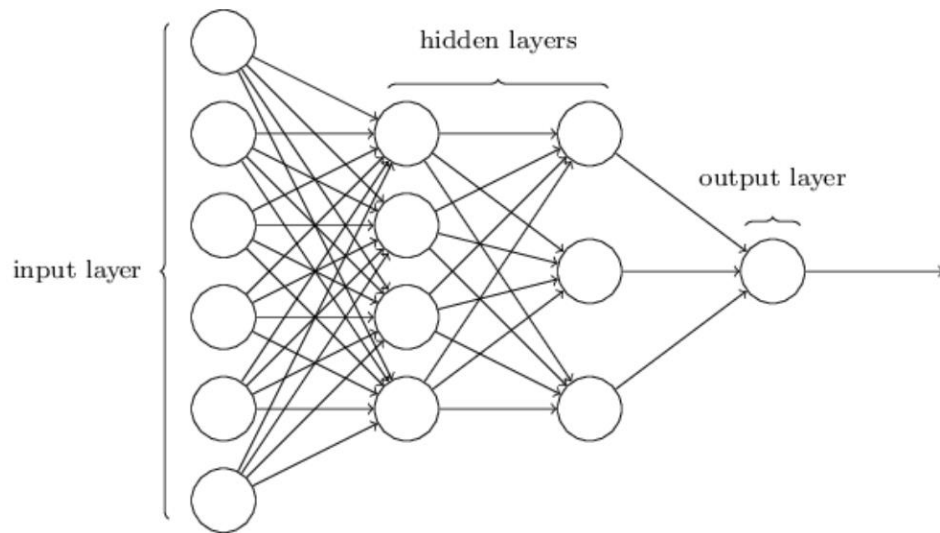


Deep Learning (CS 470, CS 570)

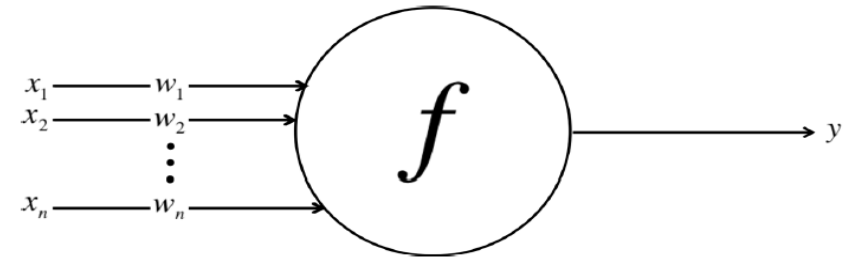
Module 3, Lecture 1: Artificial Neural Network Introduction

What is an Artificial Neural Network (ANN)?

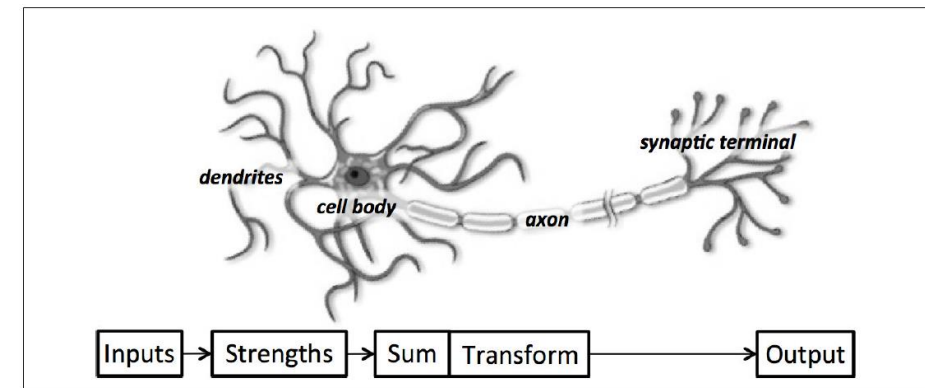
ANNs are a type of machine learning model that are loosely inspired by biological networks.



An ANN is made of nodes and weighted connections. The ANN architecture shown in the image is used for supervised learning; especially classification. The first layer of an ANN is called the input layer where feature vector is passed. Remember we represent each image as a feature vector (or data point) in our image classification example. A series of mathematical operations are performed on the features to produce a final decision in the output layer. The final decision assigns a class to the input vector. In this section, we will learn the details of ANN's operation.

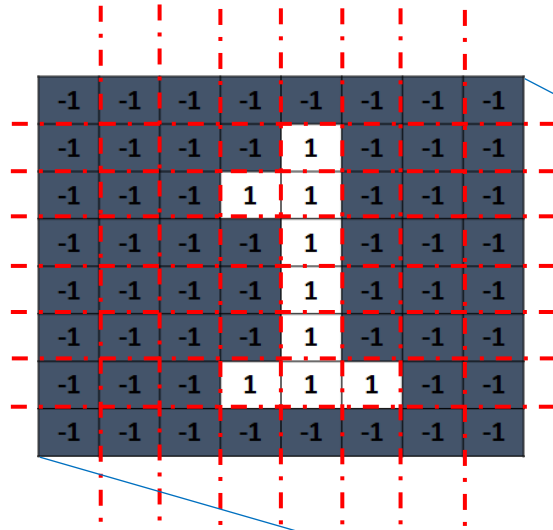


A node or perceptron takes inputs and produces an output

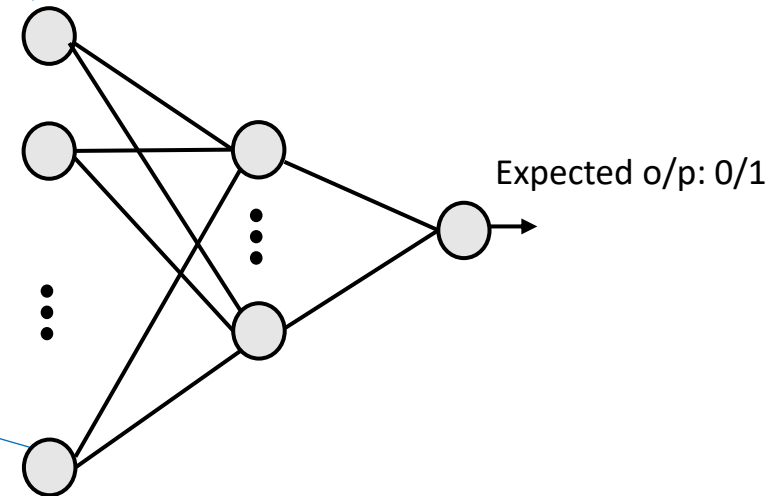
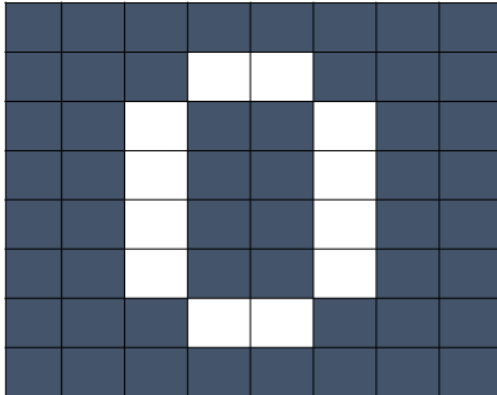


This is similar to a biological neuron that takes inputs through dendrons and produces output by a biochemical process. The output is passed to next sets of neurons through axon.

