EE488C Special Topics in EE
<Deep Learning and AlphaGo>

Sae-Young Chung

Lecture 3

September 4, 2017

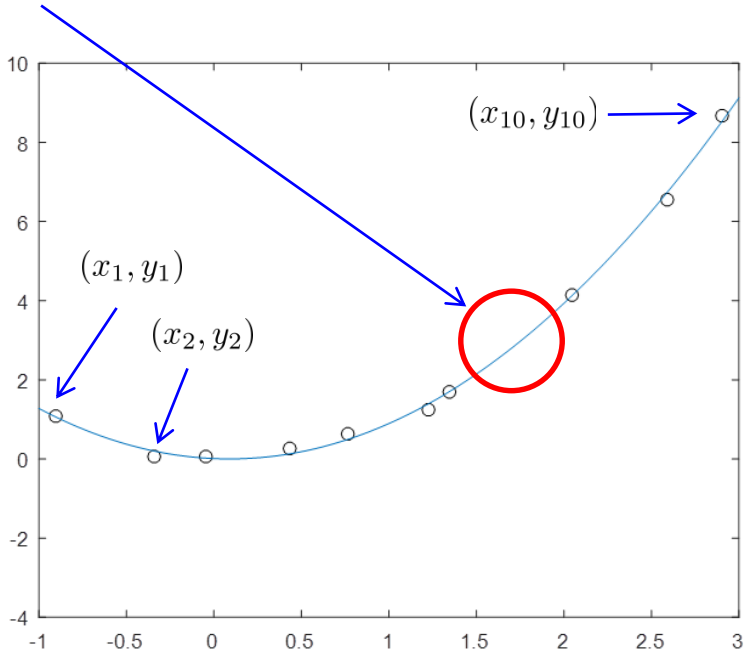
KAIST

Chap. 1 Introduction

- Machine learning examples
  - Regression
  - Classification
- Perceptron
- Representation learning
- Multilayer perceptron
- Universal approximation theorem

Machine Learning

- Learning from data without explicit programming
- Simple toy example) Curve fitting using polynomial ***regression***
  - Requires a model (polynomial, exponential, etc.)
  - Parameters of the model (coefficients of the polynomial) are found automatically from data (experience) via optimization
  - ***Can predict unseen samples!!! (generalization from experience)***



Least Square Solution

- Training examples

$$(x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_m, y_m)$$

- Model for polynomial regression

$$y_i = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \ldots + \theta_n x_i^n + z_i, \quad i = 1, \ldots, m$$

- Matrix form: $\mathbf{y} = X\theta + \mathbf{z}$, i.e.,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix} + \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix}$$

  - X is called a Vandermonde matrix

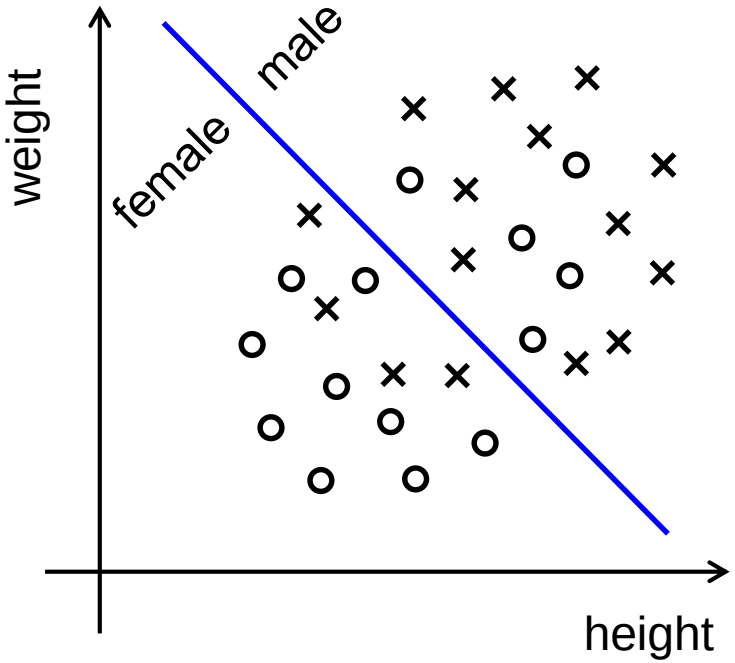
- Goal: to find the least square solution θ that minimizes $\|\mathbf{z}\|^2$ assuming $n < m$

- Answer

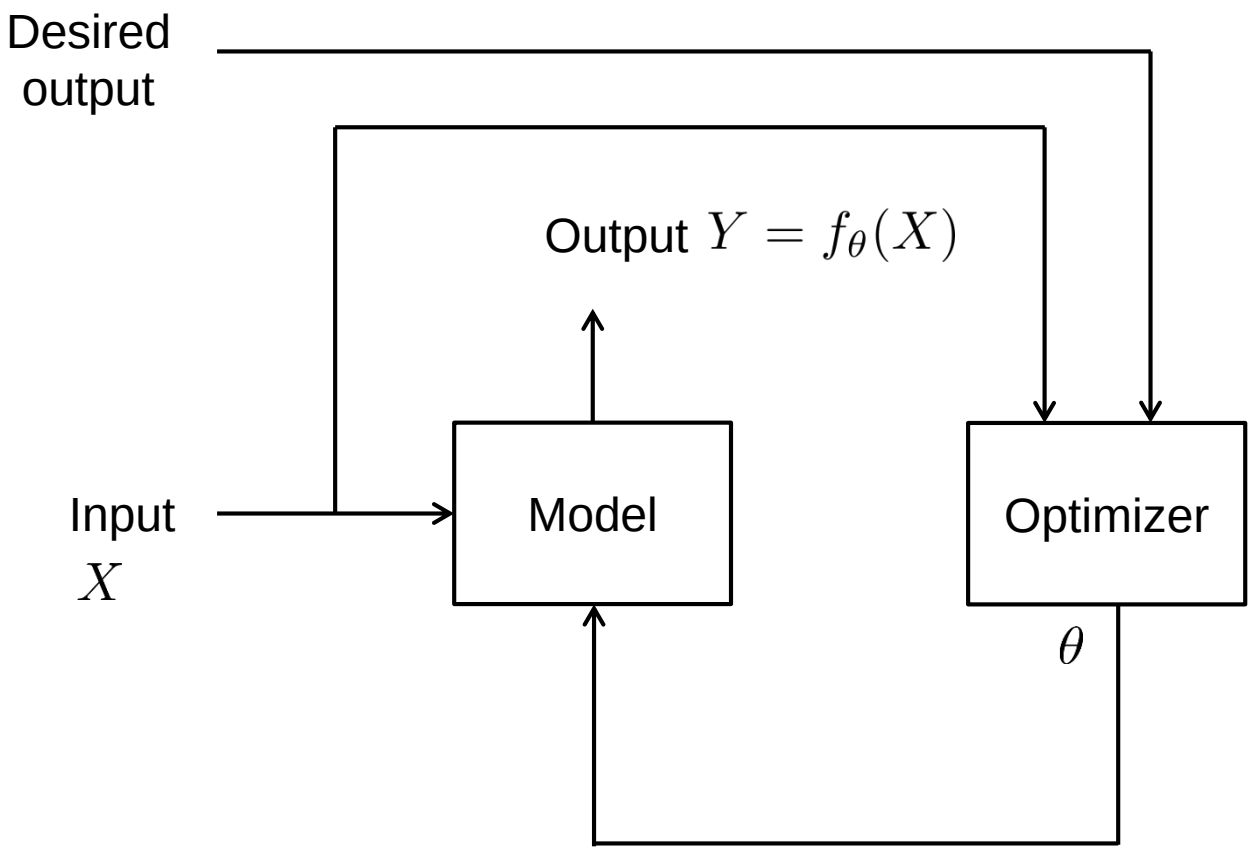
$$\theta^* = (X^T X)^{-1} X^T \mathbf{y}$$

