PadhAI Week 2: MP Neuron & Perceptron

by Manick Vennimalai

Greetings, these are my notes for week 2 of the PadhAI course, covering the topics of MP Neuron and Perceptron.

2.1: MP Neuron

Artificial Neuron:

- Takes in a series of inputs[$x_0..x_n$], adds weights [$\theta_0..\theta_n$], applies a function like the sigmoid function and returns the transformed output

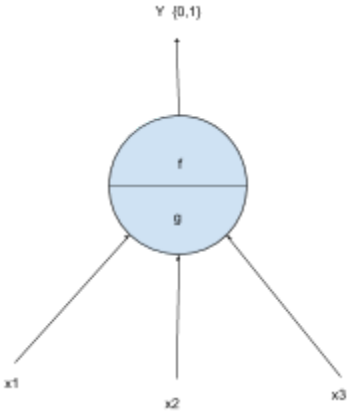
McCulloch-Pitts Neuron:

- The early model on an artificial neuron is introduced by Warren McCulloch(neuroscientist) and Walter Pitt(logician) in 1943
- The McCulloch-Pitts Neuron is also known as a linear threshold gate

We are going to be applying the 6-jars framework to learn about the MP neuron

2.1.1: MP Neuron Model

What is the mathematical model?



1. Inputs belong to the discrete set of values {0,1}
2. g aggregates the inputs and function f takes a decision based on these aggregations
3. These inputs can be excitatory or inhibitory
   a. y = 0 if x_i is inhibitory (outputs zero, sort of an override), else
   b. g(x) = $\Sigma^n_{i=1}x_i$
   c. y = f(g(x))
      i. y = 1 if g(x) >= b
      ii. y = 0 if g(x) < b
      iii. Where b is a threshold value
      iv. b is a parameter, it is adjusted with the aim of maximizing the number of correct predictions

2.1.2: MP Neuron Data and Task

What kind of data and tasks can MP neuron process

1. Consider the example of detecting whether the cricketer is out by LBW
2.

Pitch in line(x_1)	Impact(x_2)	Missing Stumps(x_3)	Is it LBW(y)
1	0	0	0
0	1	1	0
1	1	1	1
0	1	0	0

3. We are interested in finding out the relationship between y and x_i
4. Here we use y = ($\Sigma^{3}_{i=1} x_i$ >= b)
   a. y = 1 if g(x) >= b
   b. y = 0 if g(x) < b
5. In case our data has non-boolean inputs, we can convert them to a boolean form
6. For example, consider the following boolean-ised phone spec data
7.

	phone 1	phone 2	phone 3	phone 4	phone 5	phone 6	phone 7	phone 8	phone 9	phone 10
Launch (within 6 months) x_1	0	1	1	0	0	1	0	1	1	0
Weight (<160g) x_2	1	0	1	0	0	0	1	0	0	1
Screen Size (< 5.9in) x_3	1	0	1	0	1	0	1	0	1	0
Dual sim x_4	1	1	0	0	0	1	0	1	0	0
Internal mem(>= 64gb, 4gb ram) x_5	1	1	1	1	1	1	1	1	1	0
NFC x_6	0	1	1	0	1	0	1	1	1	0
Radio x_7	1	0	0	1	1	1	0	0	0	0
Battery (>= 3500mAh) x_8	0	0	0	1	0	1	0	1	0	0
Price? (> 20k) x_9	0	1	1	0	0	0	1	1	1	0
Liked (y)	1	0	1	0	1	1	0	1	0	0

2.1.3: MP Neuron Loss

How do we compute the loss

1. Consider the previous example

	phone 1	phone 2	phone 3	phone 4	phone 5	phone 6	phone 7	phone 8	phone 9	phone 10
Launch (within 6 months) x_1	0	1	1	0	0	1	0	1	1	0
Weight (<160g) x_2	1	0	1	0	0	0	1	0	0	1
Screen Size (< 5.9in) x_3	1	0	1	0	1	0	1	0	1	0
Dual sim x_4	1	1	0	0	0	1	0	1	0	0
Internal mem(>= 64gb, 4gb ram) x_5	1	1	1	1	1	1	1	1	1	0
NFC x_6	0	1	1	0	1	0	1	1	1	0
Radio x_7	1	0	0	1	1	1	0	0	0	0
Battery (>= 3500mAh) x_8	0	0	0	1	0	1	0	1	0	0
Price? (> 20k) x_9	0	1	1	0	0	0	1	1	1	0
Liked (y)	1	0	1	0	1	1	0	1	0	0
Prediction $\hat{y}$	1	0	0	1	1	1	1	0	0	0
loss	0	0	1	-1	0	0	-1	1	0	0
Square error	0	0	1	1	0	0	1	1	0	0