Example: How ANN Works

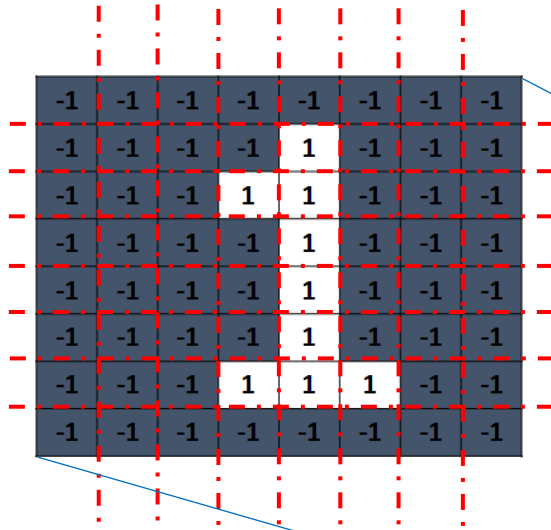


All the pixels are passed to i/p layer

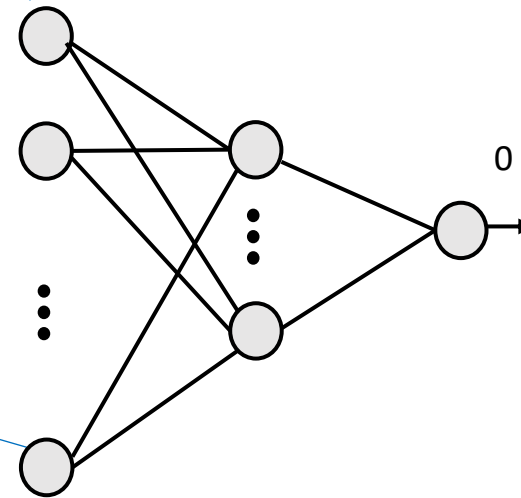
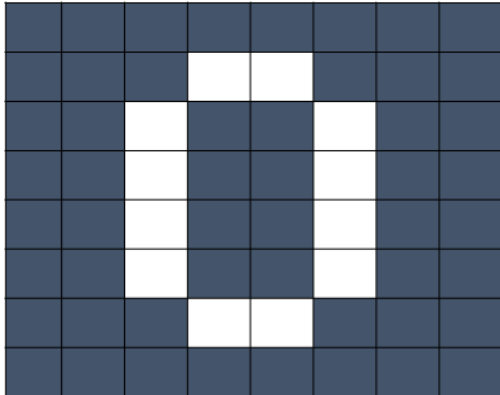


This example illustrates the working philosophy of a simple ANN architecture that aims to classify images of two digits ('0' and '1').

Example: How ANN Works



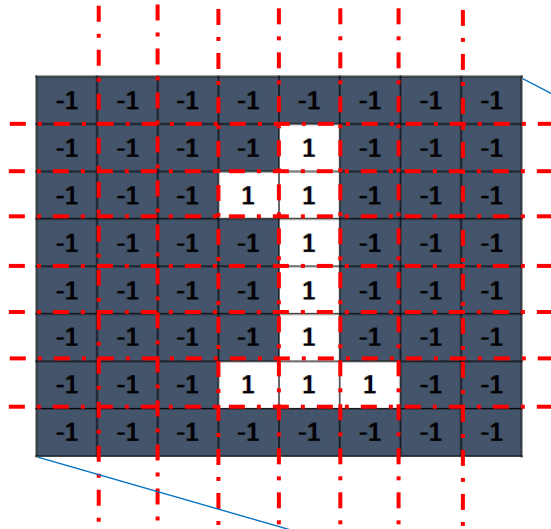
All the pixels are passed to i/p layer



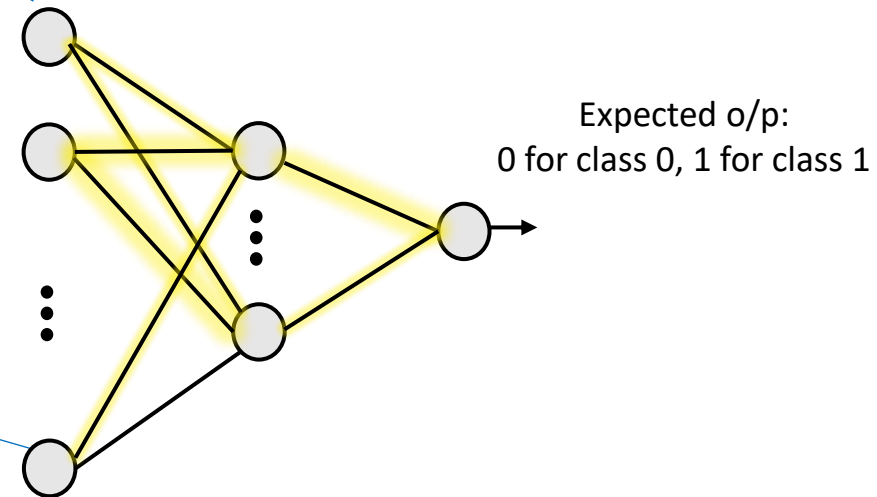
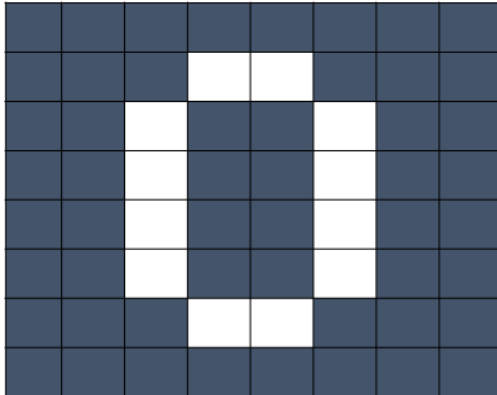
Expected o/p:
0 for class 0, 1 for class 1

Random initialization of connection weights

Example: How ANN Works



All the pixels are passed to i/p layer



Expected o/p:
0 for class 0, 1 for class 1

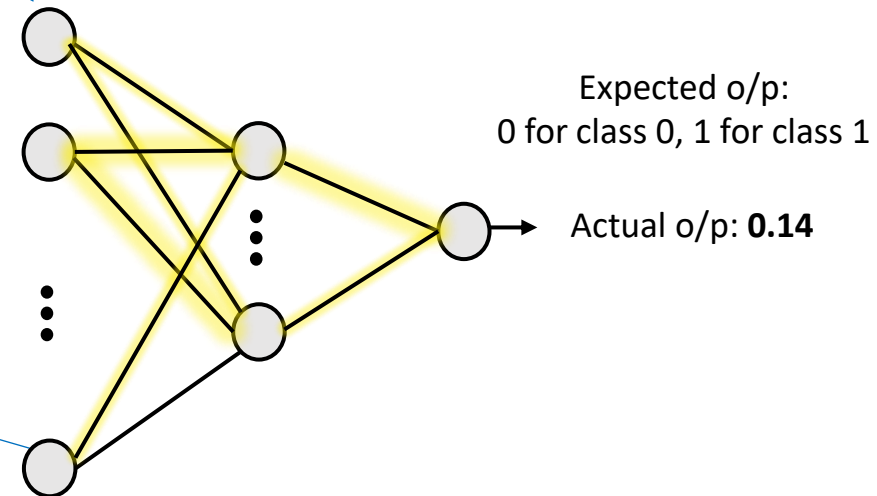
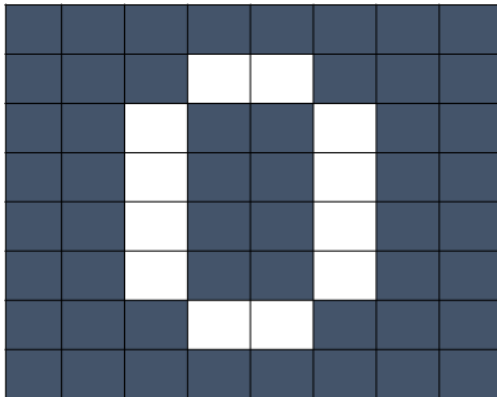
Random initialization of connection weights

