CSE 475: Statistical Methods in AI **Monsoon 2019**

SMAI-M-2019 17: Neural Networks: Multi Layer Perceptrons

Lecturer: C. V. Jawahar *Date: 10 Oct, 2019*

17.103 Introduction

A neural network (or more precisely artificial neural network (ANN)) consists of a set of neurons arranged in a specific manner. Based on the mathematical model of the neuron, their organisation and the learning algorithm employed, the network becomes capable of solving various pattern recognition tasks. A neuron typically takes inputs from various neurons and outputs its activation state. Inputs are usually weighed and added within a neuron. Let w_{kj} is the weight of connection between jth and kth neurons. Adder within the neuron sums the weighted inputs. Output is a nonlinear activation function of this weighted sum. Activation function also limits the value of the output signal to some finite value

Neural netwrok as a computational model has many contrasting difference with the traditional ones including Sequential Vs. Parallel. CPUs are complex architectures while neurons are simple structures. Neurons are typically 5 to 6 order slower than silicon gates. While Number of instructions per sec. is a popular measure of complexity in the traditional computing, number of interconnections or number of weights has very strong role in the neural netwroks. Traditionally, we formulate a mathematical model and validate with real data. In NN learning, we identify the model directly from the data

17.103.1 Biological Neural Networks

Neural networks are inspired by our brains. The human brain has about 10^{11} neurons and 10^{14} synapses. A neuron consists of a soma (cell body), axons (sends signals), and dendrites (receives signals). A synapse connects an axon to a dendrite. Given a signal, a synapse might increase (excite) or decrease (inhibit) electrical potential. Neurons are densely interconnected with each other. Connections between neurons need not be simple and layered. Each neuron could be receiving inputs from approximately 10^5 synapses.

A neuron receives signals from connected neurons via dendrites. The cell body sums up the received signals. The neuron then fires if the combined weighted input exceeds a threshold. By neuron firing, we mean that it generates an output which is sent out through axons to the next set of neurons. If the weighted sum is below the threshold, no response signal is generated by the neuron (i.e., the output is zero or neuron did not fire). The threshold decides whether a neuron fires or not is called activation function. In other words, a neuron fires when its electrical potential reaches a threshold. Learning might occur by changes to synapses.

17.103.2 Brief History of ANNs

Neural network research has rich history. There are biological and cognitive motivation for understanding the neural netwroks. There are also engineering interest in understanding ANNs as a mathematical tool to model the complex systems. Neural networks have also gone through many ups and downs. They played big role in the success of deep learning and modern AI.

17.104 Artificial Neural Networks

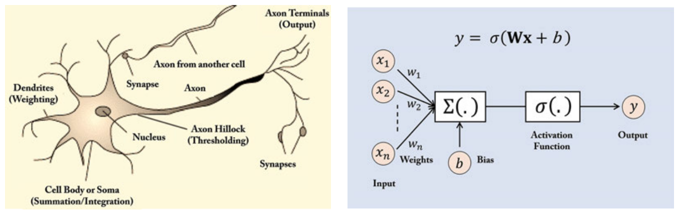

Figure 17.14: Biological and Arficial Neuron Models

17.104.1 Neuron Model

The computational model used popularly today is a simple weighted addition of inputs passing through a nonlinear function. i.e.,

$$y = \phi(\sum_{i=1}^{d} w_i x^i)$$

17.105 Feed Forward and Feedback Networks

Neural networks can be broadly classified as feed forward and feedback. In the case of feedforward networks, the signal flow (or information flow) takes place only in one direction. i.e., forward. If we look at this network as a graph, they are directed acyclic graphs. Examples include MLP and CNN. While in the case of feedback networks, output signal is fedback to the network, or sometime the same neuron. Examples include RNNs with popular implementations like LSTM and GRU. Due to the feedback nature of the connections, it some time becomes nontrivial to analyze, model or even understand such networks.

17.106 Multi Layer Perceptrons

17.106.1 Single Layer Perceptrons

We know about the linear classifiers that classify based on the simple rule:

$$sign(\mathbf{w}^T\mathbf{x}).$$

Such a classifier classifies a sample into positive class if $\mathbf{w}^T\mathbf{x}$ is ≥ 0 and negative class if $\mathbf{w}^T\mathbf{x}$ negative. We also know the perception algorithm that learns the $\mathbf{w}$ for a linear separable problem.

