Perceptron
A linear Classifier

Dr. Rizwan Ahmed Khan

Outline

Reference Books

Reference books for this lecture:

- Chapter 4: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.

Reference Books

Reference books for this lecture:

- Chapter 4: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 5: Pattern Classification, R. DUDA et al., Wiley Interscience, latest edition.

Reference Books

Reference books for this lecture:

- Chapter 4: Machine Learning, Tom MITCHELL, McGraw Hill, latest edition.
- Chapter 5: Pattern Classification, R. DUDA et al., Wiley Interscience, latest edition.
- Chapter 3: Pattern Recognition, S. Theodoridis et al.,Academic Press, 4^{th} or latest edition.

Section Contents

Issues with K-Nearest Neighbors

- Although k-nearest neighbor is a strong classifier and can achieve good results if the number of training samples (n) are very large, but one issue that restricts to use it (for practical reason) is:

Issues with K-Nearest Neighbors

- Although k-nearest neighbor is a strong classifier and can achieve good results if the number of training samples (n) are very large, but one issue that restricts to use it (for practical reason) is:

What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$

Issues with K-Nearest Neighbors

- Although k-nearest neighbor is a strong classifier and can achieve good results if the number of training samples (n) are very large, but one issue that restricts to use it (for practical reason) is:

What is computational complexity of K-Nearest Neighbors

1. Compare query data / test data to all training examples.
2. Training Complexity : $\mathcal{O}(1)$
3. Test Complexity : $\mathcal{O}(nd)$, where n = number of training instances and d = dimensions of training data. It's linear time algorithm and that is not good!
4. Result: K-Nearest Neighbors is **slow**.

- For practical application, test time is more important that train time.

Section Contents

Historical Context

- The first artificial neural network (ANN) was invented in 1958 by psychologist Frank Rosenblatt, called Perceptron.
- It was intended to model how the human brain processed visual data and learned to recognize objects.
- Press Conference in 1958: "the embryo of an electronic computer that [the US Navy (funding agency)] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence".

Historical Context

- The first artificial neural network (ANN) was invented in 1958 by psychologist Frank Rosenblatt, called Perceptron.
- It was intended to model how the human brain processed visual data and learned to recognize objects.
- Press Conference in 1958: "the embryo of an electronic computer that [the US Navy (funding agency)] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence".
- In 1969 it was proved that Perceptron could not be trained for non-linearly separable data (i.e. XOR problem). This lead to field of neural network research to stagnate for many years (almost quarter of a century – A.I winter).

Historical Context

1958 New York Times...

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

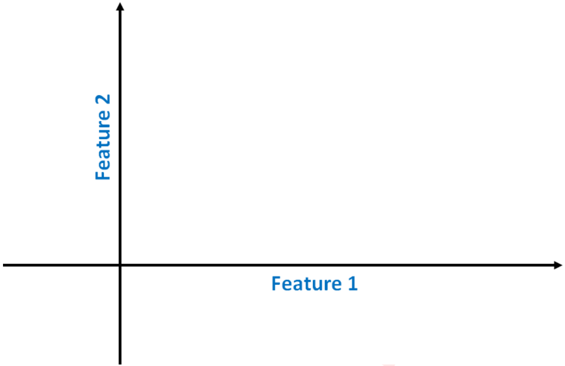
Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

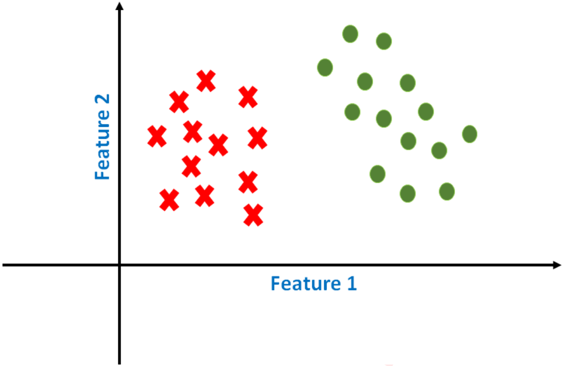
Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

Assumption



Assumption

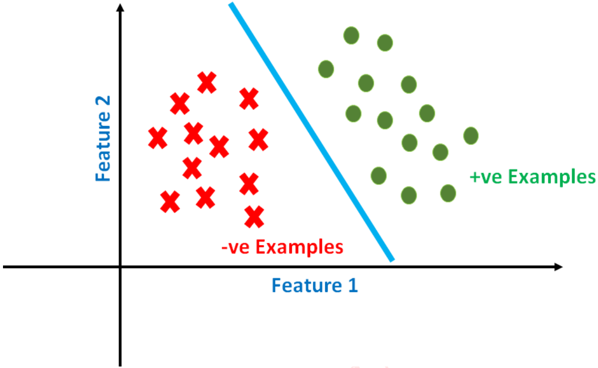


Assumptions or Bias:

- Binary classification

$$y_i \in \{-1, +1\}$$

Assumption



Assumptions or Bias:

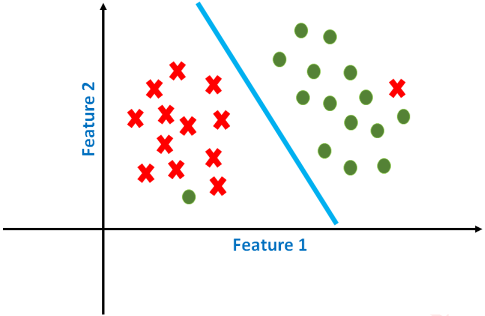
- Binary classification

$$y_i \in \{-1, +1\}$$

- There must be a hyperplane that linearly separates the data (one class from the other).

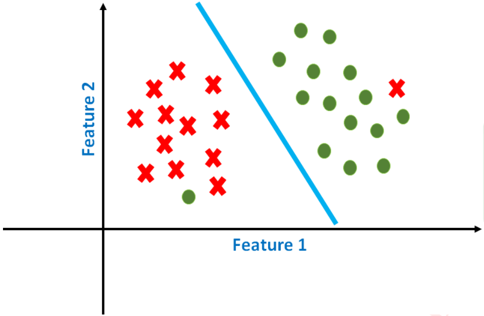
- All data points from one class lie on one side of hyperplane.

Assumption



- What will happen in this case? Now data is not linearly separable.

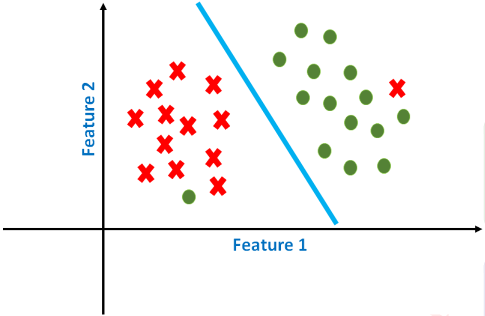
Assumption



- What will happen in this case? Now data is not linearly separable.
- In high dimensional space data points tend to be far away from each other (difficult to visualize).

In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Assumption



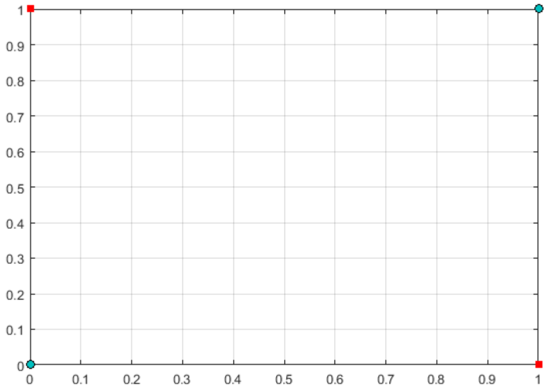
- What will happen in this case? Now data is not linearly separable.
- In high dimensional space data points tend to be far away from each other (difficult to visualize).

In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

In essence Perceptron is opposite of k-NN as k-NN works better in low dimensional spaces (rem: curse of dimensionality) while Perceptron assumption holds in high dimensional spaces.

Assumption : Data in higher dimensional space

XOR Problem



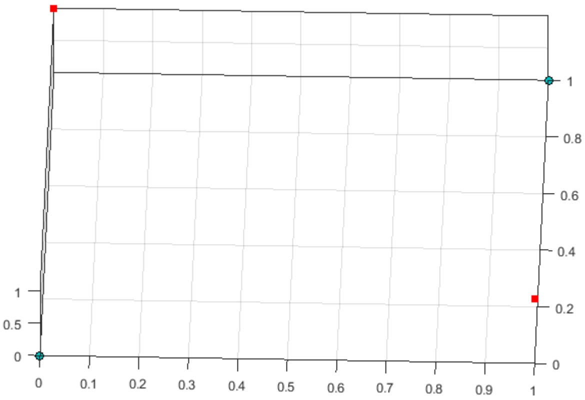
Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Assumption : Data in higher dimensional space

XOR Problem



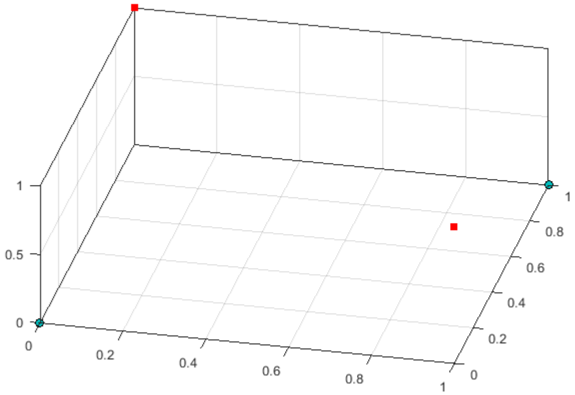
Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Assumption : Data in higher dimensional space

XOR Problem



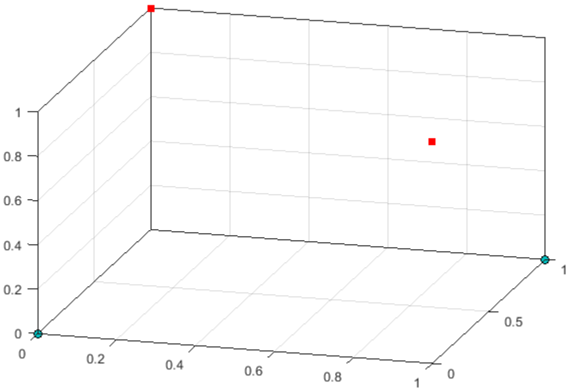
XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Assumption : Data in higher dimensional space