Inputs multiplied by connection weights and passed to next layer

Example: How ANN Works

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	1	-1	-1
-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	1	1	-1

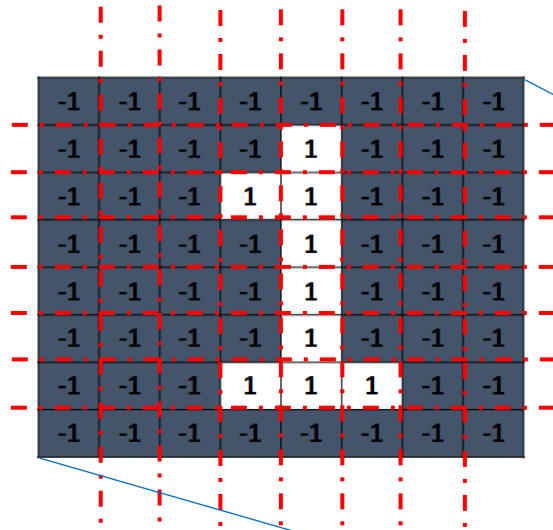
All the pixels are passed to i/p layer



Random initialization of connection weights

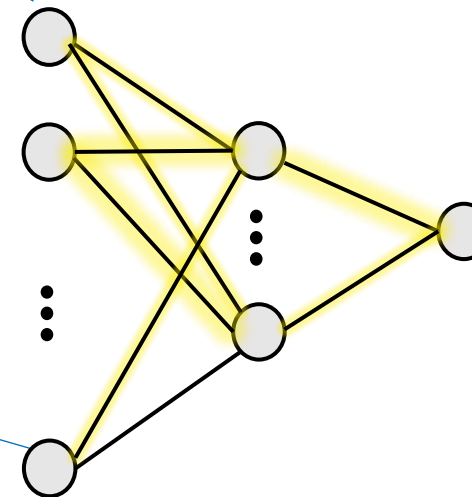
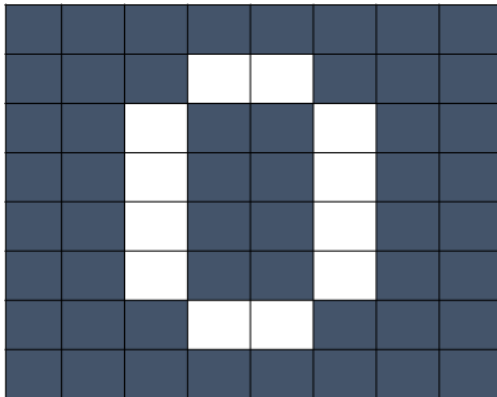
Inputs multiplied by connection weights and passed to next layer

Example: How ANN Works



-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

All the pixels are passed to i/p layer



Expected o/p:
0 for class 0, 1 for class 1

Actual o/p: **0.14**

Error= |expected -actual o/p| = **0.86**

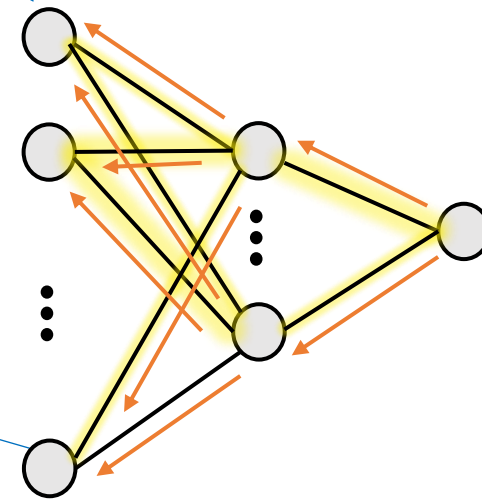
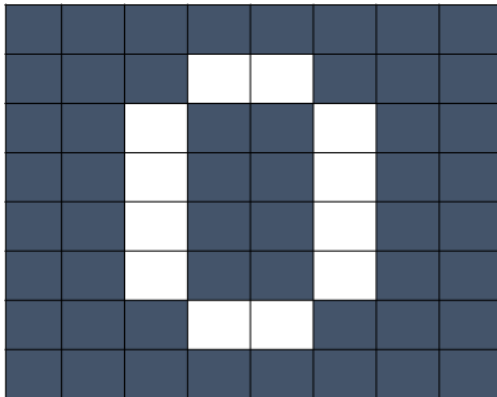
Random initialization of connection weights

Inputs multiplied by connection weights and passed to next layer

Example: How ANN Works

-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

All the pixels are passed to i/p layer



Expected o/p:
0 for class 0, 1 for class 1

Actual o/p: **0.14**

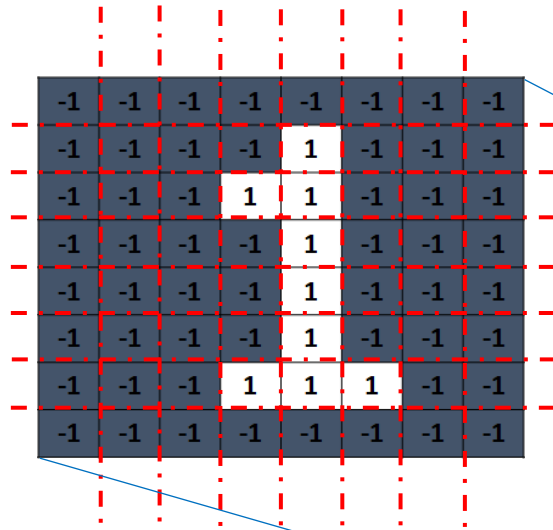
Error= |expected –actual o/p| = **0.86**

Backpropagation: depending on the error update the connection weights

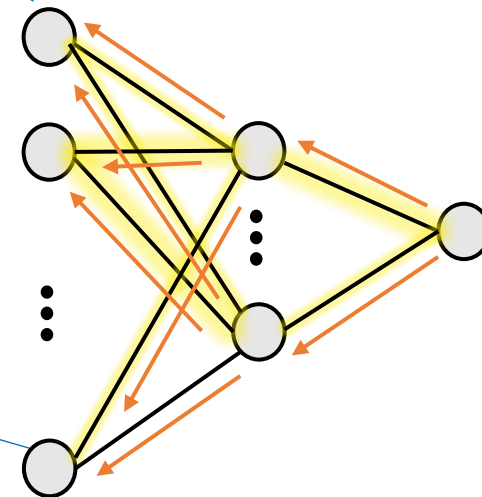
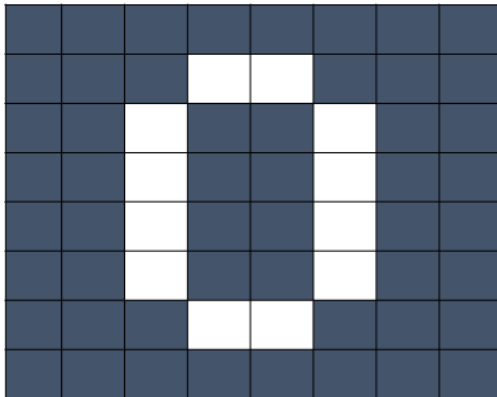
Random initialization of connection weights

