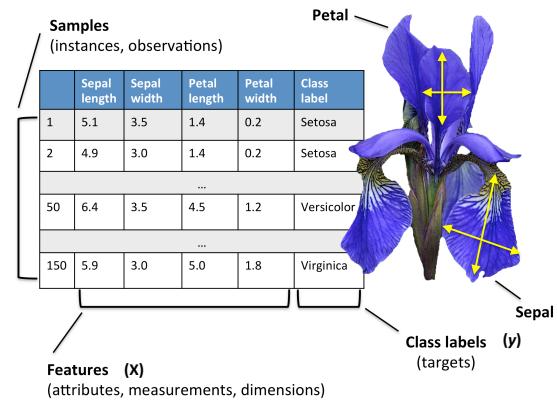


# Basic Terminology



## Perceptron Model and Adaline Model

### Terminology & Notation

A neural network layer with five neurons (A-E) receiving input from four features (Sepal Length, Sepal Width, Petal Length, Petal Width).

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	4.6	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.4	3	1.1	0.1	Iris-setosa
11	4.9	3.1	1.5	0.1	Iris-setosa
12	5.4	3.7	1.5	0.2	Iris-setosa
13	4.8	3.4	1.6	0.2	Iris-setosa
14	4.8	3	1.4	0.1	Iris-setosa
15	4.3	3	1.3	0.1	Iris-setosa
16	5.8	4	1.2	0.2	Iris-setosa
17	5.7	4.4	1.5	0.4	Iris-setosa
18	5.4	3.9	1.3	0.4	Iris-setosa
19	5.1	3.5	1.4	0.3	Iris-setosa
20	5.7	3.8	1.7	0.3	Iris-setosa
21	5.1	3.8	1.5	0.3	Iris-setosa
22	5.4	3.4	1.7	0.2	Iris-setosa
23	5.1	3.7	1.5	0.4	Iris-setosa
24	4.6	3.6	1	0.2	Iris-setosa
25	5.1	3.3	1.7	0.5	Iris-setosa

Feature Table

Feature # 1	Sepal Length
Feature # 2	Sepal Width
Feature # 3	Petal Length
Feature # 4	Petal Width

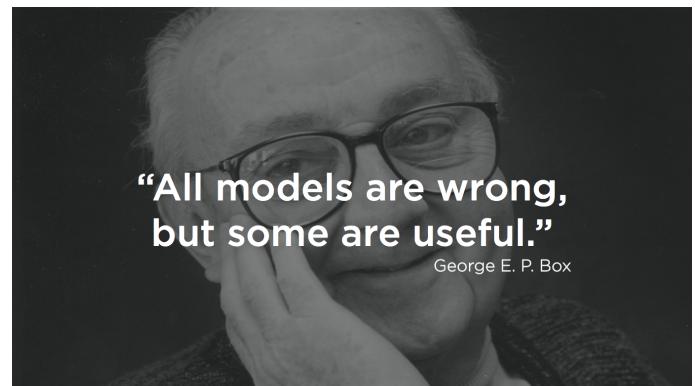
INPUT: The features of the 150 samples can be written as a  $150 \times 4$  matrix.

Target: [Setosa, Versicolor, Virginica]

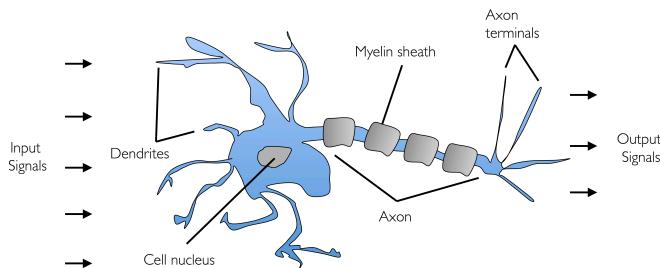
TARGET: The targets/labels of the 150 samples can be written as a 150-dimensional column vector.

