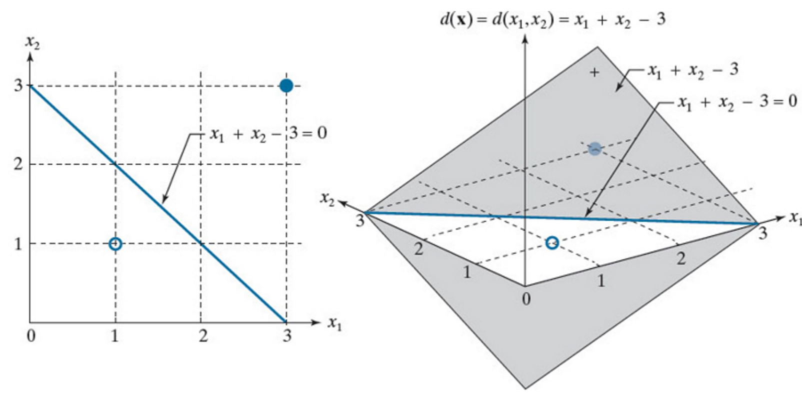
Lecture 24

- Arrangement of Final Exam

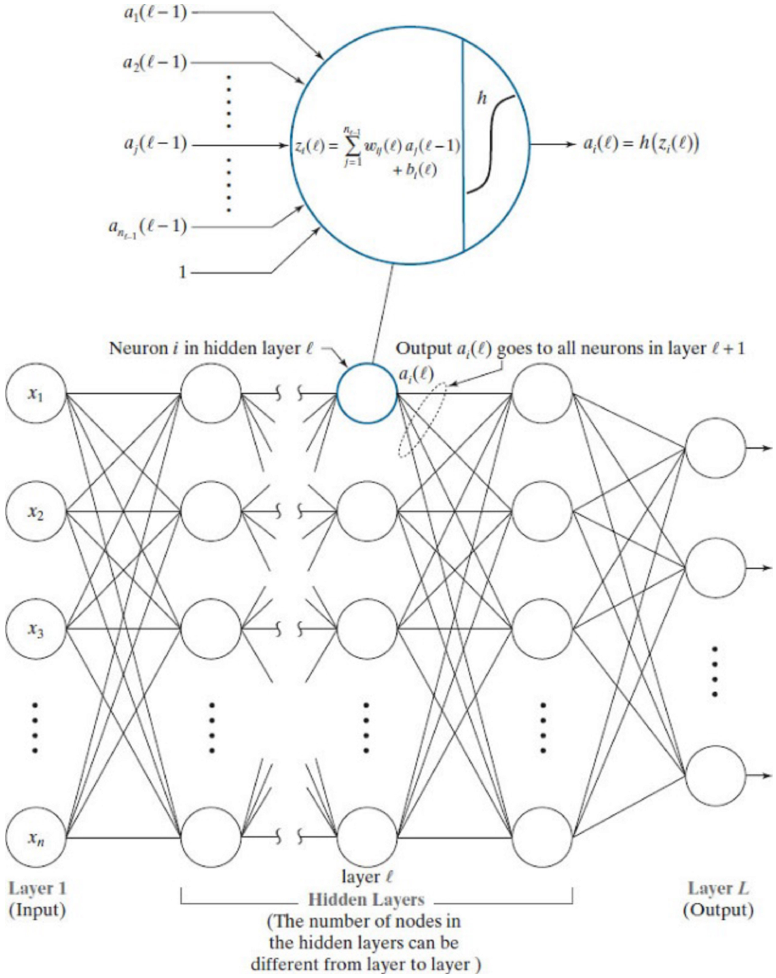
Perceptron in sklearn

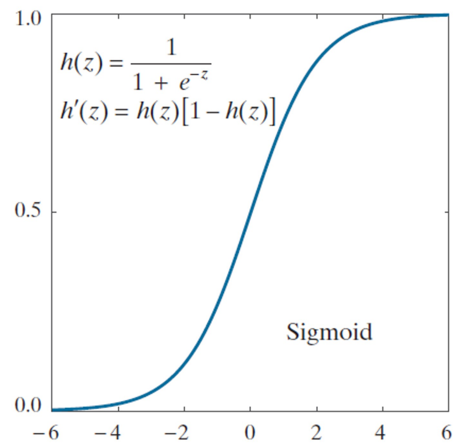
```
"""
'perceptron_demo.py'

"""

import numpy as np
from sklearn.linear_model import Perceptron
x1 = np.array([3, 3])
x2 = np.array([1, 1])
X = np.vstack((x1, x2)) # Features are along the row
y = np.array([1,2])
clf = Perceptron()
clf.fit(X, y)
clf.coef_
clf.intercept_
clf.score(X, y)
```

