DATA SCIENCE

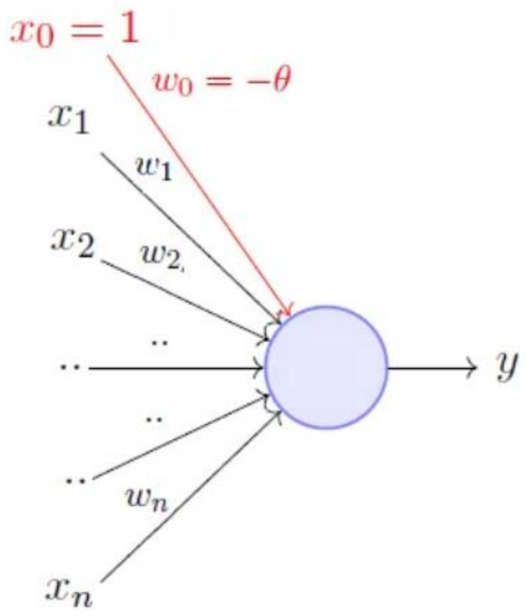
By: Michel Samir Zaki

Perceptron Learning Algorithm (PLA)

- A single perceptron can only be used to implement **linearly separable** functions
- It takes both real **inputs** and associates a set of **weights** to them, along with a **bias**
- Multiply all input values with corresponding weight values and then add them to determine the **weighted sum**. $\sum w_i * x_i + b$

$\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + \ldots + w_n * x_n$. Add another essential term called bias 'b' to the weighted sum to improve the model performance. $\sum w_i * x_i + b$.

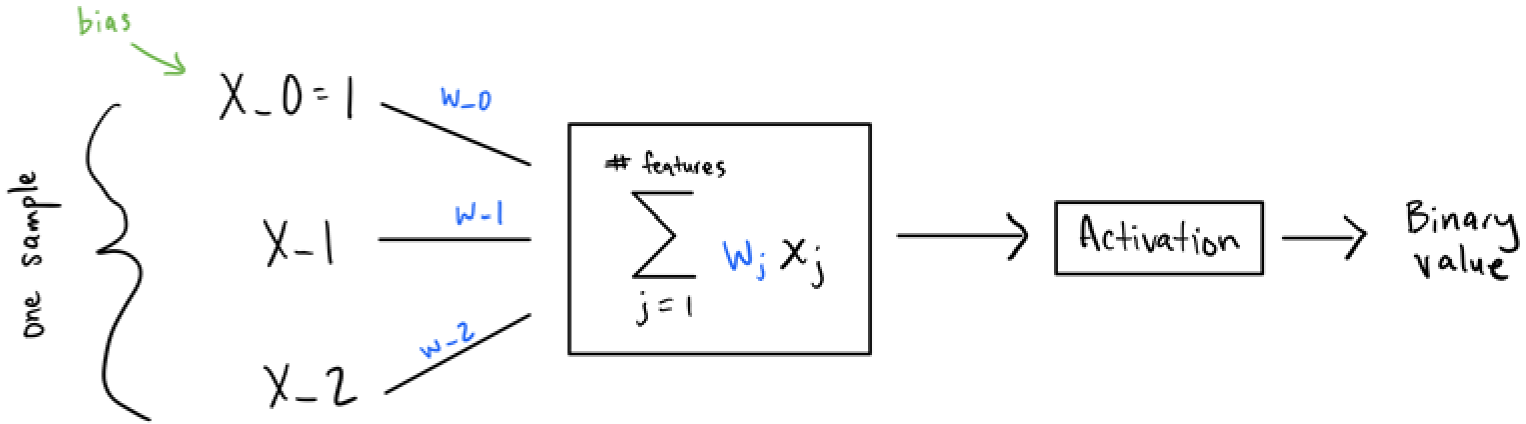
Perceptron Learning Algorithm (PLA)



A more accepted convention,

$$
\begin{aligned}
y &= 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0 \\
&= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0
\end{aligned}
$$

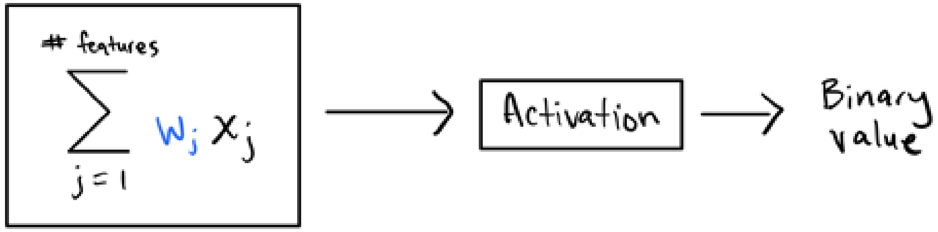
$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$



