INTRODUCTION TO NEURAL NETWORKS

Lecture 5. Feedforward Networks and Forward Propagation

Dr. Jingxin Liu

School of AI and Advanced Computing

Xi'an Jiaotong-Liverpool University
西交利物浦大學

Table of Contents

Feed-forward Neural Networks

A feed-forward neural network (FNN) is an ANN wherein connections between the nodes do NOT form a cycle.

No feedback connections (Recurrent Neural Networks)

A feed-forward neural network defines a **mapping** from **input $\boldsymbol{x}$** to **output $\boldsymbol{y}$** as:

$$y = f(x; \boldsymbol{W})$$

A ***N* layer FNN** is represented as a function composition:

$$f(x; \boldsymbol{W}) = f^N\left(f^{N-1}\left(\dots f^2\left(f^1(x)\right)\right)\right)$$

where f^1 is called the first layer, f^2 is the second layer, etc.

Feed-forward Neural Networks

- x is called the **input layer**.
- The final layer f^N is called the **output layer.**
- The layers in between, f^1 … f^{N-1} or h^1 … h^{N-1} are called **hidden layers**.

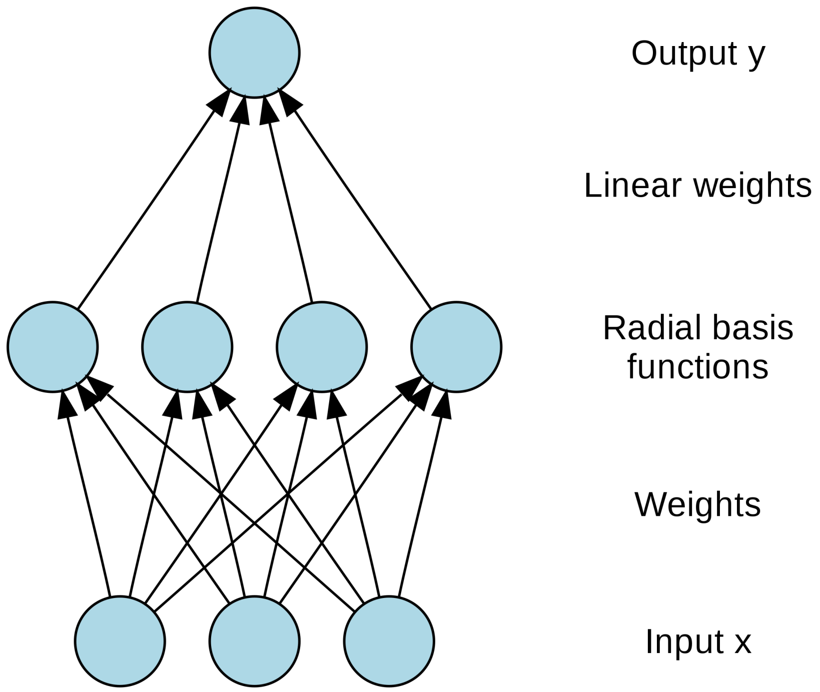
Feed-forward neural networks:

- ❖ Single-layer perceptron
- ❖ **Radial basis function network**
- ❖ **Multi-layer perceptron**
- ❖ Convolutional neural network

Sometimes *multi-layer perceptron* is used loosely to refer to any feed-forward neural network.