XOR Problem



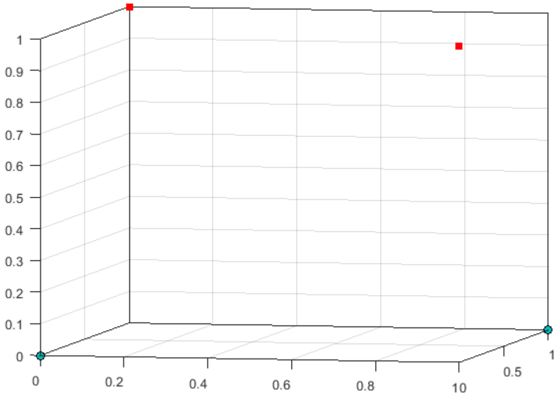
Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Assumption : Data in higher dimensional space

XOR Problem

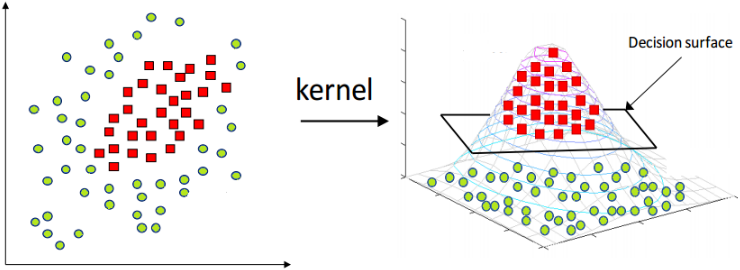


XOR in 3D: `https://www.youtube.com/watch?v=5KIYu3zKvqo`

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR in 2D is not linearly separable but in 3D it is.
- In low dimensional spaces linear separability doesn't hold for long but in high dimensional space it almost holds i.e. (kernel trick).

Mapping data in higher dimensional space: Kernel function



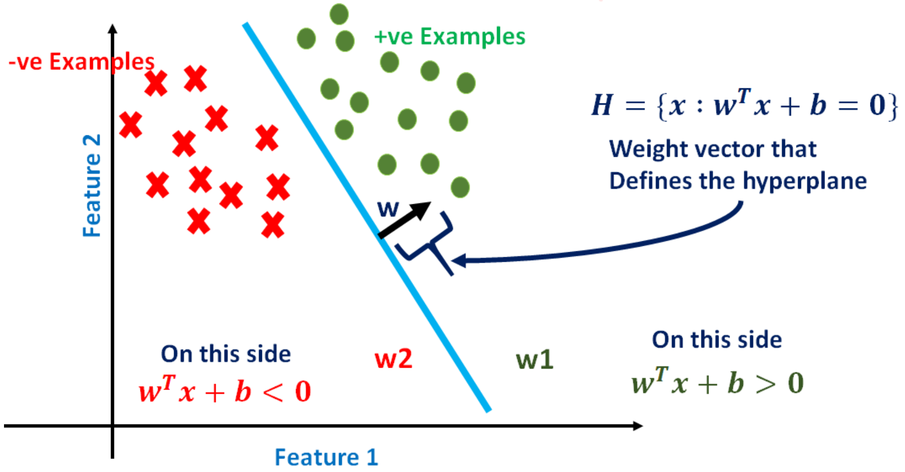
Video showing XOR data from 2D to 3D. [1]

Kernel Trick [2]

- There is a little trick that can be done to transform data (change the data point without changing the data point) in a such away that it become linearly separable.
- Define function Φ that will take data point and change its dimension.

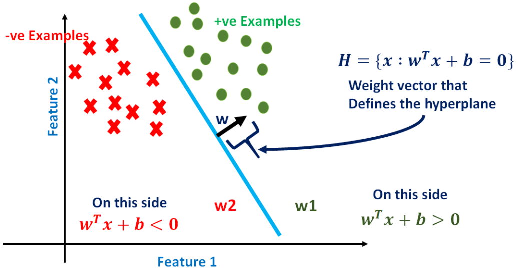
[1] https://youtu.be/5KIYu3zKvqo
[2] Later in the course during lecture on SVM

Classifier Visualization : Defining hyperplane



- In case of difficulty in understanding equation of a hyperplane, refer Section 7.

Classifier Calculus



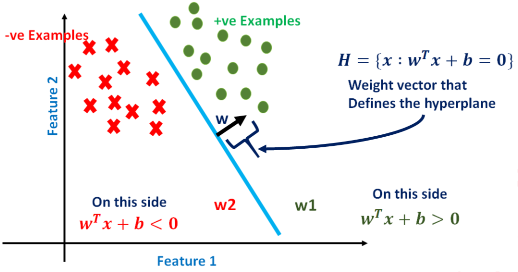
- Assuming that hyperplane exists that linearly separates data according to labels, Perceptron algorithm tries to find it.
- Mathematically hyperplane can be given by:

$$\mathcal{H} = \{x : (\bar{\mathbf{w}}^\top \bar{\mathbf{x}} + b) = 0\}$$

where: b is the bias term (without the bias term, the hyperplane that $\mathbf{w}$ defines would always have to go through the origin).

- Learning a perceptron involves choosing values for weights $\mathbf{w}$.

Classifier Calculus



- What to do at test time? (unknown sample $\mathbf{x}_i$)

$$h(x_i) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$

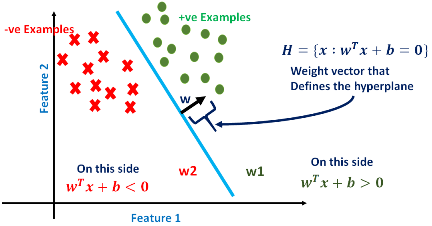
OR

$$\mathbf{w}^\top \mathbf{x} + b > 0 \quad \forall \text{ x in class1, +ve Examples}$$

$$\mathbf{w}^\top \mathbf{x} + b < 0 \quad \forall \text{ x in class2, -ve Examples}$$

- This means test time speed is constant. It's very fast.

Classifier Calculus



- Dealing with b separately is difficult (difficult for mathematical proofs and for programming), thus this term can be merged with weight vector w. Under this convention:

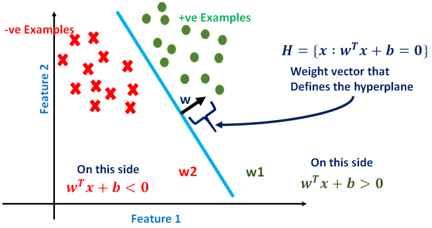
$$\mathbf{x}_i \quad \text{becomes} \quad \begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$$

$$\mathbf{w} \quad \text{becomes} \quad \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

- We can verify:

$$\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}^\top = \mathbf{w}^\top \mathbf{x}_i + b$$

Classifier Calculus



$\mathbf{x}_i$ becomes $\begin{bmatrix} \mathbf{x}_i \\ 1 \end{bmatrix}$

$\mathbf{w}$ becomes $\begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

Now we can say:

$$\mathcal{H} = \{x : (\bar{\mathbf{w}}^\top \mathbf{x}) = 0\}$$

Rem: We absorbed b with w, in essence b is offset and w is orientation of hyperplane.

Section Contents

Perceptron Learning Algorithm

Algorithm 1 Perceptron Learning Algorithm

Result: Learned Hyperplane / Decision Boundary

initialization $\vec{w} = 0$

while *TRUE* **do**

  missClassification $= 0$

  for $(x_i, y_i) \in D$ **do**

    if $y_i(\vec{w}^\top \vec{x_i}) \leq 0$ **then**

      $\vec{w} \leftarrow \vec{w} + y\vec{x}$

      $missClassification \leftarrow missClassification + 1$

    end

  end

  if *missClassification = 0* **then**

    break

  end

end

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \tag{1}$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \tag{2}$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \tag{3}$$

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \tag{1}$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
  And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \tag{2}$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \tag{3}$$

By combining Equations 2 and 3, we can write:

$$y_i(\vec{w}^\top \vec{x_i}) \geq 0 \tag{4}$$

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \tag{1}$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
  And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \tag{2}$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \tag{3}$$

By combining Equations 2 and 3, we can write:

$$y_i(\vec{w}^\top \vec{x_i}) \geq 0 \tag{4}$$

Proof:

1. $y_i(\vec{w}^\top \vec{x_i}) \geq 0$, $y_i = +1$ for +ve samples
   $+1(\vec{w}^\top \vec{x_i}) \geq 0 \implies (\vec{w}^\top \vec{x_i}) \geq 0$
   same as Equation 2

Perceptron Learning Algorithm

- In algorithm, what this statement specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0 \qquad (1)$$

- Remember: We are dealing binary classification $y_i \in \{-1, +1\}$
And

$$\vec{w}^\top \mathbf{x} > 0 \quad \forall \text{ +ve Examples} \qquad (2)$$

$$\vec{w}^\top \mathbf{x} < 0 \quad \forall \text{ -ve Examples} \qquad (3)$$

By combining Equations 2 and 3, we can write:

$$y_i(\vec{w}^\top \vec{x_i}) \geq 0 \qquad (4)$$

Proof:

1. $y_i(\vec{w}^\top \vec{x_i}) \geq 0$, y_i = +1 for +ve samples
$+1(\vec{w}^\top \vec{x_i}) \geq 0 \implies (\vec{w}^\top \vec{x_i}) \geq 0$
same as Equation 2
2. $y_i(\vec{w}^\top \vec{x_i}) \geq 0$, y_i = -1 for -ve samples
$-1(\vec{w}^\top \vec{x_i}) \geq 0 \implies (\vec{w}^\top \vec{x_i}) \leq 0$
same as Equation 3

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

This shows a misclassification!

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

  This shows a misclassification!

- $$\vec{w} \leftarrow \vec{w} + y\vec{x}$$

  This is weight update rule.

Perceptron Learning Algorithm

- Again, in perceptron learning algorithm, what this statement (refer Equation 1) specifies?

$$y_i(\vec{w}^\top \vec{x_i}) \leq 0$$

  This shows a misclassification!

- $$\vec{w} \leftarrow \vec{w} + y\vec{x}$$

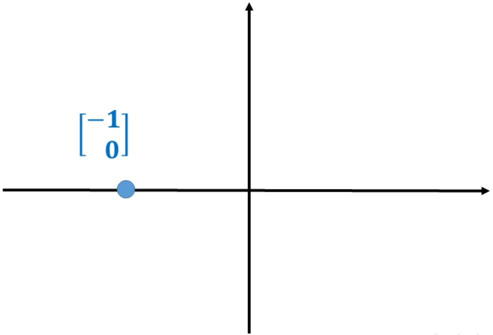
  This is weight update rule.

