Neural Networks

Lokesh Mohanty

April 22, 2023

Contents

1 Introduction

If $h_\theta(x)$ is our hypothesis function, then

$$h_\theta(x) = w_2^T \sigma_1 \left(w_1^T x\right)$$

where w_1, w_2 are the parameter matrices for layer 1 and 2 and σ_1 is the activation function of layer 1. By definition σ is an asymptotic function $\left(\text{i.e., } \sigma(x) = \begin{cases} 0, x \to -\infty \\ 1, x \to \infty \end{cases}\right)$.

$$x \in \mathbb{R}^d, y \in \mathbb{R}^k$$
$$w_1 \in \mathbb{R}^{d \times l}, w_2 \in \mathbb{R}^{l \times k}$$

$w_{jk}^l \to$ weight from k^{th} neuron in $(l-1)^{th}$ layer to j^{th} neuron in the l^{th} layer

In neural networks we just use a complex hypothesis function which can be a universal approximator. We can have a single or multiple layers of neurons which together forms the hypothesis function. This is also called a *Multi-Layer Perceptron (MLP)*. If all the neurons are connected, it is called a *Fully connected neural network.*

2 Universal Approximation Theorem (UAT)

Suppose $f(x) : \mathbb{R}^d \to \mathbb{R}^k$

$$\begin{aligned} h_\theta(x) &= w_2^T \sigma_1(w_1^T x) \\ w_1 \text{ and } w_2 \text{ s.t. } &|f(x) - h_\theta(x)| \leq \epsilon \\ \epsilon &> 0, \delta > 0 \end{aligned}$$

3 Error-back Propagation (Chain rule)

We need to find the hypothesis function by training the parameters. This is done by empirical risk minimization. We use a method *Error-back propagation* to train the parameters.

Activation Notations:

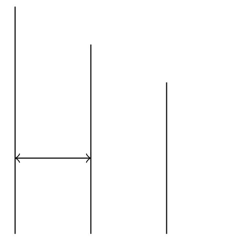
- $a^l = \sigma\left(w^l a^{l-1} + b^l\right)$
- a_j^l = activation of j^{th} neuron in l^{th} layer
- $a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$
- $z^l \triangleq w^l a^{l-1} + b^l$

$$
\begin{aligned}
\frac{\partial \hat{R}}{\partial w_{jk}^l} &= ? \\
\delta_j^L &= \frac{\partial \hat{R}}{\partial z_j^L} \\
&= \frac{\partial \hat{R}}{\partial a_j^L} . \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial \hat{R}}{\partial a_j^L} \sigma'(z_j^L) \\
\implies \delta_j^L &= 2(a_j^L - y_j)\sigma'(z_j^L)
\end{aligned}
$$

$$
\begin{aligned}
\hat{R}(h) &= \sum_{i=1}^{n} \left(h_\theta(x_i) - y_i\right)^2 \\
h_\theta(x_i) &= a_i^L,\, a^L = \sigma\left(w_2^T \sigma\left(w_1^T x + b_1\right)\right) \\
\implies \hat{R} &= \sum_{j=1}^{k} \left(y_j - a_j^L\right)^2 \\
\implies \frac{\partial \hat{R}}{\partial a_j^L} &= 2\left(a_j^L - y_j\right) \\
\frac{\partial \hat{R}}{\partial w_{jk}^l} &= a_k^{l-1} . \delta_j^l
\end{aligned}
$$

3.1 Error-back Propagation (In matrix-vector form)

Date: 23/03/2023



Calculating in Matrix-Vector form

$$\begin{aligned}
h_\theta(x)/a_3 &= \sigma(w_3 a_2)\\
a_2 &= \sigma(w_2 a_1)\\
a_1 &= \sigma(w_1 x)
\end{aligned}$$

$$\begin{aligned}
\hat{R}(h) &= \frac{1}{2}\|h_\theta(x) - y\|_2^2\\
w^{t+1} &= w^t - \alpha\frac{\partial \hat{R}(h)}{\partial w}
\end{aligned}$$

$\frac{\partial \hat{R}}{\partial w_3}$

$$\begin{aligned}
\frac{\partial \hat{R}}{\partial w_3} &= (a_3 - y).\frac{\partial a_3}{\partial w_3}\\
&= (a_3 - y) \odot \sigma'(w_3 a_2).\frac{\partial (w_3 a_2)}{\partial w_3}\\
&= (a_3 - y) \odot \sigma'(w_3 a_2).a_2^T\\
\delta_3 &\triangleq (a_3 - y) \odot \sigma'(w_3 a_2)\\
\implies \frac{\partial \hat{R}}{\partial w_3} &= \delta_3 a_2^T
\end{aligned}$$