Machine Learning

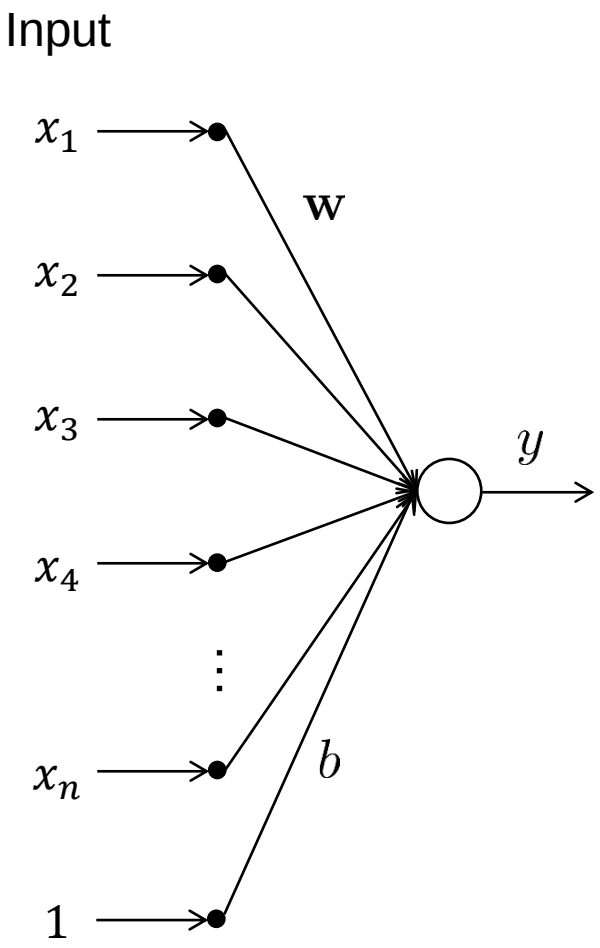
- Another example) Binary ***classification*** using a linear classifier



(Supervised) Learning System



Perceptron



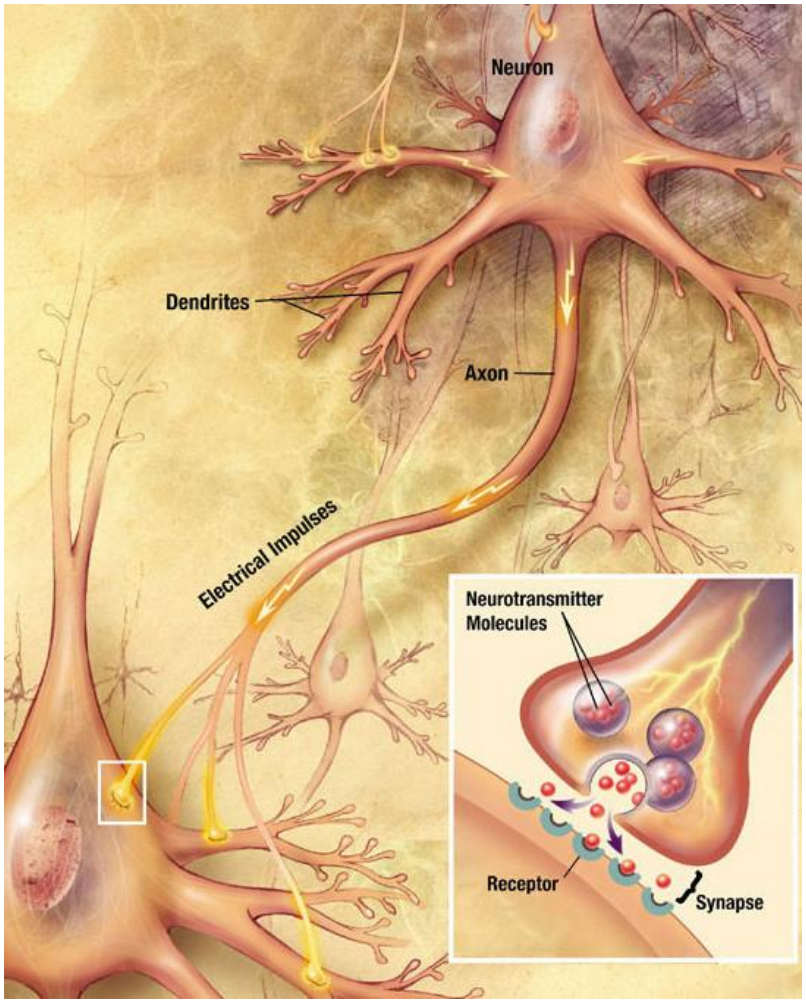
Simple artificial neuron capable of learning to perform binary classification

$$y = f(\mathbf{w} \cdot \mathbf{x} + b)$$

$$f(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\theta = (\mathbf{w}, b)$$

Rosenblatt, 1958



https://en.wikipedia.org/wiki/Neuron

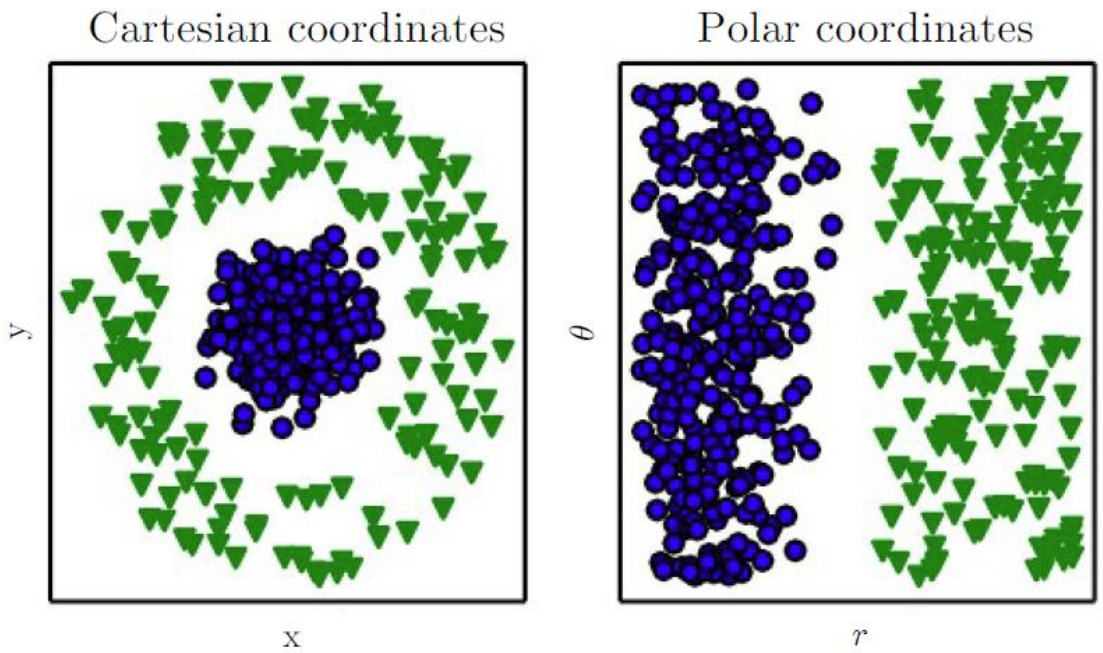


Fig. 1.1 Example of different representations

Feature: an information-bearing quantity contained in the representation

In this example, (r, θ) is a feature vector in polar coordinate representation. (x, y) is a feature vector in Cartesian coordinate representation.