  1. if misclassified sample is from $+1$ class then add in $\vec{w}$ amount proportional to $\vec{x}$
  2. if misclassified sample is from -1 class then subtract in $\vec{w}$ amount proportional to $\vec{x}$

- The algorithm belongs to a more general algorithmic family known as reward and punishment schemes.

Example : Perceptron Learning Algorithm

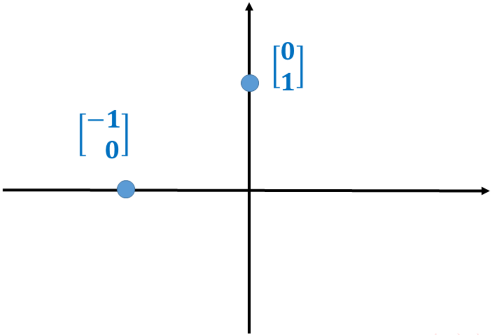
- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$

Example : Perceptron Learning Algorithm

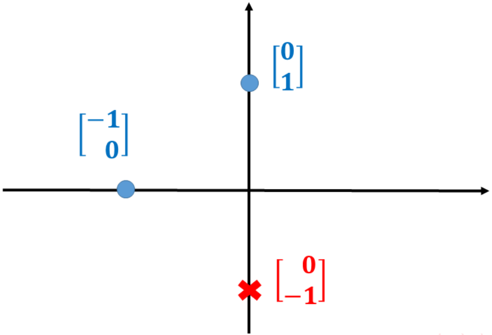
- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix}-1\\0\end{bmatrix}$
  2. $\begin{bmatrix}0\\1\end{bmatrix}$

Example : Perceptron Learning Algorithm

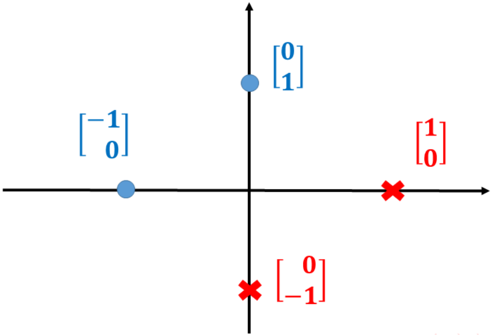
- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$
  2. $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$
  3. $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$

Example : Perceptron Learning Algorithm

- Design a linear classifier using the perceptron algorithm



- Consider four data points (first two points belong to class $w1$, while other two belongs to class $w2$):
  1. $\begin{bmatrix}-1\\0\end{bmatrix}$
  2. $\begin{bmatrix}0\\1\end{bmatrix}$
  3. $\begin{bmatrix}0\\-1\end{bmatrix}$
  4. $\begin{bmatrix}1\\0\end{bmatrix}$
- Consider initial weight vector is chosen as $w(0) = \begin{bmatrix}0\\0\\0\end{bmatrix}$ in extended 3D space i.e. merged w and b.

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T \vec{x}$

$$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 0 \text{ (Miss-classification, result should be } > 0 \text{ for } w1 \text{ samples)}$$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix}-1\\0\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix}0 & 0 & 0\end{bmatrix}\begin{bmatrix}-1\\0\\1\end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

- update rule, $w : \vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{1}) \leftarrow \begin{bmatrix}0\\0\\0\end{bmatrix} + (1)\begin{bmatrix}-1\\0\\1\end{bmatrix} = \begin{bmatrix}-1\\0\\1\end{bmatrix}$$ (This is updated w)

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix}-1\\0\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix}0 & 0 & 0\end{bmatrix} \begin{bmatrix}-1\\0\\1\end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

- update rule, $w: \vec{w} \leftarrow \vec{w} + y\vec{x}$

$\vec{w(1)} \leftarrow \begin{bmatrix}0\\0\\0\end{bmatrix} + (1)\begin{bmatrix}-1\\0\\1\end{bmatrix} = \begin{bmatrix}-1\\0\\1\end{bmatrix}$ (This is updated w)

2. Consider second data point $\begin{bmatrix}0\\1\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

1. Consider first data point $\begin{bmatrix}-1\\0\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix}0 & 0 & 0\end{bmatrix}\begin{bmatrix}-1\\0\\1\end{bmatrix} = 0$ (Miss-classification, result should be > 0 for $w1$ samples)

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

$\vec{w(1)} \leftarrow \begin{bmatrix}0\\0\\0\end{bmatrix} + (1)\begin{bmatrix}-1\\0\\1\end{bmatrix} = \begin{bmatrix}-1\\0\\1\end{bmatrix}$ (This is updated w)

2. Consider second data point $\begin{bmatrix}0\\1\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$= \begin{bmatrix}-1 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix} = 1 > 0$ (Correct as $\vec{w}^T\vec{x} > 0$ for $w1$ samples, no update in $\vec{w(1)}$ required, $\vec{w(2)} = \vec{w(1)}$)

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$[-1\ 0\ 1] \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$ (Miss-classification, result should be < 0 for $w2$ samples)

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T \vec{x}$

$[-1\ 0\ 1] \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$ (Miss-classification, result should be < 0 for $w2$ samples)

- update rule, $w : \vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{3}) \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$ (This is updated $w(\vec{3})$)

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$$ (Miss-classification, result should be < 0 for $w2$ samples)

- update rule, $w: \vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{3}) \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$ (This is updated $w(\vec{3})$)

4. Consider fourth data point $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0$$ (Miss-classification, result should be < 0 for $w2$ samples)

- update rule, $w : \vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w(\vec{3}) \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$ (This is updated $w(\vec{3})$)

4. Consider fourth data point $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \text{-1} < 0$$

Example : Perceptron Learning Algorithm

3. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$[-1\ 0\ 1]\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = 1 > 0 \text{ (Miss-classification, result should be} < 0 \text{ for } w2 \text{ samples)}$$

- update rule, $w : \vec{w} \leftarrow \vec{w} + y\vec{x}$

$$w\vec{(3)} \leftarrow \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \text{ (This is updated } w\vec{(3)}\text{)}$$

4. Consider fourth data point $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$[-1\ 1\ 0]\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \text{-1} < 0$$

(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, no update in $w\vec{(3)}$ required, $w\vec{(4)} = w\vec{(3)}$)

Example : Perceptron Learning Algorithm

5. One loop on dataset is completed in which misclassification were encountered, now again go through dataset (loop will only stop if there is no misclassification). Consider $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

5. One loop on dataset is completed in which misclassification were encountered, now again go through dataset (loop will only stop if there is no misclassification). Consider $\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 1 > 0 \quad (\text{Correct as } \vec{w}^T\vec{x} > 0 \text{ for } w1 \text{ samples}, w(\vec{5}) = w(\vec{4}))$$

6. Consider second data point $\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

5. One loop on dataset is completed in which misclassification were encountered, now again go through dataset (loop will only stop if there is no misclassification). Consider $\begin{bmatrix}-1\\0\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix}-1 & 1 & 0\end{bmatrix}\begin{bmatrix}-1\\0\\1\end{bmatrix} = 1 > 0 \ \ (\text{Correct as } \vec{w}^T\vec{x} > 0 \text{ for } w1 \text{ samples}, \ w(\vec{5}) = w(\vec{4}))$$

6. Consider second data point $\begin{bmatrix}0\\1\\1\end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix}-1 & 1 & 0\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix} = 1 > 0 \ (\text{Correct as } \vec{w}^T\vec{x} > 0 \text{ for } w1 \text{ samples}, \ w(\vec{6}) = w(\vec{5}))$$

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \text{-1} < 0$$

(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, $w(\vec{7}) = w(\vec{6})$)

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T\vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \text{-1} < 0$$

(Correct as $\vec{w}^T\vec{x} < 0$ for $w2$ samples, $w(\vec{7}) = w(\vec{6})$)

- Since for four consecutive steps no correction is needed, all points are correctly classified and the algorithm terminates. Final weight vector $w = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T$.

Example : Perceptron Learning Algorithm

7. Consider third data point $\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, find $\vec{w}^T \vec{x}$

$$\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \text{-1} < 0$$

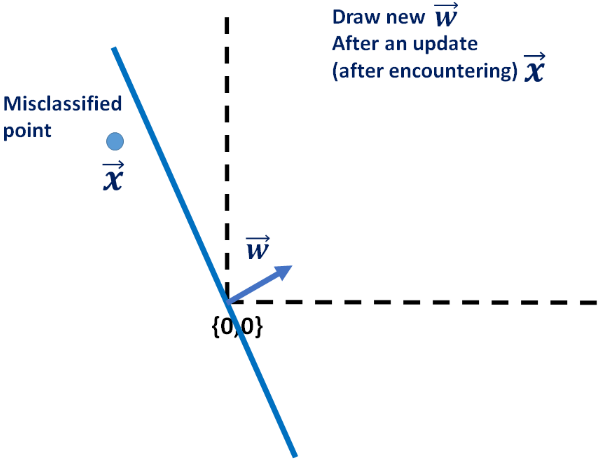
(Correct as $\vec{w}^T \vec{x} < 0$ for $w2$ samples, $w(\vec{7}) = w(\vec{6})$)

- Since for four consecutive steps no correction is needed, all points are correctly classified and the algorithm terminates. Final weight vector $w = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T$.

- That is the resulting linear classifier that correctly separates all data points. This line has slope = 1 and intercept = 0, how?