Note: it is x_1 not x-1 and so on

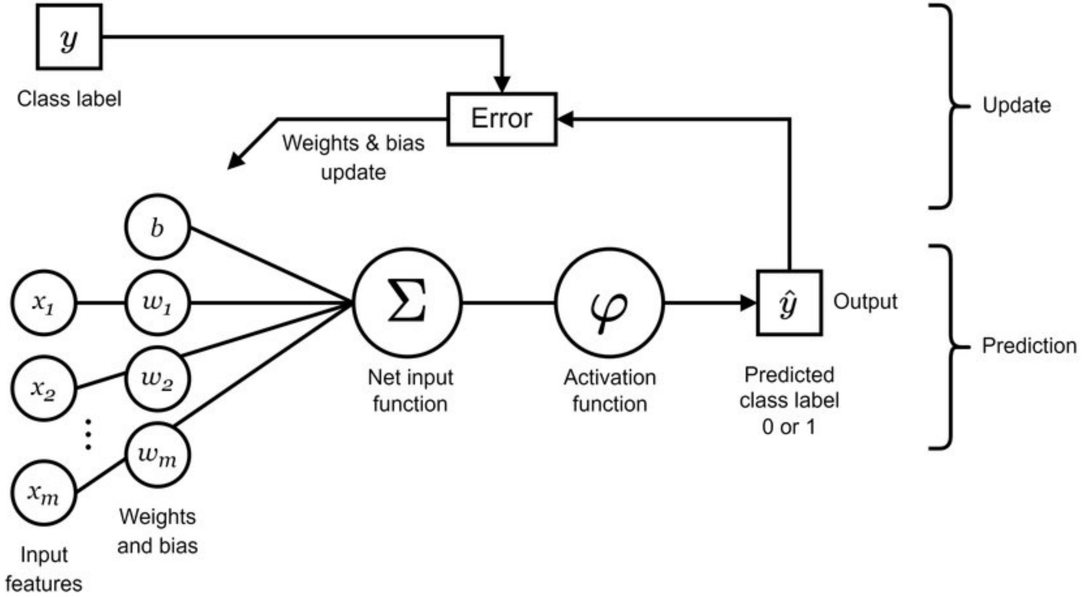
Perceptron Learning Algorithm (PLA)

- Apply an **activation function** (typically a step or sign function) to the weighted sum to determine the output of the perceptron



- Compare the **predicted output** with the desired output for the input example.

- Adjust the weights and biases based on the error between the **predicted output and the desired output.**



Perceptron Learning Algorithm (PLA)

- Binary classification labels generally appear as $-1,1$ or $0,1$.
- The activation function used in the perceptron model will depend on which set of binary labels you choose.

- If you choose (0, 1), you will need to use the **Heaviside step function** as your activation function.
- Otherwise, you will use the **sign function**. (-1, 1)

Perceptron Learning Algorithm (PLA)

To get a prediction from the perceptron model, you need to implement $\text{step}\left(\sum_{j=1}^{n} w_j x_j\right)$. Recall that the vectorized equivalent of $\text{step}\left(\sum_{j=1}^{n} w_j x_j\right)$ is just $\text{step}(w \cdot x)$, the dot product of the weights vector w and the features vector x.

- **Dot Product of Two Vectors**

$$\mathbf{w} = [w_0, w_1, w_2, ..., w_n]$$
$$\mathbf{x} = [1, x_1, x_2, ..., x_n]$$

$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{i=0}^{n} w_i * x_i$$

Pocket Algorithm

Perceptron learning algorithm function $f(x)$ is represented as the product of the input vector (x) and the learned weight vector (w). In mathematical notion, it can be described as:

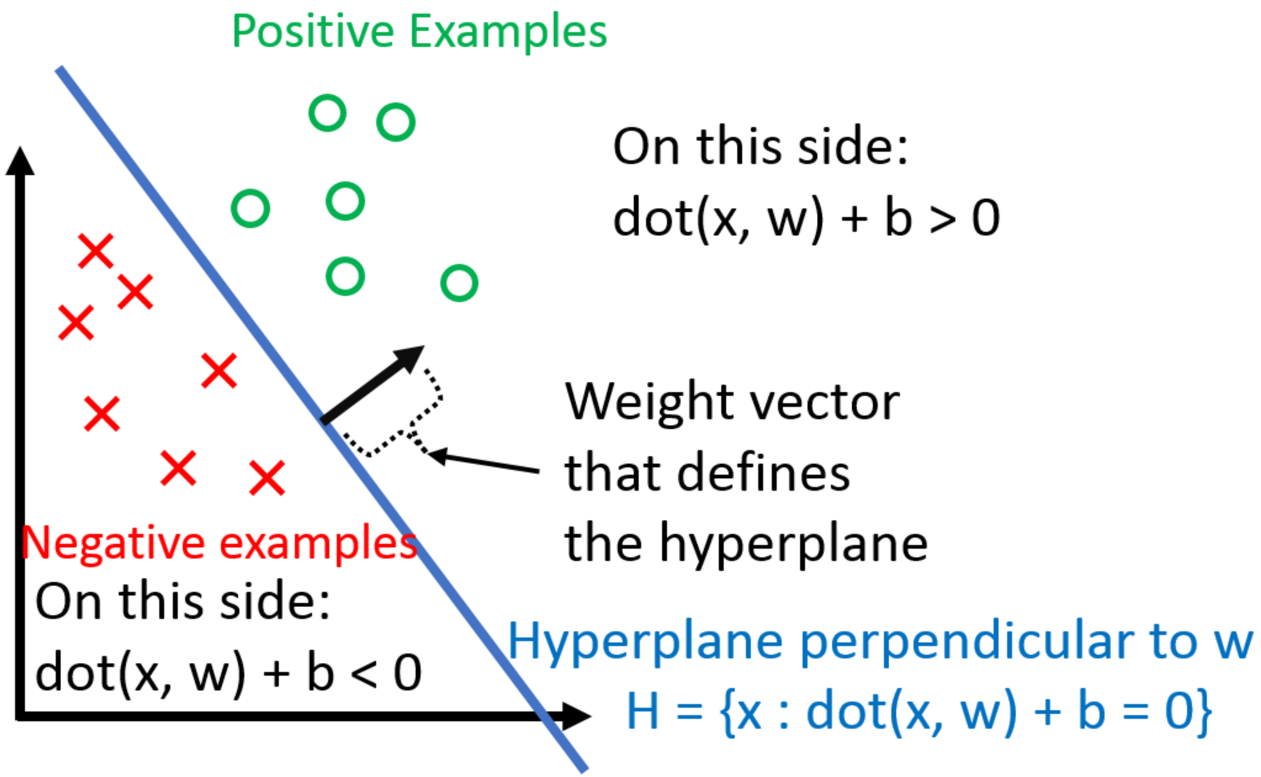
$$f(x) = 1, if w.x + b > 0 \, f(x) = 0, otherwise$$

Where–

- w represents the weight vector which consists of a set of real-valued weights.
- b represents the bias vector.
- x represents the input vector which consists of the input feature values.

Perceptron Learning Algorithm (PLA)

- We aim to find the w vector that can perfectly classify positive and negative inputs in a dataset.
- w is initialized with a random vector.
- We are then iterative overall positive and negative samples (P U N). (P union N)
- Now, if an input x belongs to P, (w.x) should be greater than or equal to 0.
- And if x belongs to N, (w.x) should be lesser than or equal to 0.
- **Only when these conditions are not met, we update the weights**



Perceptron Learning Algorithm (PLA)

- **Update weights**

Our goal is to find the **w** vector that can perfectly classify positive inputs and negative inputs in our data.

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

Perceptron Learning Algorithm (PLA)

Update weights form Geometry perspective:

Angle Between Two Vectors

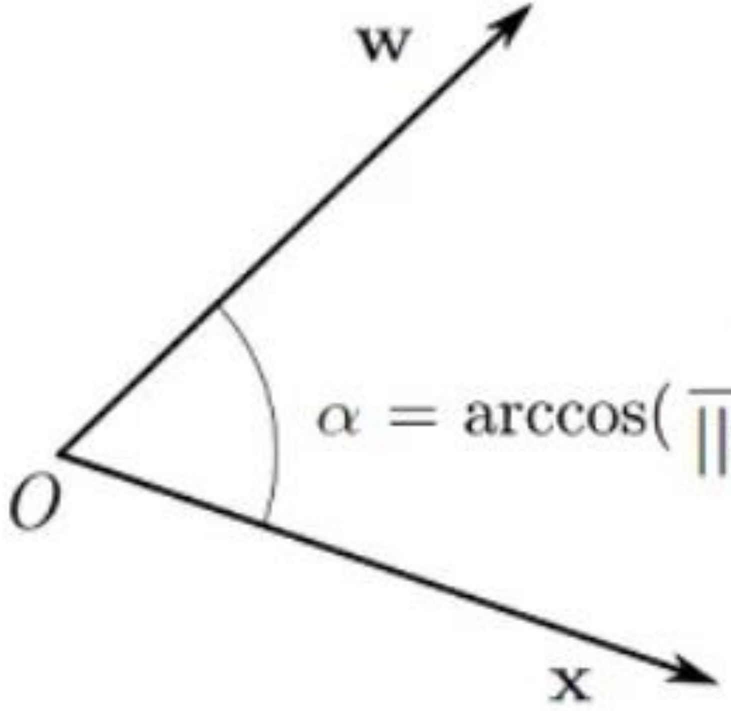
Dot product can be computed differently if only you knew the angle between the vectors and their individual magnitudes:

$$\mathbf{w}^T\mathbf{x} = ||\mathbf{w}||\,||\mathbf{x}||\, cos\alpha$$

Update weights form Geometry perspective:

Angle Between Two Vectors

$$\cos\alpha = \frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||||\mathbf{x}||}$$



$$\alpha = \arccos\left(\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||||\mathbf{x}||}\right)$$

Perceptron Learning Algorithm (PLA)

Update weights form Geometry perspective:

Angle Between Two Vectors

when x belongs to P, we want w.x > 0. That means that the angle between w and x should be less than 90 because the cosine of the slope is proportional to the dot product.

$$\cos\alpha = \frac{\mathbf{w}^T\mathbf{x}}{\|\mathbf{w}\|\|\mathbf{x}\|} \mid \cos\alpha \propto \mathbf{w}^T\mathbf{x}$$

$$if\mathbf{w}^T\mathbf{x} > 0 \Rightarrow \cos\alpha > 0 \Rightarrow \alpha < 90$$

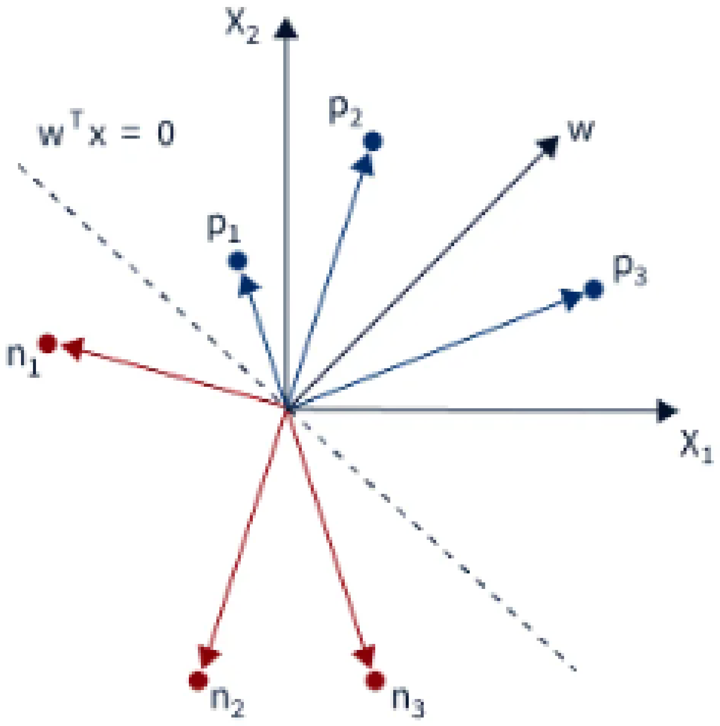
Similarly,

$$if\mathbf{w}^T\mathbf{x} < 0 \Rightarrow \cos\alpha < 0 \Rightarrow \alpha > 90$$

Perceptron Learning Algorithm (PLA)

Update weights form Geometry perspective:

So whatever the w vector may be, as long as it makes an angle less than 90 degrees with the positive example data vectors (x ∈ P) and an angle more than 90 degrees with the negative example data vectors (x ∈ N), we are good



wTx=0: is the clasifier line
w: is the vector w (perpendicular to line)
x1 and x2 is the 2D Plane
Angle < 90 between w and any P point
Angle > 90 between w and any N point