2. Take the square of the difference to ignore the sign.
3. cost/loss = $\Sigma_i(y_i - \hat{y}_i)^2$

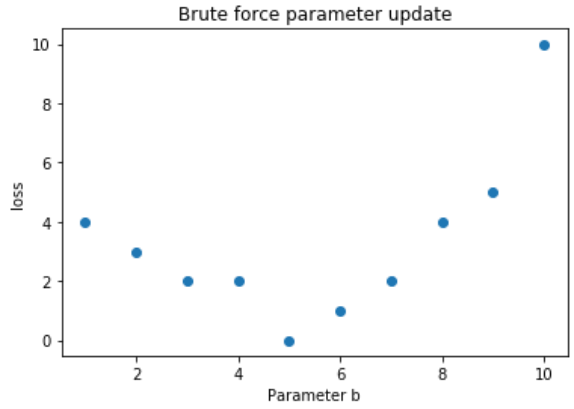
2.1.4: MP Neuron Learning Algorithm

How do we train our model

1.

	phone 1	phone 2	phone 3	phone 4	phone 5	phone 6	phone 7	phone 8	phone 9	phone 10
Launch (within 6 months) x_1	0	1	1	0	0	1	0	1	1	0
Weight (<160g) x_2	1	0	1	0	0	0	1	0	0	1
Screen Size (< 5.9in) x_3	1	0	1	0	1	0	1	0	1	0
Dual sim x_4	1	1	0	0	0	1	0	1	0	0
Internal mem(>= 64gb, 4gb ram) x_5	1	1	1	1	1	1	1	1	1	0
NFC x_6	0	1	1	0	1	0	1	1	1	0
Radio x_7	1	0	0	1	1	1	0	0	0	0
Battery (>= 3500mAh) x_8	0	0	0	1	0	1	0	1	0	0
Price? (> 20k) x_9	0	1	1	0	0	0	1	1	1	0
Liked (y)	1	0	1	0	1	1	0	1	0	0
Prediction $\hat{y}$	?	?	?	?	?	?	?	?	?	?

2. $\hat{y}$ = ($\Sigma^{n}_{i=1} x_i >= b$)
3. cost/loss = $\Sigma_i (y_i - \hat{y}_i)^2$
4. In this case, we have only one parameter, so we can afford to use brute force search.
   a. Here, consider we have n features
   b. b can only range from 0 to n, else it would be a pointless parameter
   c. b has discrete values only, as the inputs are also discrete values



2.1.5: MP Neuron Evaluation

1. Training Data

	phone 1	phone 2	phone 3	phone 4	phone 5	phone 6	phone 7	phone 8	phone 9	phone 10
Launch (within 6 months) x_1	0	1	1	0	0	1	0	1	1	0
Weight (<160g) x_2	1	0	1	0	0	0	1	0	0	1
Screen Size (< 5.9in) x_3	1	0	1	0	1	0	1	0	1	0
Dual sim x_4	1	1	0	0	0	1	0	1	0	0
Internal mem(>= 64gb, 4gb ram) x_5	1	1	1	1	1	1	1	1	1	0
NFC x_6	0	1	1	0	1	0	1	1	1	0
Radio x_7	1	0	0	1	1	1	0	0	0	0
Battery (>= 3500mAh) x_8	0	0	0	1	0	1	0	1	0	0
Price? (> 20k) x_9	0	1	1	0	0	0	1	1	1	0
Liked (y)	1	1	1	0	0	1	1	1	0	0
Prediction $\hat{y}$	1	1	0	1	1	1	1	0	0	0