Inputs multiplied by connection weights and passed to next layer

Example: How ANN Works



All the pixels are passed to i/p layer



Expected o/p:
0 for class 0, 1 for class 1

Actual o/p: **0.14**

Error= |expected -actual o/p| = **0.86**

Backpropagation: depending on the error update the connection weights

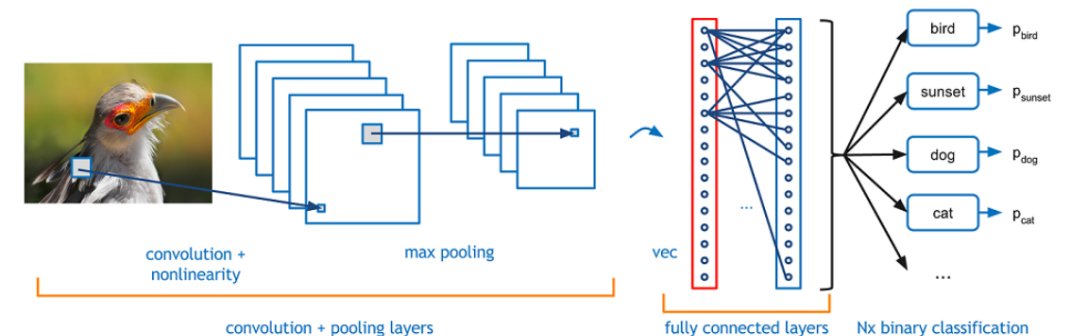
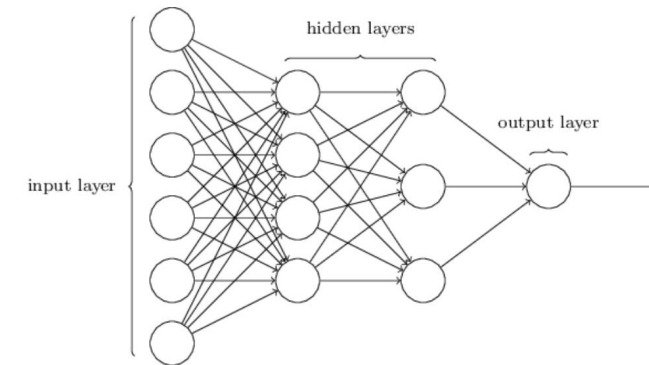
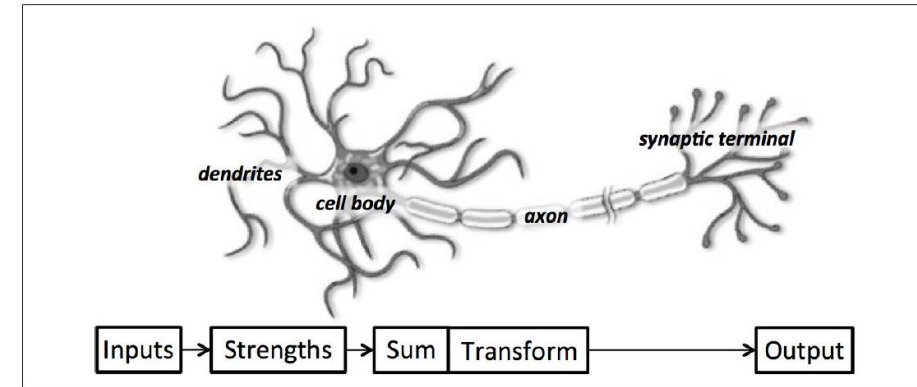
Random initialization of connection weights

Inputs multiplied by connection weights and passed to next layer

Perform the o/p value calculation, error calculation, and backpropagation iteratively for all training samples until the network weights converges.

Artificial Neural Network: a Brief History

- **1943:** Warren McCulloch and mathematician Walter Pitts modeled a neural network with electrical circuits.
- **1949:** Donald Hebb showed how learning happens in brain
- **1950s and 60s:** several attempts made to simulate neural network
- **Late 1970s and 80s:** Revival of neural network happened, but the computers were still not powerful enough
- **1986:** David Rumelhart rediscovered backpropagation
- **1995:** Yann LeCun at Bell lab demonstrated the power of CNN for handwriting recognition



Artificial Neural Network: a Brief History

Recurrent Neural Network (RNN)

- Long Short Term Memory (LSTM): in 1997 introduced by Sepp Hochreiter and Juergen Schmidhuber

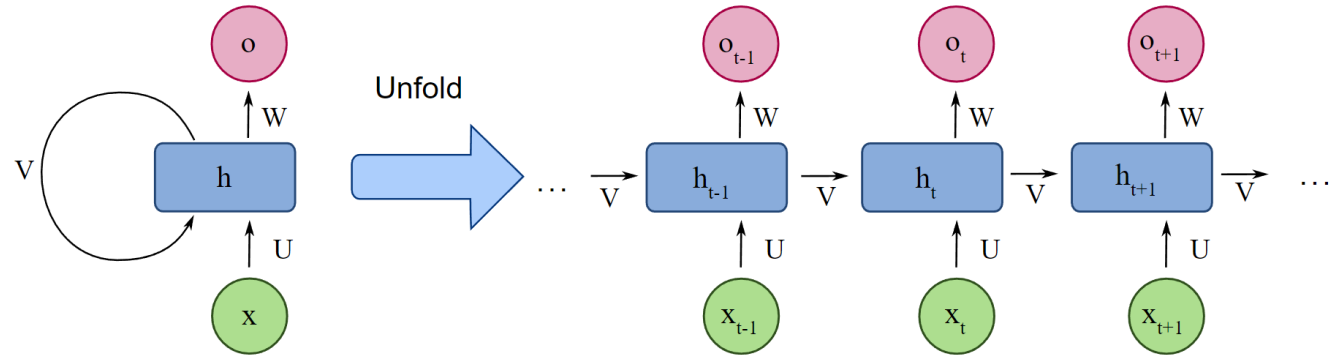
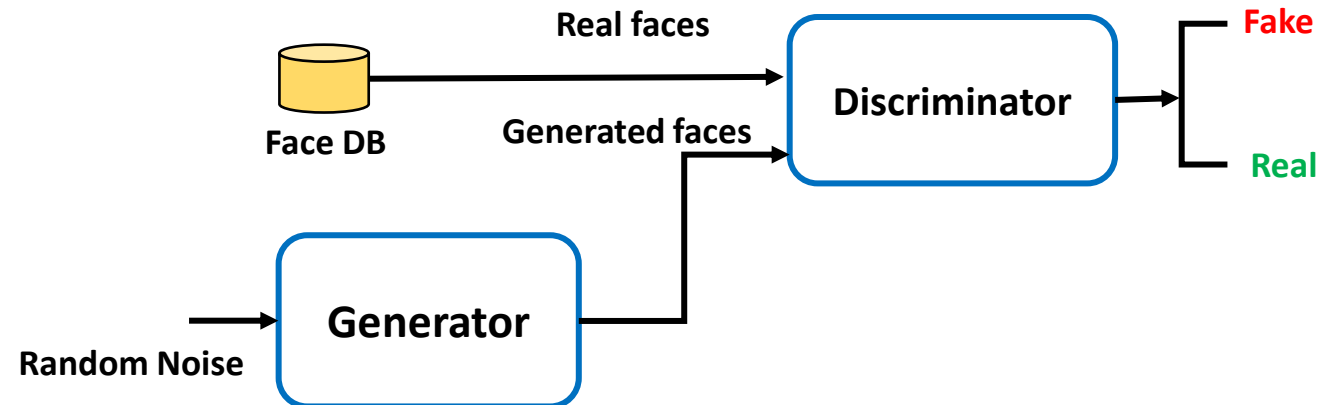


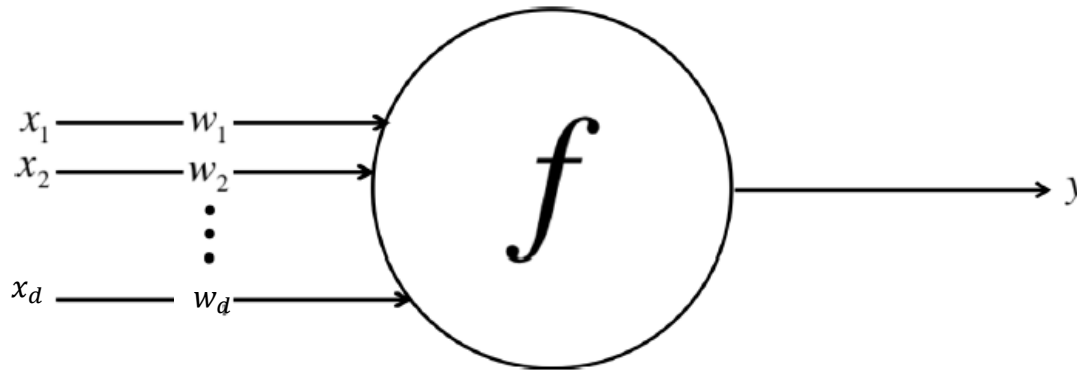
Figure: Wikipedia

Generative Adversarial Networks (GANs)