Pocket Algorithm

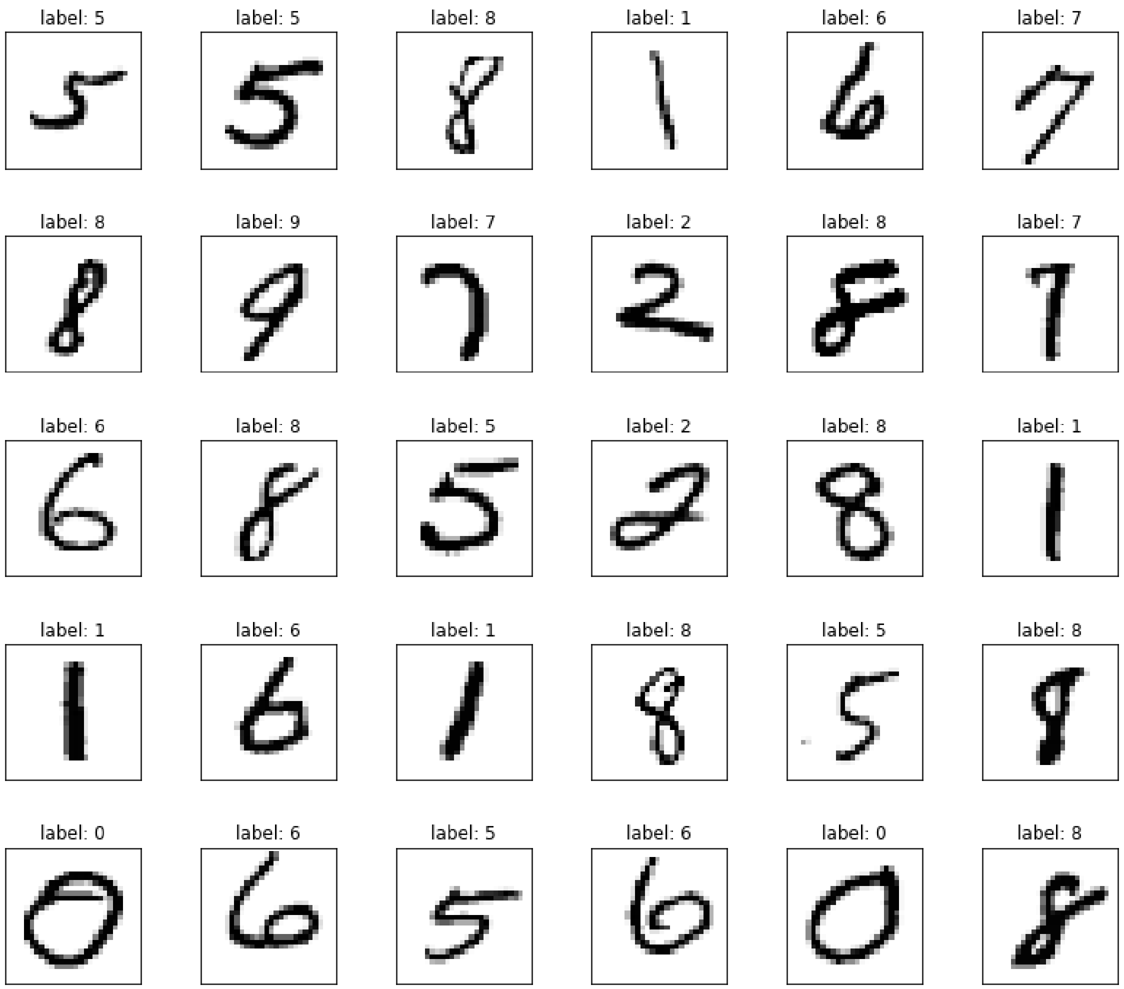
Pocket Algorithm is similar to the perceptron algorithm its just the advanced version of the perceptron. Perceptron is just for the linearly classifying data, where Pocket algorithm is used when a data is not linearly sepearable and you want to sepreate data with the minimal error. -you can implement the Pocket algorithm using your own generated data. Your data doesn't need to be linearly separable