Representation Learning

- Automated discovery of good representation from data
  - E.g., Cartesian coordinate or polar coordinate
- Model should be general enough for this to be possible
- ***Deep learning*** solves this problem by having hierarchical representations. Simpler representations are found first and then more complex representations are found based on simpler representations.

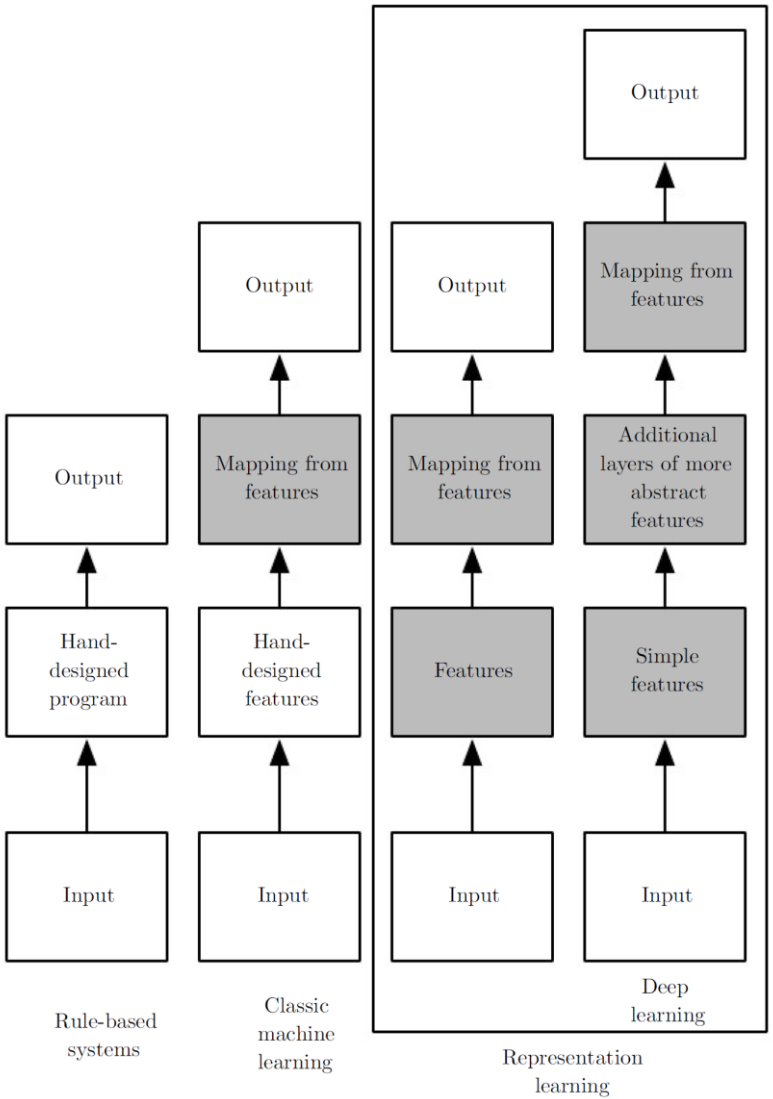
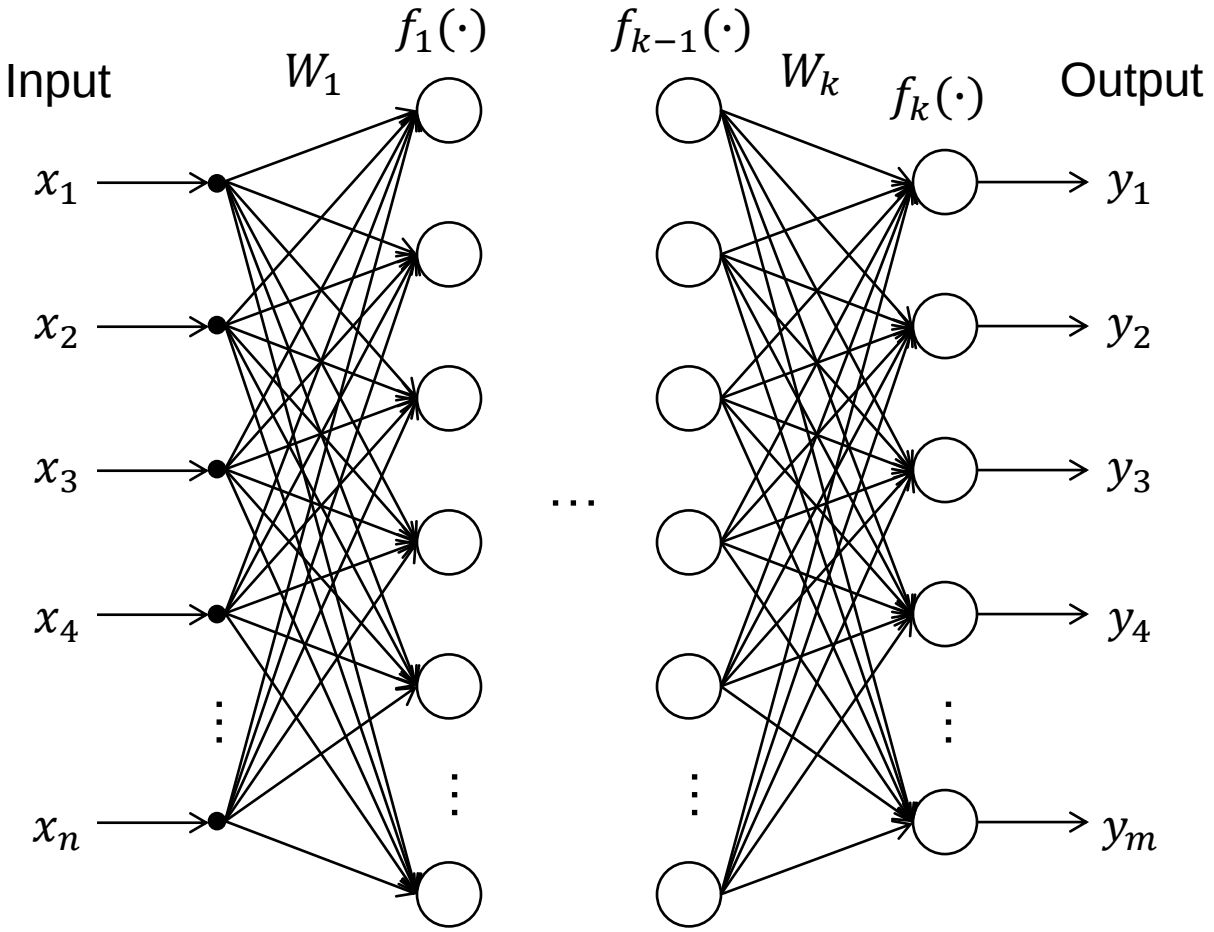


Figure 1.5: Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that are able to learn from data.