Radial Basis Function Network

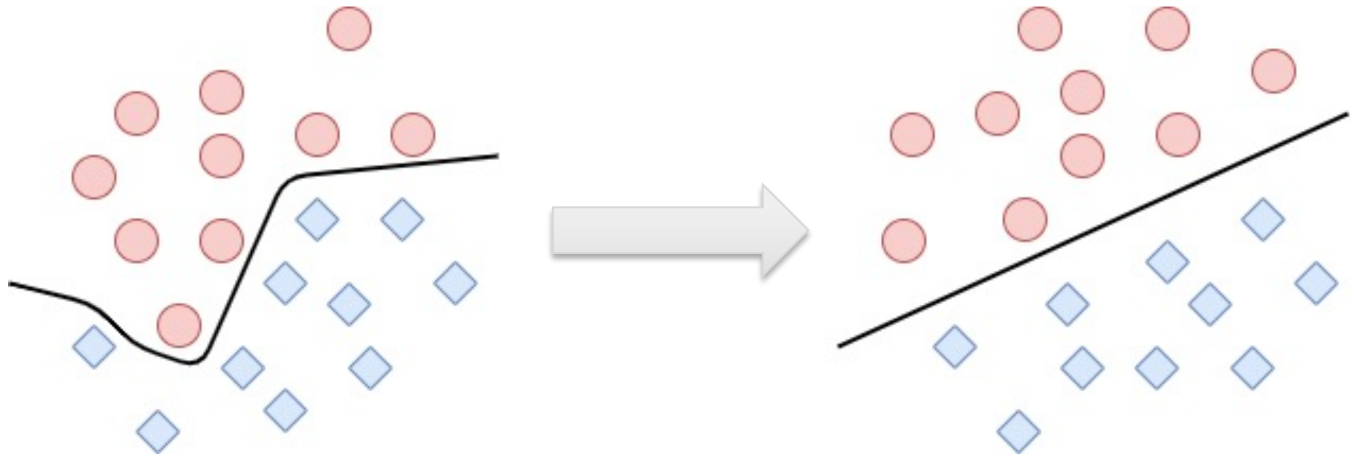
The **radial basis function network** is a special class of multi-layer feed-forward network, which has three layers: an input layer, a hidden layer uses a non-linear radial basis function as activation functions, and a linear output layer.



Radial Basis Function Network

Cover's Theorem:

Cover's Theorem on separability of patterns which states that a complex pattern-classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low dimension space.

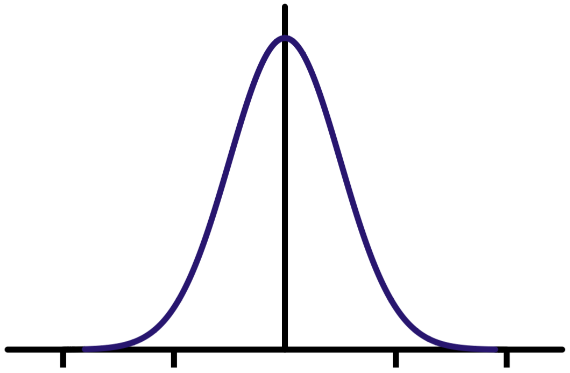


Radial Basis Function Network – RBF

Radial basis functions (RBF) are a special class of function. Their characteristic feature is that their response decreases (or increases) monotonically with distance from a central point. The centre, the spread, and the precise shape of the radial function are parameters of the model, all fixed if it is linear.

$$\varphi(x) = G(\|x - c\|)$$

where G is a decreasing function and c is the center.



Example: Gaussian function

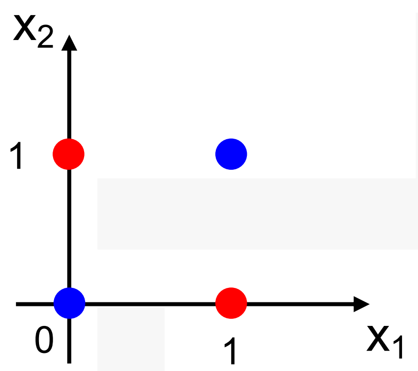
$$G(r) = \exp(-r^2/2\sigma^2)$$

Where $r = \|x - c\|$ and σ is the spread.

Sums of radial basis functions are typically used to approximate given functions. RBFs are also used as a kernel in support vector classification.

Radial Basis Function Network – XOR with RBFN

How to choose parameters to realize XOR with RBFN?



x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

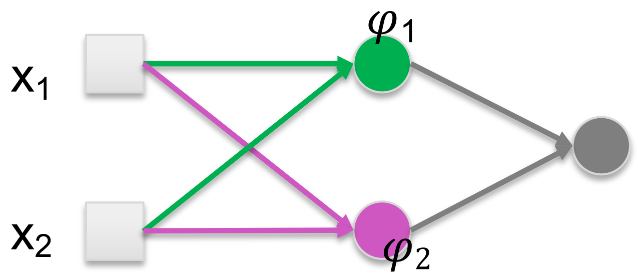
- Choose Guassian function as the hidden layer activation function:

$$G(x) = \exp(-\|x-c\|^2/2\sigma^2)$$

- Use 2 hidden units, choose centers at $c_1 = (1,0)$ and $c_2 = (0,1)$, let $2\sigma^2 = 1$.

