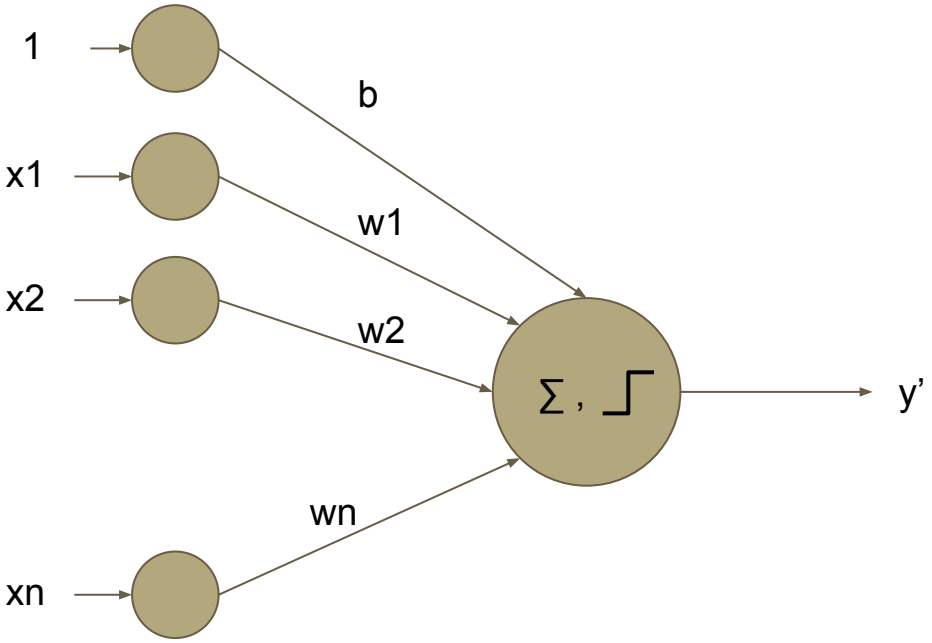
Perceptron

- Inspired by (simulation of) the human nervous system



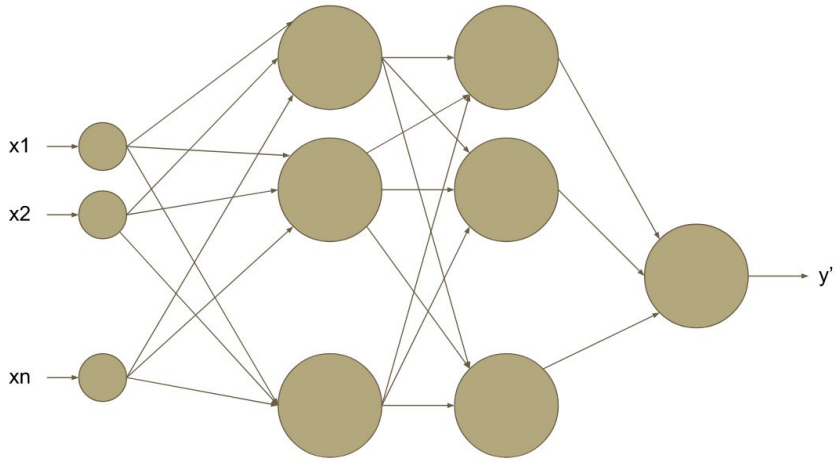
$$y' = sign(\sum_i w_i x_i + b)$$

Learning (iterative process):

- Initialize weights
- For each training item (**x**,**y**)
  - y' = g(w,x)
  - update all weights
    $w_i' = w_i + \alpha(y_i - y'_i)x_i$
- Until convergence

- Can learn (converge) in linearly separable situations
- Finds (some!) linear separation

Neural networks with hidden layers



- Very powerful in capturing arbitrary functions
  - having non-linear activation functions; careful selection to facilitate learning
- Automatic generation of (higher-level) features!
  - last level is similar to logreg on generated (relevant) high-level features, not all quadratic, cubic, ... which easily go into hundreds of thousands.
- Drawbacks
  - computationally demanding learning (recently alleviated)
  - more layers - more power - more prone to overfitting
  - black-box models