We can look at this as a single layer neural network with inputs as $x^1, x^2, \ldots, x^d$ and the weights (on the edges) $w_1, w_2, \ldots w_d$ gets multiplied and added.

There are two computations that are happening with in the neuron. (1) Multiply and add (i.e., compute $\mathbf{w}^T\mathbf{x}$. (2) pass through the nonlinearity $\phi()$, also known as the activation. In this case the nonlinearity $\phi()$ is

$$\phi(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

In this case $\phi(\cdot)$ is practically the $sign()$ function.

Single Layer Perceptron or simple perceptron has only one layer and the above mentioned activation function. This can classify samples into two class with a linear decision boundary.

Activation function: This nonlinearity $\phi()$ is often called activation function. A disadvantage of the above activation function is that it is not differentiable. A logistic/sigmoid function is often used as the nonlinearity

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

As shown in the figure 17.15, this function ranges between 0 and 1.

Figure 17.15: Sigmoid or Logistic Function $\phi(x) = \frac{1}{1+e^{-x}}$

17.106.2 MLP

In Multi Layer Perceptron (MLP), we connect or concatenate multiple perceptrons and model a complex function class. We can connect many single layer neural networks to form a multi layer neural network. A typical MLP has

- **Nodes and Edges** Network has nodes (where simple computations take place and edges where information flow happen.
- **Layered architecture.** Network is understood and represented in layers.
- **Dense Connections** Successive layers are often fully connected.

One way to appreciate a MLP/Neural network as a learniable function from X to Y. Eventually this network models the transformation of the input $\mathbf{x}_i$ to $\mathbf{y}_i$. From this point of view, MLPs can be used for either regression or classification. In other words, $\mathbf{y}$ could be a single intiger, real number or multiple integers/real numbers.

17.106.3 MLP for Classification

MLPs are very good for the classification. In the case of binary classification, one can design a network with only one output neuron. This neuron could output the probability being class-1. And a complement of this could be class-2. This can be achieved by keeping a sigmoid or tanh at the output neuron.

What about multiclass classification? Assume there are four classes, how many neurons are required in the output space? One solution could be to use two neurons and encode the classes as 00, 01, 10 and 11. However, this

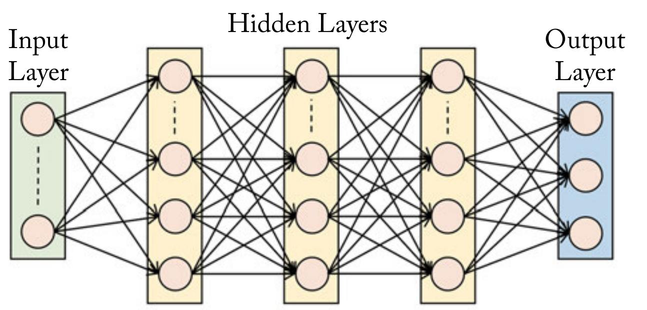

Figure 17.16: A typical MLP with three hidden layers.

assumes a structure at the output space (eg. a specific class is larger than the other). This is not preferred. The preferred solution is to have four output neuron and each outputting the probability (or confidence) for certain class. What if two classes are fired at a time.? To avoid this, one can use softmax at the output layer.

In other words, in the case of classification, it is the practice to represent the class-ID as a one hot vector. i.e., if there are C classes, there will be C neurons in the output. The the desired class is p, then we represent the output as a C dimensional vector with zero every where except at pth location. Output is often a softmax

$$\frac{e^{x^j}}{\sum_{i=1}^{K} e^{x^i}}$$

17.106.4 MLP for Regression

MLPs are popular for regression as well as classification. In the case of regression, the output neurons predict a real quantify.

In some cases, the output range need not be $[0, 1]$ or $[-1, 1]$. For example, we want to design a network that will predict the age of a person from a photograph. In such cases, output could be even a simple linear activation i.e., $x = \phi(x)$.

17.107 Notations

The layer to layer transition in a typical MLP can be understood as a matrix multiplication followed by the activation computation.

Let $\mathbf{x}$ is the input. Then the output of the next layer is represented as:

$$\mathbf{y} = \phi(\mathbf{Wx})$$

Representing the network in this manner (than using, i,j,k notations makes the representations compact.