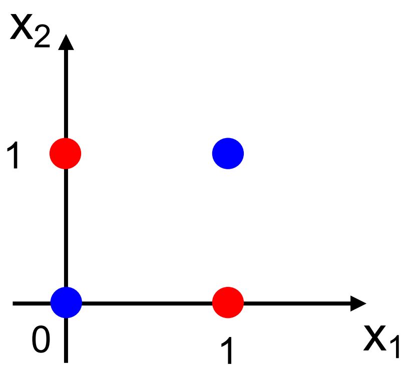
- So,

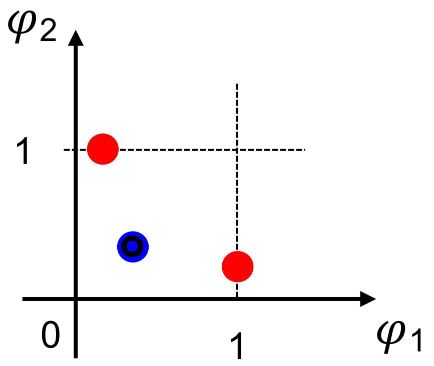
$$\varphi_1(x) = e^{-\|x-c_1\|^2}$$
$$\varphi_2(x) = e^{-\|x-c_2\|^2}$$



Radial Basis Function Network – XOR with RBFN

x_1	x_2	$\varphi_1(x)$	$\varphi_2(x)$
0	0	0.3678	0.3678
0	1	1	0.1353
1	0	0.1353	1
1	1	0.3678	0.3678