Neural network - use (forward propagation)

Use of a neural network

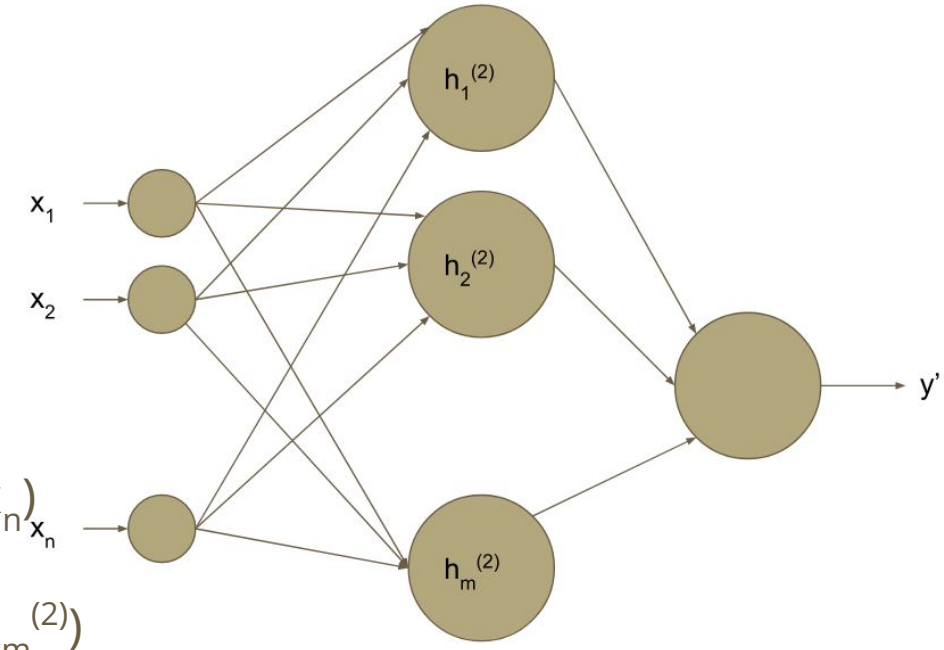
$h_1^{(2)} = g(w_{11}^{(1)}x_1 + w_{21}^{(1)}x_2 + \ldots + w_{n1}^{(1)}x_n)$

$h_2^{(2)} = g(w_{12}^{(1)}x_1 + w_{22}^{(1)}x_2 + \ldots + w_{n2}^{(1)}x_n)$

...

$h_m^{(2)} = g(w_{1m}^{(1)}x_1 + w_{2m}^{(1)}x_2 + \ldots + w_{nm}^{(1)}x_n)$

$y' = g(w_{11}^{(2)}h_1^{(2)} + w_{21}^{(2)}h_2^{(2)} + \ldots + w_{m1}^{(2)}h_m^{(2)})$



Neural networks - learning

- Two things to learn:
  - Structure: expert knowledge and experimentation
  - Parameters/weights : backpropagation (and other optimization approaches)
    - Gradient descent (consequence: step → sigmoid; error 0/1 → $(y-y')^2$)
      - Optimum can be local !
      - Weights must be initialized to random values
    - Can be done in a batch or online mode
      - One epoch : one learning iteration over training data
    - Overfitting problem - stop on check with holdout, ...
    - Computationally demanding
    - EXAMPLE

Neural networks - learning of the structure

- Fully connected
  - Number of layers, number of nodes in layers
  - Experiment & select
- Not fully connected
  - Optimal brain damage
    - Create a fully connected ANN
    - Remove a connection (or a node)
    - Retrain & test
    - If not worse, keep and repeat
  - Constructive approaches: sequential adding of units (e.g., to tackle misclassified examples)
- ! Very large networks can memorize all the training data
- Specific structures: recurrent (internal state, dynamics, memory), convolutional, ...

Neural networks - multiple classes