Artificial Intelligence

The roots of Machine Learning

Playing with the Perceptron

Extrapolating concepts,

from the Perceptron to general NNs

Playing with the Perceptron

- Objectives
  - Understand the "artificial neuron" AKA "Perceptron" as a mathematical function inspired by the biological neuron
  - It is the basic building block of neural networks
  - Code different Python implementations
  - Observe its behavior using different datasets
  - Understand the effect of some hyper-parameters

Perceptron concept

- Concept by Frank Rosenblatt, 1943
  - implementation 1957
  - Inspired by the biological neuron (a single neuron)
- It takes inputs
  - Think of a collection of values, X = (x1, ... , xn)
- Each input is weighted by a weight/coefficient
  - W = (w1, ... , wn)
- So, it does a summation of products
  - Sum = x1*w1 + ... + xn*wn
- It also adds a bias term
  - Linear Result = Sum + bias
- It computes some "activation function" of the linear result, Result = f(linear result)
- So, a Perceptron is a function: w1x1 + ... + wnxn + bias

Perceptron concept

- The classic "activation function" (F) is a "step-function"
  - Outputs 0 or 1, nothing in-between
- Perceptron Learning Algorithm: iterative process to learn W , b
- Perceptron Learning Algorithm
  - Random init for W and b
  - For each learning opportunity, for each sample
    - Multiply the weights (w) by the current sample (cs)
      - lout = np.dot(cs, w)
    - Sum the bias (b) to the resulting linear output (lout)
      - lout += b
    - Obtain a prediction for cs (pcs) using the activation function (F)
      - pcs = F(lout)
    - Compare the prediction to the real thing (y), there will be a deviation (d)
      - d = y - pcs
    - Find the adjustment (a) according to the learning rate (LR) and d
      - a = LR * d
    - Update W by summing it the current sample adjustment: W += a * cs
    - Also update the bias: b += a

Perceptron - making predictions

- A Perceptron implementation should have a "predict" method
- Accuracy is the relationship between the number of correct predictions for samples in some test set, and the test set's entire length
- If the accuracy is low, check:
  - W and b initialization
  - Learning rate
    - Too low? Too high? Relationship to number of iterations?
  - Feature scaling
    - Do 1+ feature(s) have values relatively big? They can dominate the learning.
    - Normalize (Min-Max scaling)
    - Standardize (Z-score normalization)
  - Number of iterations
  - Implementation
  - Data quality

Perceptron - making predictions

- If the accuracy is low, check the need for:
  - Normalization (SciKit's MinMaxScaler)
    - To scale the data to a fixed range (e.g. [0,1])
      - N(x) = x - min(X) / max(X) - min(X)
      - x is a sample and X the entire dataset ; operations are performed per feature
  - Standardization (SciKit's StandardScaler)
    - Mean = 0 ; Stdevp = 1
      - S(x) = x - Mean(X) / Stdevp(X)
      - x is a sample and X the entire dataset ; operations are performed per feature

References

- https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon
- https://en.wikipedia.org/wiki/Perceptron
- https://github.com/amsm/am_perceptron