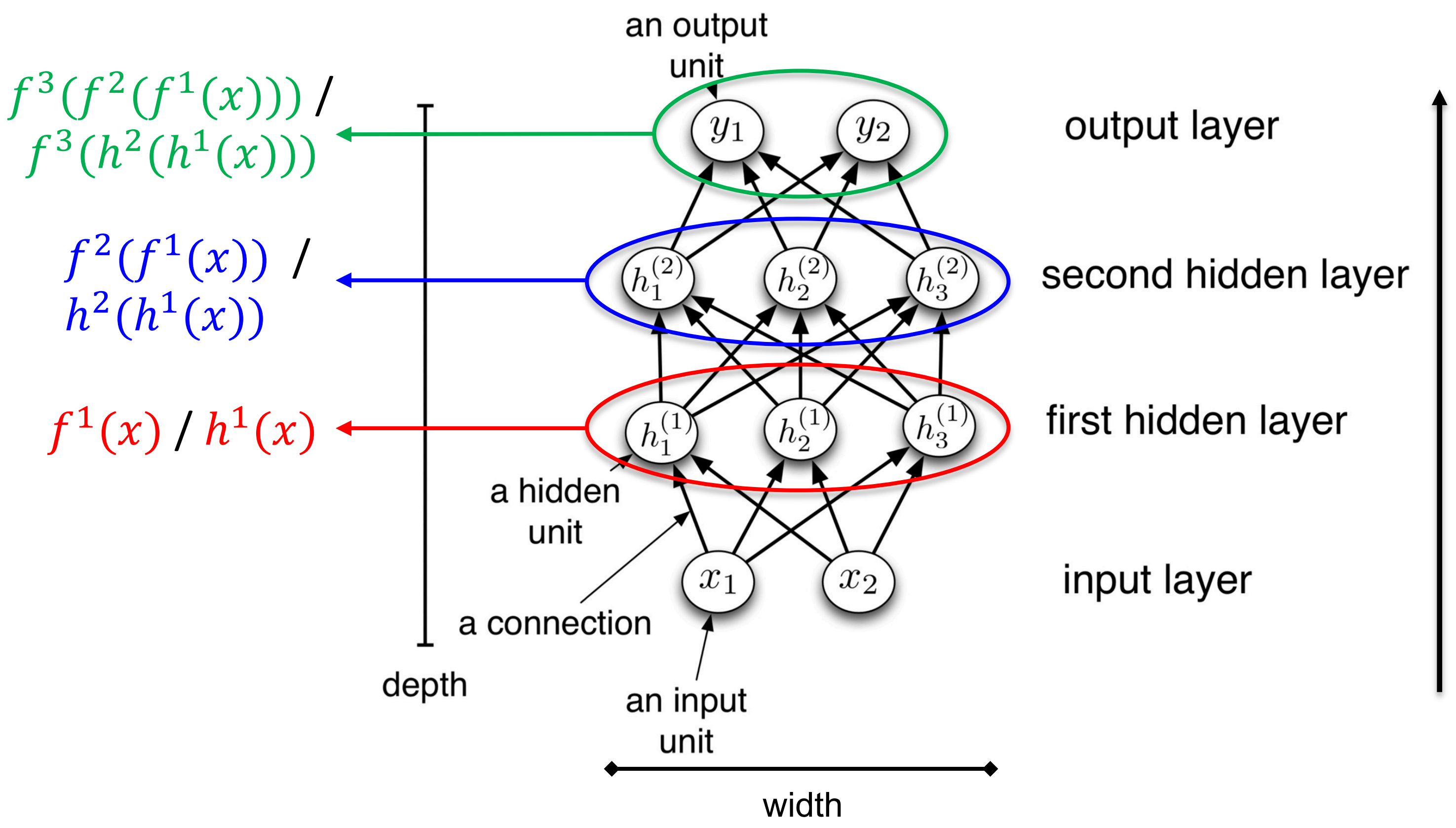


Radial Basis Function Network – Problems

- **Local network**
  Only inputs near a receptive field produce an activation

- **Not computationally efficient**
  The network requires one hidden unit (i.e. one basis function) for each training data pattern, and so for large data sets the network will become very costly to evaluate.

Multi-layer Perceptron

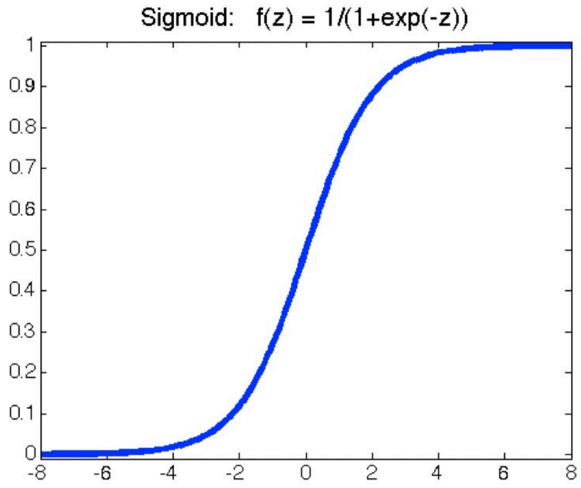


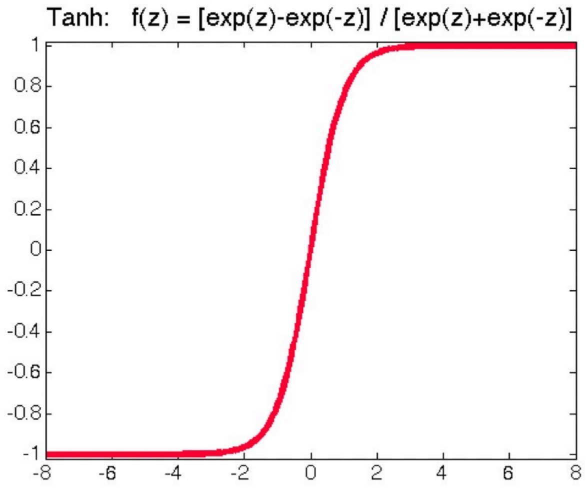
- The length of the chain of functions in neural network is called its **depth**.
- The dimensionality of the hidden layers of a neural network is called its **width**.