a b



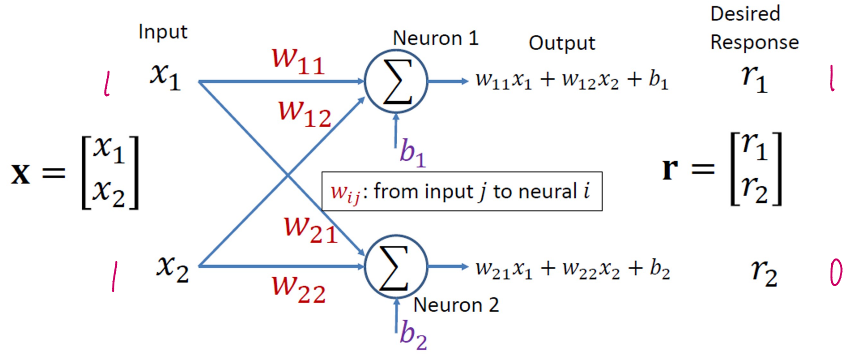

```
Out[15]: Perceptron()

clf.coef_
Out[16]: array([[-1., -1.]])

clf.intercept_
Out[17]: array([3.])

clf.score(X, y)
Out[18]: 1.0
```





- To illustrate the principle of neural network training, we start with a single layer network, without any hidden layer.
- To provide an insight into the backpropagation method, we then investigate a neural network with only one hidden layer, where the activation function for the hidden layer and the output layers is the identity linear function.
- We then look at the same three-layer network, where the activation functions are now changed to sigmoid function, and see how the derivatives of the activation function are integrated into the backpropagation processing flow.
- Next, we use the **Softmax** activation function for the final output layer, and compare the sigmoid function and softmax function in terms of the weights and biases learned.
- We then discuss implementations of training multilayer neural network in Matlab and sklearn.

Single Layer Network without Hidden Layer



$$E(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{2}\|\mathbf{r} - (\boldsymbol{W}\mathbf{x} + \boldsymbol{b})\|^2$$

$$\boxed{\begin{aligned} \boldsymbol{W}(k+1) &= \boldsymbol{W}(k) - \alpha[(\boldsymbol{W}\mathbf{x} + \boldsymbol{b}) - \mathbf{r}]\mathbf{x}^{\mathrm{T}} \\ \boldsymbol{b}(k+1) &= \boldsymbol{b}(k) - \alpha[(\boldsymbol{W}\mathbf{x} + \boldsymbol{b}) - \mathbf{r}] \end{aligned}}$$

X =

$$\begin{matrix} 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{matrix}$$

R =

$$\begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

```
alpha = 0.1;  % learning rate

% Linearly separable example
% Input data pattern
X = [1 -1 -1 1; 1 -1 1 -1];
% Response
R = [1  0  1 0; 0  1 0  1];

% Linearly non-separable example
% Input data pattern
%X = [1 -1 -1 1; 1 -1 1 -1];
% Response
%R = [1  1  0 0; 0 0  1  1];

rng('default');
Std = 0.02;

% Initial weights and biases
W2 = Std*randn(2,2);
b2 = Std*randn(2,1);
```

```
max_iter = 100;
mse = zeros(1, max_iter);

epoch = 0;
while (epoch <= max_iter)
   for i = 1: 4
      epoch = epoch + 1;
      A1 = X(:,i);
      A2 = W2*A1 + b2;

      D2 = A2 - R(:,i);
      mse(epoch) = 0.5*norm(D2)^2;

      % Update the weights and biases
      W2 = W2 - alpha*D2*A1';
      b2 = b2 - alpha*D2;
   end

end
mse(epoch)
plot(mse); grid
```