2. Test Data

	phone11	phone12	phone13	phone14
Launch (within 6 months) x_1	1	0	0	1
Weight (<160g) x_2	0	1	1	1
Screen Size (< 5.9in) x_3	0	1	1	1
Dual sim x_4	0	1	0	0
Internal mem(>= 64gb, 4gb ram) x_5	1	0	0	0
NFC x_6	0	0	1	0
Radio x_7	1	1	1	0
Battery (>= 3500mAh) x_8	1	1	1	0
Price? (> 20k) x_9	0	0	1	0
Liked (y)	0	1	0	0
Prediction $\hat{y}$	0	1	1	0

3. Accuracy = No. of correct predictions/ Total No. of predictions (¾ = 75% in test set)

2.1.6: Geometric Basics

1. Chapter on geometry basics, a brush-up.
2. $x_2 = mx_1 + c$
3. In 2D: General form $ax_1 + bx_2 + c = 0$
   a. Consider a = 2, b = 1, c = -2
   b. The intercepts are 1 and 2
   c. Consider the point (1,2), plugging it into the equation gives us the value 2
   d. If $ax_1 + bx_2 + c > 0$ then it is above the line
   e. If $ax_1 + bx_2 + c < 0$ then it is below the line
   f. If $ax_1 + bx_2 + c = 0$ then it is on the line
4. In 3D: General form $ax_1 + bx_2 + cx_3 + d = 0$
   a. If $ax_1 + bx_2 + cx_3 + d > 0$ then it is above the line
   b. If $ax_1 + bx_2 + cx_3 + d < 0$ then it is below the line
   c. If $ax_1 + bx_2 + cx_3 + d = 0$ then it is on the line

2.1.7: MP Neuron Geometric Interpretation

1. In 2D: $ax_1 + bx_2 + d = 0$
   a. $x_2 = -(a/b)x_1 - (d/b)$
   b. $x_2 = mx_1 + c$
   c. Where m = -a/b
   d. c = -d/b
2. $\hat{y}$ = ($\Sigma^{n}_{i=1}x_i$ >= b) in 2D can be rewritten as
   a. $x_1 + x_2 - b >= 0$ (decision boundary)
   b. Positive predictions(1) yield a value >= 0 and lie above the decision boundary
   c. Negative predictions(0) yield a value < 0 and lie below the decision boundary
3. This is a very restrictive model with respect to the freedom it has due to only one parameter
4. Some downsides to this model
   a. Boolean inputs and outputs
   b. The model is linear
   c. The model has a fixed slope
   d. The model has few possible intercepts(b's)

2.1.8: Summary

1. Data: All boolean inputs ☹
2. Task: Binary classification (boolean output) ☹
3. Model: Linear decision boundary, all +ve points lie above the line ane -ve points are below (minimum flexibility) ☹
4. cost/loss: mean squared error
5. Learning: brute force approach to learn best parameter b ☹

6. Evaluation: Accuracy

2.2: Perceptron

Introduction to the Perceptron, a summary.

- Data: real inputs 😃
- Task: Classification(boolean output) ☹
- Model: Weights for every input 😃, but still linear ☹
- Cost/loss: $\Sigma_i \max(0, 1 - y_i * \hat{y}_i)$ ☹
- Learning: Our first learning algorithm 😃
- Evaluation: accuracy

2.2.1: Perceptron Data and Task

What kind of data and tasks can Perceptron process

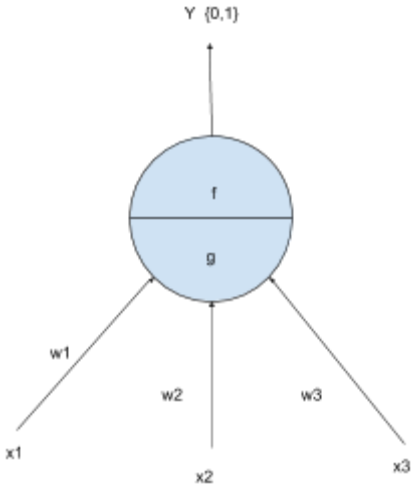
1. Perceptron can also take real inputs
2.