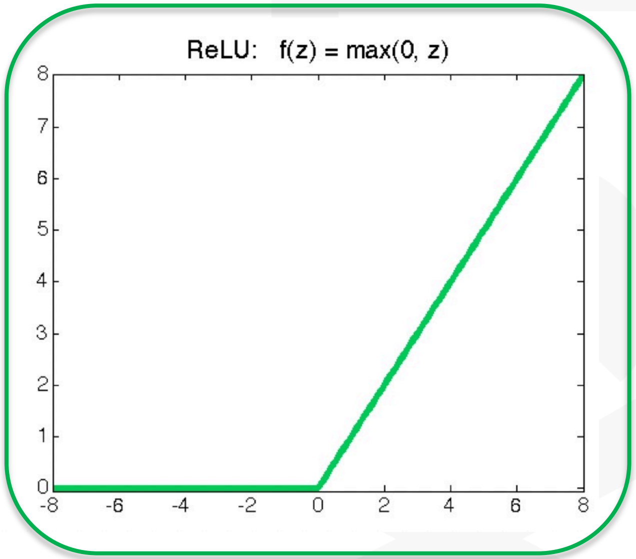
Multi-layer Perceptron – ReLU

Most commonly used activation functions in hidden layers

- Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$
- Tanh: $\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$
- ReLU (Rectified Linear Unit): $ReLU(z) = \max(0, z)$

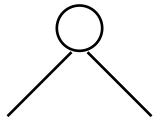
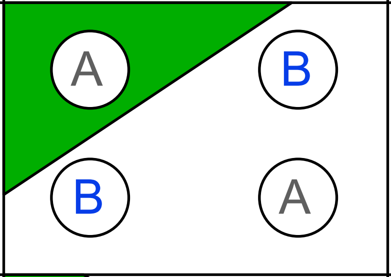
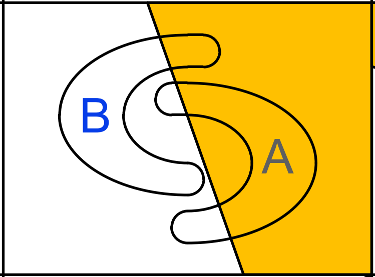
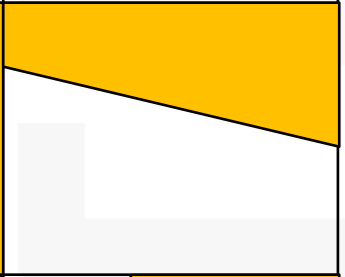
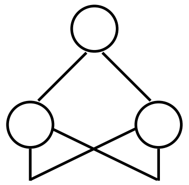
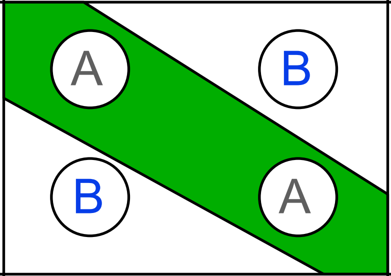
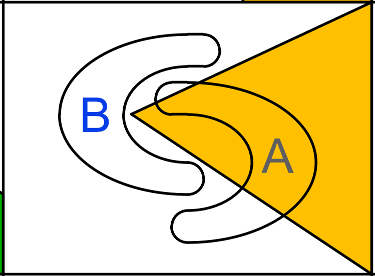
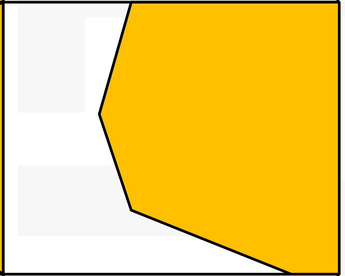
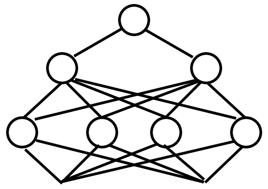
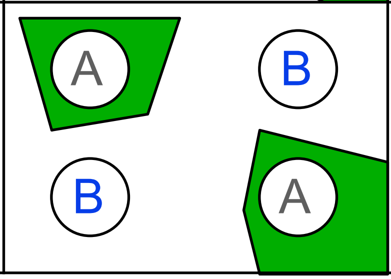
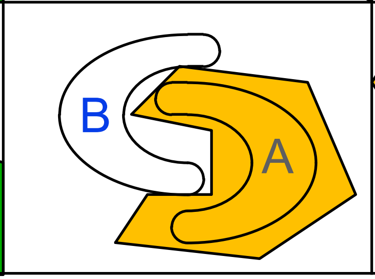
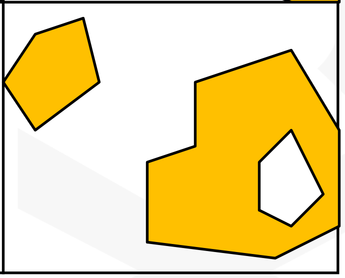






