# Fundamentals of Media Processing

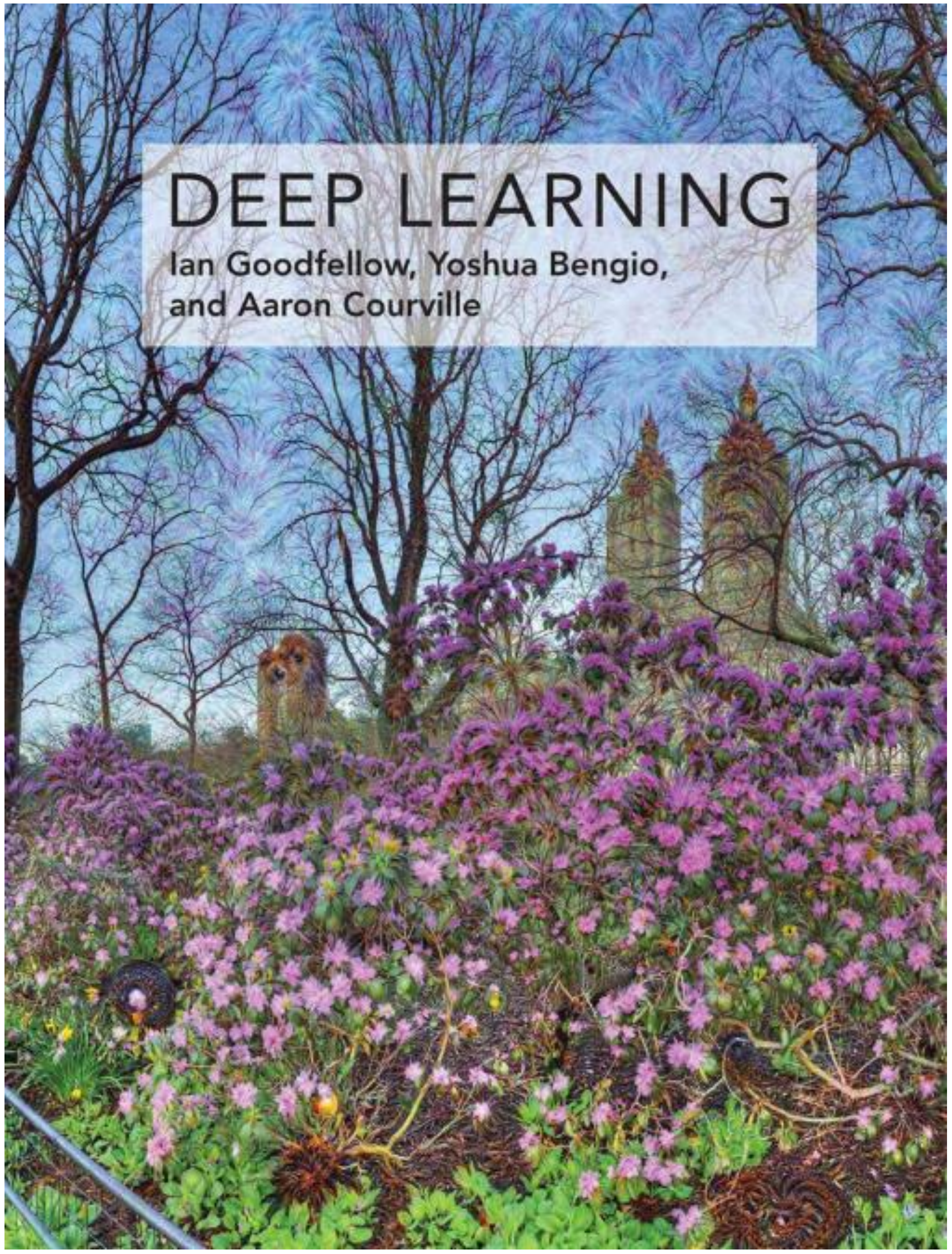
Lecturer:
池畑　諭（Prof. IKEHATA　Satoshi）
児玉　和也（Prof. KODAMA Kazuya）

Support:
佐藤　真一（Prof. SATO　Shinichi）
孟　洋（Prof. MO　Hiroshi）

## Course Overview (15 classes in total)

| | |
|---|---|
| **1-10** | **Machine Learning by Prof. Satoshi Ikehata** |
| 11-15 | Signal Processing by Prof. Kazuya Kodama |

Grading will be based on the final report.



