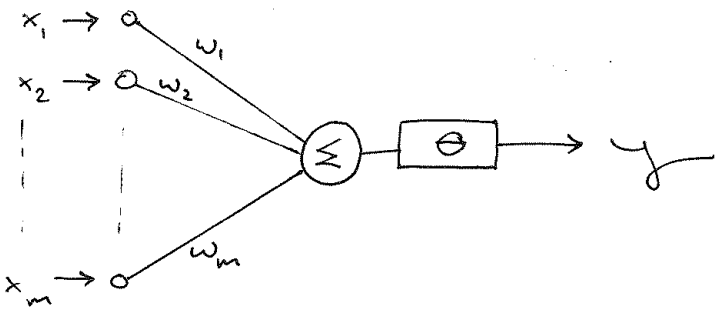
COMS 4030A

Adaptive Computation and Machine Learning

1. Perceptrons

1.1. Artificial Neurons.

An **artificial neuron** has the following structure:



The values $w_1, \ldots, w_m$ are real numbers called the **weights**.
The value of θ is a real number called the **threshold**.

An artificial neuron is a function from the set $\mathbb{R}^m$ to the set $\{0, 1\}$.

Given **input** values $x_1, \ldots, x_m$, where each $x_i \in \mathbb{R}$, the **output** y is obtained as follows.

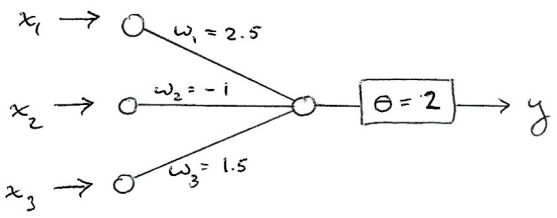
Calculate the sum $\displaystyle\sum_{i=1}^{m} x_i w_i$ and then compare its value with the **threshold** value θ:

$$\text{if } \sum_{i=1}^{m} x_i w_i > \theta \text{ then } y = 1;$$
$$\text{if } \sum_{i=1}^{m} x_i w_i \leq \theta \text{ then } y = 0.$$

In the case that $\sum_{i=1}^{m} x_i w_i > \theta$, which produces output $y = 1$, the neuron is said to **fire** or to **activate**.

A function made up of a single artificial neuron, as described above, is called a **perceptron**.

Example. Consider the following 3-input artificial neuron:



Suppose the following values are input to neuron: $x_1 = 1$, $x_2 = 3$, $x_3 = 2$.
The input can be considered a vector in $\mathbb{R}^3$, that is, $\boldsymbol{x} = (1, 3, 2)$.
The output corresponding to this input is obtained as follows:

$$\sum_{i=1}^{3} x_i w_i = 1(2.5) + 3(-1) + 2(1.5) = 2.5 \quad \text{and} \quad 2.5 > 2, \quad \text{so } y = 1.$$

If the input vector is $\boldsymbol{x} = (1.5, 1, -1)$ the corresponding output is obtained as follows:

$$\sum_{i=1}^{3} x_i w_i = (1.5)(2.5) + 1(-1) + (-1)(1.5) = 1.25 \quad \text{and} \quad 1.25 \leq 2, \quad \text{so } y = 0.$$

1.2. **Linear Discriminants.**

Consider a dataset consisting of points in $\mathbb{R}^m$ such that each point in the dataset is classified into one of two possible classes. Let 0 and 1 be the labels used to denote the two classes.

Consider the problem of finding a function that correctly classifies each input in the dataset. That is, given any input vector from the dataset, if the vector values are input into the function, then the function outputs the correct classification (i.e., 1 or 0) for that input.

The above problem is an example of a **classification** problem. We shall first consider solutions to such a problem using linear functions.