https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

Multi-layer Perceptron – Why MLP?

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
	Half Plane Bounded By Hyperplane			
	Convex Open Or Closed Regions			
	Arbitrary (Complexity Limited by No. of Nodes)			

Multi-layer Perceptron

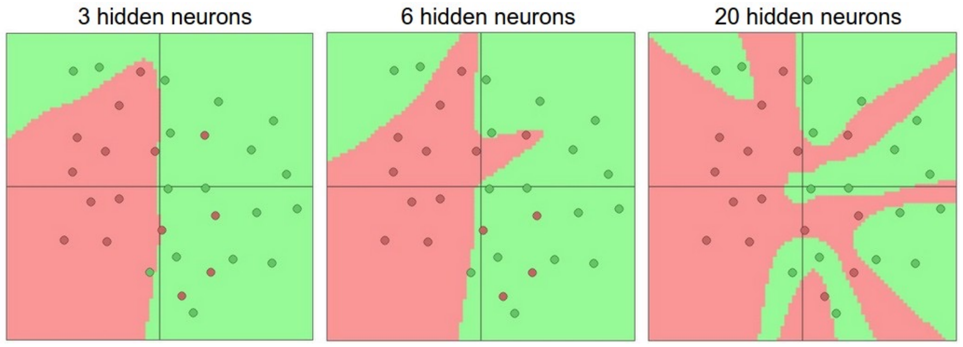
- Neural Networks with at least one hidden layer are *universal approximators* *(can represent any function)*.

  Proof in: *Approximation by Superpositions of Sigmoidal Function, Cybenko, 1989*

  This has been shown for various activation functions (thresholds, logistic, ReLU, etc.)

- The capacity of the network increases with more hidden units and more hidden layers.

  Read e. g .,: *Do Deep Nets Really Need to be Deep?* Jimmy Ba, Rich Caruana, 2013



Multi-layer Perceptron – Mode of Action

Goal: Approximate some unknown ideal function $f^*: \mathcal{X} \rightarrow \mathcal{Y}$

MLP/Feed-Forward Net: Define parametric mapping $y = f(x; \boldsymbol{W})$

Training: Opimize w to drive $f(x; \boldsymbol{W})$ closer to f^*

Training Data: Pairs of x and y

Only **output** is specified by training data! Network is free to do anything with its **hidden layers**. The network itself decide how to modify these layers to best implement an approximation.

This is referred to as **representation learning**.

Forward Propagation – How does the network compute?