$\frac{\partial \hat{R}}{\partial w_2}$

$$
\begin{aligned}
\frac{\partial \hat{R}}{\partial w_2} &= (a_3 - y).\frac{\partial a_3}{\partial w_2} \\
&= (a_3 - y) \odot \sigma'(w_3 a_2).\frac{\partial (w_3 a_2)}{\partial w_2} \\
&= \delta_3.\frac{\partial (w_3 a_2)}{\partial w_2} \\
&= w_3^T \delta_3.\frac{\partial a_2}{\partial w_2} \\
&= w_3^T \delta_3 \odot \sigma'(w_2 a_1).\frac{\partial (w_2 a_1)}{\partial w_2} \\
\delta_2 &\triangleq w_3^T \delta_3 \odot \sigma'(w_2 a_1) \\
\implies \frac{\partial \hat{R}}{\partial w_2} &= \delta_2 a_1^T
\end{aligned}
$$

$\frac{\partial \hat{R}}{\partial w_1}$

$$
\frac{\partial \hat{R}}{\partial w_1} = \left[w_2^T \delta_2 \odot \sigma'(w_1 x) \right] x^T
$$

In general,

$$
\begin{aligned}
\delta_L &= (h_\theta(x) - y) \odot \sigma'(w_L a_{L-1}) \\
\delta_i &= w_{i+1}^T \delta_{i+1} \odot \sigma'(w_i a_{i-1}) \\
\frac{\partial \hat{R}}{\partial w_i} &= \delta_i a_{i-1}^T \\
w_i^{t+1} &= w_i^t - \alpha \frac{\partial \hat{R}}{\partial w_i}
\end{aligned}
$$

Here, α is the learning rate which can different for different weights

4 Activation functions

- Sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Tanh function

$$\sigma(x) = tanh(x)$$

- Rectified Linear unit function (ReLU)

$$\sigma(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$

- Leaky ReLU function

$$\sigma(x) = \begin{cases} x, x > 0 \\ \alpha x, x \leq 0 \end{cases} \qquad \alpha << 1$$

5 Regularization with Neural Networks

$$\hat{R}(h) + \lambda\Omega(w)$$

Epoch: Computation of empirical risk over the whole dataset

Batch Gradient Descent: Instead of computing empirical risk over the whole dataset, we sample some $n_B (n_B << n)$ datapoints from the dataset without replacement and compute the empirical risk for these data points.

$$\hat{R}(h) = \sum_{i \in B} l(h(x_i), y_i)$$
$$B \rightarrow \text{Batch},$$
$$n_B \rightarrow \text{batch size}$$

When $n_B = 1$ it is called **on-line gradient descent** or **sequential gradient descent** or **stochastic gradient descent**. (Different definitions are given by different authors)

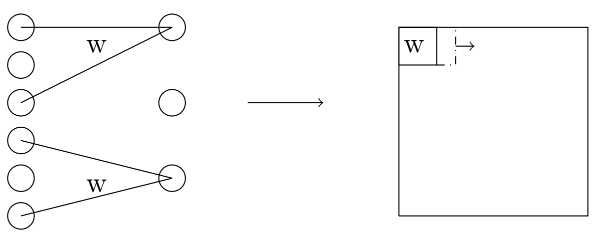
6 Convolutional Neural Networks (CNN)

Date: 23 March 2023, 28 March 2023

Imposes a strong prior on an MLP (multi-layer perceptron or feed-forward neural network) by a structural modification. This regularizes the MLP.

Notations:

- Receptive field of a neuron: the set of neurons that send information to this neuron
- **Local receptive field**: the neurons that a neuron is connected to (in brain)
- **Parameter sharing**:



Note:

- Stride: hyperparameter
- Channel: number of weight matrices in the previous layer
- Average Pooling, Max pooling(non-differentiable, heuristics are used for back propagation)

Other CNN based neural networks

- ResNet-50:
  Residual block($x + \sigma(w^T x)$) with 50 layers (residual block on every layer)

- VGG

7 Recurrent Neural Networks

Eg: Machine translation (any time series data)
Datapoint: sentence (collection of words) Cannot use MLP as every datapoint has a different dimension

Problems:

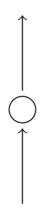
- different input lengths

- temporal dependency

Note:

- CNN: parameter sharing across space

- RNN: parameter sharing across time



Notations:

- X: a sentence

- $x^{(i)}$: a sentence

- $x_t^{(i)}$: a word (represented by a one-hot vector with length as the number of words in dictionary)

- BPTT: Back propagation through time

7.1 Problem: Vanishing gradient

Since the weight (w) is same for all words(across time), the gradient can get close to zero (i.e., the past is forgotten) during backpropagation.