Chapter 1-9 (out of 20)

**An introduction to a broad range of topics in deep learning, covering mathematical and conceptual background, deep learning techniques used in industry, and research perspectives.**

- Due to my background, I will mainly talk about “image”
- I will introduce some applications beyond this book

10/16 (Today) Introduction Chap. 1

## Basic of Machine Learning (Maybe for beginners)

10/23 Basic mathematics (1) (Linear algebra, probability, numerical computation) Chap. 2,3,4

10/30 Basic mathematics (2) (Linear algebra, probability, numerical computation) Chap. 2,3,4

11/6 Machine Learning Basics (1) Chap. 5

11/13 Machine Learning Basics (2) Chap. 5

## Basic of Deep Learning

11/20 Deep Feedforward Networks Chap. 6

11/27 Regularization and Deep Learning Chap. 7

12/4 Optimization for Training Deep Models Chap. 8

## CNN and its Application

12/11 Convolutional Neural Networks and Its Application (1) Chap. 9 and more

12/18 Convolutional Neural Networks and Its Application (2) Chap. 9 and more

## Linear Algebra

# Scalars, Vectors, Matrices and Tensors

Scalars

$$a = 1$$

(1-D) Vector

$$\boldsymbol{a} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = [0,1]^T$$

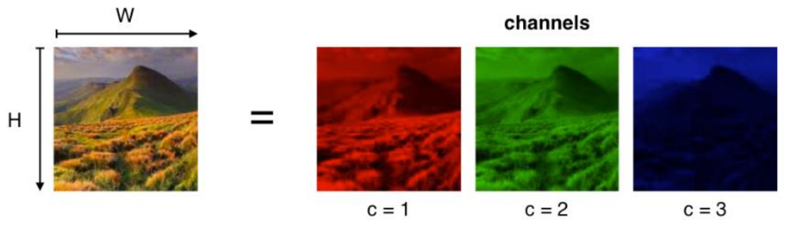
(2-D) Matrix

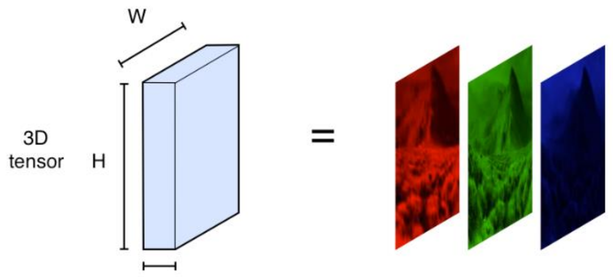
$$A = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \quad A^T = \begin{bmatrix} 0 & 2 \\ 1 & 3 \end{bmatrix} \quad A \in \mathbb{R}^{2\times 2}$$

$$A_{1,1} = 0, \quad A_{1,2} = 1, \quad A_{2,1} = 2, \quad A_{2,2} = 3$$

(N-D) Tensor

$$A_{i,j,k\ldots}$$





# Multiplying Matrices and Vectors

$\boldsymbol{x}^T\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{x}$

$A(B+C) = AB + AC$

$\boldsymbol{x} * \boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x} = |\boldsymbol{x}|^2$

$(AB)^T = B^T A^T$

| | | | |
|---|---|---|---|
| $\|\boldsymbol{x}\|_2 = 1$ | Unit vector | $\boldsymbol{x}^T\boldsymbol{y} = 0$ | Orthogonal |
| $AA^{-1} = I$ | Inverse matrix | $A^T = A$ | Symmetric matrix |
| $A^T A = AA^T = I$ | Orthogonal matrix | $\mathrm{Tr}(A) = \sum_i A_{i,i}$ | Trace operator |

| | | | |
|---|---|---|---|
| L1 Norm | $\|\boldsymbol{x}\|_{\mathbf{1}} = \sum_{\boldsymbol{i}} \|x_i\|$ | L∞ Norm | $\|\boldsymbol{x}\|_\infty = \max_{\boldsymbol{i}} \|x_i\|$ |
| L2 Norm | $\|\boldsymbol{x}\|_{\mathbf{2}} = \left(\sum_{\boldsymbol{i}} \|x_i\|^{\mathbf{2}}\right)^{\frac{\mathbf{1}}{\mathbf{2}}}$ | Frobenius Norm | $\|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2}$ |
| L0 Norm | $\|\boldsymbol{x}\|_{\mathbf{0}} = \#$ of nonzero entries | | $\|A\|_F = \sqrt{\mathrm{Tr}(AA^T)}$ |

## Matrix Decomposition

- Decomposition of matrices shows us information about their functional properties that is not obvious from the representation of the matrix

- ***Eigendecomposition***
- ***Singular value decomposition***
- LU decomposition
- QR decomposition
- Jordan decomposition
- Cholesky decomposition
- Schur decomposition
- Rank factorization
- And more…