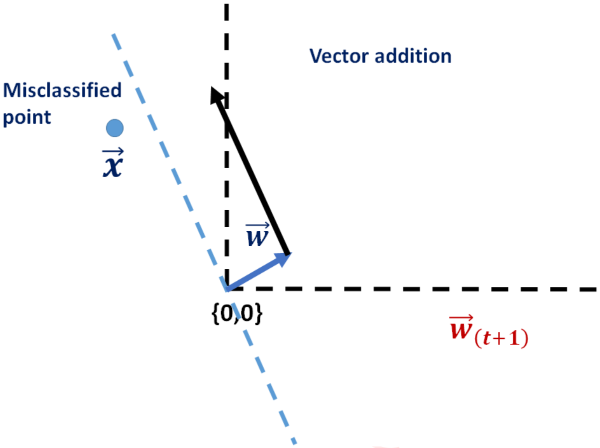
Section Contents

w update visualization



- Draw new $\vec{w}$ after encountering $\vec{x} \in w_+$, which is misclassified point.
- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

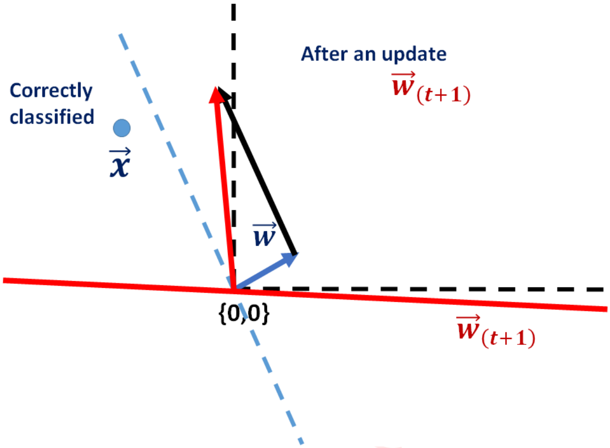
w update visualization



- Draw new $\vec{w}$ after encountering $\vec{x} \in w_+$, which is misclassified point.

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

w update visualization

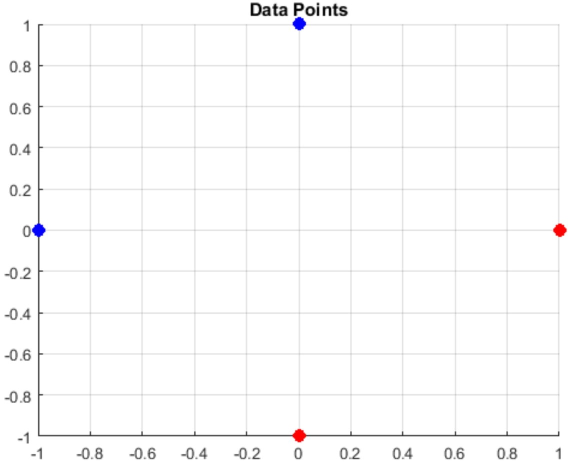


- Draw new $\vec{w}$ after encountering $\vec{x} \in w_+$, which is misclassified point.

- update rule, w : $\vec{w} \leftarrow \vec{w} + y\vec{x}$

- In our example after an update $\vec{x}$ gets correctly classified but there is no guarantee that after one update data point will be correctly classified.

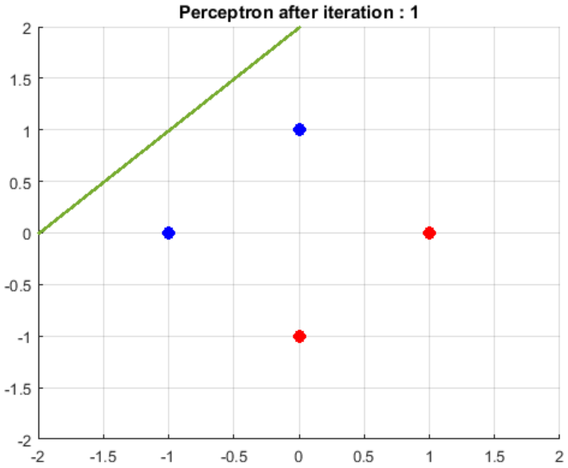
Demo 1: Perceptron Learning Algorithm



*3

[3] Matlab demo available

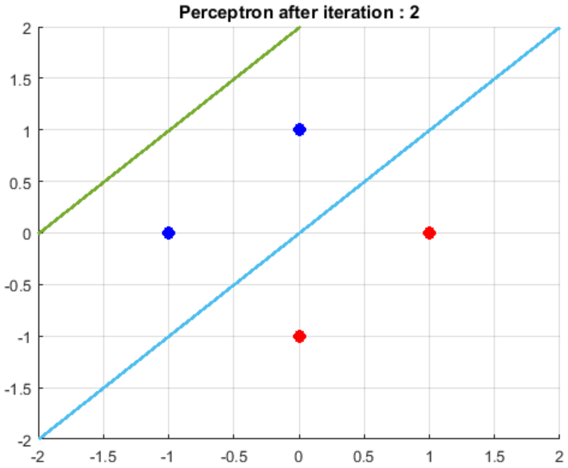
Demo 1: Perceptron Learning Algorithm



*3

[3] Matlab demo available

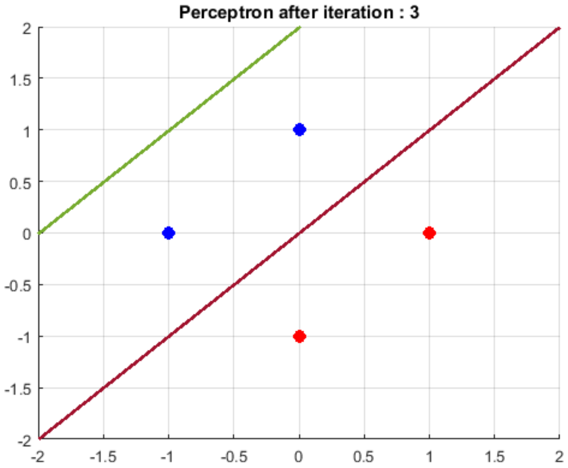
Demo 1: Perceptron Learning Algorithm



*[3]

[3] Matlab demo available

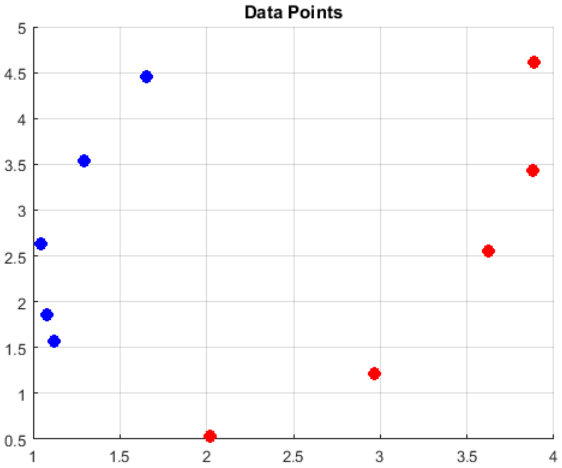
Demo 1: Perceptron Learning Algorithm



*[3]

[3] Matlab demo available

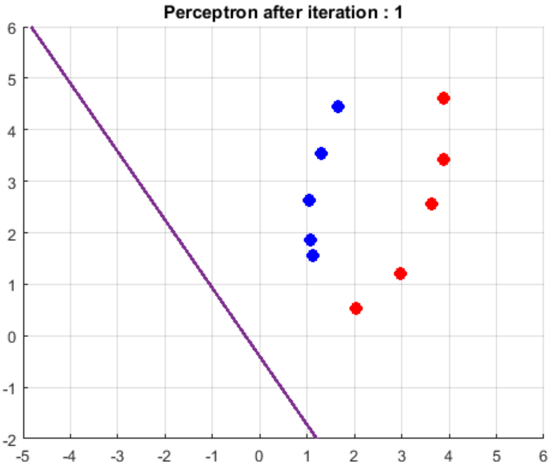
Demo 2: Perceptron Learning Algorithm



*4

[4]Matlab demo available

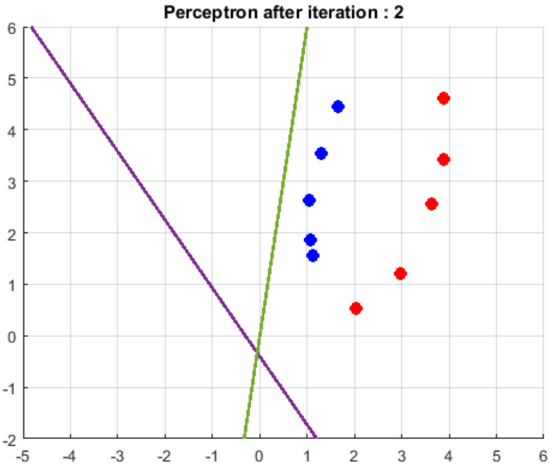
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

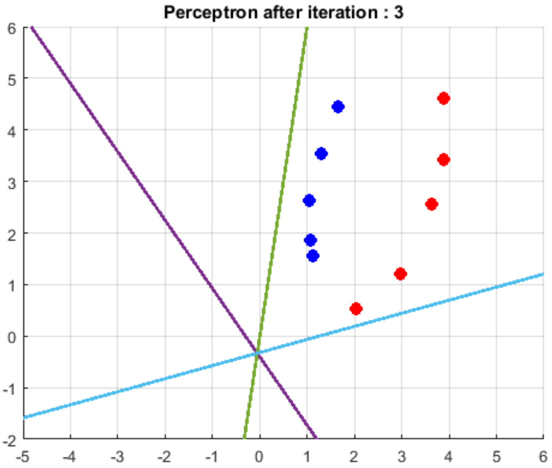
Demo 2: Perceptron Learning Algorithm



*4

4 Matlab demo available

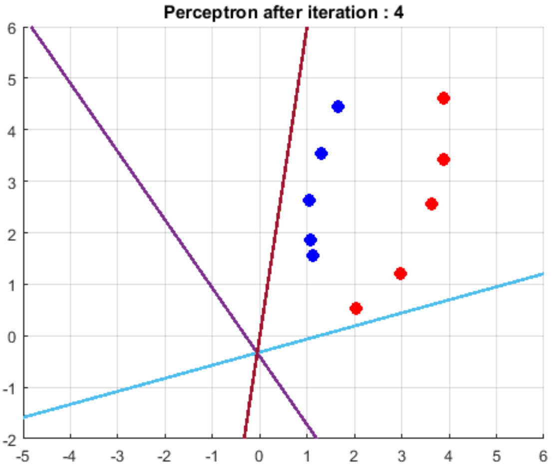
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

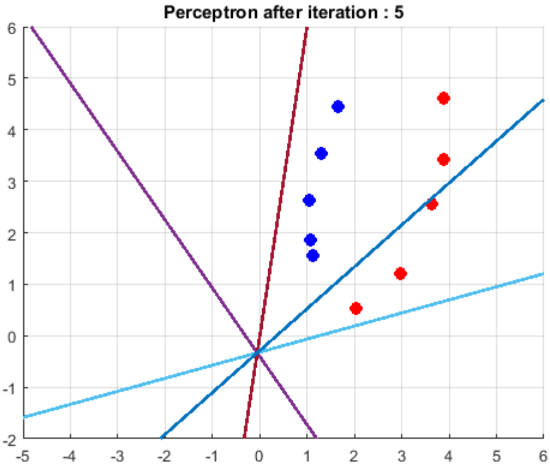
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