Multi-layer Perceptron

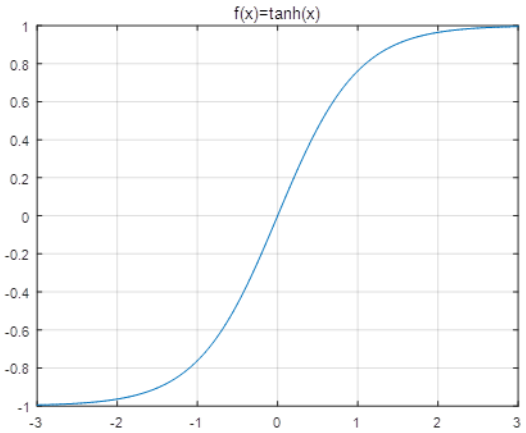


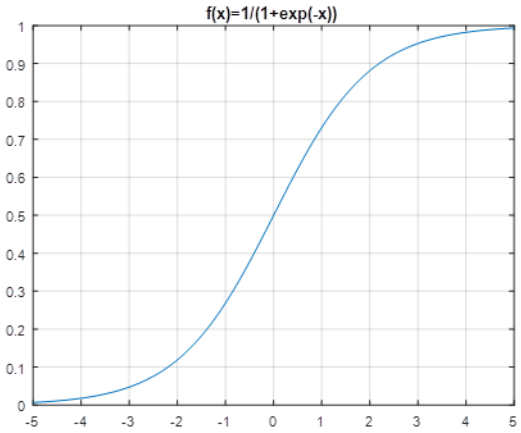
Multi-layer Perceptron

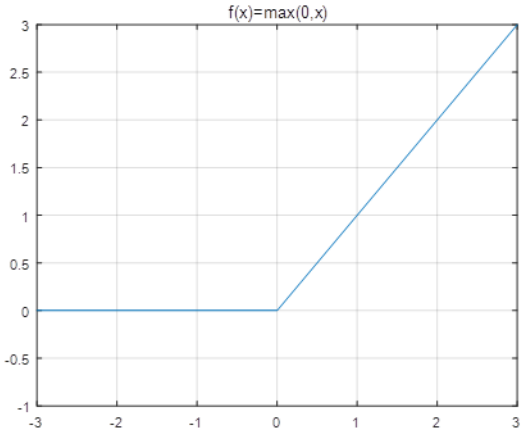
- Input-output relationship

$$\mathbf{y} = f_k(W_k f_{k-1}(W_{k-1} \cdots f_2(W_2 f_1(W_1 \mathbf{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) \cdots) + \boldsymbol{b}_k)$$

- Non-linear function (activation function): $f(\cdot)$
  - Unit step: $f(x) = 1$ if $x = 0$ and $f(x) = 0$ otherwise (original perceptron)
  - Sigmoid: $f(x) = 1/(1 + e^{-x})$
  - tanh: $f(x) = \tanh x$
  - Rectified linear unit (ReLU): $f(x) = \max(0, x)$

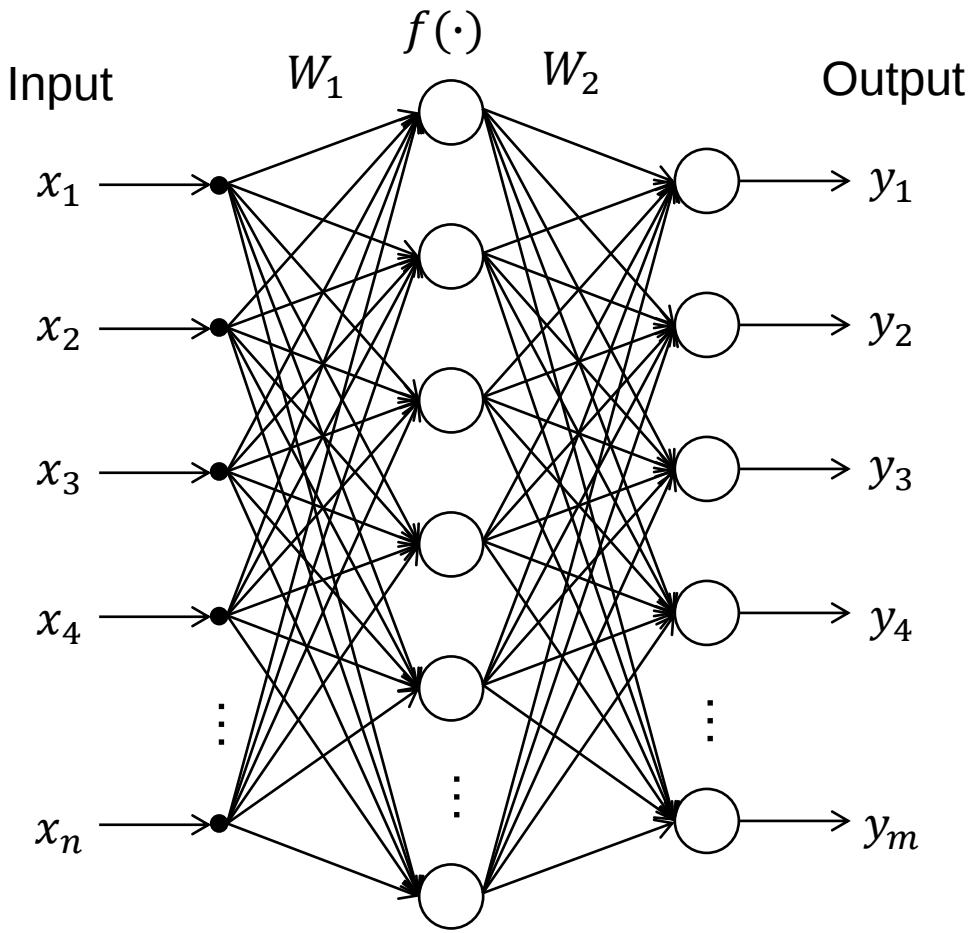






KAIST

Universal Approximation Theorem

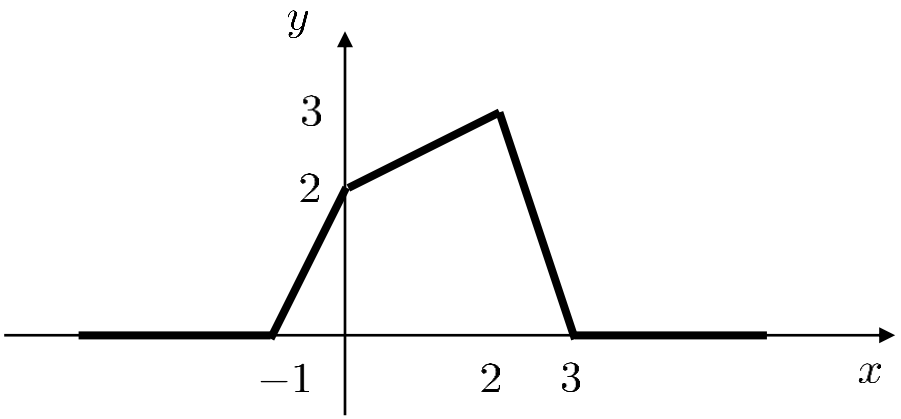


- Cybenko 1989
- A neural network with a single hidden layer with some non-linear activation function and a linear output layer can approximate any continuous function with arbitrary accuracy.