17.108 Design Choices

17.108.1 Why do we need a nonlinearity?

In typical MLPs, it is assumed that all Neurons in a layer gets connected to all the neurons in the next layer. i.e., the layer is fully connected. Say $\mathbf{x_i}$ is the representation (output of the neurons at the ith layer and $\mathbf{x_{i+1}}$ is the representation at the $i+1$ th layer, then we can compute

$$\mathbf{x_{i+1}} = \phi(\mathbf{W}\mathbf{x}_i)$$

If there was no nonlinearity, then multiple layer perceptron could have reduced to a single layer perceptron.

Example: If $\mathbf{x_2} = \mathbf{W}'\mathbf{x}_1$ and $\mathbf{x_3} = W''\mathbf{x}_2$, then we can in fact write $\mathbf{x_3} = \mathbf{W}'''\mathbf{x_1}$. For some $\mathbf{W}''' = \mathbf{W}' \cdot \mathbf{W}''$

17.108.2 Architecture

A typical MLP what is given to us is the input, output pairs $(\mathbf{x_i}, \mathbf{y_i})$ $i = 1, \ldots N$. Typically $\mathbf{x}$ and $\mathbf{y}$ are vectors. The number of neurons in the first/input layer is the dimensionality of $\mathbf{x}$. Number of neurons in the output layer is the dimensionality of $\mathbf{y}$. We have two things in our control (i) Number of hidden layers (ii) number of neurons in each hidden layer. Typically, we go for 2 or 3 hidden layers. With number of layers increasing, the network becomes deep and the learning problem becomes difficult due to issues like vanishing gradient.

The number of neurons in each layer is typically larger than the number of neurons that are required at the input or output layers. Note that typical neural networks are over parameterized. i.e., there are more neurons and weights than what is required for solving the problem. (Indeed it may be possible to prune and get a smaller network once the network is trained.)

17.108.3 Loss Function

The objective of the network is to minimze the loss functions (or objective functions) during the training. The choice of loss function may depend on the problem. However, there are a set of popular loss functions that works for most problems.

Let there are K output neurons. Let the true output/target be $\mathbf{t}$ and the prediction/output is $\mathbf{o}$.

1. **MSE** Mean Square Error (MSE) is a popular loss function

$$\mathcal{L}(\mathbf{o}, \mathbf{t}) = \frac{1}{2N} \sum_{i=1}^{K} (t_i - o_i)^2$$

2. **L1**

$$\mathcal{L}(\mathbf{o}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{K} |t_i - o_i|$$

3. **Cross Entropy**

$$\mathcal{L}(\mathbf{o}, \mathbf{t}) = \sum_{i=1}^{K} t_i \log(o_i)$$

If $\mathbf{t}$ and $\mathbf{o}$ are probability distributions (desired and output), then the cross entropy loss is equivalent to the KL divergence between these two distributions.

$$D_{KL}(\mathbf{t}||\mathbf{o}) = \mathcal{L}(\mathbf{o}, \mathbf{t}) - H(\mathbf{t})$$

$$D_{KL}(\mathbf{t}||\mathbf{o}) = \mathcal{L}(\mathbf{o}, \mathbf{t}) + \sum_i t_i \log t_i$$

$$D_{KL}(\mathbf{t}||\mathbf{o}) = \sum_i t_i \log(o_i) + \sum_i t_i \log t_i = \sum_i t_i \log(\frac{o_i}{t_i})$$

(double check equations!!)

- Q: Show that KL Divergence is non-negative
- Q: Is KL Divergence symmetric?
- Q: What is "Bhattacharya distance"?

4. **Hinge Loss**

$$\mathcal{L}(\mathbf{o}, \mathbf{t}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} \max(0, 1 - \mathbf{w}^T \mathbf{x}_i t_i)^2$$

Understand the Hinge loss as:

$$\max(0, 1 - wT.x_i.t_i)$$

where target t is ± 1 and o is the raw output (not thresholded).

- What happens when $t \cdot o$ is negative? sample is misclassified and the error is high.
- When the $t \cdot o$ is larger than 1? Loss is zero
- When $t \cdot o \in (0, 1)$?

Read: Deep Learning using Linear Support Vector Machines

5. **Regularized Losses**

$$\mathcal{L}_R = \mathcal{L} + ||\mathbf{w}||_2^2$$

$$\mathcal{L}_R = \mathcal{L} + ||\mathbf{w}||_1$$

Like in other problems, we can (and should regularize the weights).