Differs in

- Applicable matrix
- Decomposition type
- Application

# Eigendecomposition

- ***Eigenvector*** and ***eigenvalue***:

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

$\boldsymbol{v}$: Eigenvector
$\lambda$: Eigenvalue

- ***Eigendecomposition*** of a diagonalizable matrix :

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

$V$: A set of eigenvectors
$\lambda$: A set of eigenvalues

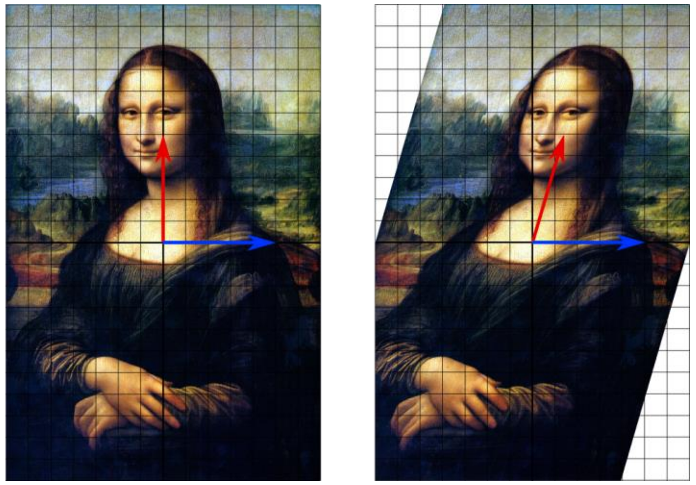
- Every symmetric matrix has a real, orthogonal eigendecomposition

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

- A matrix whose eigenvalues are all positive is ***positive definite***, positive or zero is ***positive semidefinite***

# What Eigenvalues and Eigenvectors are?

- Eigenvalues and eigenvectors feature prominently in the analysis of linear transformations. The prefix *eigen-* is adopted from the German word *eigen* for "proper", "characteristic". Originally utilized to study principal axes of the rotational motion of rigid bodies, eigenvalues and eigenvectors have a wide range of applications, for example in stability analysis, vibration analysis, atomic orbitals, facial recognition, and matrix diagonalization.



In this shear mapping the red arrow changes direction but the blue arrow does not. The blue arrow is an eigenvector of this shear mapping because it does not change direction, and since its length is unchanged, its eigenvalue is 1.

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

# Calculation of Eigendecomposition

- Simply solve a linear equation

$$A\boldsymbol{v} = \lambda\boldsymbol{v} \;\rightarrow\; (A - \lambda I)\boldsymbol{v} = 0$$

- $A - \lambda I$ must not have inverse matrix, otherwise $v = 0$

$$\det(A - \lambda I) = 0 \rightarrow \quad p(\lambda) = (\lambda - \lambda_1)^{n_1}(\lambda - \lambda_2)^2\,(\lambda - \lambda_3)^{n_3} \ldots$$

- Once $\lambda_s$ are calculated, eigenvectors are calculated by:

$$(A - \lambda I)\boldsymbol{v} = 0$$

# Singular Value Decomposition (SVD)

- Similar to eigendecomposition, but matrix need not be square. ***Singular value decomposition*** of a real matrix A is

$$A = UDV^T \qquad A\boldsymbol{u} = \sigma\boldsymbol{u} \qquad A^T\boldsymbol{v} = \sigma\boldsymbol{v}$$

- The left singular vectors of $B$ are the eigenvectors of $AA^T$
- The right singular vectors of $B$ are the eigenvectors of $A^T A$
- SVD is useful for the matrix inversion of non-square matrix (***pseudo inverse matrix***) or rank approximation (E.g., find nearest matrix whose rank is k) or solving a ***homogenous system***

## Solving Linear Systems

- Number of Eq = Number of Unknown (Determined)

$$a + 2b + 3c = 3$$
$$2a - b + c = 2$$
$$5a + b - 2c = 1$$

$$\underbrace{\begin{pmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \\ 5 & 1 & -2 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} a \\ b \\ c \end{pmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}}_{\boldsymbol{y}}$$

$$\boldsymbol{x} = A^{-1}\boldsymbol{y}$$

- Number of Eq < Number of Unknown (Underdetermined)

$$a + 2b + 3c = 3$$
$$2a - b + c = 2$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

?

- Number of Eq > Number of Unknown (Overdetermined)

$$a + 2b + 3c = 3$$
$$2a - b + c = 2$$
$$5a + b - 2c = 1$$
$$8a - b + 6c = 0$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \\ 5 & 1 & -2 \\ 8 & -1 & 6 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 1 \\ 0 \end{pmatrix}$$

?