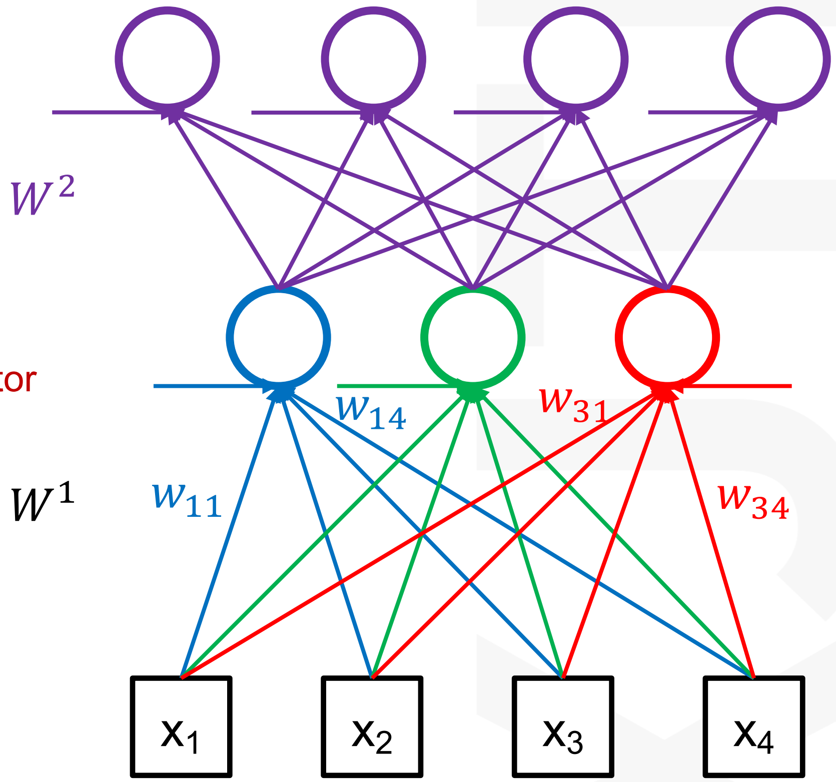
First Hidden Layer:

for the k'th neuron: $z_k = \sum_{i=1} x_i w_{ki}^1 + w_{k0}^1 = \sum_{i=0} x_i w_{ki}^1 = {W_k^1}^T \mathrm{x}$

$$h_k^1 = g_k^1 \left({W_k^1}^T \mathrm{x}\right)$$

for the whole layer: $h^1 = g^1 \left({W^1}^T \mathrm{x}\right)$

Activation function Weight matrix Input vector



Forward Propagation

Example:

$$x = \begin{bmatrix} -1 & 2 \\ 3 & 5 \\ 2 & 7 \\ 2 & 3 \end{bmatrix} \longrightarrow \text{2nd dimension feature}$$

$$W_k^{1^T} = [1 \quad 2 \quad 3 \quad 4\ ] \quad w_{k0}^1 = 5$$

$$W^{1^T} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 3 & 1 & 7 \\ 5 & -5 & 2 & 4 \end{bmatrix} \quad W_0^1 = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}$$

$$W_k^{1^T} x + w_{k0}^1 = [1 \quad 2 \quad 3 \quad 4] \begin{bmatrix} -1 & 2 \\ 3 & 5 \\ 2 & 7 \\ 2 & 3 \end{bmatrix} + [5 \quad 5]$$

$$= [19 \quad 25\ ] + [5 \quad 5]$$

$$= [24 \quad 30\ ]$$

$$W^{1^T} x + W_0^1$$

$$= \begin{bmatrix} 1 & 2 & 3 & 4 \\ -2 & 3 & 1 & 7 \\ 5 & -5 & 2 & 4 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ 3 & 5 \\ 2 & 7 \\ 2 & 3 \end{bmatrix}$$

$$+ \begin{bmatrix} 5 & 5 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 24 & 50 \\ 28 & 40 \\ -6 & 13 \end{bmatrix}$$

Forward Propagation

Consequently,

First Hidden Layer: $h^1 = g^1\,({W^1}^T \mathrm{x})$

Second Hidden Layer: $h^2 = g^2\,({W^2}^T h^1)$

nth Hidden Layer: $h^n = g^n\,({W^n}^T h^{n-1})$

Activation function   Weight matrix   Input / output of the previous layer

Cross Entropy – Information

We need some knowledge from **Information Theory**…

Information theory studies encoding, decoding, transmitting, and manipulating information.

Information / Shannon information / Self-information:

The intuition behind quantifying information is the idea of measuring how much surprise there is in an event. Those events that are rare (low probability) are more surprising and therefore have more information than those events that are common (high probability).