	phone 1	phone 2	phone 3	phone 4	phone 5	phone 6	phone 7	phone 8	phone 9
Launch (within 6 months) x_1	0	1	1	0	0	1	0	1	1
Weight (g) x_2	151	180	160	205	162	182	138	185	170
Screen Size (< 5.9in) x_3	5.8	6.18	5.84	6.2	5.9	6.26	4.7	6.41	5.5
Dual sim x_4	1	1	0	0	0	1	0	1	0
Internal mem(>= 64gb, 4gb ram) x_5	1	1	1	1	1	1	1	1	1
NFC x_6	0	1	1	0	1	0	1	1	1
Radio x_7	1	0	0	1	1	1	0	0	0
Battery (mAh) x_8	3060	3500	3060	5000	3000	4000	1960	3700	3260
Price? (k) x_9	15k	32k	25k	18k	14k	12k	35k	42k	44k
Liked (y)	1	0	1	0	1	1	0	1	0

3. Apply feature scaling to standardize real input values x' = (x-min)/(max-min)

Weight (g) x_2	0.19	0.63	0.33	1.00	0.36	0.66	0.00	0.70	0.48
Screen Size (< 5.9in) x_3	0.64	0.87	0.67	0.88	0.70	0.91	0.00	1.00	0.47
Battery (mAh) x_8	0.36	0.51	0.36	1.00	0.34	0.67	0.00	0.57	0.43
Price? (k) x_9	0.09	0.63	0.41	0.19	0.06	0.00	0.72	0.94	1.00

2.2.2: Perceptron Model



1. $\hat{y}$ = 1 if $\Sigma^{n}_{i=1} w_i x_i >= b$
2. $\hat{y}$ = 0 otherwise
3. Comparing with MP Neuron

MP Neuron	Perceptron
$\hat{y}$ = 1 if $\Sigma^{n}_{i=1} x_i >= b$ $\hat{y}$ = 0 otherwise	$\hat{y}$ = 1 if $\Sigma^{n}_{i=1} w_i x_i >= b$ $\hat{y}$ = 0 otherwise
Boolean inputs ☹	Real inputs 😃
Linear ☹	Linear ☹
Inputs are not weighted ☹	Weights for each input 😃
Adjustable threshold 😃	Adjustable threshold 😃

What do weights allow us to do?

1. Each parameter has a different effect on the output, some more, some less, some directly proportional and some inversely proportional.
2. Weights(θ/w) allow us to do this effectively.
3. x = [0, 0.19, 0.64, 1, 1, 0] features
4. w = [0.3, 0.4, -0.3, 0.1, 0.5] weights
5. x.w = $\Sigma^{n}_{i=1} w_i x_i$
6. $\hat{y}$ = 1 (if x.w >= b)
7. $\hat{y}$ = 0 otherwise

2.2.3: Perceptron Geometric Interpretation

What is the geometric interpretation of the perceptron model

1. b can now take real values, and slope can change by varying w
2. $x_1 + x_2 - b = 0$
3. $w_1x_1 + w_2x_2 - b = 0$
4. $x_2 = -(w_1/w_2)x_1 + (b/w_2)$
5. This results in more freedom than MP Neuron.
6. However, it only works with linearly separable data

2.2.4: Perceptron Loss Function

What loss function should you use for this model?

1. Consider the following training data

Weight	Screen Size	Liked(y)
0.19	0.64	1
0.63	0.81	1
0.33	0.67	0
1	0.88	0

2. Loss/cost
   a. = 0 if y = ŷ,
   b. = 1 otherwise
3. More often, it is represented using an indicator variable
   a. $L = 1_{(y \neq \hat{y})}$
   b. Or $L = 0_{(y = \hat{y})}$
4. **Q**: what is the purpose of the loss function
   a. **A**: It is to tell the model that some correction needs to be done
5. Comparing to Square Error loss function
6.

Weight	Screen Size	Liked(y)	ŷ	Perceptron loss $L = 1_{(y \neq \hat{y})}$	Sq. Error $(y - \hat{y})^2$
0.19	0.64	1	0	0	0
0.63	0.81	1	0	1	1
0.33	0.67	0	1	1	1
1	0.88	0	0	0	0

7. The Perceptron loss is almost identical to the square error loss function. For all intents and purposes in this course, it can be considered equivalent to the square error loss function.