7.2 LSTM

Date: 30/03/2023
Selectively remembers things.
GRU: Gated recurrent unit
Blog: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

$$
\begin{aligned}
(\textit{forget gate})\ f_t &= \sigma\left(W_f\left[a_{t-1}, x_t\right] + b_f\right) \\
(\textit{input gate})\ i_t &= \sigma\left(W_i\left[a_{t-1}, x_t\right] + b_i\right) \\
(\textit{output gate})\ o_t &= \sigma\left(W_o\left[a_{t-1}, x_t\right] + b_o\right) \\
(\textit{gate gate})\ g_t &= \sigma\left(W_c\left[a_{t-1}, x_t\right] + b_c\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
a_t &= o_t \odot \sigma(c_t)
\end{aligned}
$$

7.2.1 Gated Recurrent Unit (GRU)

$$
\begin{aligned}
z_t &= \sigma\left(W_z\left[a_{t-1}, x_t\right]\right) \\
\tilde{a}_t &= \sigma\left(W\left[a_{t-1}, x_t\right]\right) \\
a_t &= (1 - z_t) \odot a_{t-1} + z_t \tilde{a}_t
\end{aligned}
$$

8 Classification and Regression Trees (CART)

Decision trees:

- Non-parametric

Tree - Splitting

$$
\begin{aligned}
\text{Let, } D_k \subseteq D,\ D_k &= \{(x, y) \in S : y = k\} \\
D &= D_1 \cup D_2 \cup ... \cup D_c \\
\text{Define } P_k &= \frac{|D_k|}{|D|} \text{ [ML estimate]}
\end{aligned}
$$

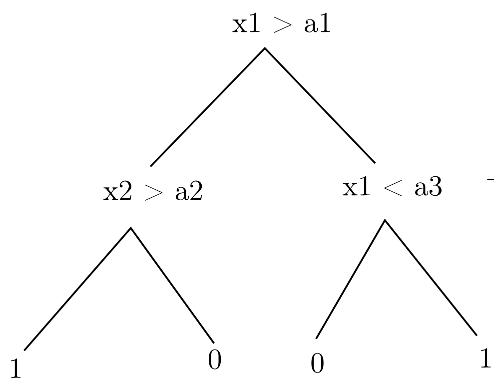


Figure 1: Tree splitting

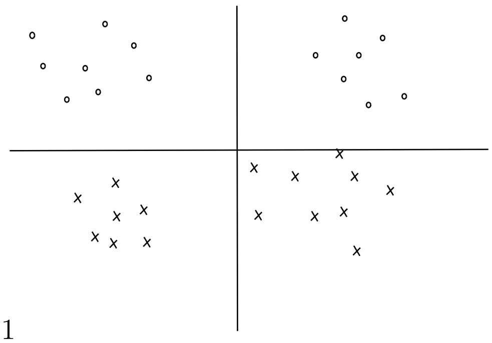


Figure 2: Data

Given a split, Let $P_1, ..., P_k$ be the corresponding probability. The split is a bad split if the distribution of a class over all the points is uniform. Hence we split the data inorder to maximize the KL-divergence between the class distribution and the uniform distribution (impure)

Maximize KL-divergence (i.e., decrease impurity)

$$\begin{aligned}
\max_p D_{KL}(p \parallel q) &= \sum_{i=1}^{c} p_i \log \frac{p_i}{q_i} \\
&= \sum (p_i \log p_i + p_i \log c) \\
&= \sum p_i \log p_i + \log c \sum p_i
\end{aligned}$$

Gini Index (another measure of impurity)

$$G(D_i) = \sum_{i=1}^{c} p_i(1 - p_i)$$

9 Notes

- Refer Prof. Sashtri's lectures for Statistial learning theory
- Reference(youtube): Cornell CS4780
- Convolution $\rightarrow$ flip the matrix then correlate (search convolution vs correlation)
- Finding $P_{y|x}$: discriminative
- Finding $P_x, P_{x|y}, P_{xy}$: generative + sampling
- Distributional shift: test and training data have different distribution

10 Homework

- Work out **BPTT** manually for RNN and GRU
- See that magnitude of gradient of loss with respect w for RNN depends on eigenvalues of w while in case of GRU it doesn't not.

11 Hacks

- Run grid search over various random seeds