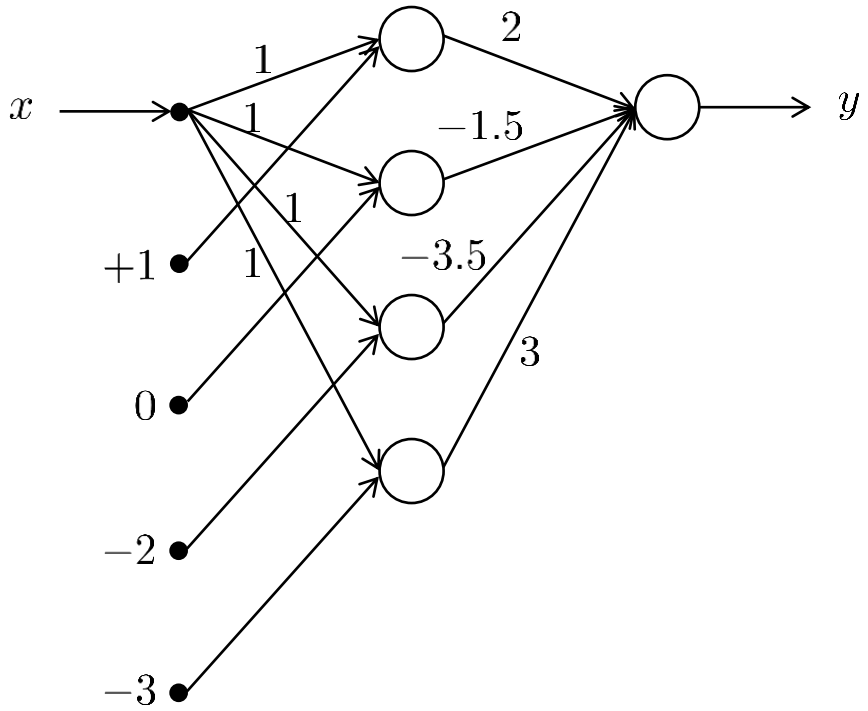
KAIST

Example

$$y = 2f(x+1) - 1.5f(x) - 3.5f(x-2) + 3f(x-3)$$



$$f(x) = \max\{0, x\}$$



KAIST

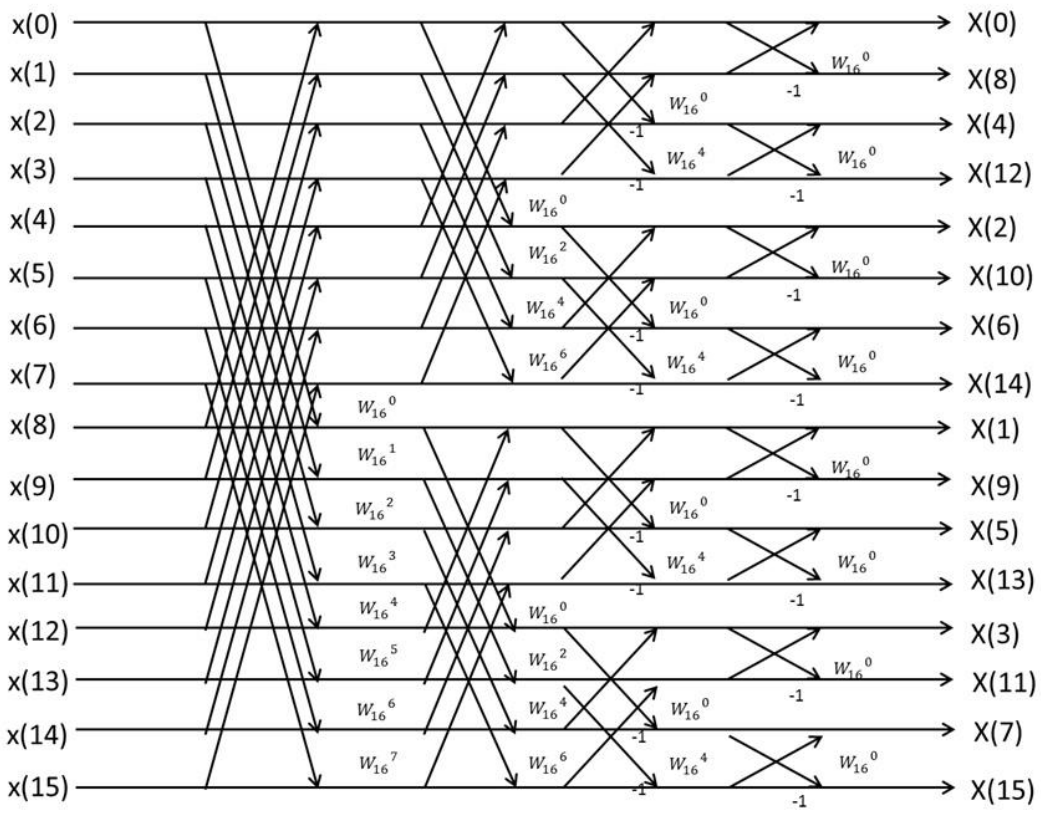
Problem

- But, we may need exponentially many neurons in the hidden layer if there is only one hidden layer, e.g., an arbitrary function mapping n binary inputs to one binary output, which requires $O(2^n)$ neurons in the hidden layer in general.
- Toy example) $y = 1$ if $(x_1, x_2, x_3) = (0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 1)$, or $(1, 1, 0)$ and $y = 0$ otherwise, which can be produced by the following neural network

$$y = f(x_3 - x_1 - x_2) + f(x_2 - x_1 - x_3) + f(x_1 + x_3 - x_2 - 1) + f(x_1 + x_2 - x_3 - 1),$$

  where $f(x) = \max\{x, 0\}$
- Having many hidden layers can reduce the minimum number of required neurons in the hidden layers drastically. This is why we want to have a deep neural architecture.

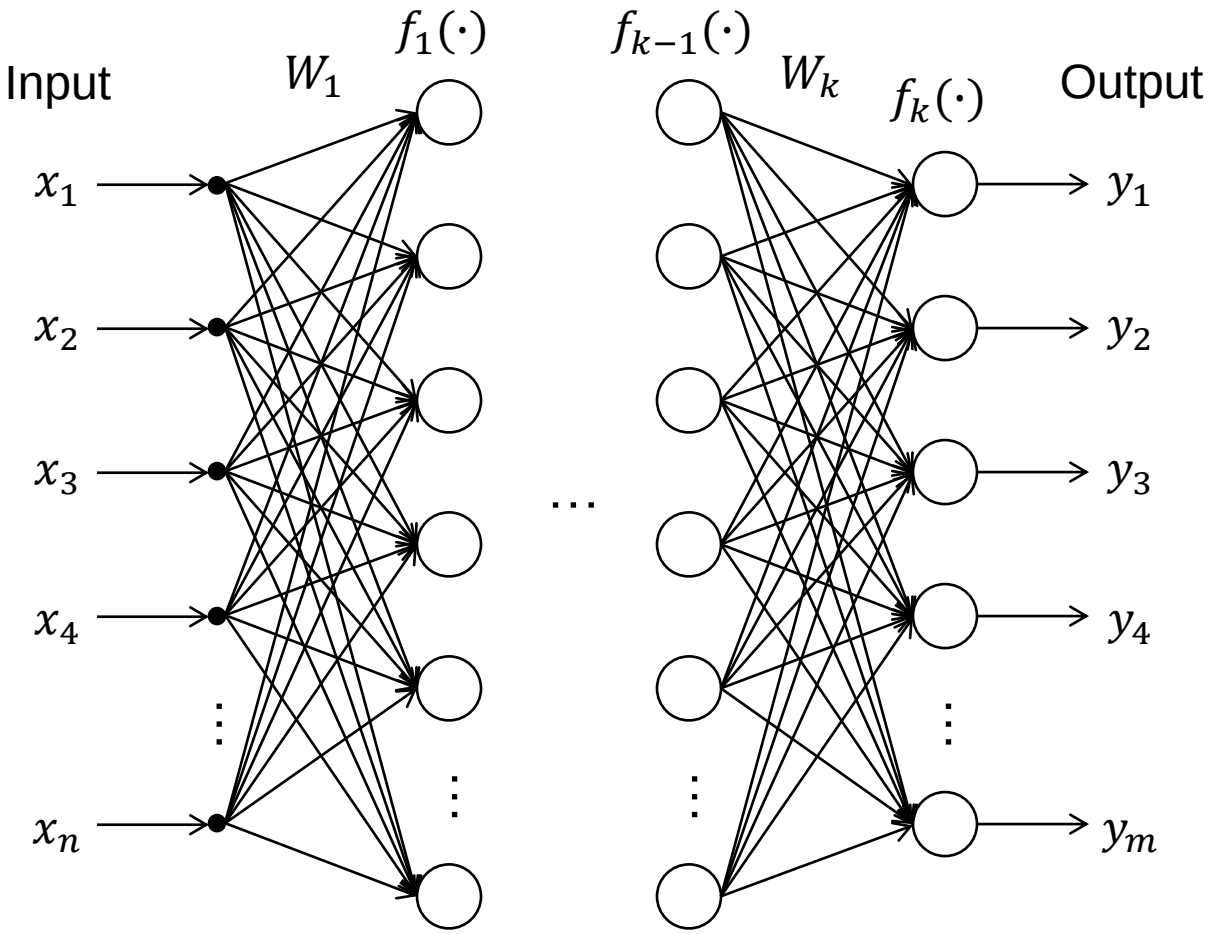
Fast Fourier Transform (FFT)