2.2.5: Perceptron Learning - General Recipe

What is the typical recipe for learning parameters of a model

1. Consider the following data

Weight x_1	Screen Size x_2	Liked(y)
0.19	0.64	1
0.63	0.81	1
0.33	0.67	0
1	0.88	0

2. Randomly initialize parameters $w_1(\theta_1)$, $w_2(\theta_2)$ and $b(\theta_0)$
3. Iterate over data:
   a. L = compute_loss(x_i)
   b. update(w_1,w_2,b,L)
   c. Repeat till satisfied, till zero loss or some defined value ε is reached.

2.2.6: Perceptron Learning Algorithm

What does the perceptron learning algorithm look like?

1. Perceptron model: $\hat{y} = \sum^{n}_{i=1} w_i x_i >= b$
   a. Can be rewritten as $w_1x_1 + w_2x_2 - b >= 0$
   b. Let $w_0 = b$ and $x_0 = 1$
   c. Further rewritten as $w_1x_1 + w_2x_2 - w_0x_0 >= 0$
   d. $\hat{y} = \sum^{n}_{i=0} w_i x_i >= 0$
   e. Can be written as $w^Tx >= 0$
   f. Where $w^Tx = w.x$
2. Perceptron Learning Algorithm
   a. P ⇒ Inputs with label 1
   b. N ⇒ Inputs with label 0
   c. Initialize w(w_0...w_n) randomly
   d. While !convergence do:
      i. Pick random x ∈ P ∪ N
      ii. If x ∈ P and $\sum^{n}_{i=0} w_i x_i < 0$ then, w = w + x; **end**
      iii. If x ∈ N and $\sum^{n}_{i=0} w_i x_i >= 0$ then, w = w - x; **end**
   e. end
   f. The algorithm converges when all the inputs are classified correctly

2.2.7: Perceptron Learning - Why it works?

What is the intuition behind the Perceptron learning algorithm

1. $W = [w_1, w_2, \ldots w_n]$
2. $X = [x_1, x_2, \ldots x_n]$
3. $\text{Cos}\ \theta = w.x/||w||x||$, here the numerator can be replaced with $\Sigma\ w_i x_i$
   a. The denominator is always positive
   b. Therefore $\text{Cos}\ \theta \propto \Sigma\ w_i x_i$
   c. As θ ranges from 0 to 180°, cos θ ranges from 1 to -1
   d. If $\cos\theta > 0$, it is an acute angle
   e. If $\cos\theta < 0$, it is an obtuse angle
4. For $x \in P$, if $w.x < 0$, then it means that the angle(α) between this x and the current w is greater than 90°, but we want $\alpha < 90°$)
   a. What happens to the new angle α_{new} when $w_{new} = w + x$
   b. $\text{Cos}\alpha_{new} \propto w_{new}^T x$
   c. $\propto (w+x)^T x$
   d. $\propto w^T x + x^T x$(always +ve)
   e. $\propto \cos\alpha + x^T x$(some +ve value)
   f. This means that the cosine is going to increase, which leads to decrease of α
5. For $x \in N$, if $w.x > 0$, then it means that the angle(α) between this x and the current w is less than 90°, but we want $\alpha > 90°$)
   a. What happens to the new angle α_{new} when $w_{new} = w - x$
   b. $\text{Cos}\alpha_{new} \propto w_{new}^T x$
   c. $\propto (w-x)^T x$
   d. $\propto w^T x - x^T x$(always +ve)
   e. $\propto \cos\alpha - x^T x$(some +ve value)
   f. This means that the cosine is going to decrease, which leads to increase of α

2.2.8: Perceptron Learning - Will it always work

Will this algorithm always work?

1. It will only work if the data is linearly separable
2. If it is not linearly separable, the algorithm will never converge (ie, predict all training examples correctly)
3. Linearly Separable: Two sets P and N of points in an n-dimensional space are called absolutely linearly separable if
   a. n+1 real numbers $w_o, w_1, \ldots w_n$ exist such that
   b. Every point$(x_o, x_1, \ldots x_n) \in P$ satisfies $\Sigma^n_{i=1} w_i x_i >= w_0$
   c. Every point$(x_o, x_1, \ldots x_n) \in N$ satisfies $\Sigma^n_{i=1} w_i x_i < w_0$

4. If the sets P and N are finite and linearly separable, the Perceptron learning algorithm will converge in a finite number of steps