# Moore-Penrose Pseudoinverse

- SVD gives the ***pseudoinverse*** of A as

$$A^+ = VD^+U^T \quad (A = UDV^T, D^+_{ii} = \frac{1}{D_{ii}})$$

- The solution to any linear systems is given by

$$\boldsymbol{x} = A^+\boldsymbol{y}$$

- If the linear system is:
  - Determined: this is same as the inverse
  - Underdetermined: this gives us the solution with the smallest norm of x
  - Overdetermined: this gives us the solution with the smallest error

## The Homogeneous System

- We may want to solve the homogenous system that is defined as

$$A\boldsymbol{x} = 0$$

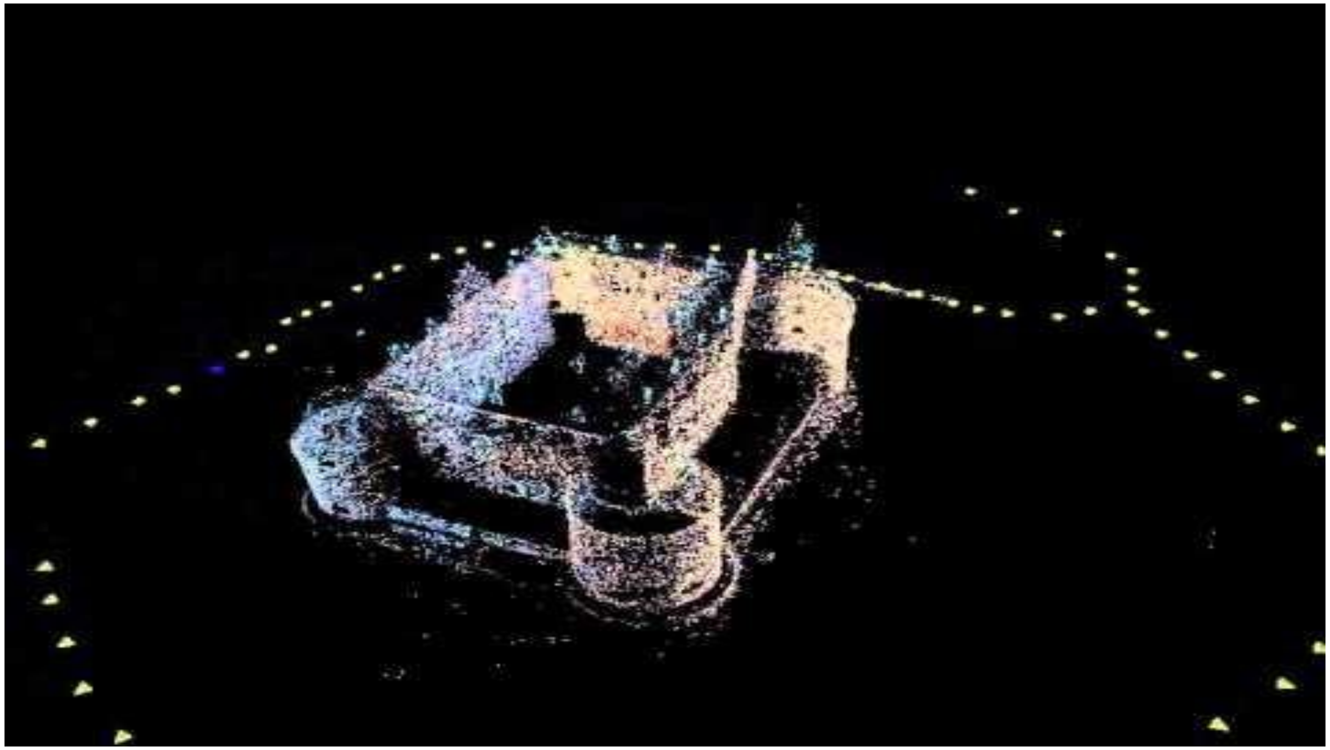
- If $\det(A^T A) = 0$, numerical solutions to a homogeneous system can be found with SVD or eigendecompositoin

$$Ax = 0 \qquad A^T A x = 0 \qquad A^T A v = \lambda v$$

  - Eigenvector corresponding to smallest eigen value (~=0)
  - Or, the column vector in V which is corresponding to the smallest singular value ($A = UDV^T$)

## Example: Structure-from-Motion

https://www.youtube.com/watch?v=i7ierVkXYa8



# Example: Structure-from-Motion

- Relative camera poses could be computed from the point correspondences on images
- Given camera pose (R,t), and camera intrinsics (K), 3D points X is given by triangulation