A **linear discriminant** for a dataset with classes as above is a function of the form

$$f(x_1, x_2, \ldots, x_m) = a_1 x_1 + a_2 x_2 + \cdots + a_m x_m + b,$$

where $a_1, \ldots, a_m, b \in \mathbb{R}$, such that for each datapoint $\boldsymbol{x} = (x_1, \ldots, x_m)$ in the dataset

$$f(x_1, x_2, \ldots, x_m) > 0 \text{ if, and only if, } \boldsymbol{x} \text{ has classification 1,}$$

$$f(x_1, x_2, \ldots, x_m) \leq 0 \text{ if, and only if, } \boldsymbol{x} \text{ has classification 0.}$$

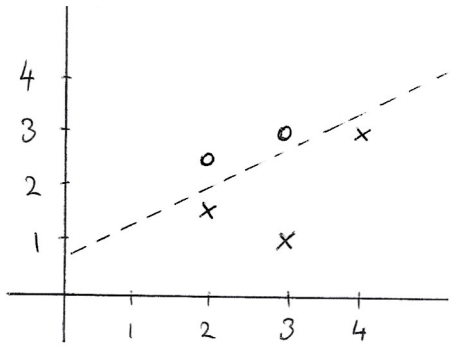
Example. Consider the following dataset:

$$D = \{((3,1),1),((2,2.5),0),((2,1.5),1),((4,3),1),((3,3),0)\},$$

which can be represented by a table, as follows:

$$X = \begin{bmatrix} 3 & 1 \\ 2 & 2.5 \\ 2 & 1.5 \\ 4 & 3 \\ 3 & 3 \end{bmatrix} \quad T = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

The values in X are input values and values in T are the corresponding classifications.
Since the input values are points in $\mathbb{R}^2$, they can also be represented on a plane as shown in the diagram below. The diagram depicts all the datapoints in D together with their classifications, where a $\circ$ indicates class 0 and a $\times$ indicates class 1.

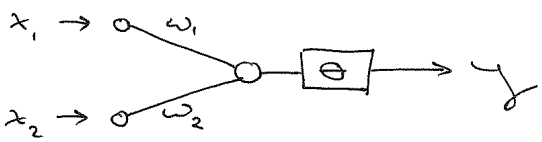


A linear discriminant for the dataset D is a function that corresponds to a straight line that separates the 1's from the 0's, (i.e., the $\times$'s from the $\circ$'s) such as the one shown by the dotted line in the diagram above.

Observe that once a linear discriminant has been found for D it can be used to predict the classification of other inputs. For example, using the above discriminant, the predicted classification for input $(3,2)$ would be 1 since the point lies below the dotted line.

Next, we show how an m-input perceptron is a linear discriminant in $\mathbb{R}^m$, and later we show how to train a perceptron as a linear discriminant for a given dataset.

First, consider a 2-input perceptron:



Recall that for input vector $\boldsymbol{x} = (x_1, x_2)$ the output y is calculated as follows:

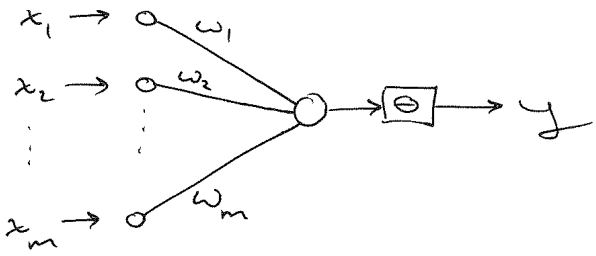
if $x_1w_1 + x_2w_2 > \theta$, then $y = 1$;
if $x_1w_1 + x_2w_2 \leq \theta$, then $y = 0$.

Another way to write this is:

if $x_1w_1 + x_2w_2 - \theta > 0$, then $y = 1$;
if $x_1w_1 + x_2w_2 - \theta \leq 0$, then $y = 0$.

Thus, the straight line with equation $x_1w_1 + x_2w_2 - \theta = 0$ (where w_1, w_2 and θ are constants) separates inputs with output $y = 1$ from those with output $y = 0$.
The function $f(x_1, x_2) = x_1w_1 + x_2w_2 - \theta$ is a linear discriminant.

Thus, a 2-input perceptron defines straight line, which is a linear discriminant in $\mathbb{R}^2$.
Similarly, a 3-input perceptron defines a plane, which is a linear discriminant in $\mathbb{R}^3$.
In general, consider an m-input perceptron:



The neuron fires, i.e., $y = 1$, if $\sum_{i=1}^{m} x_iw_i > \theta$.
The linear discriminant associated with this perceptron is $f(x_1, \ldots, x_m) = \sum_{i=1}^{m} x_iw_i - \theta$ and the equation $\sum_{i=1}^{m} x_iw_i - \theta = 0$ defines an m-**hyperplane**.

