CS 7180 Algorithmic and Statistical Aspects of Deep Learning

Mounica Subramani

Problem Set 1

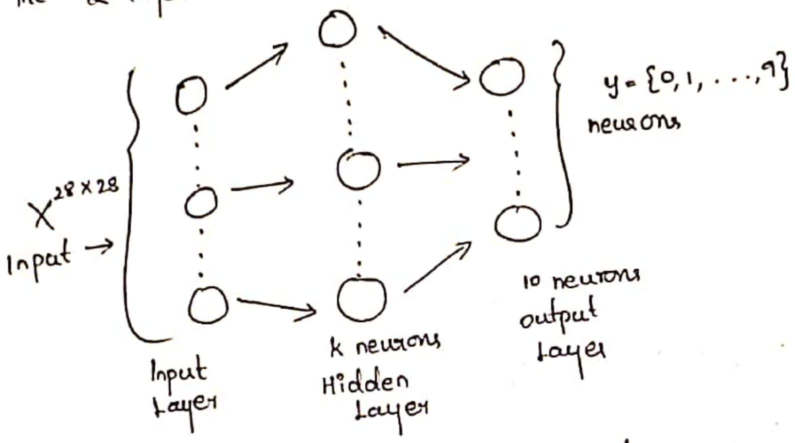
1) a) W_1 and W_2 denote the weight matrices of each layer.

Sample $\rightarrow$ $x \in \mathbb{R}^{28 \times 28}$

label $\rightarrow$ $y \in \{0, 1, 2, \dots, 9\}$

$y_{pred} = ?$

Let the 2 layer neural network architecture be like,



Equation of y^{Pred} as a function of w_1 and w_2, given input x is written as,

output from hidden layer $\Rightarrow$ $reLU(w_1 \cdot x)$ $\hookrightarrow$ input of output layer along with weights and activation.

$[W_{input-hidden}] \cdot I/P$

$$\boxed{y^{Pred} = ReLU[W_2(ReLU(W_1 \cdot x + b_1)) + b_2]}$$

x - input matrix of size 784×1

b_1 and b_2 are bias vector.

ReLU - activation function

W_1 - weight matrix of size $k \times 784$

W_2 - weight matrix of size $10 \times k$

Output of hidden layer $\Rightarrow$ $ReLU(W_1 \cdot x)$

$\Rightarrow ReLU([k \times 784] \cdot [784 \times 1])$

$X_2 = ReLU[k \times 1]$

hidden layer to output layer $= ReLU(W_2 \cdot X_2)$

$\Rightarrow ReLU([10 \times k] \cdot [k \times 1])$

$$\boxed{output = ReLU[10 \times 1]}$$

Scanned with CamScanner

1b) Given :

$(x_1, y_1) \ldots (x_m, y_m)$ where $m = 50{,}000$ – training data

$x_i \in \mathbb{R}^{28 \times 28}$

$y_i \in \{0, 1, \ldots 9\}$, $i = 1 \ldots m$

Loss of predicted label compared to correct label = ?

Here, optimization of the network refers to, reducing the loss of predicted label and bringing close to correct label.

Cross-entropy is widely used loss function for optimizing classification models. So lets minimize that.

Cross-entropy equation,

$$L(w) = -\left[\sum_{i=1}^{m} \sum_{k=1}^{10} I\{y_i = k\} \log \frac{\exp(w_k^T x_i)}{\sum_{j=1}^{10} \exp(w_j^T x_i)}\right]$$

where $I\{y_i = k\}$ – indicator function

W – w_1 and w_2 weight matrices together

k – no. of classes.

$L(w)$ – loss

Prob 2 a)

Given :

w_1, w_2 – weight matrices of the network

ReLU activation and

Quadratic loss function is used.

Training data $= (x_1, y_1) \ldots (x_m, y_m)$

$x_i \in \mathbb{R}^{\cancel{28 \times 28}}$

$y_i \in \mathbb{R}$

Training loss,

$$L(w) = \frac{\sum_{i=1}^{m} (y_i - y_{(i)}^{Pred})^2}{m}$$

$y_{(i)}^{Pred}$ can be written as $[ReLU(w_2(ReLU(w_1 \cdot x) + b_1)) + b_2)]$

[refered from prob 1 a)]

2b) Backpropagation algorithm.

Back-propagation is the practice of fine-tuning the weights of neural net based on the error rate obtained in the previous epoch)
loss.

Neural networks uses back propagation as a learning algorithm. It is used to calculate derivates quickly. The weights are updated backwards, from output towards input.

Proper tuning of weights results in lower error rates, making the model more reliable by generalizing it.

This algorithm looks for minimum value of the error function in the weight space using a technique called delta rule (or) gradient descent.

[Gradient descent is an iterative optimization algorithm for finding the minimum of a function (error function)]

The error function derivation for w_1 & w_2,

$$\frac{\partial L(w)}{\partial w_2} = \frac{\partial L(w)}{\partial y_{pred}} \cdot \frac{\partial y_{pred}}{\partial W_2}.$$

$$\frac{\partial L(w)}{\partial W_2} = \frac{\partial L(w)}{\partial O} \cdot \frac{\partial O}{\partial Z} \cdot \frac{\partial Z}{\partial W_2}$$ [O_1 and O_2].

$$\left\{\begin{array}{l} O_2 = \mathrm{ReLU}(W_2(\mathrm{ReLU}(W_1 \cdot X + b_1)) + b_2) \\ O_1 = \mathrm{ReLU}(W_1 x + b_1). \\ Z = W_1 x + b_1 \end{array}\right.$$

Forward Pass:

Backward Pass: computing gradients,

$$\frac{\partial L(w)}{\partial w_i} = \frac{\partial L(w)}{\partial O} \cdot \frac{\partial O}{\partial Z} \cdot \frac{\partial Z}{\partial w'}.$$

$$\therefore \frac{\partial L(w)}{\partial w_2} \Rightarrow \frac{\partial L(w)}{\partial O_2} \cdot \frac{\partial O_2}{\partial W_2}$$

$$\frac{\partial L(w)}{\partial W_1} = \frac{\partial L(w)}{\partial O_2} \cdot \frac{\partial O_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial Z} \cdot \frac{\partial Z}{\partial W_1}$$