Example of 16-point FFT

- FFT: fast way of computing discrete Fourier transform (DFT) using $\log n$ stages
- n: # of data size
- Complexity of FFT: $O(n \log n)$
- Complexity of one-stage DFT: $O(n^2)$
- Why faster? Because intermediate computation results are stored and used in the next stage in an efficient way.
- Likewise, having multiple hidden layers in deep learning can also help.

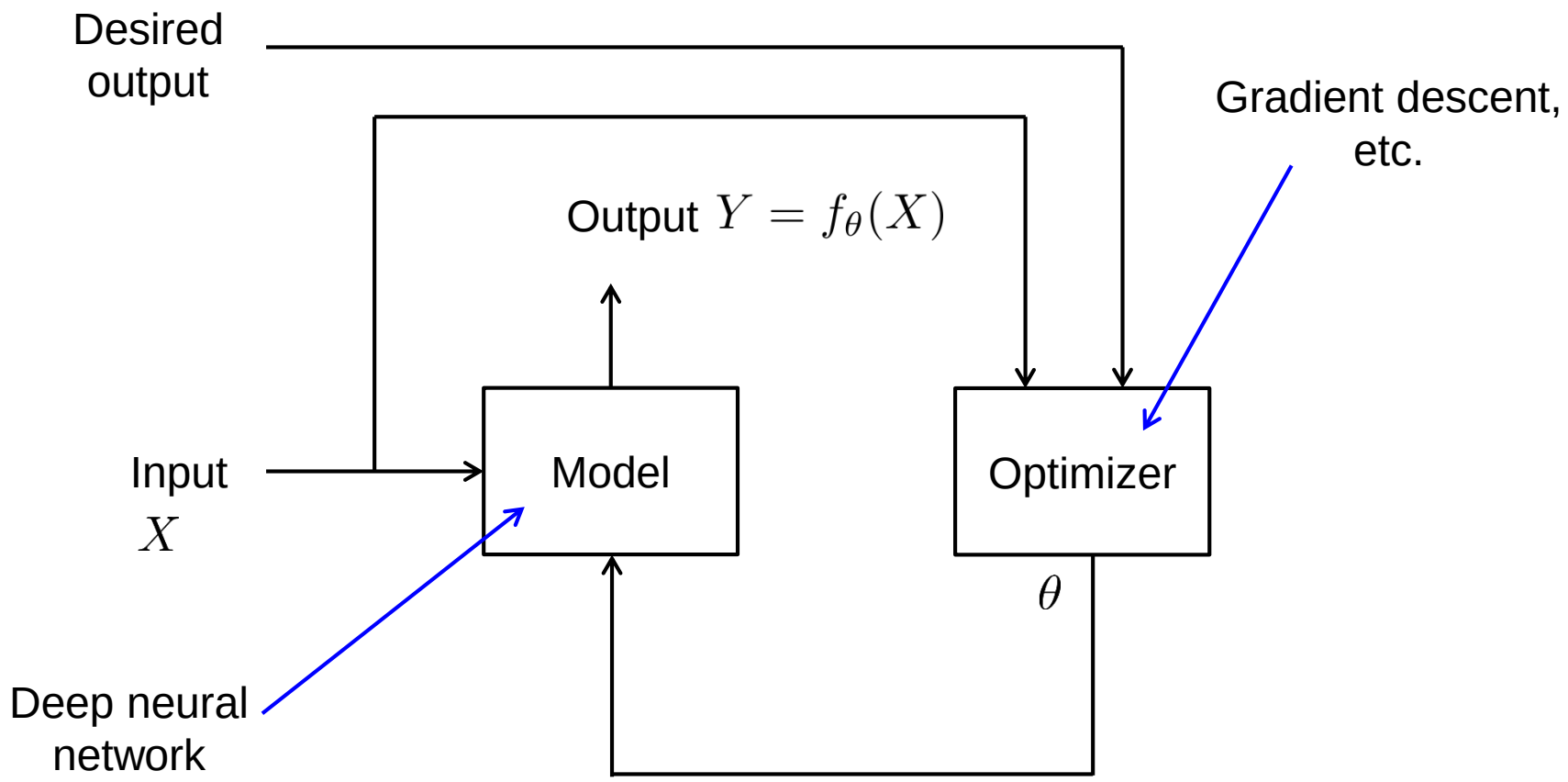
Deep Feed-forward Neural Network



Linearity and Generalization

- Linear operations in neural network, i.e., matrix multiplication by W, makes generalization from experience easier by providing smooth interpolation.

Training for DNN



Chap. 2 Linear Algebra

- Vector norm, matrix norm
- Inner product
- Orthogonal matrix
- Eigendecomposition
- Singular value decomposition
- Matrix approximation lemma

Norm

- L^p norm ($p \geq 1$): $\|\mathbf{x}\|_p := \left(\sum_i |x_i|^p\right)^{1/p}$
- L^2 norm (Euclidean norm): $\|\mathbf{x}\| := \sqrt{\sum_i |x_i|^2}$
- L^0 "norm": # of non-zero elements in $\mathbf{x}$
- L^∞ norm (maximum norm): $\|\mathbf{x}\|_\infty := \max_i |x_i|$
- Frobenius norm of a matrix: $\|A\|_F := \sqrt{\sum_{i,j} |A_{i,j}|^2}$
- Inner product between two vectors: $\mathbf{x}^T\mathbf{y} = \sum_i x_i y_i = \|\mathbf{x}\| \cdot \|\mathbf{y}\| \cos\theta$

Orthogonal Matrix

- Orthogonal matrix: real square matrix A such that $A^T A = AA^T = I$
- $\det(A^T A) = \det(A^T)\det(A) = \det(A)\det(A) = (\det A)^2$
- Since $A^T A = I$ for an orthogonal matrix, $(\det A)^2 = 1$, i.e., $\det A = \pm 1$
- If $\det A = 1$, then (proper) rotation
- If $\det A = -1$, then (proper) rotation followed by reflection (mirror image)
- 2×2 orthogonal matrix

$$\pm \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Eigendecomposition

- Eigenvalue λ and (right) eigenvector $\mathbf{v}$ of an $n \times n$ square matrix A:

$$A\mathbf{v} = \lambda\mathbf{v}$$

- There are n eigenvalues (up to multiplicity).
- Eigenvalues and eigenvectors can be complex even when A is real.
- Eigendecomposition of A

$$A = V\,\mathrm{diag}(\lambda)V^{-1}$$

  where $V = [\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(n)}]$ is a matrix whose columns are eigenvectors of A and $\lambda = [\lambda_1, \ldots, \lambda_n]^T$ is the set of corresponding eigenvalues.

