neuron was very limited.

- Could computers built from these simple units reproduce the computational power of biological brains?
- Were *biological* neurons performing logical operations?



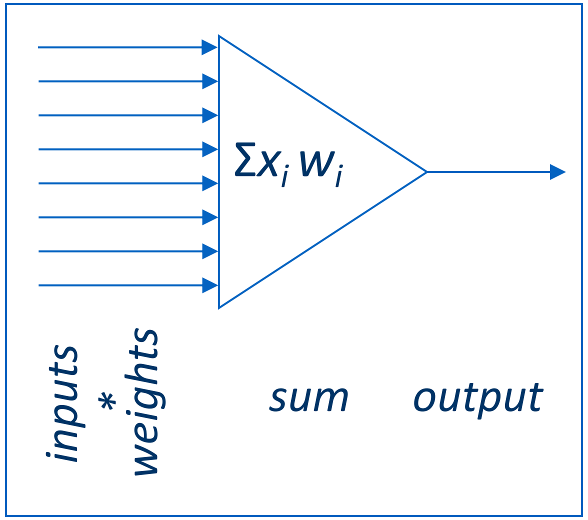
Truth Table for Logical OR

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

inputs *output*

The Perceptron

- Frank Rosenblatt (1962). *Principles of Neurodynamics*, Spartan, New York, NY.
- Subsequent progress was inspired by the invention of *learning rules* inspired by ideas from neuroscience…
- Rosenblatt's *Perceptron* could automatically learn to categorise or classify input vectors into types.



It obeyed the following rule:

If the sum of the weighted inputs exceeds a threshold, output 1, else output -1.

$$1 \text{ if } \Sigma\, input_i * weight_i > threshold$$

$$-1 \text{ if } \Sigma\, input_i * weight_i < threshold$$

Linear neurons

- The neuron has a real-valued output which is a weighted sum of its inputs

weight vector

$$\hat{y} = \sum_i w_i x_i = \mathbf{w}^T \mathbf{x}$$

Neuron's estimate of the desired output

input vector

- The aim of learning is to minimize the discrepancy between the desired output and the actual output
  - How de we measure the discrepancies?
  - Do we update the weights after every training case?
  - Why don't we solve it analytically?

A motivating example

- Each day you get lunch at the cafeteria.
  - Your diet consists of fish, chips, and milk.
  - You get several portions of each
- The cashier only tells you the total price of the meal
  - After several days, you should be able to figure out the price of each portion.
- Each meal price gives a linear constraint on the prices of the portions:

$$price = x_{fish}\, w_{fish} + x_{chips}\, w_{chips} + x_{milk}\, w_{milk}$$

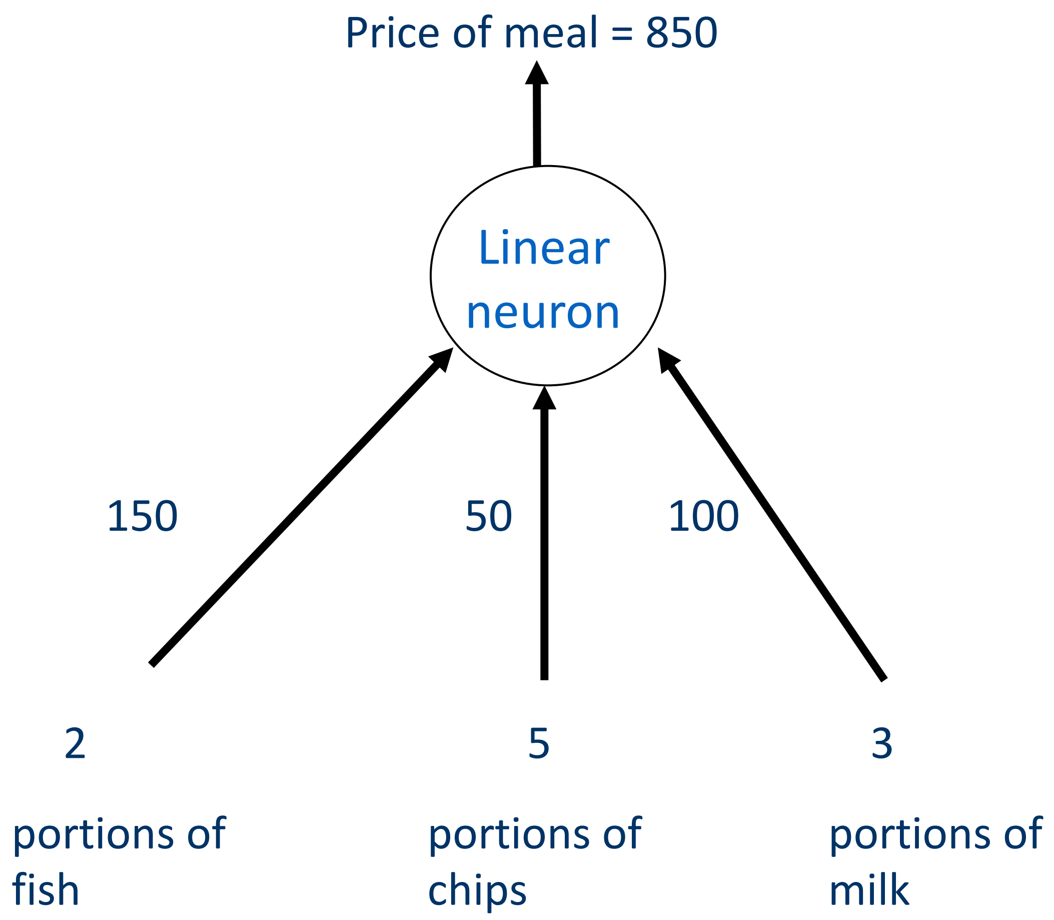
Two ways to solve the equations

- The obvious approach is just to solve a set of simultaneous linear equations, one per meal.
- But we want a method that could be implemented in a neural network.
- The prices of the portions are like the weights in of a linear neuron.

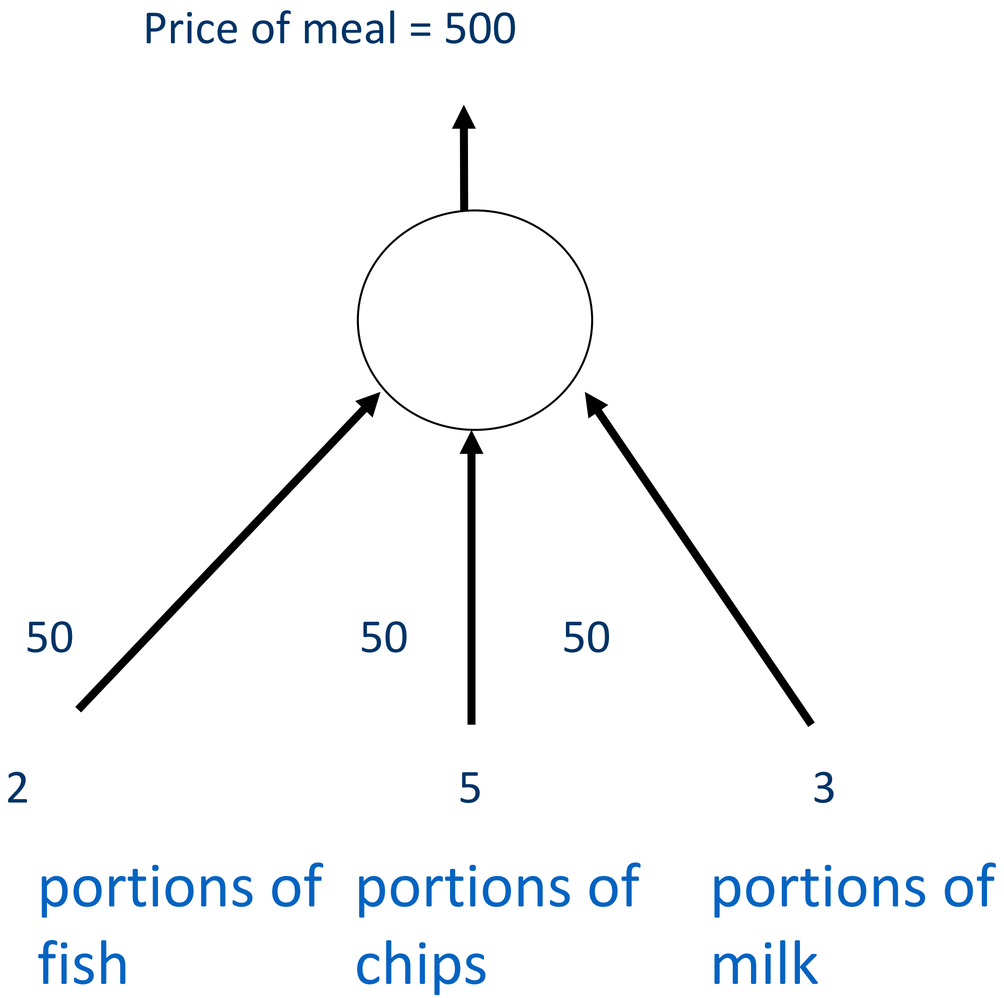
$$\boldsymbol{w} = (w_{fish} , \quad w_{chips} , \quad w_{milk})$$

- We will start with guesses for the weights and then adjust the guesses to give a better fit to the prices given by the cashier.