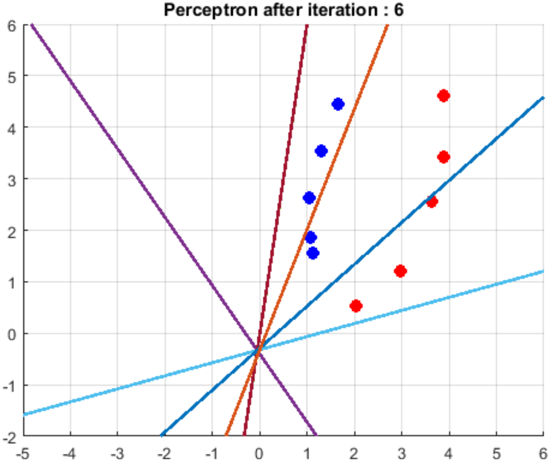
Demo 2: Perceptron Learning Algorithm



*4

4 Matlab demo available

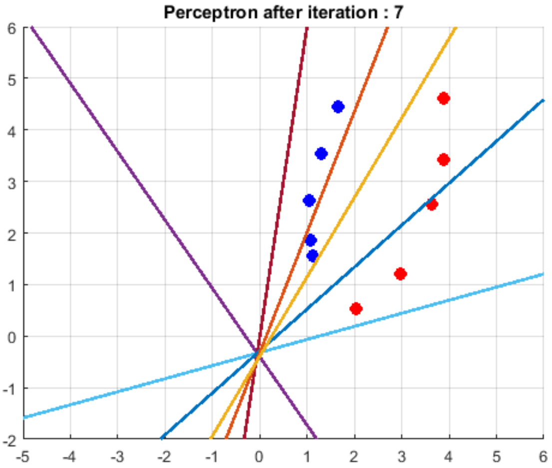
Demo 2: Perceptron Learning Algorithm



*4

4 Matlab demo available

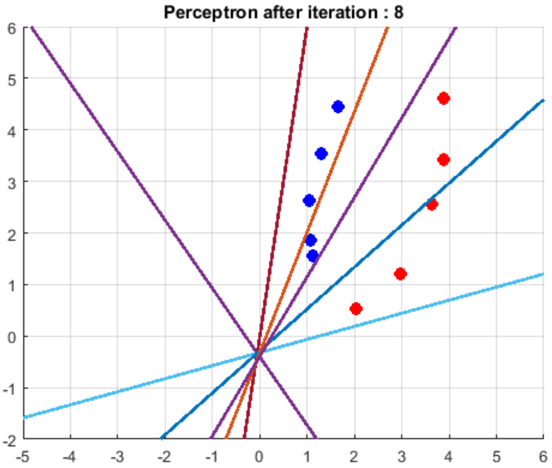
Demo 2: Perceptron Learning Algorithm



*4

[4] Matlab demo available

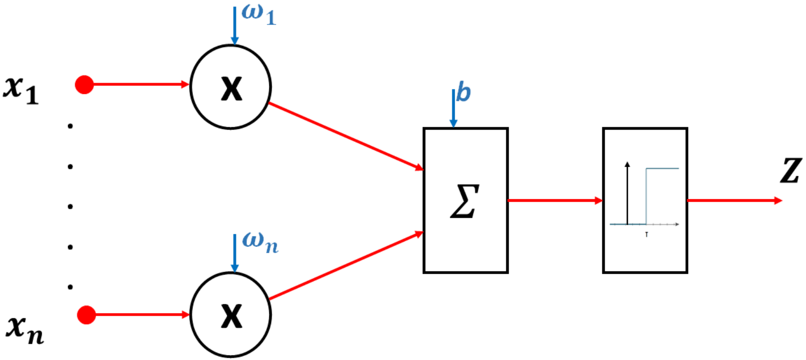
Demo 2: Perceptron Learning Algorithm



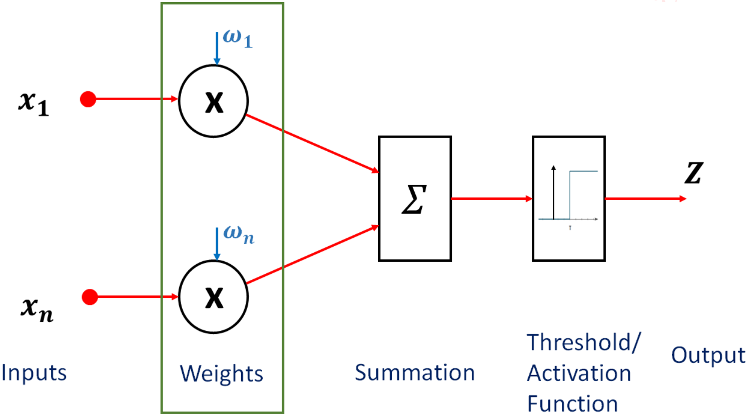
*4

[4]Matlab demo available

Perceptron or Artificial Neuron



Artificial Neuron

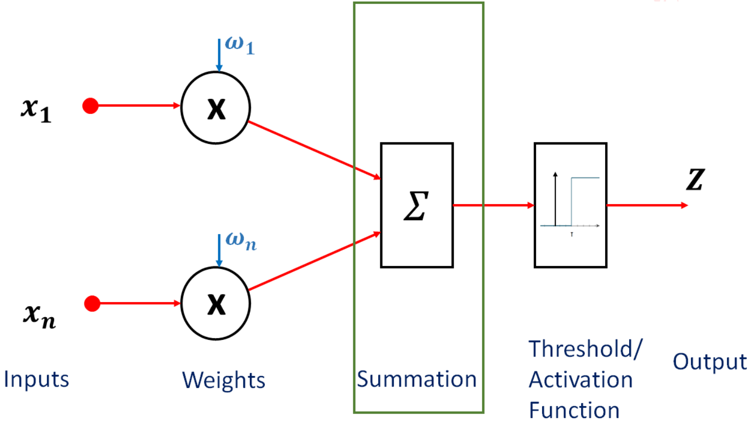


Step 1

Modeling synaptic connection.

$$x_i \times w_i$$

Artificial Neuron

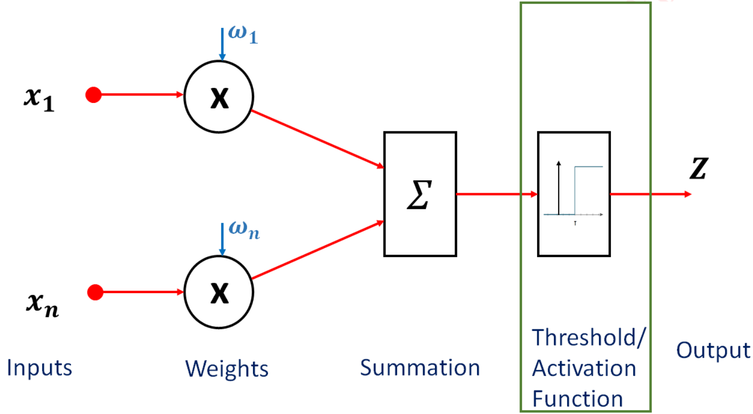


Step 2

Modeling collection of inputs

$$\sum_i x_i w_i$$

Artificial Neuron



Step 3

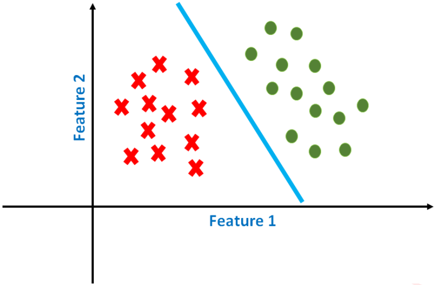
Decision, whether collective input is more than threshold to fire neuron

$$f(x) = \begin{cases} 1, if x \geq T \\ 0, otherwise \end{cases}$$

Section Contents

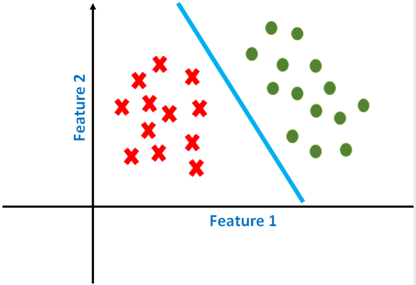
Perceptron Algorithm

- First algorithm with a strong formal guarantee of convergence.
  1. If the data is linearly separable, it will find a separating hyperplane in a finite number of updates.
  2. If the data is not linearly separable, it will loop forever.



Perceptron Algorithm

- First algorithm with a strong formal guarantee of convergence.
  1. If the data is linearly separable, it will find a separating hyperplane in a finite number of updates.
  2. If the data is not linearly separable, it will loop forever.



Perceptron Algorithm

- If $\exists \mathbf{w}$ such that $y_i(\mathbf{w}^\top \mathbf{x}) > 0 \; \forall (\mathbf{x}_i, y_i) \in D$, then Perceptron will find that $\mathbf{w}$ in finite number of steps.
  - **Condition to satisfy:**

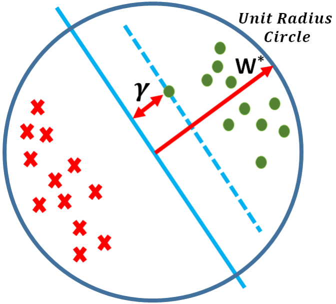
$$y(\mathbf{w}^\top \mathbf{x}) > 0 \begin{cases} y &= +1 : \; \mathbf{w}^\top \mathbf{x} > 0 \\ y &= -1 : \; \mathbf{w}^\top \mathbf{x} < 0 \end{cases} \tag{5}$$

  - **Update rule:**

$$\vec{w} \leftarrow \vec{w} + y\vec{x} \begin{cases} y &= +1 : \; \vec{w} \leftarrow \vec{w} + \vec{x} \\ y &= -1 : \; \vec{w} \leftarrow \vec{w} - \vec{x} \end{cases} \tag{6}$$

Perceptron Convergence : Setup

1. If $\exists \mathbf{w}^*$ such that $y_i(\mathbf{w}*^\top \mathbf{x}) > 0 \ \forall(\mathbf{x}_i, y_i) \in D$
2. Rescale each data point and the $\mathbf{w}^*$ such that:
   $||\mathbf{w}^*|| = 1$ and $||\mathbf{x}_i|| \leq 1 \ \forall \mathbf{x}_i \in D$
   - To get $||\mathbf{x}_i|| \leq 1$, divide all $\mathbf{x}$ by norm of max of $\mathbf{x}$.
3. Let us define the Margin (it's a constant) (the distance from the hyperplane to the closest data point) γ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i)\in D} |\mathbf{w}*^\top \mathbf{x}_i|$

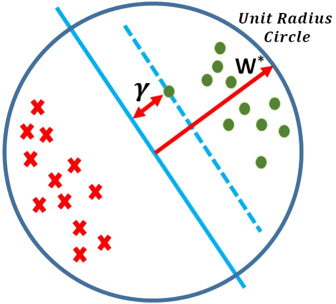


Note:

$\mathbf{w}^*$ is one of the hyperplane that separates data and point no. 2 elaborates on how data is scaled to be confined in unit radius circle. This helps in proof of convergence.