- Eigendecomposition does not always exist.
- If A is real and symmetric (i.e., $A^T = A$), then its eigenvalues are all real and can be decomposed as follows

$$A = Q\Lambda Q^T$$

  where Q is an orthogonal matrix and $\Lambda = \mathrm{diag}(\lambda)$.

Singular Value Decomposition

- A real $m \times n$ matrix A can be written as

$$A = UDV^T$$

  where U and V are orthogonal matrices of sizes $m \times m$ and $n \times n$, respectively, and D is an $m \times n$ diagonal matrix with non-negative entries.

- D contains the singular values of A (typically in non-increasing order).

Matrix Approximation Lemma

Theorem. *The solution to the following problem*

$$\min_{\hat{A}:\ \mathrm{rank}(\hat{A})\le r} \|A-\hat{A}\|_F$$

is given by

$$\hat{A}^* = U_1\Sigma_1 V_1^T$$

where U_1 *is* $m\times r$, V_1 *is* $n\times r$, *and* Σ_1 *is* $r\times r$ *satisfying the singular value decomposition* $A = U\Sigma V^T = \begin{pmatrix} U_1 & U_2 \end{pmatrix}\begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}\begin{pmatrix} V_1 & V_2 \end{pmatrix}^T$. *The singular values* $\{\sigma_1,\ldots,\sigma_{\min\{m,n\}}\}$ *in* $\begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}$ *are in non-increasing order.* $\hat{A}^*$ *is unique if and only if* $\sigma_r \neq \sigma_{r+1}$.

Proof. *Let* B *be an* $m\times n$ *matrix and* $D = U^TBV$, *then* $B = UDV^T$. *Then, we have*

$$\begin{aligned}\|A-B\|_F^2 &= \|U\Sigma V^T - UDV^T\|_F^2 \\ &= \|\Sigma - D\|_F^2 = \sum_i |\sigma_i - D_{ii}|^2 + \sum_{i\neq j}|D_{ij}|^2 \ge \sum_i |\sigma_i - D_{ii}|^2\end{aligned}$$

Chap. 3 Probability & Info Theory

- Why probability?
  - Truly random (e.g., quantum mechanical randomness)
  - Lack of observation (e.g., without knowing the precise initial velocity and position of a dice, you cannot predict its outcome precisely even if it follows Newtonian dynamics)
  - Lack of model (e.g., even with precise information about the initial velocity and position of a dice, you cannot predict its outcome precisely without exact model of environment – movement of air molecules and conditions of floor and wall)
  - Lack of computational resource (e.g., even with precise information about the initial velocity and position of a dice and precise model of the environment, you cannot predict its outcome precisely since doing so requires a prohibitively large amount of calculations)

Notations

- Random variable: X
- Value of a random variable: x
- Alphabet set of X: $\mathcal{X}$, i.e., $X \in \mathcal{X}$

$$\text{Outcome of dice rolling, } \mathcal{X} = \{1, 2, 3, 4, 5, 6\}$$

- Probability mass function (pmf): $p_X(x) = \Pr(X = x)$ or simply $p(x)$
- Probability density function (pdf): $p(x)$
- Joint pmf: $p(x, y)$
- Conditional probability: $p(y|x) = \Pr(X = x|Y = y) = \frac{p(x,y)}{p(y)}$
- Chain rule of conditional probability

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \ldots p(x_n|x_1, \ldots, x_{n-1})$$

- X and Y are independent if $p(x, y) = p(x)p(y)$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$
- X and Y are conditionally independent given Z if

$$p(x, y|z) = p(x|z)p(y|z) \text{ for all } x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$$

- Expectation: $\mathbb{E}_{X\sim p}[f(X)] = \sum_x p(x)f(x)$, $\mathbb{E}_{X\sim p}[f(X)] = \int p(x)f(x)dx$
- Variance: $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$, $\text{Var}(f(X)) = \mathbb{E}[(f(X) - \mathbb{E}[f(X)])^2]$
- Covariance: $\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))]$
- Covariance matrix of a random vector $\mathbf{X}$

$$\text{Cov}(\mathbf{X})_{i,j} = \text{Cov}(X_i, X_j)$$

Common Distributions

- Bernoulli: $X = 1$ with probability p and $X = 0$ with probability $1 - p$

$$\mathbb{E}[X] = p, \ \mathrm{Var}(X) = p(1-p)$$

- Gaussian (normal) distribution

$$\mathcal{N}(X; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- Multivariate normal distribution

$$\mathcal{N}(\mathbf{X}; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right)$$

- Exponential distribution

$$p(X; \lambda) = \mathbf{1}_{x \geq 0} \lambda \exp(-\lambda x)$$

- Laplace distribution

$$\mathrm{Laplace}(X; \mu, \gamma) = \frac{1}{2\gamma} \exp\left(-\frac{|x-\mu|}{\gamma}\right)$$

Common Functions

- Logistic function (logistic curve, sigmoid function)

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Useful when limiting the value between 0 and 1, e.g, when output needs to represent the parameter p of a Bernoulli variable.

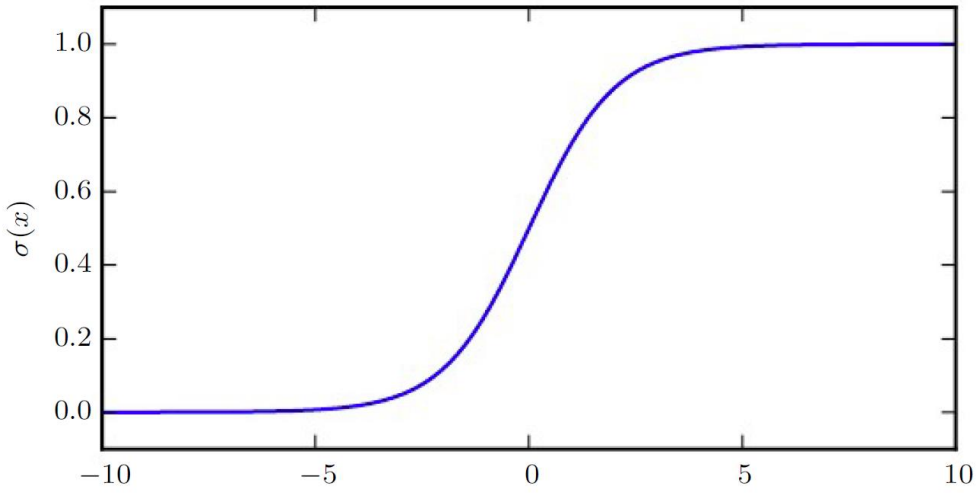


Figure 3.3: The logistic sigmoid function.