Pocket Algorithm

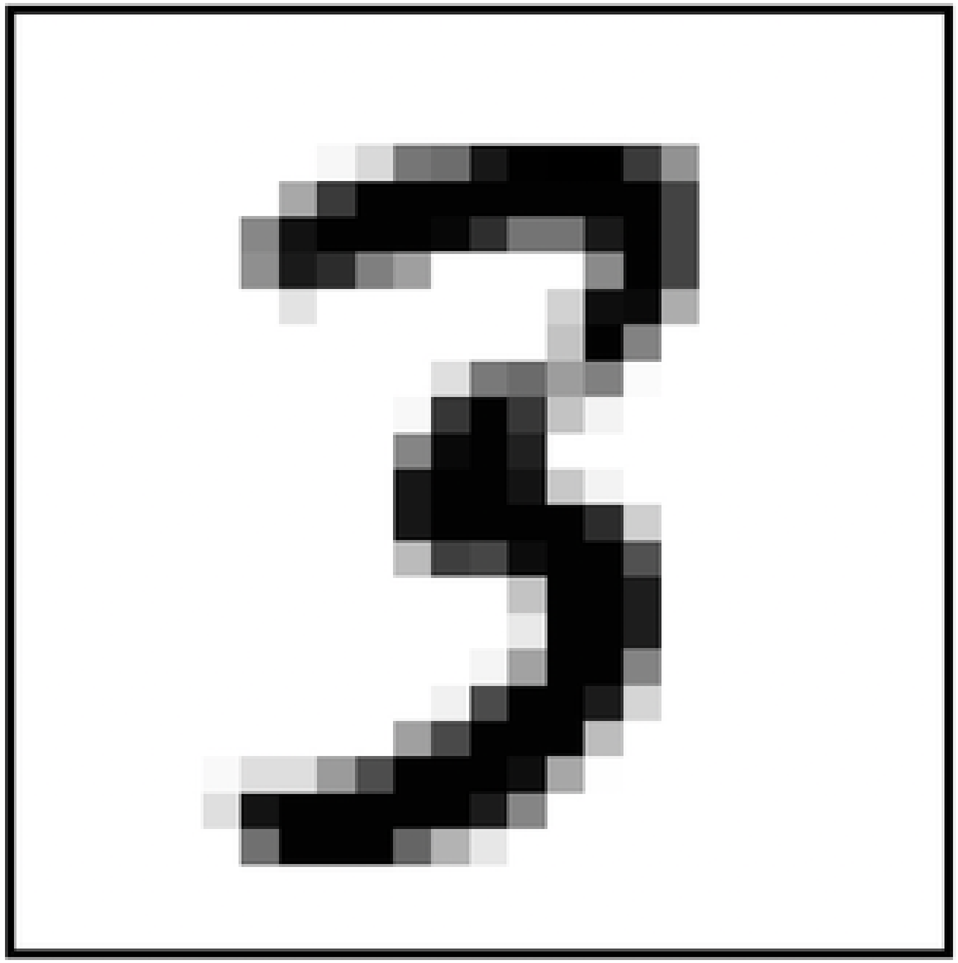
1. Initialize the pocket weight vector, W_{pocket}, to 0 or small random numbers and use this weight vector as the initialized weight vector, W_0 of Perceptron Learning Algorithm.
2. For each training iteration, perform the following sub-steps:
    1. Run the training step of Perceptron Learning Algorithm to obtain the updated weight vector, W_t, where t indicates the current iteration.
    2. Evaluate W_t by comparing the number of misclassification on the entire sample set with the number of misclassification performed by W_{pocket}.
    3. If W_t is better than W_{pocket}, replace W_{pocket} to W_t.
3. Return W_{pocket} when the training iteration terminates.

Mnist Dataset

- The dataset contains 70,000 grayscale images of handwritten digits from 0 to 9
- Each image is a 28x28 pixel square
- All images are labelled with the respective digit they represent, indicating which digit (0–9) is written in each image.
- There are 70,000 images, and each images has 784 (28*28) features.
- Each image is 28*28 pixels , and each feature simply represents one pixel’ s intensity from 0 (white) to 255 (Black)



Mnist Dataset



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	8	38	137	146	232	254	255	255	197	109	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	87	197	253	253	253	253	253	253	253	253	188	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	120	237	253	253	253	248	209	139	139	230	253	188	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	112	229	210	128	96	0	0	0	0	117	253	188	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	45	241	245	82	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	253	125	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	32	134	148	98	127	4	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	7	201	253	200	59	12	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	122	246	253	223	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	233	253	253	236	55	11	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	233	253	253	253	253	210	48	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	68	193	185	243	253	253	173	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	59	253	253	226	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	253	253	226	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	93	253	253	123	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	14	180	253	253	228	42	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	95	181	253	253	253	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	6	33	33	100	178	253	253	253	241	88	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	32	236	253	253	253	253	253	228	122	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	143	253	253	253	154	76	24	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


Thank You