Perceptron Convergence : Setup

1. If $\exists \mathbf{w}^*$ such that $y_i(\mathbf{w}*^\top \mathbf{x}) > 0 \ \forall (\mathbf{x}_i, y_i) \in D$
2. Rescale each data point and the $\mathbf{w}^*$ such that:
   $||\mathbf{w}^*|| = 1 \quad$ and $\quad ||\mathbf{x}_i|| \leq 1 \ \ \forall \mathbf{x}_i \in D$
   - To get $||\mathbf{x}_i|| \leq 1$, divide all $\mathbf{x}$ by norm of max of $\mathbf{x}$.
3. Let us define the Margin (it's a constant) (the distance from the hyperplane to the closest data point) γ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i) \in D} |\mathbf{w}*^\top \mathbf{x}_i|$



Note:

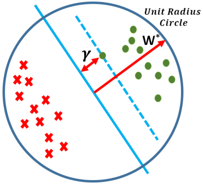
$\mathbf{w}^*$ is one of the hyperplane that separates data and point no. 2 elaborates on how data is scaled to be confined in unit radius circle. This helps in proof of convergence.

Theorem

If all of the above holds, then the Perceptron algorithm makes at most $1/\gamma^2$ mistakes before it converges.

Perceptron Convergence : Setup

- $\mathbf{w}$ is initial hyperplane that we have (let's say all zeros)
- $\mathbf{w}^*$ is a separating hyperplane that we want to obtain.
- Keeping previous definition, consider the effect of an update $(\mathbf{w} + y\mathbf{x})$ on the two terms:
  1. $\mathbf{w}^\top \mathbf{w}^*$
  2. $\mathbf{w}^\top \mathbf{w}$,



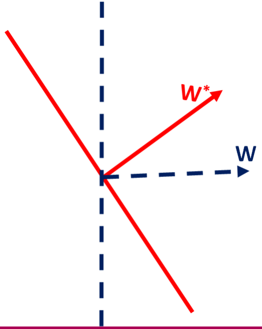
Why these two terms?

1. First Term ($\mathbf{w}^\top \mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top \mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.

Perceptron Convergence : Two Terms

Why these two terms?

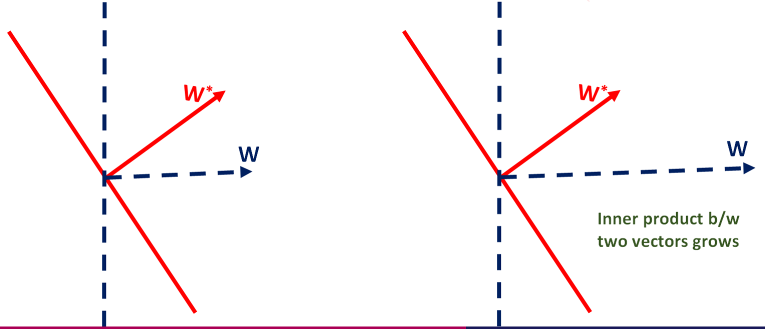
1. First Term ($\mathbf{w}^\top\mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top\mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.



Perceptron Convergence : Two Terms

Why these two terms?

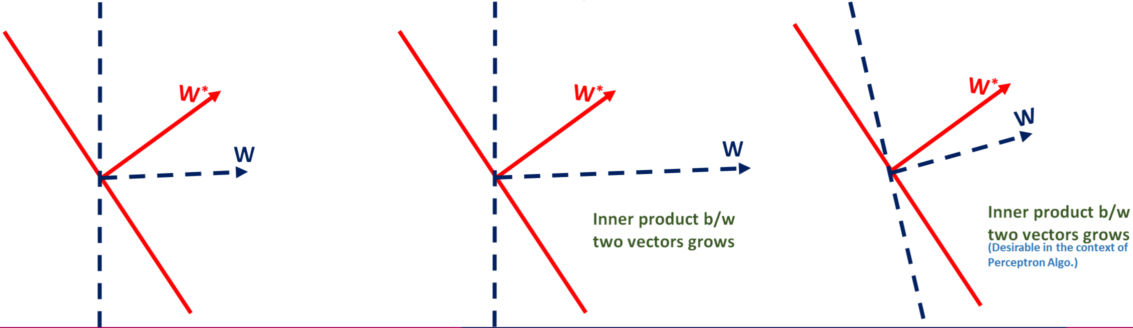
1. First Term ($\mathbf{w}^\top \mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top \mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.



Perceptron Convergence : Two Terms

Why these two terms?

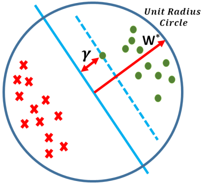
1. First Term ($\mathbf{w}^\top\mathbf{w}^*$): Calculates how closer $\mathbf{w}$ is getting to $\mathbf{w}^*$, inner product.
2. Second term ($\mathbf{w}^\top\mathbf{w}$): This is required in order to understand that increase in first term is not due to scaling (first term can grow even if hyperplanes are not getting close but getting scaled i.e. scaled by 2) but these hyperplanes are actually getting closer i.e. $\mathbf{w}$ is tilting towards $\mathbf{w}^*$. So it is required that this term should not grow fast.



Perceptron Convergence : First Term

Two facts, in case $\mathbf{w}$ gets updated

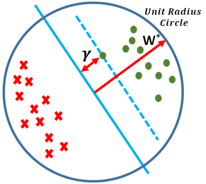
1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



Perceptron Convergence : First Term

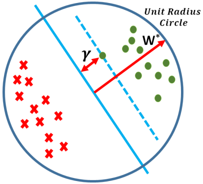
Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:

Perceptron Convergence : First Term



Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

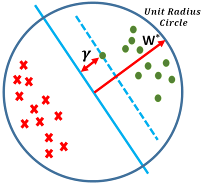
- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:
  -

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + \underbrace{y(\mathbf{x}^\top \mathbf{w}^*)}_{>0 \text{ or } \geq\gamma} \underbrace{\geq \mathbf{w}^\top \mathbf{w}^* + \gamma}_{Resultant} \quad (7)$$

Perceptron Convergence : First Term

Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

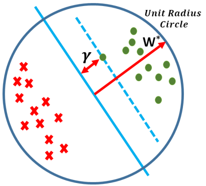


- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:
  -

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + \underbrace{y(\mathbf{x}^\top \mathbf{w}^*)}_{>0 \text{ or } \geq \gamma} \underbrace{\geq \mathbf{w}^\top \mathbf{w}^* + \gamma}_{Resultant} \tag{7}$$

the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ
or
$y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$

Perceptron Convergence : First Term



Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (first term)**, which is $\mathbf{w}^\top \mathbf{w}^*$:
  - $$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + \underbrace{y(\mathbf{x}^\top \mathbf{w}^*)}_{>0 \text{ or } \geq \gamma} \underbrace{\geq \mathbf{w}^\top \mathbf{w}^* + \gamma}_{Resultant} \tag{7}$$

    the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ
    or
    $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$

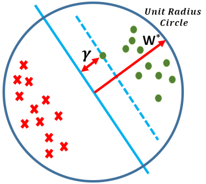
Conclusion-1

This means that for each update, $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ i.e. $\mathbf{w}^\top \mathbf{w}^* + \gamma$.

Perceptron Convergence : Second Term

Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

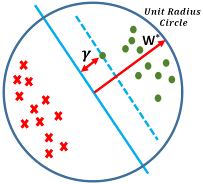


- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (second term)**, which is $\mathbf{w}^\top \mathbf{w}$:

Perceptron Convergence : Second Term

Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

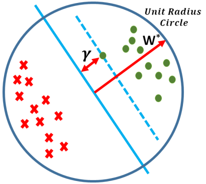


- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (second term)**, which is $\mathbf{w}^\top \mathbf{w}$:
  -
$$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2}_{=1} \underbrace{(\mathbf{x}^\top \mathbf{x})}_{\leq 1} \underbrace{\leq \mathbf{w}^\top \mathbf{w} + 1}_{Resultant} \tag{8}$$

Perceptron Convergence : Second Term

Two facts, in case $\mathbf{w}$ gets updated

1. $y(\mathbf{x}^\top\mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise update wouldn't happen.
2. $y(\mathbf{x}^\top\mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.



- How this update ($\vec{w} \leftarrow \vec{w} + y\vec{x}$) **effects (second term)**, which is $\mathbf{w}^\top\mathbf{w}$:
  -
$$(\mathbf{w}+y\mathbf{x})^\top(\mathbf{w}+y\mathbf{x}) = \mathbf{w}^\top\mathbf{w} + \underbrace{2y(\mathbf{w}^\top\mathbf{x})}_{<0} + \underbrace{y^2}_{=1}\underbrace{(\mathbf{x}^\top\mathbf{x})}_{\leq 1}\underbrace{\leq \mathbf{w}^\top\mathbf{w}+1}_{Resultant} \tag{8}$$
  - The inequality follows from the fact that:
    - $2y(\mathbf{w}^\top\mathbf{x}) < 0$ as we had to make an update, meaning $\mathbf{x}$ was misclassified.
    - $0 \leq y^2(\mathbf{x}^\top\mathbf{x}) \leq 1$ as $y^2 = 1$ and $\mathbf{x}^\top\mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$, data was scaled to have max. norm of 1)

Conclusion-2

This means that for each update, $\mathbf{w}^\top\mathbf{w}$ grows by at most 1, i.e. $\mathbf{w}^\top\mathbf{w}+1$.

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| =$$

1 (Data scaled)

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top\mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top\mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top\mathbf{w} \leq M$ as $\mathbf{w}^\top\mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top\mathbf{w}^* = \underbrace{|\mathbf{w}^\top\mathbf{w}^*|}_{Abs.Val.}$$

- What do we know about $\mathbf{w}^\top\mathbf{w}$?