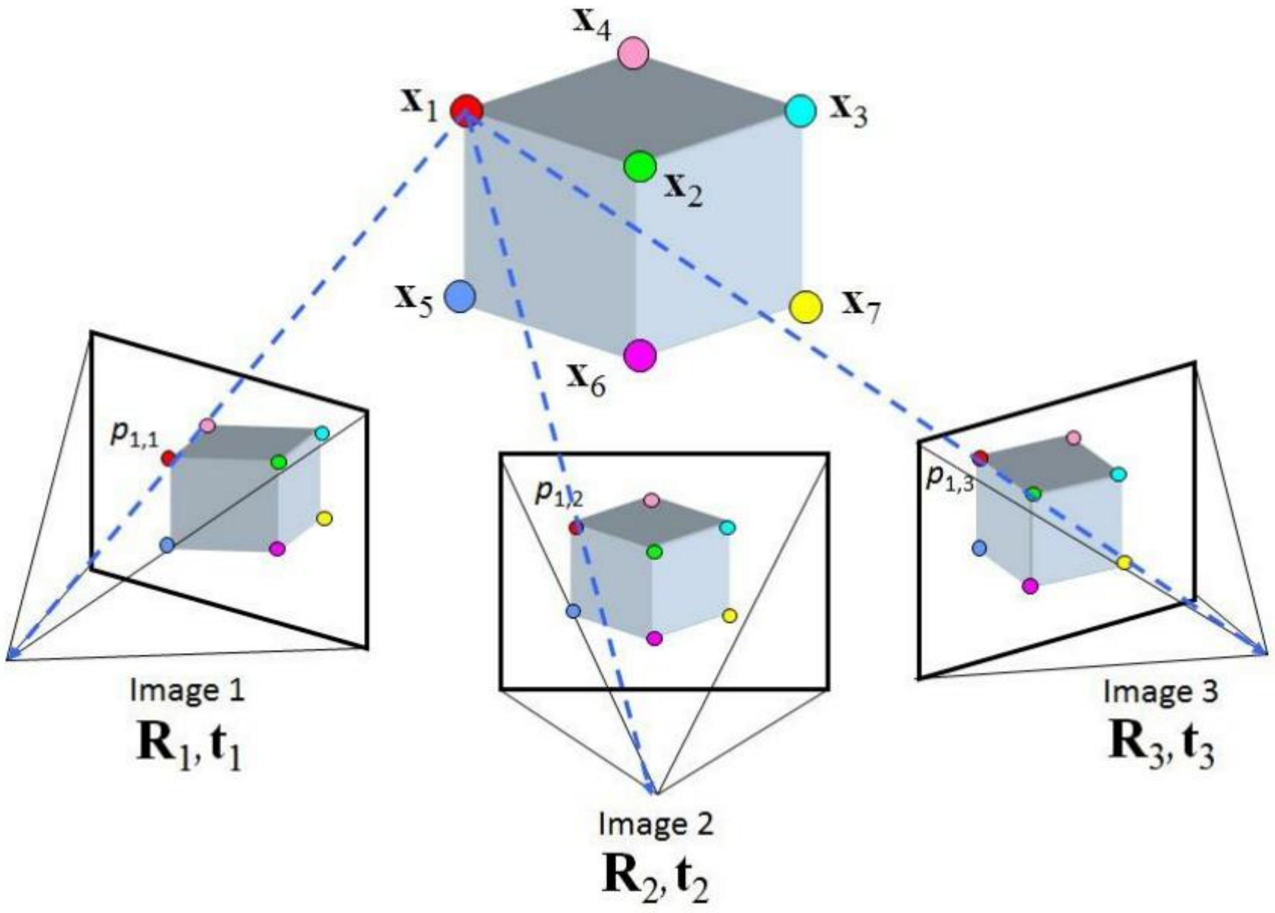
$$x_1 = K_1[R_1|T_1]\tilde{X}$$

$$x_2 = K_2[R_2|T_2]\tilde{X}$$

$$\downarrow$$

$$AX = 0$$

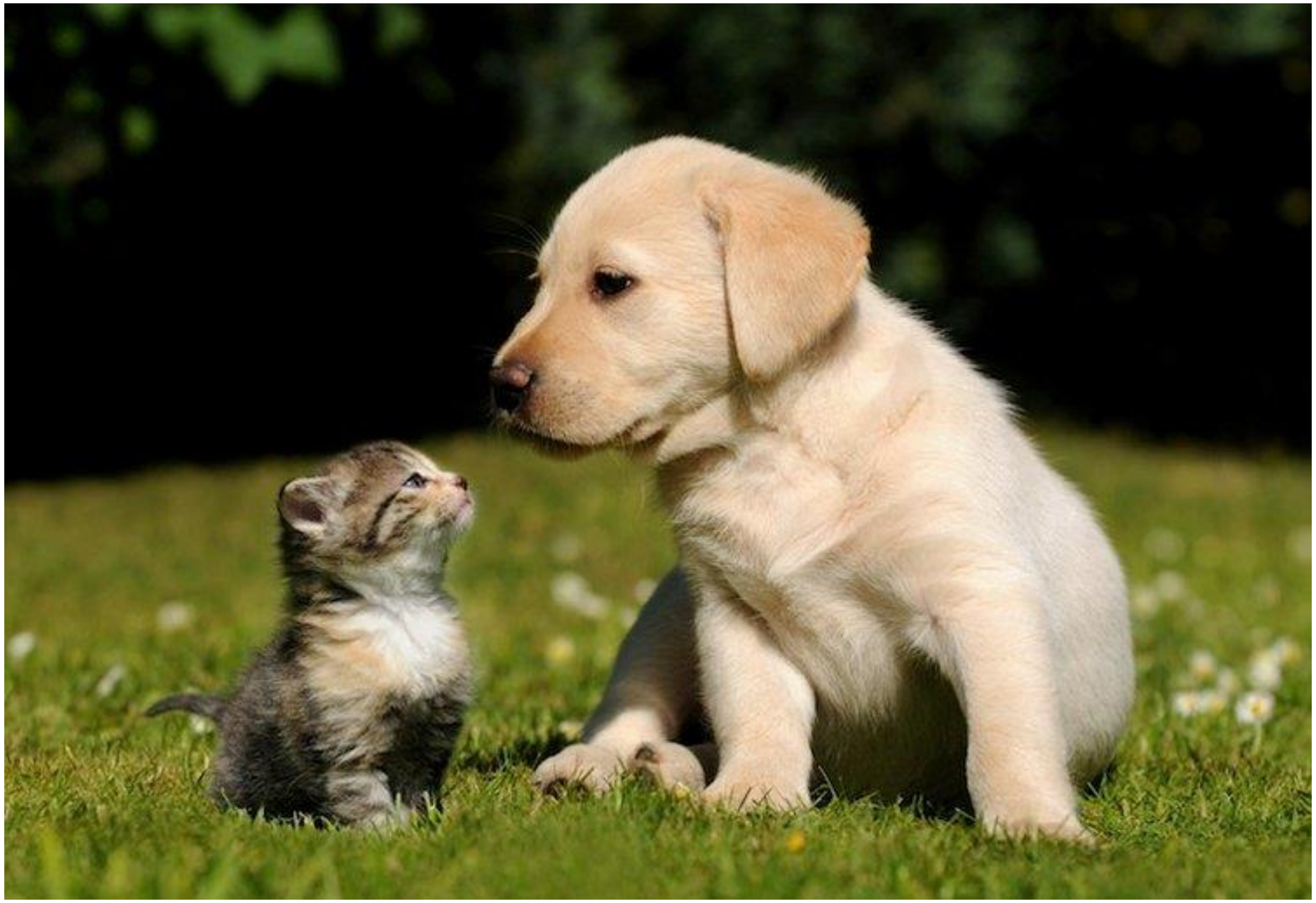
Homogenous system



## Probability and Information Theory

## Why probability?

Most real problem is not deterministic.



Which is this picture about Dog or Cat?

