Computation Graph and Back Propagation

S. Dakurah

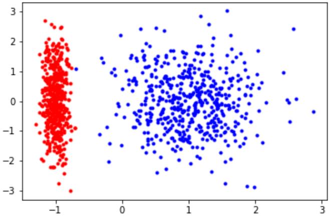
Setup & Implementation
02/12/2020

Outline

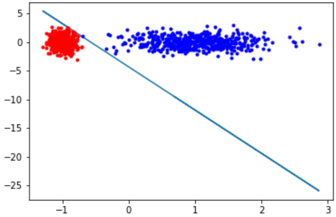
Outline

Linearly Separable Case

- Apply the perceptron algorithm[1].



(a) Linearly Separable



(b) Perceptron Decision Line

Figure

[1] single layer neural network

Outline

Non-Linearly Separable Case

- The perceptron breaks down for non-linearly separable case.
- Apply a MLP with good approximation.

Problem Set Up

- Given an input matrix X and a response vector y.

$$X = \begin{bmatrix} X_{11} & \dots & X_{1m} \\ X_{21} & \dots & X_{2m} \\ . & \dots & . \\ X_{n1} & \dots & X_{nm} \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}$$

Network Architecture

1. Decide on the network architecture.

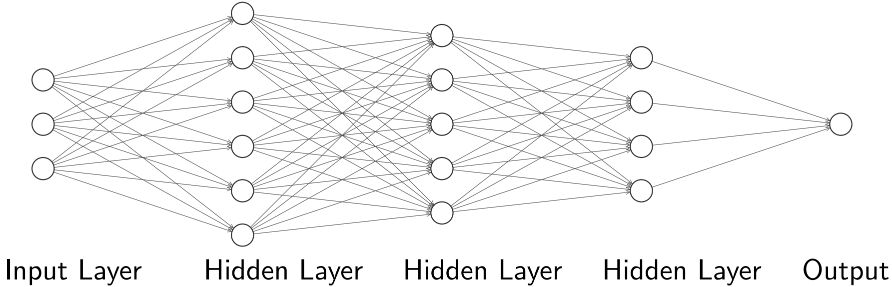


Figure: Network Architecture

Network Architecture

- The network architecture assumes a 3-feature input.
- Three (3) hidden layers.
- Five (5) neurons, four (4) neurons, and three (3) neurons excluding the bias term in the first, second and third layers respectively.

Feed-forward Computations

1. Compute the activations.
   Activations for hidden layer 1.

$$a_j^{[0]} = \sum_{i=1}^{m=3} w_{ji}^{(0)} X_i + w_{j0}^{(0)} \quad j = 1, ..., 6$$

2. Choose an activation function $h(.)$.[2] This transforms the activations into a new set of inputs.
   Common activation functions:
   - Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
   - Tanh[3]: $tanh(x) = 2\sigma(2x) - 1$
   - ReLU[4]: $max(0, x)$
   - Leaky ReLU: $max(ax, x)$, where a is a small positive constant.

'

[2] This should be differentiable and non-linear

[3] Scaled sigmoid

[4] Dying neuron problems

Feed-forward Computations

3. Transform activation ouput:
   Inputs for hidden layer 1.

$$z_j^{[1]} = h(a_j^{[0]}) \quad j = 1, ..., 6$$

4. Compute activations for hidden layer 1

$$a_j^{[1]} = \sum_{i=1}^{m=6} w_{ji}^{(1)} z_i^{[1]} + w_{j0}^{(1)} \quad j = 1, ..., 5$$

5. Transform the activations into inputs for hidden layer 2.

$$z_j^{[2]} = h(a_j^{[1]}) \quad j = 1, ..., 5$$

6. Compute activations for hidden layer 2.

$$a_j^{[2]} = \sum_{i=1}^{m=6} w_{ji}^{(2)} z_i^{[2]} + w_{j0}^{(2)} \quad j = 1, ..., 4$$

Feed-forward Computations

7. Transform activations into input for the third layer.

$$z_j^{[3]} = h(a_j^{[2]}) \quad j = 1, .., 4$$

8. Compute the last activation

$$a_j^{[3]} = \sum_{i=1}^{m=6} w_{ji}^{(3)} z_i^{[3]} + w_{j0}^{(3)} \quad j = 1$$

Complete Computation Graph

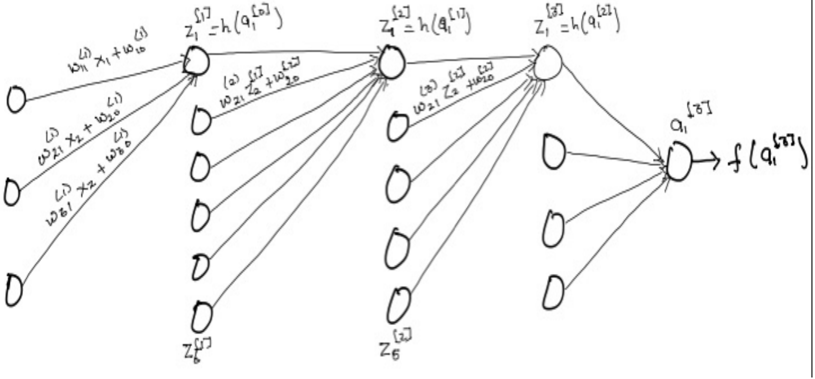


Figure: Computation Graph

Summary of CG

- Each layer has $[J \times K]$ weigths, where J is the number of neurons in the previous layer and K is the number of neurons in the current layer.
- This simple three layer network will have $[3 \times 6] + [6 \times 5] + [5 \times 4] + [4 \times 1] = 72$ parameters!

Outline

Error Backpropagation

1. Define a cost/loss function $L()$ that quantifies how much your outputs deviates from the target.
2. Compute the gradient $\frac{\partial L}{\partial w^{[3]}}$ for the last set of weights.
3. Apply chain rule to get weigths for previous layers.
4. Using the delta rule, perform the update $w^{[k]} = w^{[k]} - \alpha \frac{\partial L}{\partial w^{[k]}}$, where α is the learning rate.

Outline

References

1. Pattern Recognition and Machine Learning $\sim$ M. Bishop
   - Chaper 5.1 & 5.3
2. The Elements of Statistical Learning $\sim$ Hastie et al.
   - Chaper 4.5