**"All models are wrong,  
but some are useful."**

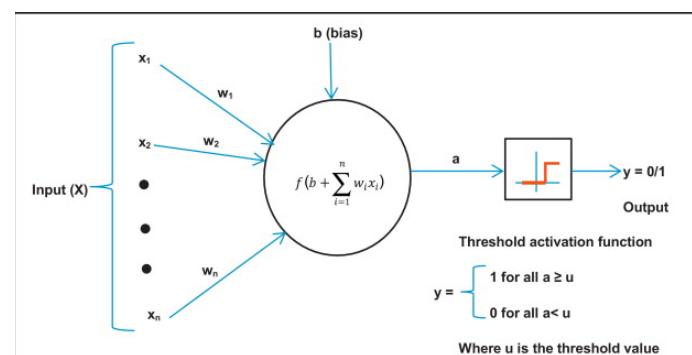
George E. P. Box



### Biology Neuron



### Perceptron Model



x: input w: weight

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

$$Z = w_1 x_1 + \dots + w_m x_m$$

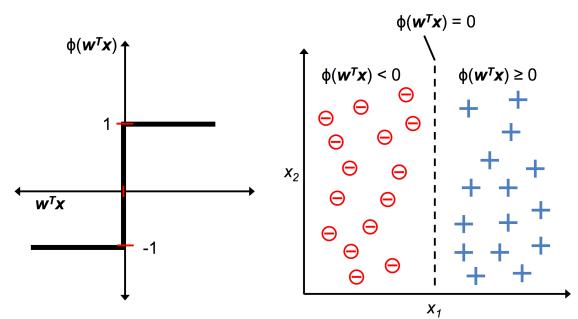
Unit step function

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

$$Z = w_0 x_0 + w_1 x_1 + \dots + w_m x_m = \mathbf{w}^T \mathbf{x}$$

Add a bias unit and the step function becomes

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



## The Perceptron Learning Rule

1. Initialize the weights to 0 or small random numbers.
2. For each training sample  $\mathbf{x}^{(i)}$ :
  - a. Compute the output value  $\hat{y}$ .
  - b. Update the weights.

Here, the output value is the class label predicted by the unit step function that we defined earlier, and the simultaneous update of each weight  $w_j$  in the weight vector  $\mathbf{w}$  can be more formally written as:

$$w_j := w_j + \Delta w_j$$

The value of  $\Delta w_j$ , which is used to update the weight  $w_j$ , is calculated by the perceptron learning rule:

$$\Delta w_j = \eta (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)}$$

## Exercise: Perceptron Learning

- Calculate the weight updates over 1 epoch for given the last two samples in the Iris dataset.

	0	1	2	3	4
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

## Adaline: Adaptive Linear Neuron

- Cost function is used to evaluate how well our current model fits the given sample data.
- In Adaline, we try to minimize a cost function: **Sum of Squared Errors**

$$J(\mathbf{w}) = \frac{1}{2} \sum_i (y^{(i)} - \phi(z^{(i)}))^2$$

Where:

$$\phi(\mathbf{w}^T \mathbf{x}) = \mathbf{w}^T \mathbf{x}$$
 is the identity activation function

## The Adaline Learning Rule

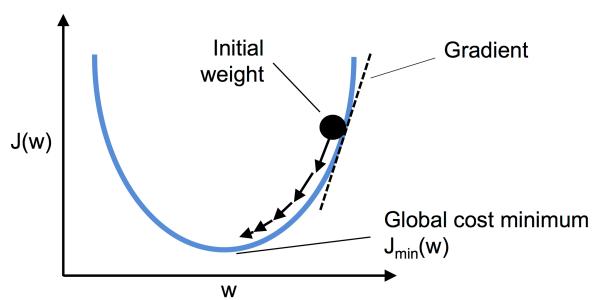
Based on the partial derivative of the cost function with respect to each weight:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i (y^{(i)} - \phi(z^{(i)})) x_j^{(i)}$$

Weights are updated simultaneously so we have:

$$\mathbf{w} := \mathbf{w} + \Delta \mathbf{w}$$

Use Gradient Descent to implement Adaline



## Perceptron V.S. Adaline

- Both are classifiers for binary classification
- Both have a linear decision boundary
- Perceptron uses the class labels (predicted output) to learn model coefficients
- Adaline uses continuous predicted values (from the net input) to learn the model coefficients.
- In their original form, Perceptron updates the weights every sample, and Adaline updates the weights based on the whole sample set.