## Three possible sources of uncertainty

- Inherent stochasticity in the system

  e.g., Randomly shuffled card

- Incomplete observability

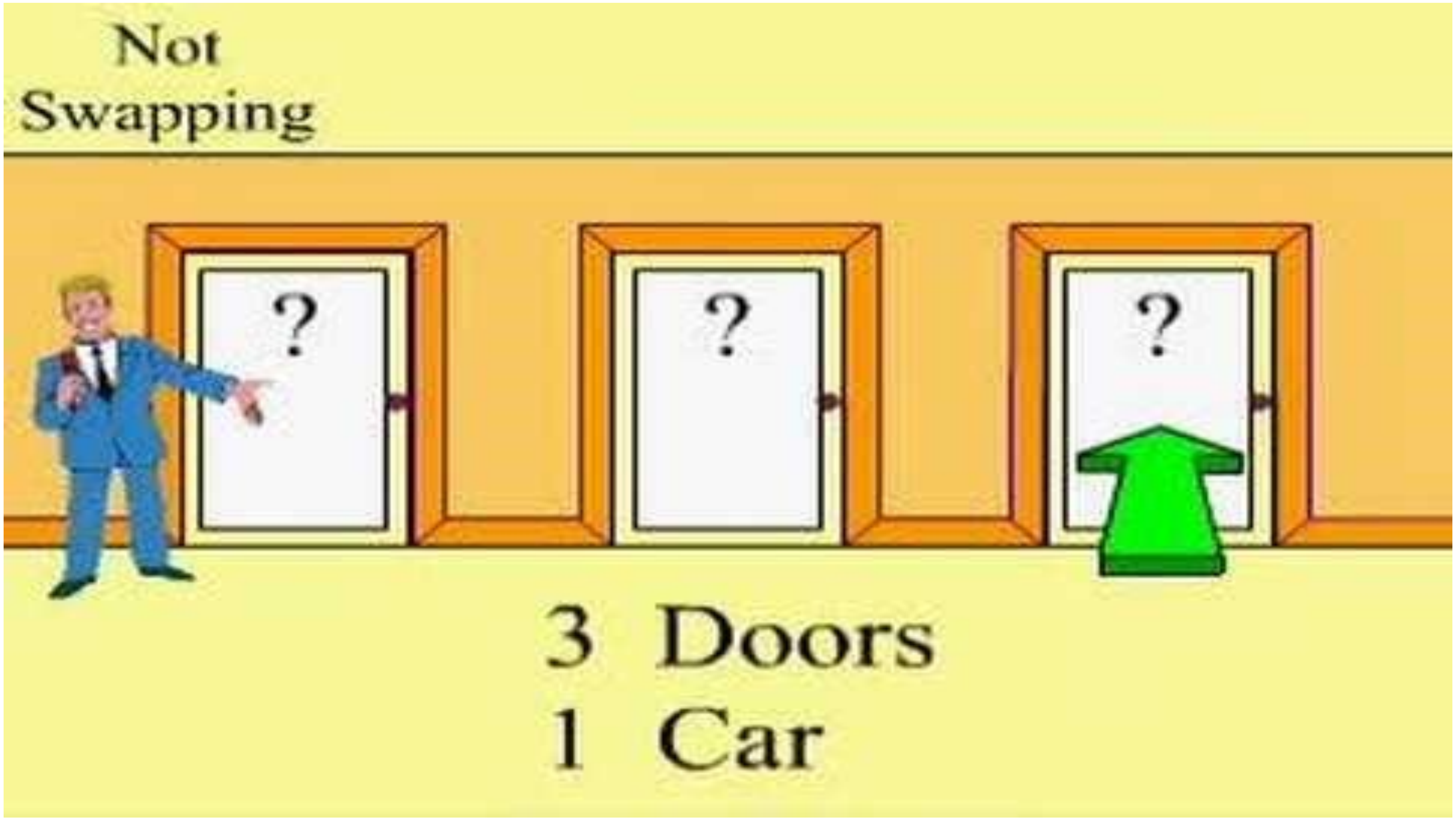
  e.g., Monty Hall problem

- Incomplete modeling

  e.g., A robot that only sees the discretized space

- A ***random variable*** is a variable that can take on different values randomly. e.g., $x \in \mathrm{x}$

# Monty Hall problem



## Discrete case: Probability Mass Function

$$P(x)$$

- The domain of *P* must be the set of all possible states of *x*
  - $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$. An impossible event has probability 0 and no state can be less probable than that. Likewise, an event that is guaranteed to happen has probability 1, and no state can have a greater chance of occurring
  - $\sum_{x\in \mathrm{x}} P(x) = 1$. We refer to this property as being ***normalized***. Without this property, we could obtain probabilities greater than one by computing the probability of one of many events occurring

- Example
  - Dice : P(*x*)=1/6, *x* is an event where f(*x*)=1,2,3,4,5,6

## Continuous case: Probability Density Function

$$p(x)$$

- The domain of $p$ must be the set of all possible states of x
  - $\forall x \in \mathrm{x}, p(x) \geq 0$. Note that we do not require $p(x) \leq 1$
  - $\int p(x)dx = 1$
  - <u>Does not give the probability of a specific state directly</u>

e.g., p(0.0001) + p(0.0002) +….. >= 100%!

- Example
  - What is the probability that randomly selected value within [0,1] is more than 0.5?

# Marginal Probability

- The probability distribution over the subset
  - $\forall x \in \mathrm{x}, \mathrm{P}(x) = \sum_y P(x,y)$ (Discrete)
  - $p(x) = \int p(x,y)dy$ (Continuous)

Table: The statistics about the student

| | Male | Female |
|---|---|---|
| Tokyo | 0.4 | 0.3 |
| Outside Tokyo | 0.1 | 0.2 |

Q. How often students come from Tokyo?

## Conditional Probability

- The probability of an event, given that some other event has happened
  - $P(y|x) = P(y,x)/P(x)$
  - $P(y,x) = P(y|x)P(x)$
- Example: The boy and girl problem

Mr. Jones has two children. One is a girl. What is the probability that the other is a boy?