The cashier’s brain



A model of the cashier's brain with arbitrary initial weights



- Residual error = 350
- The learning rule is:

$$\Delta w_i = \varepsilon \, x_i \, (y - \hat{y})$$

- With a learning rate ε of 1/35, the weight changes are +20, +50, +30
- This gives new weights of 70, 100, 80
- Notice that the weight for chips got worse!

Behavior of the iterative learning procedure

- Do the updates to the weights always make them get closer to their correct values? No!
- Does the online version of the learning procedure eventually get the right answer? Yes, if the learning rate gradually decreases in the appropriate way.
- How quickly do the weights converge to their correct values? It can be very slow if two input dimensions are highly correlated (e.g. ketchup and chips).
- Can the iterative procedure be generalized to much more complicated, multi-layer, non-linear nets? YES!

Deriving the delta rule

- Define the error as the squared residuals summed over all training cases:

$$E = \tfrac{1}{2}\sum_n (y_n - \hat{y}_n)^2$$

- Now differentiate to get error derivatives for weights

$$\frac{\partial E}{\partial w_i} = \tfrac{1}{2}\sum_n \frac{\partial \hat{y}_n}{\partial w_i}\frac{\partial E_n}{\partial \hat{y}_n}$$

$$= -\sum_n x_{i,n}\,(y_n - \hat{y}_n)$$

- The batch delta rule changes the weights in proportion to their error derivatives summed over all training cases

$$\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i}$$

Perceptron Convergence

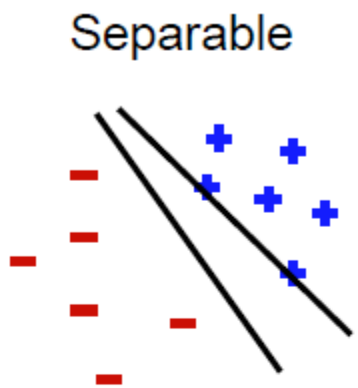
- Perceptron Convergence Theorem:
  - If there exist a set of weights that are consistent (i.e., the data is linearly separable) the Perceptron learning algorithm will converge
- How long would it take to converge?
- Perceptron Cycling Theorem:
  - If the training data is not linearly separable the Perceptron learning algorithm will eventually repeat the same set of weights and therefore enter an infinite loop
- How to provide robustness, more expressivity?

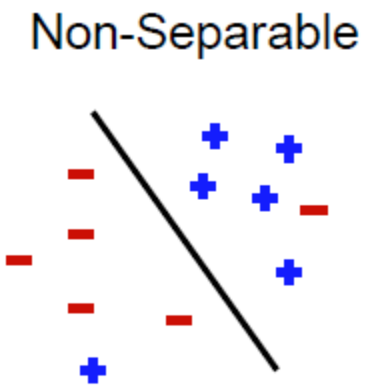
Properties of Perceptron

- Separability: Some parameters get training set perfectly

- Convergence: If training set is separable, perceptron will converge

- (Training) Mistake bound:
  Number of mistakes $< \frac{1}{\gamma^2}$
  - where $\gamma = \min_{t,u} |x^{(t)} u|$
    and $\|u\|_2 = 1$
    - Note we assume **x** Euclidean length **1**, then **γ** is the minimum distance of any example to plane **u**



