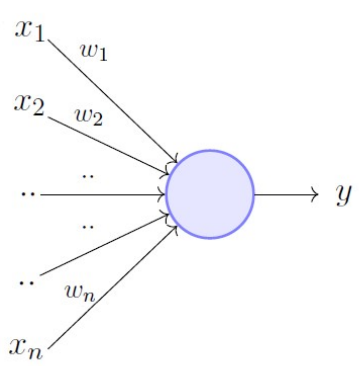
The Perceptron Learning Algorithm

Learning And Machine Learning

Learning is the ability to improve one's behavior with experience. In focusing on the experience aspect of learning, machine learning can be be defined as building computers, systems that automatically improve with experience. A typical algorithm solves a problem by giving a computer data and a program to give a suitable output. Machine learning involves giving the data and examples of outcomes to the computer to output a model for getting suitable outputs.



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

Synopsis of The Perceptron

The Perceptron learning algorithm is an algorithm that is used for supervised learning of binary clusters.The supervised learning involves giving conditions(inputs) with documented outputs. An allusion of learning and the binary clusters means that the choice of output has only two outcomes. The Perceptron was invented in 1957 by Frank Rosenblatt Ph.D. The idea is that the Perceptron works just like the neuron of a nervous system. Each neuron receives thousands of signals from other neurons, connected via synapses. Once the sum of the signals being received surpasses a certain threshold, a response is sent through the axon.

Composition of the Perceptron Learning Algorithm

The implementation of the perceptron learning algorithm involves using a collection of features to answer a question that has two choices; binary cluster. And the algorithm learns to make these choices from being exposed to previous data collected with resultant outcomes with one of the two choices. So what we have actualy is: $\mathcal{X} \subseteq \mathbb{R}^d$ and $d \in \mathbb{N}$ be the input space, and let $\mathcal{Y} = \{-1, 1\}$ x: Input customer information that is used to make credit decision.

- $f : \mathcal{X} \to \mathcal{Y}$: *Unknown target* function that is the ideal formula for credit approval.
- $\mathcal{X}$: *Input space* consisting of all possible input x.
- $\mathcal{Y}$: *Output space* consisting of no or yes credit approval.
- $\mathcal{D}$: *Data set* of tuples in input-output examples of the form (x_i, y_i), where $f(x_i) = y_i$ and $i \in \mathbb{N}$.
- $\mathcal{A}$: Learning algorithm which uses D to pick a formula (hypothesis) $g : \mathcal{X} \to \mathcal{Y}$ so that $g \approx f$, where $g \in \mathcal{H}$. Here $\mathcal{H}$ is the *hypothesis space*.

For $h \in \mathcal{H}$, $h(x)$ gives different weights to the different coordinates of x. This reflects the relative importance of each coordinate to the credit decision. The combinded weighted coordinates form a credit score which is compared to some threshold, say $theta$.