2.2.9: Perceptron Evaluation

How do you check the performance of the perceptron model

1. Training data

	phone 1	phone 2	phone 3	phone 4	phone 5	phone 6	phone 7	phone 8	phone 9
Launch (within 6 months) x_1	0	1	1	0	0	1	0	1	1
Weight (g) x_2	0.19	0.63	0.33	1.00	0.36	0.66	0.00	0.70	0.48
Screen Size (< 5.9in) x_3	0.64	0.87	0.67	0.88	0.70	0.91	0.00	1.00	0.47
Dual sim x_4	1	1	0	0	0	1	0	1	0
Internal mem(>= 64gb, 4gb ram) x_5	1	1	1	1	1	1	1	1	1
NFC x_6	0	1	1	0	1	0	1	1	1
Radio x_7	1	0	0	1	1	1	0	0	0
Battery (mAh) x_8	0.36	0.51	0.36	1.00	0.34	0.67	0.00	0.57	0.43
Price? (k) x_9	0.09	0.63	0.41	0.19	0.06	0.00	0.72	0.94	1.00
Liked (y)	1	0	1	0	1	1	0	1	0

2. Test data

	phone 10	phone 11	phone 12	phone 13
Launch (within 6 months) x_1	1	0	0	1
Weight (g) x_2	0.23	0.34	0.44	0.54
Screen Size (< 5.9in) x_3	0.74	0.93	0.34	0.42
Dual sim x_4	0	1	0	0
Internal mem(>= 64gb, 4gb ram) x_5	1	0	0	0
NFC x_6	0	0	1	0
Radio x_7	1	1	1	0
Battery (mAh) x_8	1	1	1	0
Price? (k) x_9	0	0	1	0
Liked (y)	0	1	0	0

Prediction (ŷ)	0	1	1	0

3. Accuracy = ¾ = 75%

2.2.10: Perceptron Summary

When will you use perceptron

1. Data: real inputs
2. Task: Classification (Boolean output)
3. Model: $\Sigma^{n}_{i=0} w_i x_i >= 0$
4. Cost/loss: $\Sigma_i 1_{(yi \neq \hat{y}i)}$
5. Learning Algorithm: Randomly assign and adjust w and b iteratively till convergence
6. Evaluation: Accuracy
7. How does this tie into final project
8. Perceptron can be used for image detection, to detect if

2.1.11: Perceptron: Toy Example

Perceptron Learning Algorithm in action

1. Dataset

x_1	x_2	y
-1	-1	0
-5	-2.5	0
-7.5	7.5	0
10	7.5	1
-2.5	12.5	0
5	10	1
5	5	1

2. P ⇒ Green points, N ⇒ Red points
3. Decision boundary line is given by $w_1x_1 + w_2x_2 - b >= 0$
   a. Or $x_2 = -(w_1/w_2)x_1 + (b/w_2)$
   b. Can be rewritten as $x_2 = mx + c$
   c. Where $m = -(w_1/w_2)$ and $c = (b/w_2)$
4. Initialize w randomly
   a. $w_1 = 1.00$, $w_2 = 1.00$, $b = 5.00 \Rightarrow m = -1.00$, $c = 5.00$
5. The line is $x_2 = -x_1 + 5$
6. While !convergence do:

a. Pick random $x \in P \cup N$
b. If $x \in P$ and $\sum_{i=0}^{n} w_i x_i < 0$ then, $w = w + x$; **end**
c. If $x \in N$ and $\sum_{i=0}^{n} w_i x_i >= 0$ then, $w = w - x$; **end**
d. Consider $x = [x_0;x_1;x_2]$ and $w = [w_0;w_1;w_2]$, where x_0 bias term is always 1
e. On the 5th training example, condition c isn't satisfied, so we recalculate w = w - x
f. $w_1 = 3.5$, $w_2 = -11.5$, $b = 4.00 \Rightarrow m = 0.3$, $c = -0.35$
g. The line changes, causing a new error on the 6th training example, so we calculate w = w+x
h. $w_1 = 8.5$, $w_2 = -1.5$, $b = 5.00 \Rightarrow m = 5.67$, $c = -3.33$
i. The resulting line predicts all examples perfectly, thus convergence is reached