1.3. **Training a Perceptron.**

Consider a dataset D containing datapoints of the form $(\boldsymbol{x}, t)$, where each $\boldsymbol{x}$ is a point in $\mathbb{R}^m$ and each t is either 1 or 0. We call t the **target** for input $\boldsymbol{x}$.

The following method is used to train the weights and threshold of a perceptron so that it outputs the correct target for inputs from the dataset.

Perceptron Training Algorithm

Create an m-input perceptron by randomly choosing weights $w_1, \ldots, w_m$
and randomly choosing a threshold value θ.
(Usually, the random values are chosen from a fixed range such as $[-1, 1]$.)

Choose a **learning rate** η, where $0 < \eta < 1$.
(Usually, we choose η between 0.001 and 0.1.)

while (stopping condition is not satisfied)
    for each datapoint $(\boldsymbol{x}, t)$ in the set D, do the following:
        feed $\boldsymbol{x}$ into the perceptron, to get output y
        update each weight w_i using the rule: $w_i \leftarrow w_i + \eta(t - y)x_i$
        update the threshold θ using the rule: $\theta \leftarrow \theta - \eta(t - y)$

The final weights obtained by the Perceptron Training Algorithm give the trained perceptron. It should be the case that for 'most of' the training data, the output from the perceptron matches the target, i.e., if $(\boldsymbol{x}, t)$ is in the dataset and $\boldsymbol{x}$ is input into the perceptron, then the output y should be the same as the target t.

One complete execution of the above **for** loop (over all datapoints in D) is called an **epoch**. During an epoch, the order in which the datapoints are input can make a difference for learning the weights since the last datapoints have more impact than the first ones. A good idea is to randomize the order in which the datapoints are input at the start of each epoch.

1.4. **Stopping Conditions.**

The Perceptron Training Algorithm can be set to stop if there are no changes to any weights during an epoch. Alternately, since this may not happen, the algorithm can be set to stop after a fixed number of epochs or after some specified time has elapsed.

Another way to determine if the perceptron has trained for long enough is to calculate the **loss** (or **error**) of the perceptron on the dataset.

Consider a dataset $D = \{(\boldsymbol{x}_k, t_k) \mid 1 \leq k \leq N\}$ that has N datapoints, where each $\boldsymbol{x}_k \in \mathbb{R}^m$ and each $t_k \in \{0, 1\}$. Suppose that a perceptron is being trained on this dataset.
At the end of an epoch of training, the loss is calculated as follows.
For each datapoint $(\boldsymbol{x}_k, t_k)$ in D, feed the input $\boldsymbol{x}_k$ into the perceptron to get the output y_k and calculate the loss L using the formula:

$$L = \sum_{k=1}^{N} |t_k - y_k|.$$

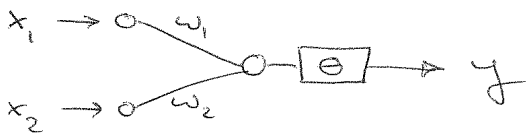
The above sum counts the number of times the perceptron misclassifies an input.
Ideally, we want $L = 0$ since this means that the perceptron gives the correct output for every input.

If, after some epoch of the Perceptron Training Algorithm, L does become 0, then training can stop. Since this may not happen, the training may be set to stop when L gets below some fixed value. Since even this is not guaranteed, one can stop when the change in the loss between epochs, i.e., ΔL, is close to 0. This means that the perceptron has stopped improving itself.

It is useful to keep a record of the L values and plot the value of L against the number of epochs to see how it changes with learning.

1.5. **Where do the perceptron learning rules come from?**

The update rules in the last two lines of the Perceptron Training Algorithm are known as the 'perceptron learning rules'. To understand how these rules are obtained, consider a 2-input perceptron:



Recall that for input vector $\boldsymbol{x} = (x_1, x_2)$ the output y is calculated as follows:

if $x_1 w_1 + x_2 w_2 > \theta$, then $y = 1$;
if $x_1 w_1 + x_2 w_2 \leq \theta$, then $y = 0$.