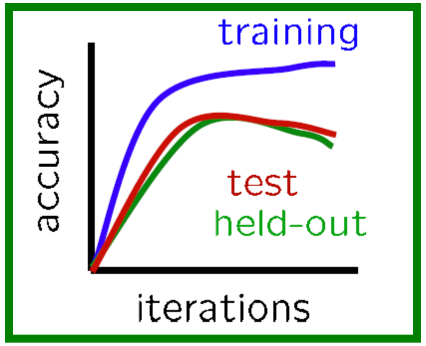


Updating the Learning Rate

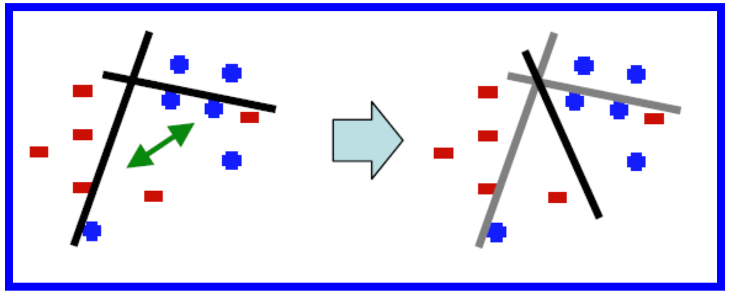
- Perceptron will oscillate and won't converge
- When to stop learning?
  1. Slowly decrease the learning rate η
     - A classic way is to: $\boldsymbol{\varepsilon} = c_1/(t + c_2)$
     - But, we also need to determine constants $\mathbf{c_1}$ and $\mathbf{c_2}$
  2. Stop when the training error stops chaining
  3. Have a small test dataset and stop when the test set error stops decreasing
  4. Stop when we reached some maximum number of passes over the data

Issues with Perceptrons

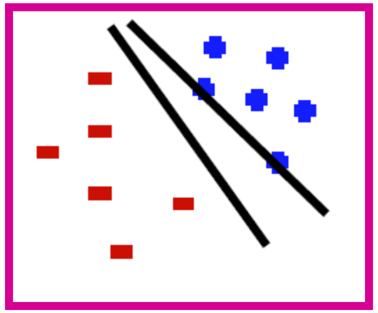
- Overfitting:



- Regularization: If the data is not separable weights dance around



- Mediocre generalization:
  - Finds a “barely” separating solution



Problem with the Perceptron

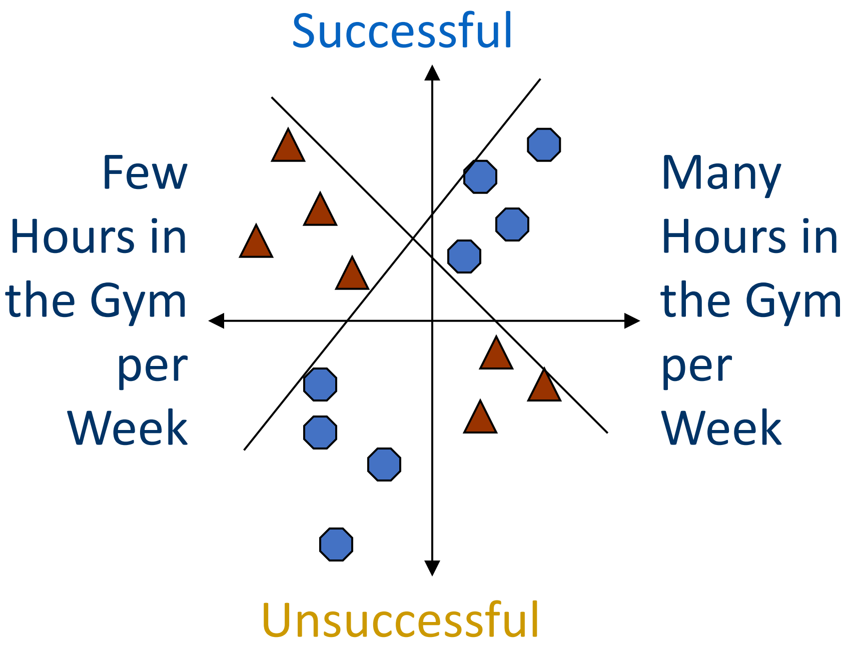
- Can only learn linearly separable tasks.
- Cannot solve any ‘interesting problems’-linearly nonseparable problems e.g. exclusive-or function (XOR)-simplest nonseparable function.

X_1	X_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

The Fall of the Perceptron

- Before long researchers had begun to discover the Perceptron's limitations.
- Unless input categories were "linearly separable", a perceptron could not learn to discriminate between them.
- Unfortunately, it appeared that many important categories were not linearly separable.
- E.g., those inputs to an XOR gate that give an output of 1 (namely 10 & 01) are not linearly separable from those that do not (00 & 11).
- Marvin Minsky & Seymour Papert (1969). *Perceptrons*, MIT Press, Cambridge, MA.

The Fall of the Perceptron



Footballers

Academics

Simple relationship:

Academics = Successful XOR Gym

In this example, a perceptron would not be able to discriminate between the footballers and the academics…

This failure caused the majority of researchers to walk away…..