- **Low Probability Event**: High Information (*surprising*).
- **High Probability Event**: Low Information (*unsurprising*).

$$Information(x) = I(x) = -\log(p(x))$$

base-2 logarithm   probability of the event x

Cross Entropy – Information

Examples:

- Roll a dice, and the result is 1 ($p = 1/6$).

$$I(X) = -log_2\left(\frac{1}{6}\right) = 2.585$$

- 'You are an undergraduate student.' ($p = 1$).

$$I(X) = -log_2(1) = 0$$

bit is the unit of information.

$$I('0010') = -log_2\big(p('0010')\big) = -log_2\left(\frac{1}{2^4}\right) = 4 \text{ bits}$$

base 2 = bits   base 10 = Hartleys   base e = nats

Cross Entropy – Entropy

Self-information only measures the information of a single discrete event, we need a more generalized measure for any random variable of either **discrete or continuous distribution**.

Entropy:

The entropy of a random variable is the average level of 'information', 'surprise', or 'uncertainty' inherent to the variable's possible outcomes.

Given a discrete random variable X, with possible outcomes $x_1, x_2, \dots, xn$, which occur with probability $P(x_1), P(x_2), \dots, P(x_n)$, the entropy of X is formally defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot logP(x_i)$$

Entropy is the expected value of the self-information of the distribution $E(I(X))$.

Cross Entropy – Entropy

Examples:

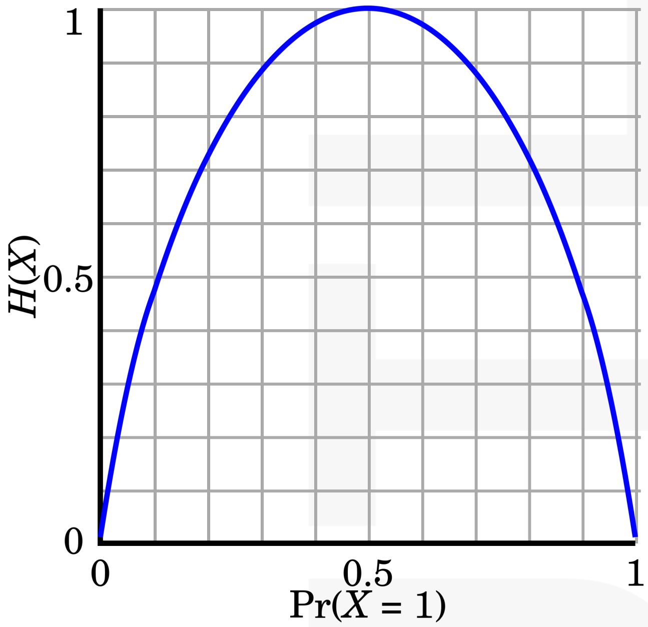
- tossing a coin, if heads and tails both have equal probability ½ (Bernoulli distribution)

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot logP(x_i)$$

$$= -\sum_{i=1}^{2} \frac{1}{2} \cdot log\frac{1}{2}$$

$$= -\sum_{i=1}^{2} \frac{1}{2} \cdot (-1)$$

$$= 1$$



Cross Entropy

The **cross entropy** measures the divergence between the estimating distribution q and the true distribution p for a random variable X.

For discrete probability distribution p and q with the same X,

$$H(p,q) = -\sum_{x \epsilon X} p(x) \cdot \log q(x)$$

$$= \underbrace{H(p)}_{\text{Entropy of } p} + \underbrace{D_{KL}(p \parallel q)}_{\text{Kullback-Leibler divergence, divergence of } p \text{ from } q}$$

Cross Entropy

Examples:

- Consider a 3-class classification.
  GrounTruth $p = [1,0,0]$
  Predicted result 1: $q_1 = [0.5, 0.2, 0.3]$
  Predicted result 2: $q_2 = [0.8, 0.1, 0.1]$

$$H(p, q_1) = -(1\times \log 0.5 + 0\times \log 0.2 + 0\times \log 0.3) = 1$$

$$H(p, q_2) = -(1\times \log 0.8 + 0\times \log 0.1 + 0\times \log 0.1) = 0.32$$