Observe that $x_1 w_1 + x_2 w_2$ is the dot-product of the vectors $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{w} = (w_1, w_2)$, that is, $\boldsymbol{x} \cdot \boldsymbol{w} = x_1 w_1 + x_2 w_2$.

Thus, the computation above can be written as

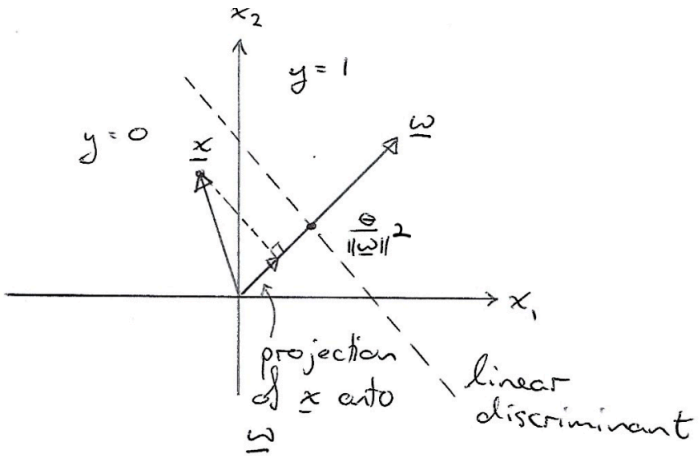
if $\boldsymbol{x} \cdot \boldsymbol{w} > \theta$, then $y = 1$;
if $\boldsymbol{x} \cdot \boldsymbol{w} \leq \theta$, then $y = 0$.

Plot vectors $\boldsymbol{x}$ and $\boldsymbol{w}$ in $\mathbb{R}^2$, and calculate the **projection** of $\boldsymbol{x}$ onto $\boldsymbol{w}$, which is the vector

$$\frac{\boldsymbol{x} \cdot \boldsymbol{w}}{||\boldsymbol{w}||^2} \boldsymbol{w}, \quad \text{where } ||\boldsymbol{w}|| = \sqrt{w_1^2 + w_2^2}.$$

Note that $\boldsymbol{x} \cdot \boldsymbol{w} > \theta$ iff $\dfrac{\boldsymbol{x} \cdot \boldsymbol{w}}{||\boldsymbol{w}||^2} > \dfrac{\theta}{||\boldsymbol{w}||^2}$, so $y = 1$ if the projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$ lies above the point $\dfrac{\theta}{||\boldsymbol{w}||^2}$ along $\boldsymbol{w}$, and $y = 0$ if the projection lies below that point.

Thus, the linear discriminant given by the perceptron is the line perpendicular to $\boldsymbol{w}$ at the point $\dfrac{\theta}{||\boldsymbol{w}||^2}$ along $\boldsymbol{w}$, as shown in the diagram below. Points above the dotted line produce output $y = 1$ and points below the line produce output $y = 0$.



To return to the perceptron update rules, consider a dataset D of datapoints of the form $(\boldsymbol{x}, t)$, where $\boldsymbol{x} \in \mathbb{R}^2$ and $t \in \{0, 1\}$. Consider a fixed datapoint $(\boldsymbol{x}, t)$ for which the existing perceptron produces the wrong output for input $\boldsymbol{x}$, i.e., $y \neq t$. We want to update the weights w_1, w_2 and θ of the perceptron so that if $\boldsymbol{x}$ is fed into the perceptron again, the output matches the target.

Suppose $\boldsymbol{x}$ has target $t = 1$, but when fed through the perceptron it outputs $y = 0$.

