Linear Data Analysis
Classification - Single Artificial Neuron

Cain Susko

Queen's University
School of Computing

March 21, 2022

a Artificial Neurons

Artificial Neural Networks are made up of Artificial Neurons. We can think of operations with Neural Networks as finding a hyper plane of separation. There is also a biological analogy that can help explain neural networks and their neurons.

b Data Flow and Computation for Neurons

Simple Model The simple model of Data flow within a NN is made up of:

inputs	x_j
weights	w_j
bias / threshold	β

Within this simple model an neuron will fire when the voltage or input exceeds the bias. This can be represented as the equation:

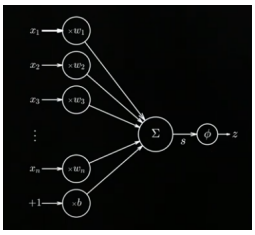
$$\sum_{j=1}^{n} x_j w_j \geq \beta$$

or

$$\sum_{j=1}^{n} x_j w_j + \beta \geq 0$$

The output of a Neuron is it's **activation function** ϕ

Data Flow Through a Neuron For each neuron within a neural network, it may have many inputs, which are first weighted, then summed and fed as input to the activation function ϕ (if the neuron has one)



Linear Algebra An observation is an input vector $\vec{x}_i$. The weighted sum of this observation is:

$$u_i = u(\vec{x}_i) = \vec{x}_i^\top \vec{w} + \beta$$

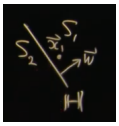
Thus, a neuron with this input of a weighted sum only fires if and only if:

$$u_i \geq 0$$

When training a Neural Network a user will provide labels for each initial data vector. For this course the labelling will be binary with each label $y_i \in \{0, 1\}$. This differs from other machine learning literature which may use $y_i = \pm 1$.

c Hyperplane of Separation for Neurons

This Section will explore how a Neural Network Does Hyperplane Operations. Recall a Hyperplane $\mathbb{H}$ is:



Where $\vec{w}$ is the weight vector associated with $\mathbb{H}$. S_1, S_2 are the 2 sides of $\mathbb{H}$. The question is how does the Neural Network decide which side $\vec{x}_1$ is on.
In terms of Neural Networks, a hyperplane $\mathbb{H}$ is $\vec{w}, \beta$ which are its axis and bias. Therefore, the pseudo-distance from $\mathbb{H}$ for each point $\vec{x}_i$ is:

$$u_i = \vec{x}_i^\top \vec{w} + \beta$$

Such that:

$$u_i \geq 0 \rightarrow \vec{x}_i \in S_1$$

$$u_i < 0 \rightarrow \vec{x}_i \in S_2$$

We then Augment the weight–and thus input vector as well–such that:

$$\hat{w} = \begin{bmatrix} \vec{w} \\ \beta \end{bmatrix} \quad \hat{x}_i = \begin{bmatrix} \vec{x}_i \\ 1 \end{bmatrix}$$

Thus, the previous computation with these augmented values would be made faster with the equation:

$$u_i = \hat{x}_i^\top \hat{w} = \vec{x}_i^\top \vec{w} + 1 \times \beta$$

Not only does it make it faster but in out code the bias scalar is better integrated into the equation in the augmented vector.

General Activation A General Activation function ϕ maps a real number to another real number

$$\phi : \mathbb{R} \to \mathbb{R} \text{ or } z = \phi(u)$$

Such that the linear algebra interpretation of ϕ is the **score** of the input for each neuron. The score is then put through a quantization function, maps the score to a set of 'labels':

$$q : \mathbb{R} \to \{0, 1\} \text{ or } q(\phi(u_i)) = \{0, 1\}$$

Therefore, The classification function for points on 2 sides of a hyperplane implemented with a Neural Network is:

$$H(u_i) =^{def} \begin{cases} 1 \text{ if } u \geq 0 \\ 0 \text{ if } u < 0 \end{cases}$$

Where u_i is the pseudo-distance from a point $\vec{x}_i$ to the hyperplane $\mathbb{H}$. Such that if $\vec{x}_i$ is on the negative side of $\mathbb{H}$ then u_i will be negative. Additionally, when $\vec{x}_i$ is on the positive side of the hyperplane, u_i is positive. This is the reasoning behind the classification function H, or the activation function of the neuron.

Binary Classification

- each $\hat{x}_i$ has a label $y_i \in \{0, 1\}$
- each label representes the 2 sides of the hyperplane such that:

$$y_i = \begin{cases} 0 \text{ if } \hat{x}_i \in S_2 \\ 1 \text{ if } \hat{x}_i \in S_1 \end{cases}$$

- thus, the ideal match is:

$$(y_i = 0) \leftrightarrow (\hat{x}_i^\top \hat{w} < 0)$$
$$(y_i = 1) \leftrightarrow (\hat{x}_i^\top \hat{w} \geq 0)$$

Learning Outcomes

Students should now be able to:

- Augment data vectors for neural computation
- Use augmented vectors to find pseudo distance
- Quantize pseudo-distance as the activation of the neuron