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \text{ as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top\mathbf{w}}}_{\text{Definition of norm}}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$

- What do we know about $\mathbf{w}^\top \mathbf{w}$?
- $\mathbf{w}^\top \mathbf{w}$ grows by at most 1 Conc-2.

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$

$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$

$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$

- What do we know about $\mathbf{w}^\top \mathbf{w}$?
- $\mathbf{w}^\top \mathbf{w}$ grows by at most 1 Conc-2.
- So, after **M** updates:

$$= \sqrt{\mathbf{w}^\top \mathbf{w}} \leq \sqrt{M}$$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

After M updates, the following two inequalities must hold:

1. $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$ as $\mathbf{w}^\top \mathbf{w}^*$ grows by at least γ, so after M updates it must be at least $M\gamma$
2. $\mathbf{w}^\top \mathbf{w} \leq M$ as $\mathbf{w}^\top \mathbf{w}$ grows by at most 1

$$M\gamma \leq \mathbf{w}^\top \mathbf{w}^* = \underbrace{|\mathbf{w}^\top \mathbf{w}^*|}_{Abs.Val.}$$
$$\leq \underbrace{||\mathbf{w}^\top||.||\mathbf{w}^*||}_{\text{Cauchy-Schwarz inequality}} = ||\mathbf{w}^\top|| \quad \text{as } ||\mathbf{w}^*|| = 1 \text{ (Data scaled)}$$
$$= \underbrace{\sqrt{\mathbf{w}^\top \mathbf{w}}}_{\text{Definition of norm}}$$ [5]

- What do we know about $\mathbf{w}^\top \mathbf{w}$?
- $\mathbf{w}^\top \mathbf{w}$ grows by at most 1 Conc-2.
- So, after **M** updates:

$= \sqrt{\mathbf{w}^\top \mathbf{w}} \leq \sqrt{M}$

Interesting find

$M\gamma \leq \sqrt{M}$

[5] Cauchy-Schwarz inequality: For two vectors, their inner products is less than equal to product of their norms

Perceptron Convergence : Final Step

We proved

$$M\gamma \leq \sqrt{M}$$

Perceptron Convergence : Final Step

We proved

$$M\gamma \leq \sqrt{M}$$

- Solve for **M**:

Perceptron Convergence : Final Step

We proved

$M\gamma \leq \sqrt{M}$

- Solve for **M**:

$$M\gamma \leq \sqrt{M} \tag{9}$$

$$M^2\gamma^2 \leq M \tag{10}$$

$$M \leq \frac{1}{\gamma^2} \tag{11}$$

- This proof made Frank Rosenblatt famous. Such a strong result!

Perceptron Convergence : Final Step

We proved

$M\gamma \leq \sqrt{M}$

- Solve for **M**:

$$M\gamma \leq \sqrt{M} \tag{9}$$

$$M^2\gamma^2 \leq M \tag{10}$$

$$M \leq \frac{1}{\gamma^2} \tag{11}$$

- This proof made Frank Rosenblatt famous. Such a strong result!

Perceptron Algorithm Convergence

$M \leq \frac{1}{\gamma^2}$: This means number of updates **M** is bounded from above by a constant. So algorithm wouldn't make more mistakes than constant $\frac{1}{\gamma^2}$ (smallest distance between data point $\mathbf{x}$ and $\mathbf{w}^*$) before finding a linear separating hyperplane.

Section Contents

Interesting Facts

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in “Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms” published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book “Perceptrons”, co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

*[6]

[6] Image from Pattern Recognition and Machine Learning Book by Christopher Bishop

Interesting Facts

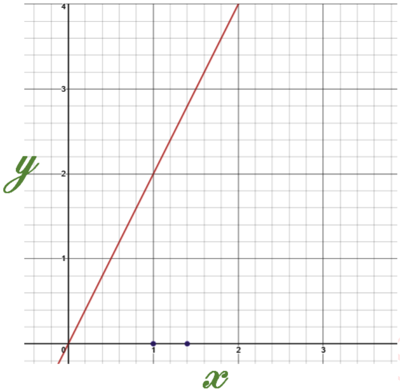
Figure 4.8 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a 20×20 array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

*[7]

[7] Image from Pattern Recognition and Machine Learning Book by Christopher Bishop

Section Contents

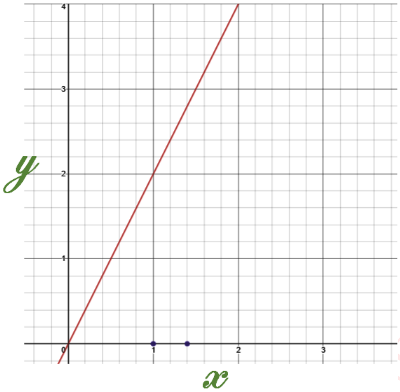
Equation of a line



Equation of a line:

$$y = mx + c$$

Equation of a line

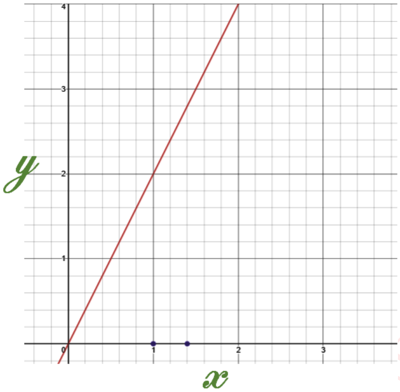


Equation of a line:

$$y = mx + c$$

$$m = \frac{rise}{run}$$

Equation of a line



Equation of a line:

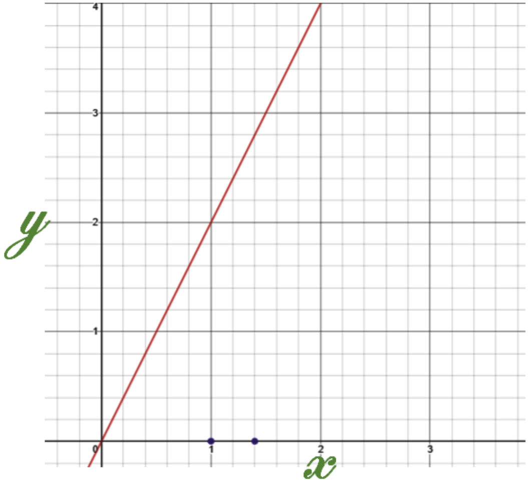
$$y = mx + c$$

$$m = \frac{rise}{run}$$

- m = slope
- c = y-intercept

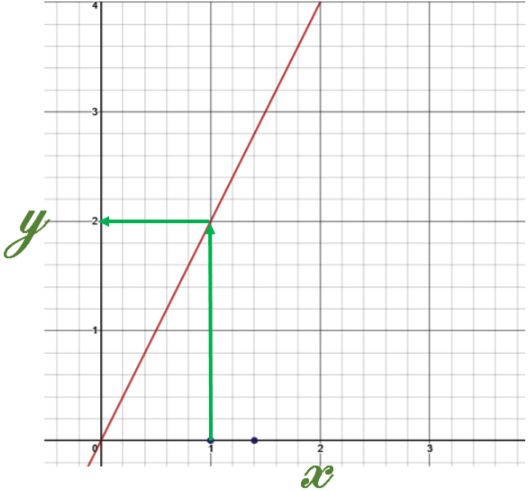
Equation of a line: Slope

Derivative / Slope Recap



Equation of a line: Slope

Derivative / Slope Recap



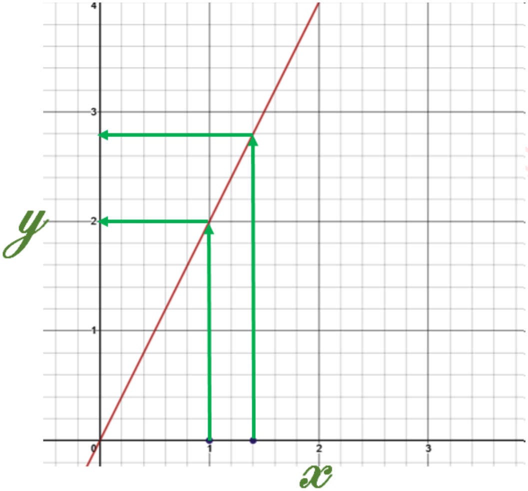
- Consider

$$f(x) = 2(x) \;\; or \;\; y = 2x$$

- if $x = 1$ then $f(x) = 2$

Equation of a line: Slope

Derivative / Slope Recap



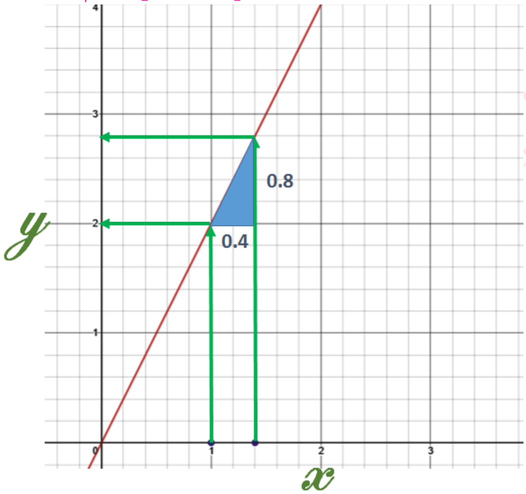
- Consider

$$f(x) = 2(x) \ \ or \ \ y = 2x$$

- if $x = 1$ then $f(x) = 2$
- if $x = 1.4$ then $f(x) = 2.8$

Equation of a line: Slope

Derivative / Slope Recap



- Consider

$$f(x) = 2(x) \;\; or \;\; y = 2x$$

- if $x = 1$ then $f(x) = 2$
- if $x = 1.4$ then $f(x) = 2.8$
- Slope $(\frac{dy}{dx})$ of $f(x)$ is 2.
  $\frac{dy}{dx} = \frac{height}{width}$
  $\frac{0.8}{0.4} = 2$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

This equation (ref Equation 12) is same as slope form of a line $y = mx + c$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

This equation (ref Equation 12) is same as slope form of a line $y = mx + c$

$$ax + by + c = 0 \tag{13}$$

$$y = \underbrace{-\frac{c}{b}}_{c\ or,\ y-intercept} - \underbrace{\frac{a}{b}}_{m\ or,\ slope} x \tag{14}$$

Equation of a line: General Form (2D)

$$ax + by + c = 0 \tag{12}$$

This equation (ref Equation 12) is same as slope form of a line $y = mx + c$

$$ax + by + c = 0 \tag{13}$$

$$y = \underbrace{-\frac{c}{b}}_{c\ or,\ y-intercept} - \underbrace{\frac{a}{b}}_{m\ or,\ slope} x \tag{14}$$

- If axis are x_1 and x_2, then $ax + by + c = 0$ can be written as:

$$ax_1 + bx_2 + c = 0 \tag{15}$$

Equation of a line: General Form (2D)

- Get rid of a and b as well, since we may need to write equation in n dimensions and then in this case we will run out of alphabets. Thus, Equation 15 can be written as:

$$w_1x_1 + w_2x_2 + w_0 = 0 \tag{16}$$

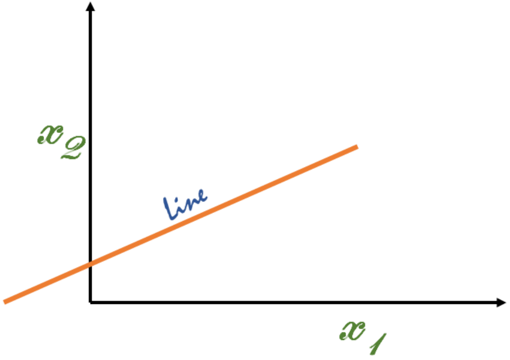
- What about in $3D$?