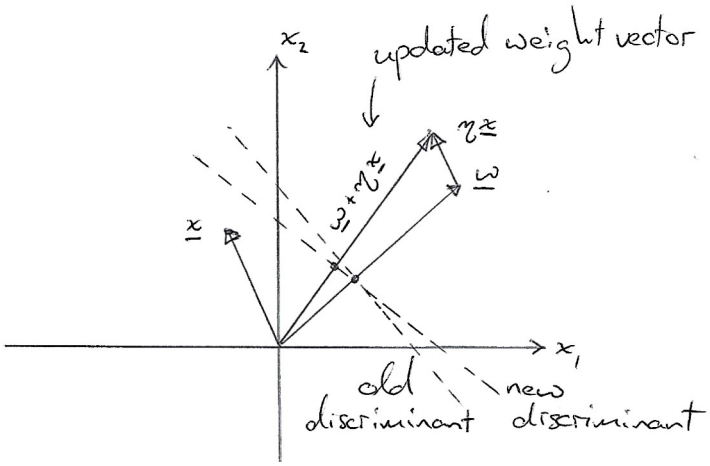
The output $y = 0$ means that $\dfrac{\boldsymbol{x} \cdot \boldsymbol{w}}{||\boldsymbol{w}||^2} \leq \dfrac{\theta}{||\boldsymbol{w}||^2}$.

To change the output to $y = 1$, we must change the weights w_i so that $\dfrac{\boldsymbol{x} \cdot \boldsymbol{w}}{||\boldsymbol{w}||^2} > \dfrac{\theta}{||\boldsymbol{w}||^2}$.

This is achieved by moving the vector $\boldsymbol{w}$ incrementally in the direction of $\boldsymbol{x}$, i.e.,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \boldsymbol{x}.$$

By changing $\boldsymbol{w}$ in this way, the linear discriminant moves so as to try include the point $\boldsymbol{x}$ on the correct side of the line, as illustrated in the diagram below. (Recall that $0 < \eta < 1$.)



Since we only want to change the weights if $\boldsymbol{x}$ is misclassified, we add the term $(t - y)$ into the update rule above:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta(t - y)\boldsymbol{x}.$$

Then, if $t = y$, the weights in $\boldsymbol{w}$ are not changed.
If $t = 1$ and $y = 0$, then $\boldsymbol{w}$ will move towards $\boldsymbol{x}$.
If $t = 0$ and $y = 1$, then the update is $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\boldsymbol{x}$, i.e., the weight vector moves away from $\boldsymbol{x}$.

Since $\boldsymbol{w} = (w_1, w_2)$ and $\boldsymbol{x} = (x_1, x_2)$, the update rule can be written as

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \leftarrow \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \eta(t - y) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

which can be written as

$$w_i \leftarrow w_i + \eta(t - y)x_i, \quad \text{for each } i.$$

The above rule then generalises to perceptrons with m inputs.

Next, consider the case of θ.

Suppose that input $\boldsymbol{x}$ has target $t = 1$ but the perceptron gives output $y = 0$.

The value of θ is changed using the update rule:

$$\theta \leftarrow \theta - \eta.$$

By making this change, the point $\frac{\theta}{||\boldsymbol{w}||^2}$ along $\boldsymbol{w}$ that determines the linear discriminant moves downwards to try include $\boldsymbol{x}$ on the correct side of the line.

In the case where $t = 0$ and $y = 1$ we use the update rule:

$$\theta \leftarrow \theta + \eta.$$

Then the point $\frac{\theta}{||\boldsymbol{w}||^2}$ along $\boldsymbol{w}$ that determines the linear discriminant moves upwards to try include $\boldsymbol{x}$ on the correct side of the line.

Both update rules can be described using one rule: $\theta \leftarrow \theta - \eta(t - y)$.

EXERCISES

(1) Consider a 3-input perceptron with weights $w_1 = 2.5$, $w_2 = -3$, $w_3 = 1.5$ and threshold $\theta = 2$.
Find the output of the perceptron for the following input vectors:
(a) $\boldsymbol{x} = (-1, 2, 4)$
(b) $\boldsymbol{x} = (2, -1, -2)$

(2) Using the perceptron in exercise (1), do one update of the weights w_1, w_2, w_3 and threshold θ as follows: Suppose the input vector $\boldsymbol{x} = (1, 1, 2)$ from your dataset has target $t = 0$. When $\boldsymbol{x}$ is fed into the perceptron the output we get is $y = 1$, which differs from the target t. Update all the weights and threshold using the rules in the PERCEPTRON TRAINING ALGORITHM. Use $\eta = 0.1$.