- Approve if

$$\sum_{i=1}^{d} w_i x_i > \theta$$

- Deny if

$$d$$

In [82]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('iris_data.csv')
df = df[0:100]
#shuffled the data to include all all observations
df = df[0:100].sample(frac=1).reset_index(drop=True)
print(df)
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.2	2.2	4.5	1.5	versicolor
1	6.6	2.9	4.6	1.3	versicolor
2	5.1	3.8	1.5	0.3	setosa
3	4.9	3.6	1.4	0.1	setosa
4	5.1	2.5	3.0	1.1	versicolor
5	5.1	3.3	1.7	0.5	setosa
6	6.1	3.0	4.6	1.4	versicolor
7	6.0	2.2	4.0	1.0	versicolor
8	6.0	2.9	4.5	1.5	versicolor
9	5.8	4.0	1.2	0.2	setosa
10	4.3	3.0	1.1	0.1	setosa
11	5.6	2.5	3.9	1.1	versicolor
12	5.0	2.0	3.5	1.0	versicolor
13	6.1	2.8	4.0	1.3	versicolor
14	5.1	3.4	1.5	0.2	setosa
15	5.7	2.8	4.5	1.3	versicolor
16	5.5	3.5	1.3	0.2	setosa
17	6.3	3.3	4.7	1.6	versicolor
18	5.8	2.7	4.1	1.0	versicolor
19	5.0	3.3	1.4	0.2	setosa
20	4.8	3.1	1.6	0.2	setosa
21	6.1	2.8	4.7	1.2	versicolor
22	5.0	3.0	1.6	0.2	setosa
23	4.8	3.0	1.4	0.1	setosa
24	4.8	3.4	1.9	0.2	setosa
25	6.0	2.7	5.1	1.6	versicolor
26	4.6	3.2	1.4	0.2	setosa
27	5.7	2.9	4.2	1.3	versicolor
28	6.8	2.8	4.8	1.4	versicolor
29	6.9	3.1	4.9	1.5	versicolor
..	...	...	...	...	...
70	5.7	3.0	4.2	1.2	versicolor
71	5.2	4.1	1.5	0.1	setosa
72	6.4	2.9	4.3	1.3	versicolor
73	6.3	2.3	4.4	1.3	versicolor
74	5.0	3.4	1.5	0.2	setosa
75	5.7	2.8	4.1	1.3	versicolor
76	5.5	2.4	3.8	1.1	versicolor
77	5.4	3.9	1.7	0.4	setosa
78	4.9	3.1	1.5	0.2	setosa
79	6.4	3.2	4.5	1.5	versicolor
80	4.5	2.3	1.3	0.3	setosa
81	6.7	3.1	4.7	1.5	versicolor
82	4.9	3.0	1.4	0.2	setosa
83	5.8	2.6	4.0	1.2	versicolor
84	5.0	3.6	1.4	0.2	setosa
85	4.6	3.4	1.4	0.3	setosa
86	6.7	3.0	5.0	1.7	versicolor
87	6.1	2.9	4.7	1.4	versicolor
88	5.0	2.3	3.3	1.0	versicolor
89	6.5	2.8	4.6	1.5	versicolor
90	6.7	3.1	4.4	1.4	versicolor
91	5.5	2.4	3.7	1.0	versicolor
92	6.6	3.0	4.4	1.4	versicolor
93	6.0	3.4	4.5	1.6	versicolor
94	7.0	3.2	4.7	1.4	versicolor
95	5.2	3.5	1.5	0.2	setosa
96	4.8	3.0	1.4	0.3	setosa
97	5.6	3.0	4.5	1.5	versicolor
98	5.0	3.4	1.6	0.4	setosa
99	5.1	3.8	1.9	0.4	setosa

[100 rows x 5 columns]

In [83]:
```
# get sepal length and petal lenght from the iris data set
X = [np.array([1.0, df['SepalLength'][i], df['PetalLength'][i]]) for i in range(99)
]

# Convert the species label to a numeric valua
make_int = lambda label: 1 if label == 'setosa' else -1
Y = [make_int(df['Species'][i]) for i in range(99)]
```

In [84]:
```
 #Set perceptron hypothesis: h(x) = sign(w^T*x)
def h(w, x):
    if w @ x > 0:
        return 1
    else:
        return -1
```

In [85]:
```
def Perceptron(x, y, iterations = 1000):

    w = np.random.rand(3)       #initializing the weights
    n = len(x)

    for _ in range(iterations):
        i = np.random.randint(n)
        if h(w, x[i]) != y[i]:
            w += y[i]*x[i]

    return w
```

In [87]:
```
# Iterate the perceptron learning algorithm 1000 times
w = Perceptron(X, Y, 10000)
```

In [88]:
```
def predict(w, i):
    if h(w, X[i]) == 1:
        return 'Setosa'
    else:
        return 'Versicolor'
```

In [89]:
```
predict(w, 20)
```

Out[89]:
```
'Setosa'
```