Lecture 19

Name: Harsh Sanjay Roniyar

Roll Number: 22B3942

`Assignment 3 has been released`

Recap:

- Hidden Layers
- Universal Approximation (Cybenko's) Theorem
- Activation Functions
    - Softmax is generalization of sigmoid for multi-class classification
    - e.g. $\text{Softmax}\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$, where $0 \leq p_j \leq 1$ and $\sum_{j=1}^{c} p_j = 1$ with

$$p_j = \frac{e^{x_j}}{\sum_{i=j}^{c} e^{x_j}}$$

    - The CE loss is $\text{CE} = -\sum_{j=1}^{c} t_j \log p_j$, where t_j is the target class. This is an extension of the binary cross entropy loss discussed earlier (when $c = 2$).

Multi-Class SVM

Choice 1:

Class i vs Class j SVMs $\implies$ $\binom{c}{2}$ SVMs

Choice 2:

Class i vs Rest SVMs $\implies$ c SVMs

$\arg\max\limits_{j} \left(w_j^T x + b_j\right)$ would give us the decided class and j is the class.

Basic Structure of a Neural Network

- Output Layer
- Hidden Layer(s)

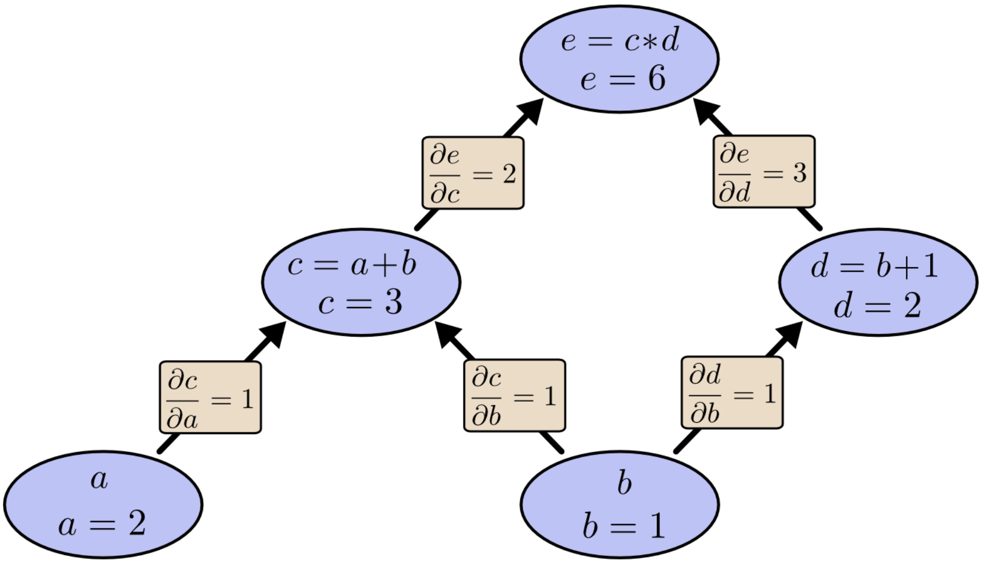
- Input Layer

Backpropagation

as a method to update the gradient weights

Chain Rule of Differentiation

Computation Graph



Source: Neural Networks - Harsh S Roniyar

Vector Valued Functions

$$f(x) = \begin{bmatrix} f_1(\bar{x}) \\ f_2(\bar{x}) \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \end{bmatrix}$$

Jacobians

The Jacobian for a function $f(x)$ as defined above would be -

$$J(f) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix}$$

Issues with Gradient Descent

- Need to find good step-size
- Lots of computation before each update
- Can get stuck in local minima

Summary of GD

Loop until stopping criterion

$$\text{loss } l \leftarrow 0,\ \nabla l \leftarrow 0$$

Loop over training samples:

$$l_i \leftarrow \text{loss}(\theta, x_i, t_i)$$

$$\nabla l_i \leftarrow \text{grad}(\theta, x_i, t_i)$$

$$\nabla l \mathrel{+}= \nabla l_i \ ; l+ = l_i$$

$$\theta \leftarrow \theta - \eta \nabla l$$