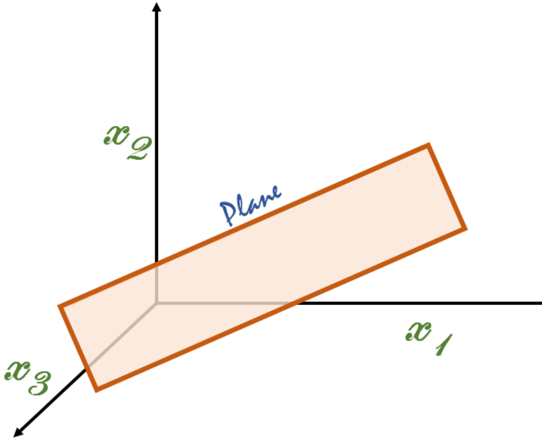
Plane in $3D$

- Equivalent of a line in $2D$ is a plane in $3D$.
- Idea is same. Line separates data in $2D$ surface, while plane separates data in $3D$ volume.

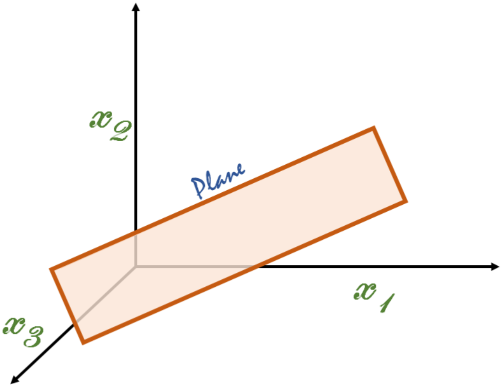
Plane in $3D$

- Equivalent of a line in $2D$ is a plane in $3D$.
- Idea is same. Line separates data in $2D$ surface, while plane separates data in $3D$ volume.





Plane in $3D$



- What about plane in nD?

- Extending Equation 16 to write equation of a plane in $3D$:

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0 = 0 \tag{17}$$

Plane in nD

- Plane in n dimensions is called hyperplane.
- Equation of a plane nD can be formulated easily from Equations 16 and 17.

$$w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = 0 \tag{18}$$

Plane in nD

- Plane in n dimensions is called hyperplane.
- Equation of a plane nD can be formulated easily from Equations 16 and 17.

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n = 0 \tag{18}$$

- Is there a more concise way to write this equation?

Plane in nD

- Plane in n dimensions is called hyperplane.
- Equation of a plane nD can be formulated easily from Equations 16 and 17.

$$w_0 + w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n = 0 \tag{18}$$

- Is there a more concise way to write this equation?

$$w_0 + \sum_{i=1}^{n} w_ix_i = 0 \tag{19}$$

- Above form is summation form / notation of an equation. Is there a vector form to write this equation?

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.
- Vector w has dimensions of $1 \times n$ ($w_{1 \times n}$)

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.
- Vector w has dimensions of $1 \times n$ ($w_{1\times n}$)
- Vector x has dimensions of $n \times 1$ ($x_{n\times 1}$)

Vector notation of a plane in nD

- Vector notation of a plane in nD

$$w_0 + \underbrace{[w_1, w_2, w_3, \cdots, w_n]}_{w\ vector} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{x\ vector} = 0 \tag{20}$$

- This equation, Equation 20 is exactly same as Equation 19.
- Vector w has dimensions of $1 \times n$ ($w_{1\times n}$)

- Vector x has dimensions of $n \times 1$ ($x_{n\times 1}$)

- Multiplication of vector w & vector x will give scalar or 1×1 matrix (multiplication of a row vector with a column vector).

Vector notation of a plane in nD

- In ML literature, as a standard, vector are written as column vector i.e. $\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}$

Taking Equation 20, and using standard notation, we can write:

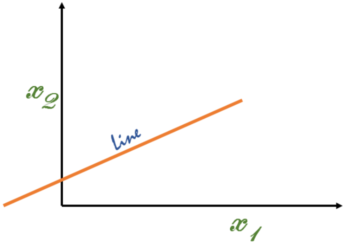
$$w_0 + \bar{w}^\top \bar{x} = 0 \tag{21}$$

- This is standard form of hyperplane equation!

Hyperplane equation with reference to line equation

- Equation of plane in 2D :

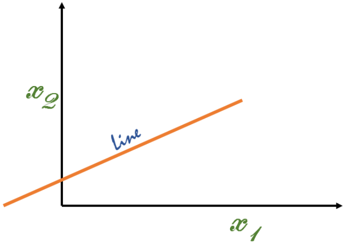
$$w_1x_1 + w_2x_2 + w_0 = 0$$



Hyperplane equation with reference to line equation

- Equation of plane in 2D :

$$w_1x_1 + w_2x_2 + w_0 = 0$$



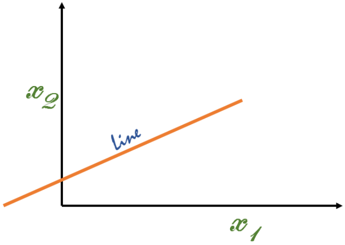
- Rearrange:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

Hyperplane equation with reference to line equation

- Equation of plane in 2D :

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$



- Rearrange:

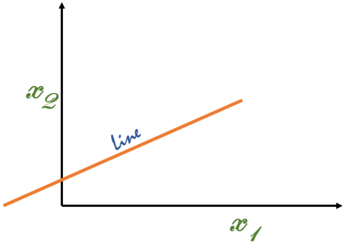
$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

- Can you find correspondence of this equation with $y = mx + c$

Hyperplane equation with reference to line equation

- Equation of plane in 2D :

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$



- Rearrange:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

- Can you find correspondence of this equation with $y = mx + c$

$$\underbrace{x_2}_{y} = -\underbrace{\frac{w_0}{w_2}}_{c} - \underbrace{\frac{w_1}{w_2}}_{m} x_1 \tag{22}$$

Hyperplane passing through origin

As we have seen:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

- If this line passes through origin then $c = 0$ or $w_0 = 0$. Then Equation 16 will become:

$$w_1 x_1 + w_2 x_2 = 0 \tag{23}$$

- In 3D (Plane)

Hyperplane passing through origin

As we have seen:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2}x_1$$

- If this line passes through origin then $c = 0$ or $w_0 = 0$. Then Equation 16 will become:

$$w_1x_1 + w_2x_2 = 0 \tag{23}$$

- In 3D (Plane)

$$w_1x_1 + w_2x_2 + w_3x_3 = 0 \tag{24}$$

- In nD (Hyperplane)

Hyperplane passing through origin

As we have seen:

$$x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$$

- If this line passes through origin then $c = 0$ or $w_0 = 0$. Then Equation 16 will become:

$$w_1 x_1 + w_2 x_2 = 0 \tag{23}$$

- In 3D (Plane)

$$w_1 x_1 + w_2 x_2 + w_3 x_3 = 0 \tag{24}$$

- In nD (Hyperplane)

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n = 0 \tag{25}$$

Hyperplane passing through origin

- Vector form of equation of hyperplane passing through origin:

$$\bar{w}^{\top}\bar{x} = 0 \tag{26}$$

- Vector form of equation of hyperplane not passing through origin:

$$w_0 + \bar{w}^{\top}\bar{x} = 0 \tag{27}$$

Geometric interpretation of Hyperplane

- Consider hyperplane that passes through origin, so Equation would be $\bar{w}^\top \bar{x} = 0$, where

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \; and \; x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$w \cdot x = w^\top x = ||w|| \, ||x|| \, cos\theta_{w,x} \tag{28}$$

Geometric interpretation of Hyperplane

- Consider hyperplane that passes through origin, so Equation would be $\bar{w}^{\top}\bar{x} = 0$, where

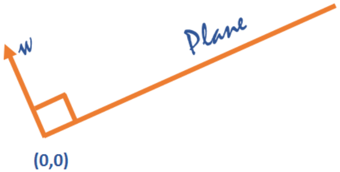
$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} \; and \; x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$w \cdot x = w^{\top} x = ||w|| \, ||x|| \, cos\theta_{w,x} \tag{28}$$

- According to definition of hyperplane passing through origin $\bar{w}^{\top}\bar{x} = 0$. This will only be true if vector w and x are orthogonal i.e ($cos(90) = 0$).

Geometric interpretation of Hyperplane

$$||w||\,||x||\,cos\theta_{w,x} = 0$$



- As w and x vectors are orthogonal

- Usually vector w is taken as vector perpendicular ($\perp$) to the hyperplane as well, for all data points / vector x lie on the plane.

Geometric interpretation of Hyperplane

$$||w||\,||x||\,cos\theta_{w,x} = 0$$

- As w and x vectors are orthogonal

- Usually vector w is taken as vector perpendicular ($\perp$) to the hyperplane as well, for all data points / vector x lie on the plane.
- Often hyperplane is defined by a unit vector $\hat{w} = \frac{w}{||w||}$ (e.g. $w \perp hyperplane$).

Geometric interpretation of Hyperplane

- Often hyperplane is defined by a unit vector $\hat{w} = \frac{w}{||w||}$ (e.g. $w \perp hyperplane$).