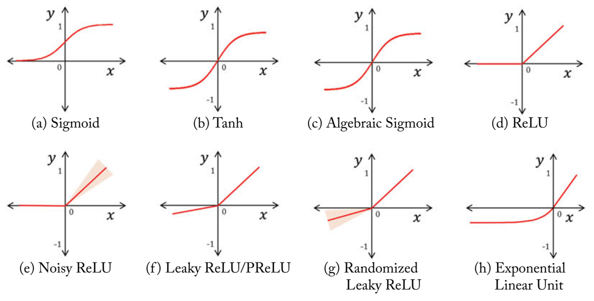

Figure 17.17: A set of popular activations/nonlinearities used in todays neural networks

17.108.4 Activation Functions

Here are some examples of popular activation functions:

1. **Sigmoid**

$$y = \frac{1}{1 + e^{-x}}$$

2. **tanh**

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

3. **ReLu**

$$y = \max(0, x)$$

4. **Leakly Relu**

$$y = \begin{cases} x & \text{if } x > 0 \\ cx & \text{if } x \leq 0 \end{cases}$$

5. **Noisy ReLu**

$$y = \max(0, x + \epsilon)$$

$$\epsilon = \mathcal{N}(0, \sigma(x))$$

6. **Exponential Linear Unit**

$$y = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

- Q: Write the expression for the other two activations.
- Q: Write expressions for $\phi'(x)$

17.109 Expressive Power of MLP

17.109.1 AND and OR

Consider the problem of implementing "AND" and "OR" with single layer perceptrons.

x^1	x^2	AND	OR	EXOR
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	1	1	0

Let us implement these logics with a single layer neural network with an activation $\phi(x) = 1$ iff $x \geq 0$. With the introduction of bias, what we want to lean is $w_0, w_1 and w_2$.

It can be seen that the following weights satisfy:

- **AND**: $w_0 = w_1 = w_2 =$
- **OR**: $w_0 = w_1 = w_2 =$

Q: Are they the only possible weights that satisfy?

Q: Can we get a valid solution with no bias?

Q: Can you find weights corresponding to NAND and NOR?

Q: Here, we used $0, 1$ logic. Assume we use $-1, 1$ representation, can you redesign the networks?

If we plot this data, we can see that these are linearly separable.

17.109.2 EXOR

However, that is not true for EXOR. It can be easily seen that no solution of the form $w_2x^2 + w_1x^1 + w_0$ is going to work for EXOR.

17.109.3 Representational Power

17.110 Learning

The key question is on "How do we train the neural network." This is possibly more important than how do we decide the number of neurons in each layer or number of layers for a beginner.

Even for an experiences, the learning process is not that simple. Experience in carefully doing an experiment is required to get the best.

The popular algorithm for training neural networks is called "Error Backpropagation Algorithm" or popularly known as "backpropagation algorithm (BP)".

Homeworks

1. Design a neural network each with two inputs and one output to implement AND, OR, ExOR, NAND and NOR logics with +1,-1 convention. Neurons have an activation of *sign*(). Write Truthtables, Draw architectures, write weights and any other associated details.

2. We know how to design a network that can solve ExOR problem. Let us consider an extension of it from 2 input to four, and cast the problem as "parity" problem. (i.e., whether the number of 1s are even or odd.). Write the truth table. Design a network with *sign*() activations. (if required, write some code while designing!).

3. Consider a sequential data (like price of a vegitable over months). This vary over time and often is a sequence. However, there is some pattern in this sequence. Let us make a simple assumption and create a data as:

$$\begin{aligned} x(n) = \alpha_1 x(n-1) + \alpha_2 x(n-2) + \alpha_3 x(n-3) \\ + \alpha_4 x(n-4) + \alpha_5 x(n-5) + N(0, \sigma^2) \end{aligned}$$

   Choose different values for α_i (including negative) and also appropriately taking care of initial conditions, plot this data (notice some trend, patterns in the data!)

   Now we would like to model this problem as a regression problem that minimize

$$\mathcal{E}(\mathbf{w}) = \sum_k (x(k) - \sum_{i=1}^{d} w_i x(k-i))^2$$

   And solve this problem using SLP.

   (a) Find situations where we are able to reliably estimate the model (and of course do prediction of priced in the future) in presence of noise. Show true and one step ahead predicted data plotted toghether.

   (b) Show some situations where this prediction fails. Why?

   (c) Start with a data where reliable estimate of α is feasible. Now create a plot of $\mathcal{E}$ vs d. What do we observe here?