- Introduced by Ian Goodfellow in 2014



Artificial Neural Network: Perceptron



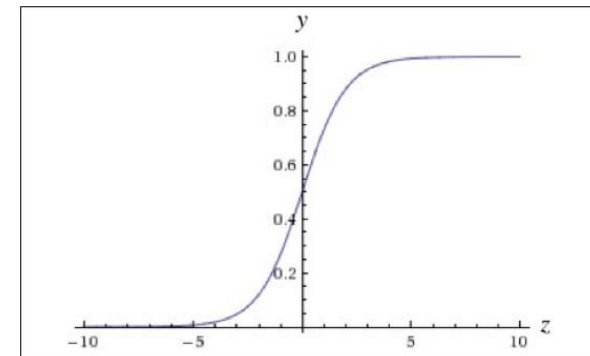
If we take a closer look on a node of an ANN, it looks like the picture above. It has a set of inputs coming through a set of a weighted edges. The input (z) to the node is defined by the 'input' equation. The node process the input with a mathematical function $f()$ and produces the output y . The function $f()$ can be different for different ANN but a popular choice is Sigmoid function. Sigmoid is a nonlinear function defined by the equation in the right. Non-linearity of $f()$ allows the neural network to have non-linear decision boundary.

$$\text{Input: } z = \sum_{i=1}^d w_i x_i + b$$

$$\text{Output: } y = f(z)$$

$f()$ is often a nonlinear function:

$$\text{Sigmoid : } y = f(z) = \frac{1}{1 + e^{-z}}$$



Artificial Neural Network: Perceptron

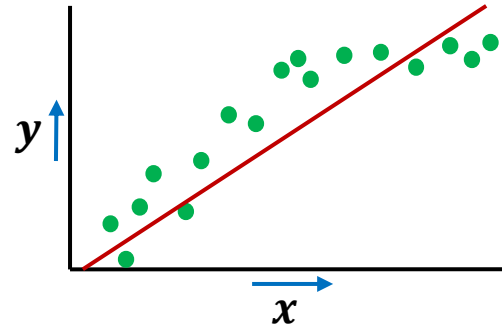
Below are two special cases of perceptron depending on the definition of $f()$.

In case of ① the perceptron behaves like a linear regressor.

In case of ② the perceptron behaves like a linear classifier.