Common Functions

- Softplus function

$$\zeta(x) = \log(1 + \exp(x))$$

Softened version of $\max(0, x)$

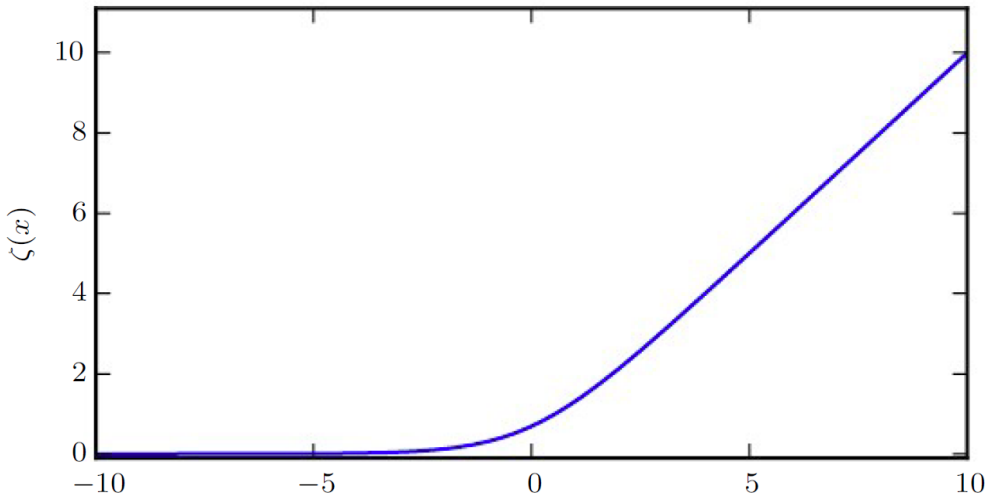


Figure 3.4: The softplus function.

Information Theory

- About knowing the limits of anything involving information.
  - Communication
  - Computing
  - Control
  - Inference
  - Machine learning
  - Statistics
  - Stock market
  - …
- About designing systems that can approach such limits closely.
  - LZ algorithm, Huffman coding, MP3, JPEG, MPEG, …
  - Reed-Solomon, Hamming, Turbo, LDPC codes
  - Dirty paper coding, rateless codes, network coding
  - Wireless communication, relaying
  - …

Entropy

- How do we measure the amount of information, the amount of uncertainty or randomness?
  - Entropy
- Examples
  - Entropy of the result of tossing of a fair coin: 1 bit
  - Entropy of the result of tossing of two fair coins: 2 bit
  - Entropy of the result of tossing of a fair dice?
  - Entropy of the result of tossing of a bent coin?

Entropy

- How much information is contained in a n-bit i.i.d. (indepedent and identically distributed) Bernoulli(p) sequence?
  - 00010000010001000000010000010 (n bits long)
- Clearly n bits if $p = 1/2$.
- What if $p \neq 1/2$?
  - Count the number k of 1's in the source.
  - Specify k and also the index of the source among $\binom{n}{k}$ candidate sequences with k 1's.
- We need $\sim \log_2 k + \log_2 \binom{n}{k}$ bits.
- If n is large, this approaches $nH(p)$, where $H(p)$ is the binary entropy function given by

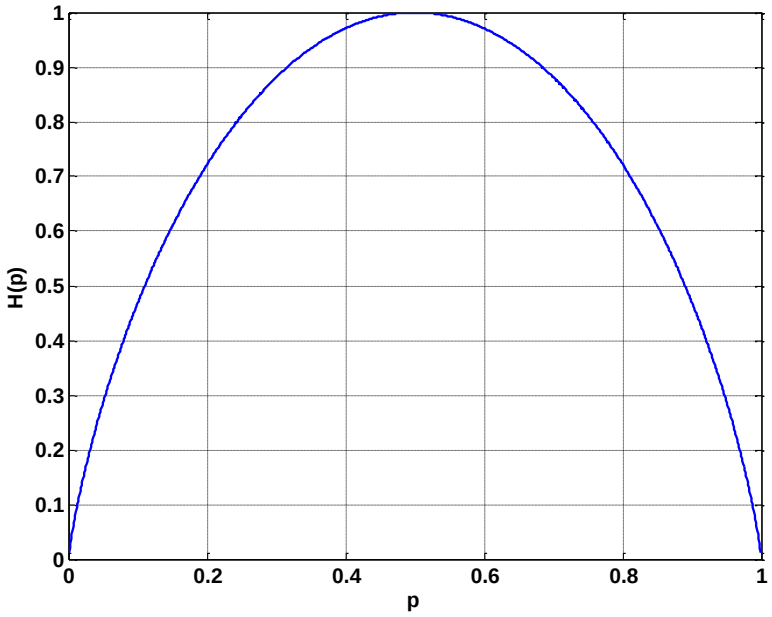
$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

Binary Entropy Function

- Binary entropy function

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

- $H(p)$ is monotonically increasing for $0 \leq p \leq 1/2$.
- $0 \leq H(p) \leq 1$.
- $H(1-p) = H(p)$.
- $H(p)$ is maximized when $p = 1/2$.
- $H(0) = H(1) = 0$ and $H(1/2) = 1$.
- $H(1/3) \sim 0.92$ [bits]



Entropy Function

- For m-ary i.i.d. (memoryless) source, we need

$$H(p_1, \ldots, p_m) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

bits on average to describe each source symbol (asymptotically), where p_k is the probability of the k-th alphabet.

Entropy of a Random Variable

- Entropy of a random variable X over alphabet $\mathcal{X} = \{x_1, \ldots, x_m\}$

$$H(X) = -E[\log_2 p(X)] = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) = -\sum_{k=1}^{m} p_k \log_2 p_k \quad [\text{bits}],$$

  where $X = x_k$ with probability p_k, $k = 1, \ldots, m$.

- Entropy is measured in bits (if log to the base 2 is used).
- Entropy function is non-negative.
- Entropy does not depend on the values of a random variable, but only on its pmf.
- Continuous version: differential entropy

Joint Entropy, Conditional Entropy

- Joint entropy

$$H(X,Y) = -E[\log p(X,Y)] = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log p(x,y)$$

- Conditional entropy

$$\begin{aligned} H(Y|X) &= -E[\log p(Y|X)] = -\sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log p(y|x) \\ &= -E_X[E_{Y|X}[\log p(Y|X)|X]] = -\sum_{x\in\mathcal{X}} p(x)\sum_{y\in\mathcal{Y}} p(y|x)\log p(y|x) \end{aligned}$$

- Chain rule

$$\begin{aligned} &\log p(X,Y) = \log p(X) + \log p(Y|X) \longrightarrow H(X,Y) = H(X) + H(Y|X) \\ &H(X,Y|Z) = H(X|Z) + H(Y|X,Z) \end{aligned}$$

Relative Entropy (KL Divergence)

- Relative entropy (Kullback-Leibler (KL) divergence) between two pmf's $p(x)$ and $q(x)$.

$$D(p\|q) = \mathbb{E}_{X\sim p(x)}\left[\log \frac{p(X)}{q(X)}\right] = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}$$

- Properties
  - Always non-negative: $D(p\|q) \geq 0$
  - Not symmetric in general: $D(p\|q) \neq D(q\|p)$
  - Does not satisfy triangle inequality in general
- Cross-entropy

$$H(p,q) = H(p) + D(p\|q) = -\mathbb{E}_{X\sim p}[\log q(X)]$$

Mutual Information

- Mutual information

$$I(X;Y) = D(p(X,Y)\|p(X)p(Y)) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$= \sum_{x,y} p(x,y) \log \frac{p(x|y)}{p(x)} = \sum_{x,y} p(x,y) \log p(x|y) - \sum_{x,y} p(x,y) \log p(x)$$

$$= H(X) - H(X|Y)$$

  - Reduction in the uncertainty of X due to the knowledge of Y.

- (Optional reading) If you are interested in learning more, then read Chapter 2 of Cover & Thomas, Elements of Information Theory, 2nd ed.

Reading Assignment

- Chapter 4 of DL book