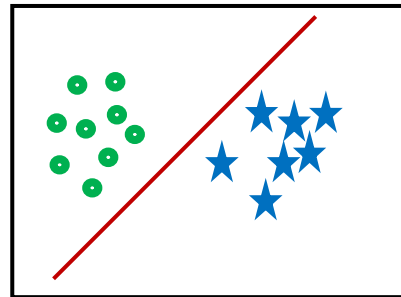
①

when: $y = f(z) = z$



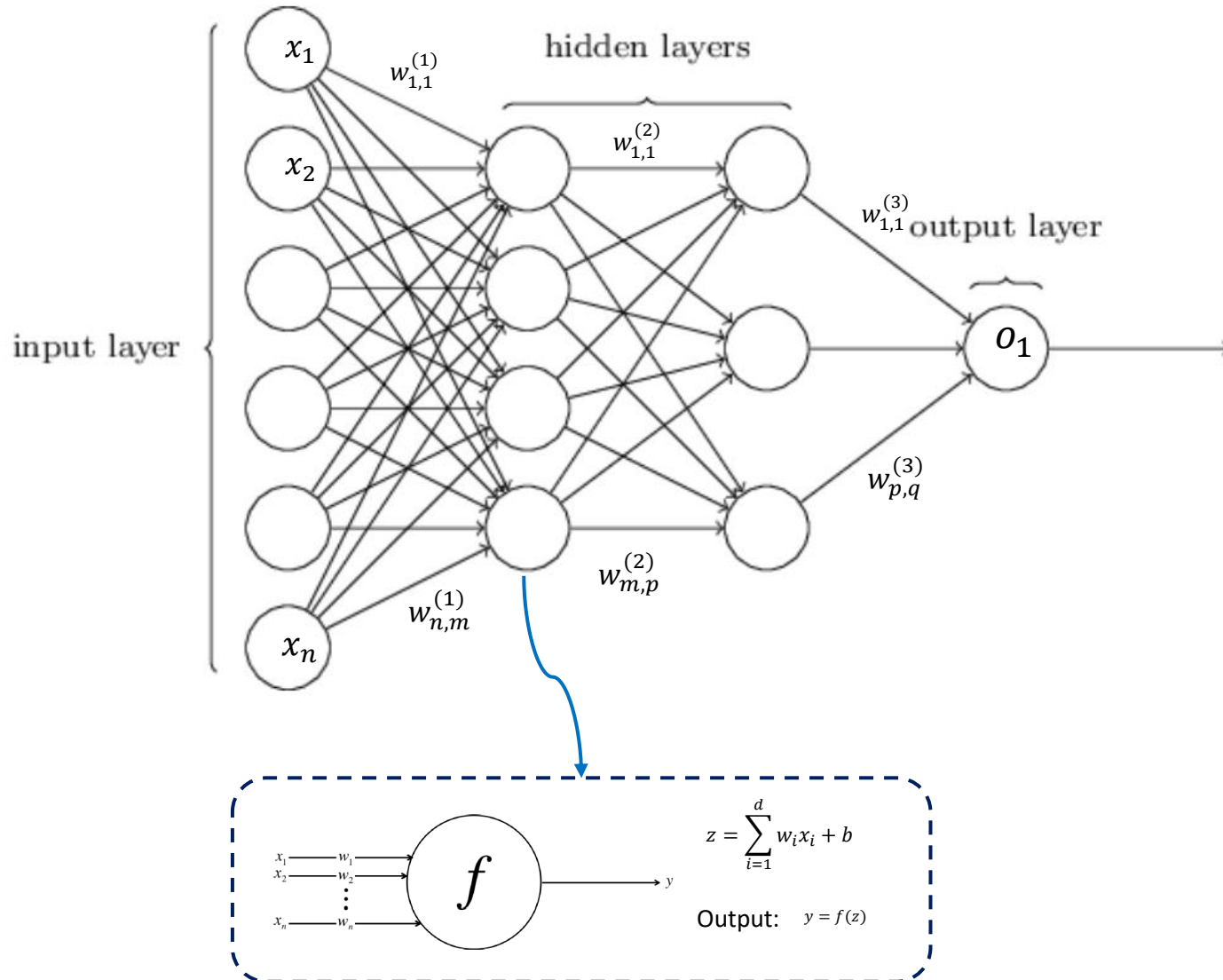
②

when: $y = f(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$



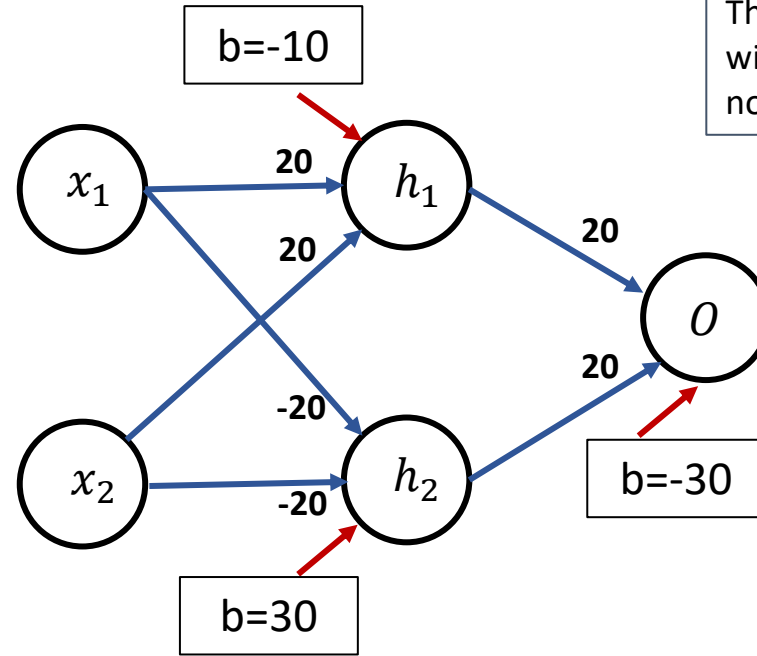
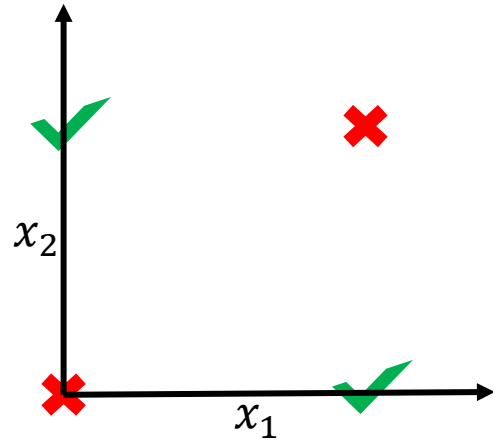
Multilayer Perceptron

Now that we know how a node of an ANN behaves, let us take a look at the whole architecture. This type of networks are called Multi Layer Perceptron (MLP).

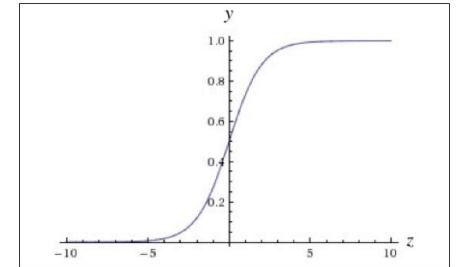


- Each node in previous layer is connected to all nodes of the next layer
- Each connection associated with a weight
- Connection weights are adjusted during learning/optimization
- Hyperparameters: number of nodes in each layer, number of layers
 - Hyperparameters are selected using cross-validation (we will cover it in a future class)

Multilayer Perceptron: XOR Problem



This example proves that an MLP with a non-linear $f()$ can solve a non-linear classification problem.



Sigmoid :
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

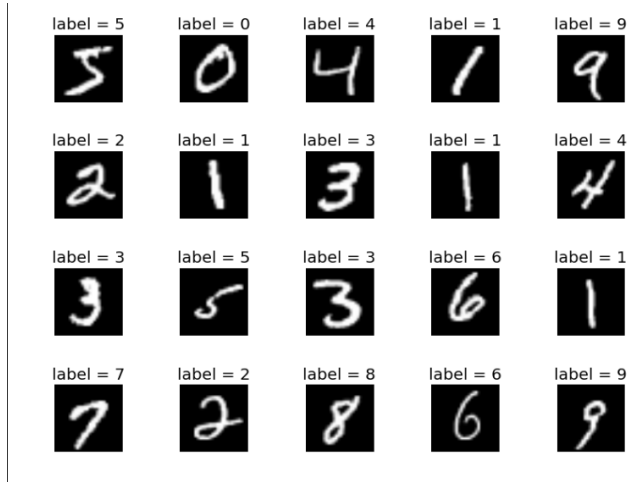
When $x_1 = 0, x_2 = 0$

$$h_1 = \sigma(20 * 0 + 20 * 0 - 10) \approx 0 \quad h_2 = \sigma(-20 * 0 - 20 * 0 + 30) \approx 1 \quad o = \sigma(20 * 0 + 20 * 1 - 30) \approx 0$$

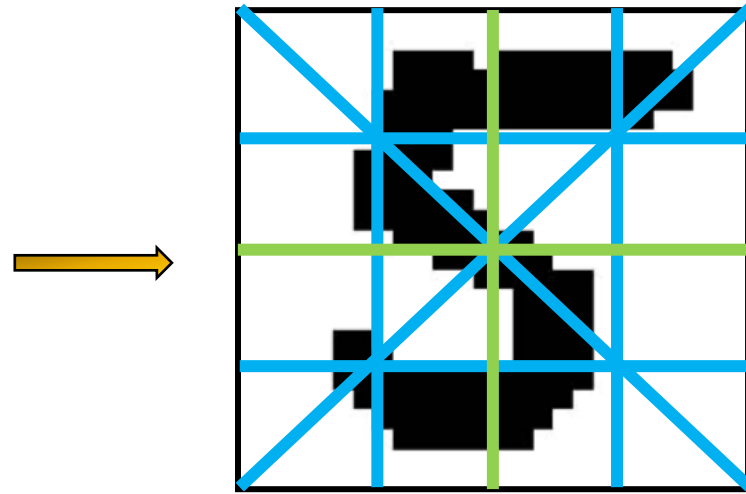
When $x_1 = 1, x_2 = 0$

$$h_1 = \sigma(20 * 1 + 20 * 0 - 10) \approx 1 \quad h_2 = \sigma(-20 * 1 - 20 * 0 + 30) \approx 1 \quad o = \sigma(20 * 1 + 20 * 1 - 30) \approx 1$$

A Practical Example: MLP for Handwriting Recognition



MNIST data set



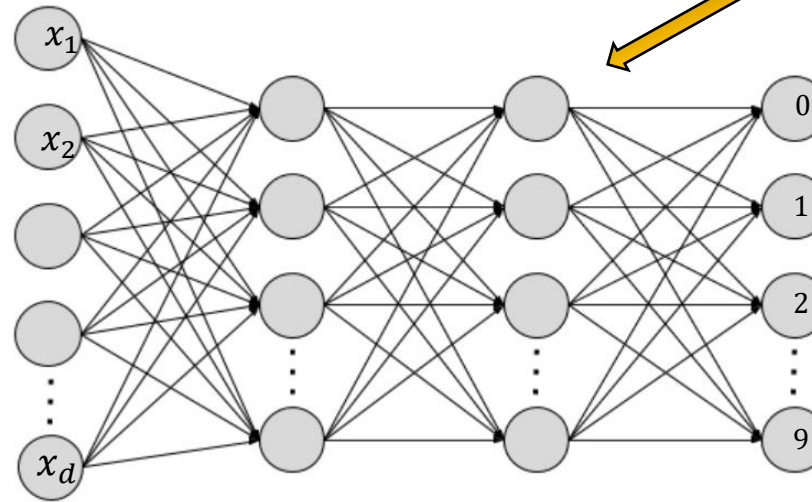
Shadow features

$$\mathbf{x} = \{x_1, x_2, \dots, x_d\}$$

Feature vector

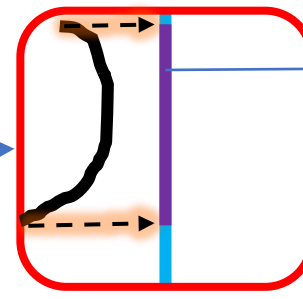
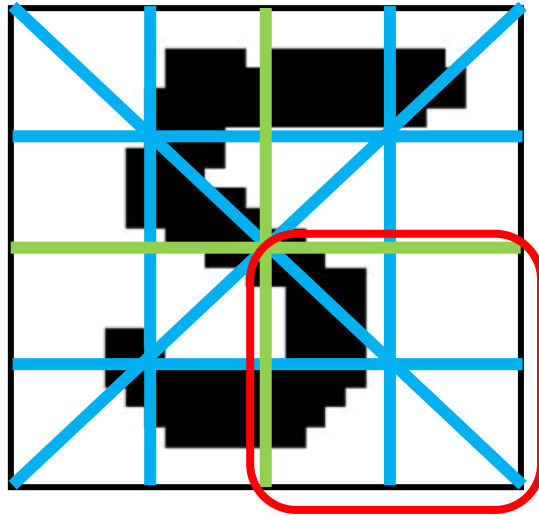
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \vdots \\ \mathbf{x}^N \end{bmatrix} \quad \text{Training set}$$

Now we know how a neural network such as an MLP functions, let us take a look at a real-world hand written digit classification problem. MNIST data set contains many image samples of handwritten digits 0-9. In this example we perform feature extraction for each image to represent it as a feature vector and feed the vector in the input layer of a MLP. The MLP is trained on the training set and validated against test dataset. The training procedure involves backpropagation that is explained in the next lecture.

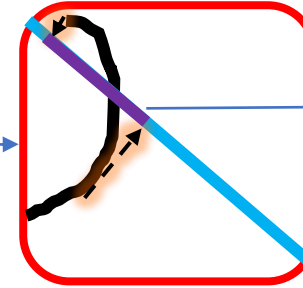


MLP classifier

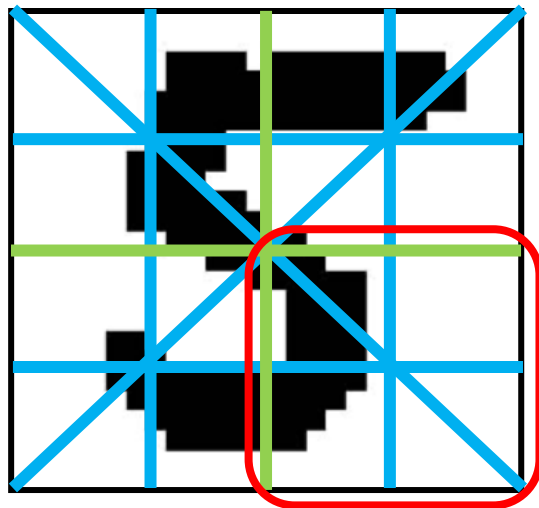
Feature extraction



$length = x_1$



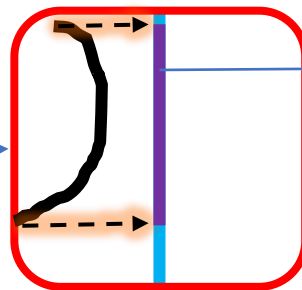
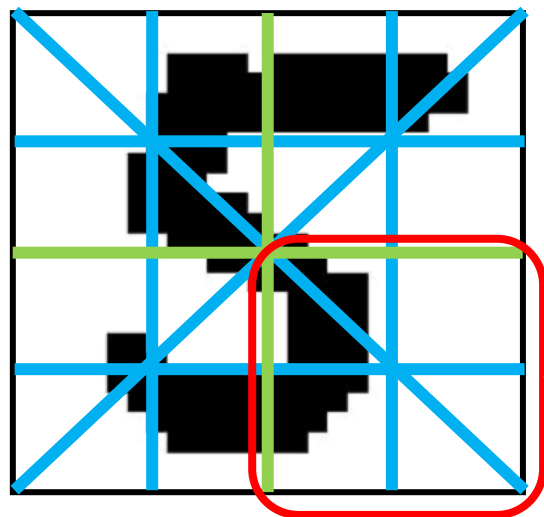
$length = x_2$



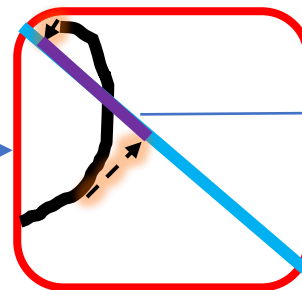
x_1
 x_2
 x_3

Shadow feature: Each image is overlapped with line segments in different direction. Next, a segment of the digit image is projected on each line and the length of the projections is estimated. This step is analogous to casting shadow by the digit segment on the line and measuring the shadow length, and that explains the name of the feature. Now the projection length is estimated for digit segment in all four image quadrants as well as on the whole image. The length of the projections forms a feature vector that represents some characteristics of the digit.

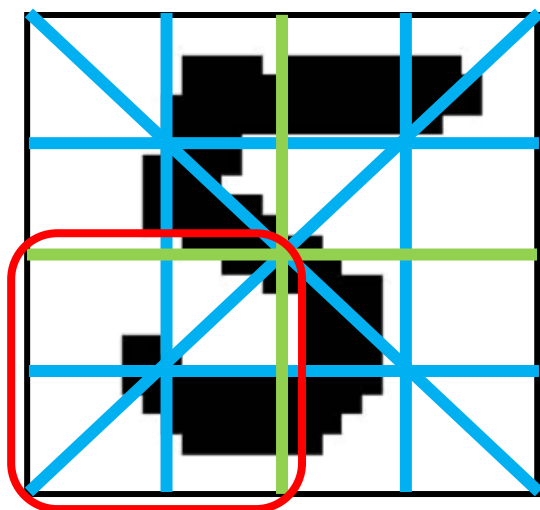
Feature extraction



$length = x_1$

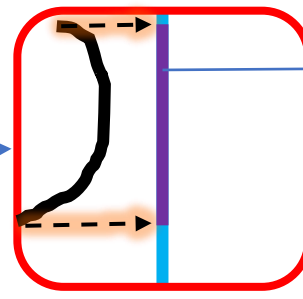
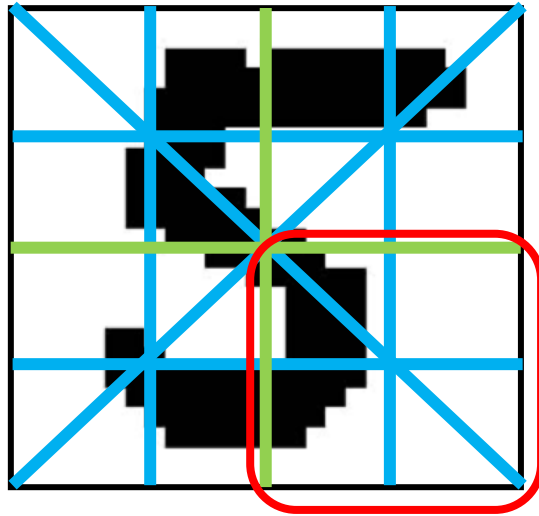


$length = x_2$

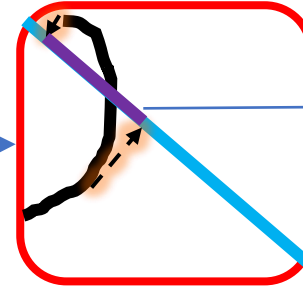


x_1
 x_2
 x_3
 x_4
 x_5
 x_6

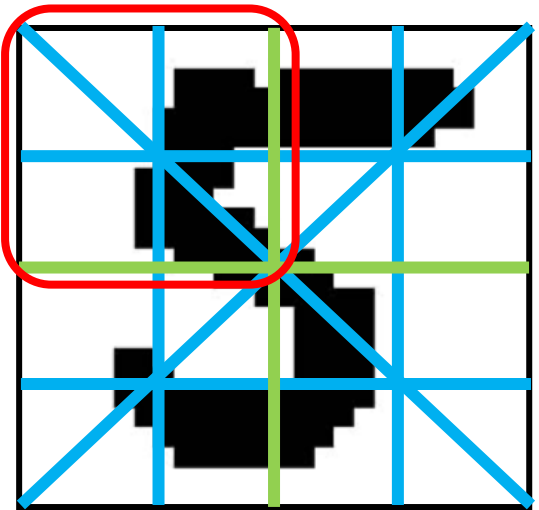
Feature extraction



$length = x_1$

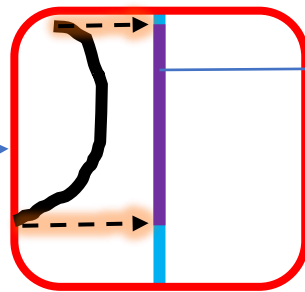
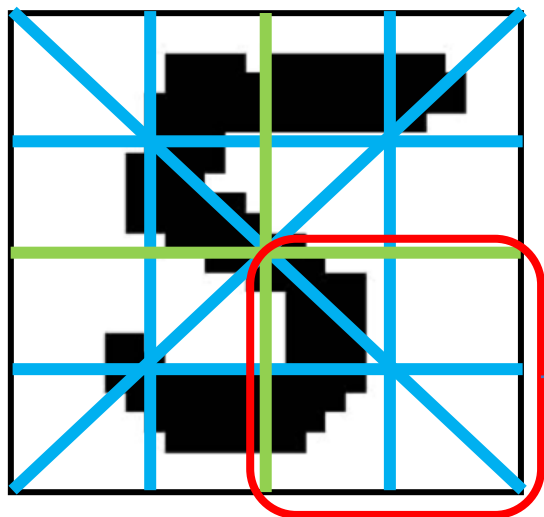


$length = x_2$

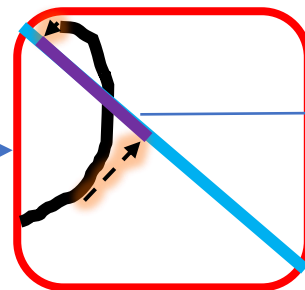


x_1
 x_2
 x_3
 x_4
 x_5
 x_6
 x_7
 x_8
 x_9

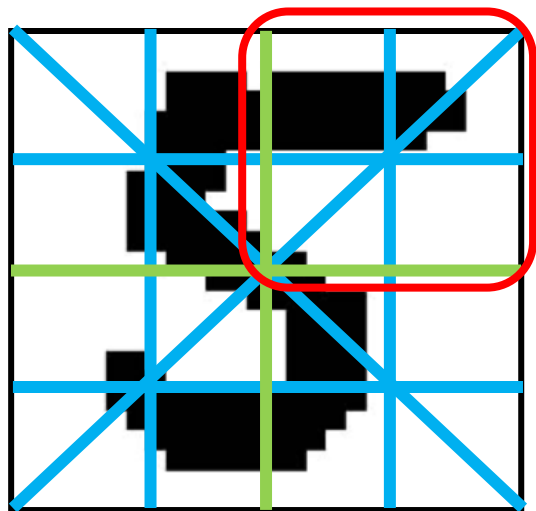
Feature extraction



$length = x_1$

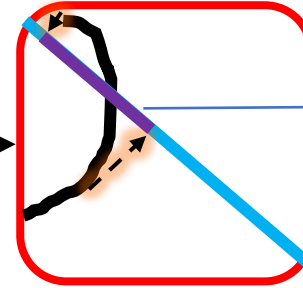
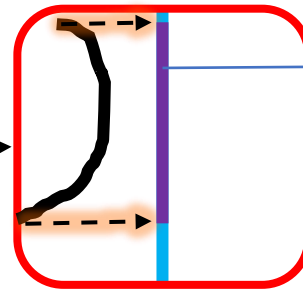
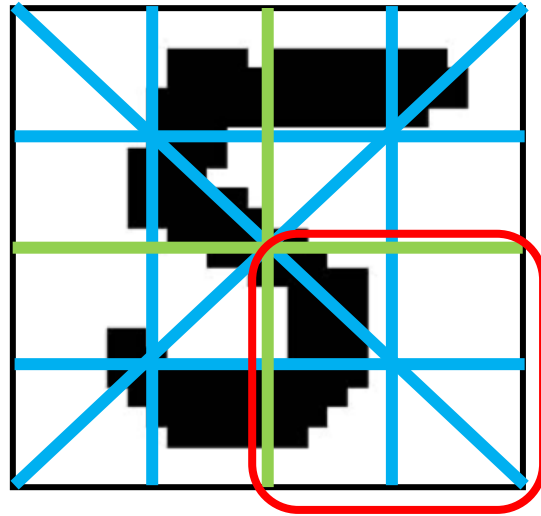


$length = x_2$

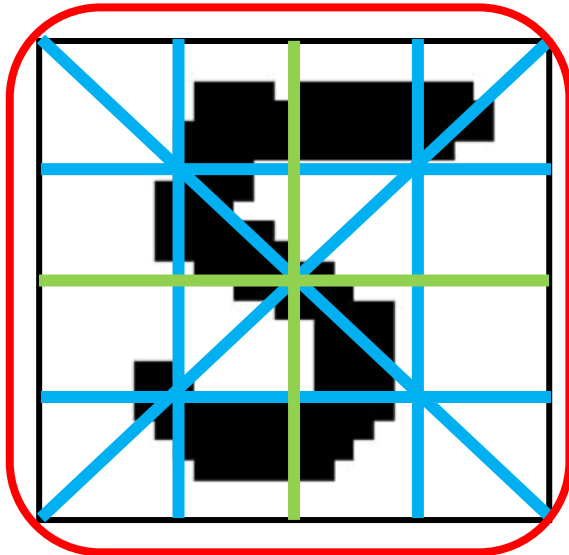


x_1
 x_2
 x_3
 x_4
 x_5
 x_6
 x_7
 x_8
 x_9
 x_{10}
 x_{11}
 x_{12}

Feature extraction

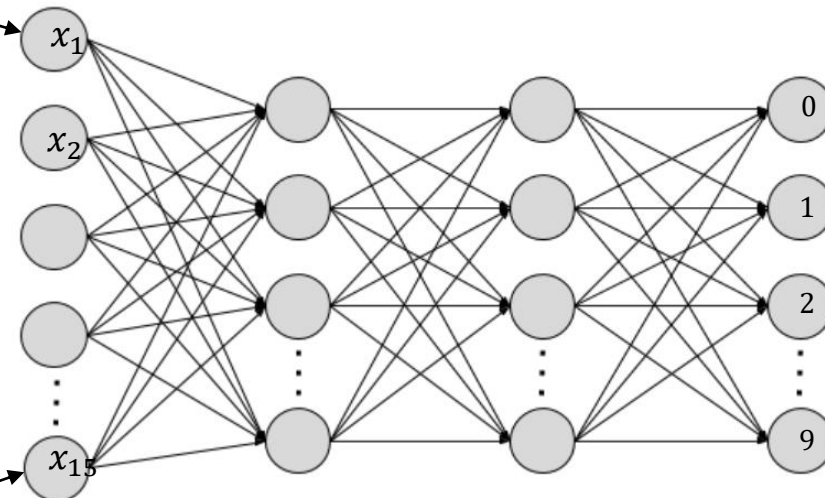


This completes the formation of the feature vector for a sample image. The steps are repeated for each image to form the corresponding feature vector. The next lecture will explain how the MLP training is performed.



x_1
 x_2
 x_3
 x_4
 x_5
 x_6
 x_7
 x_8
 x_9
 x_{10}
 x_{11}
 x_{12}
 x_{13}
 x_{14}
 x_{15}

Classification



Additional Readings

Multilayer perceptron:

[Article 1](#)

[Article 2](#)