- Each child is either male or female.
- Each child has the same chance of being male as of being female.
- The sex of each child is independent of the sex of the other.

## Conditional Probability

[Boy/Boy, Boy/Girl, Girl/Boy, Girl/Girl]

- $P(y)$: The probability that "The other is a boy"
- $P(x)$: The probability that "One is a girl"
- $P(y|x)$: The probability that "The other is a boy" when "One is a girl"
- $P(y,x)$: The probability that "One is a girl, the other is a boy"

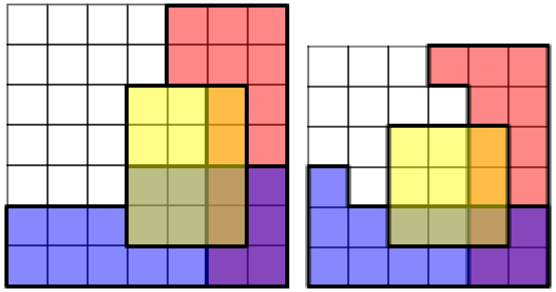
$$P(y|x) = \frac{P(y,x)}{P(x)} = \frac{2/4}{3/4} = \frac{2}{3}$$

## Chain rule of Probability

- $P\left(x^{(1)},\ldots,x^{(n)}\right)=P\left(x^{(1)}\right)\Pi_{i=2}^{n}P\left(x^{(i)}|x^{(1)},\ldots,x^{(i-1)}\right)$
- $P(a,b,c)=P(a|b,c)P(b,c)$
- $P(b,c)=P(b|c)P(c)$
- $P(a,b,c)=P(a|b,c)P(b|c)P(c)$

# Independence and Conditional Independence

- The random variables x and y are ***independent*** if their probability distribution can be expressed as a product of two independent factors
  - $\forall x \in \mathrm{x}, y \in \mathrm{y},\ p(x, y) = p(x)p(y)$
- Two random variables x and y are ***conditionally independent*** given random variable z if the conditional probability distribution over x and y factorizes in this way for every value of z
  - $\forall x \in \mathrm{x}, y \in \mathrm{y}, z \in \mathrm{z}, p(x, y|z) = p(x|z)p(y|z)$



$P(R_{ed} \cap B_{lue}|Y_{ellow}) = P(R|Y)P(B|Y)$

## Expectation, Variance and Covariance

- The ***expectation*** of some function f(*x*) with respect to P(*x*) or p(*x*) is mean value that f takes on when *x* is drawn from P or p

  - $\mathrm{E}_{x\to P}[f(x)] = \sum_x P(x)f(x)$
  - $\mathrm{E}_{x\to p}[f(x)] = \int p(x)f(x)dx$

- The ***variance*** gives how much the values of a function of a random variable *x* vary as we sample different values of *x* from its probability distribution

  - $\mathrm{Var}[f(x)] = \mathrm{E}[(f(x) - \mathrm{E}[f(x)])^2]$

- The ***covariance*** gives some sense of how much two values are linearly related to each other, as well as the scale of these variables:

  - $\mathrm{Cov}[f(x), g(y)] = \mathrm{E}[(f(x) - \mathrm{E}[f(x)])(g(y) - \mathrm{E}[g(y)])]$
  - $\mathrm{Cov}(\boldsymbol{x})_{i,j} = \mathrm{Cov}(x_i, x_j)$ Covariance matrix for $n \times n$ matrix

# Common Probability Distribution (1)

- Bernoulli distribution

$$P(1) = \Phi \qquad P(0) = 1 - \Phi \qquad P(x) = \phi^x(1-\Phi)^{1-x}$$

- Gaussian (Normal) distribution

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

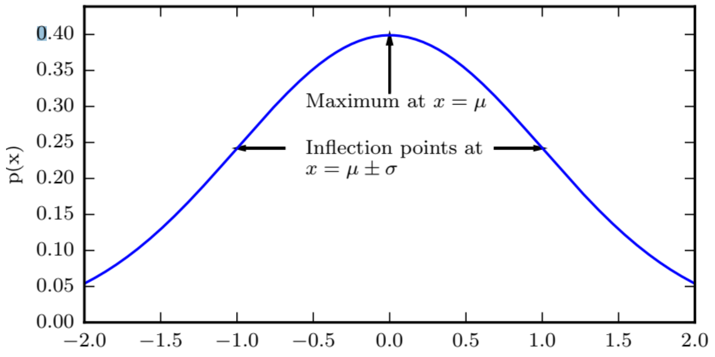


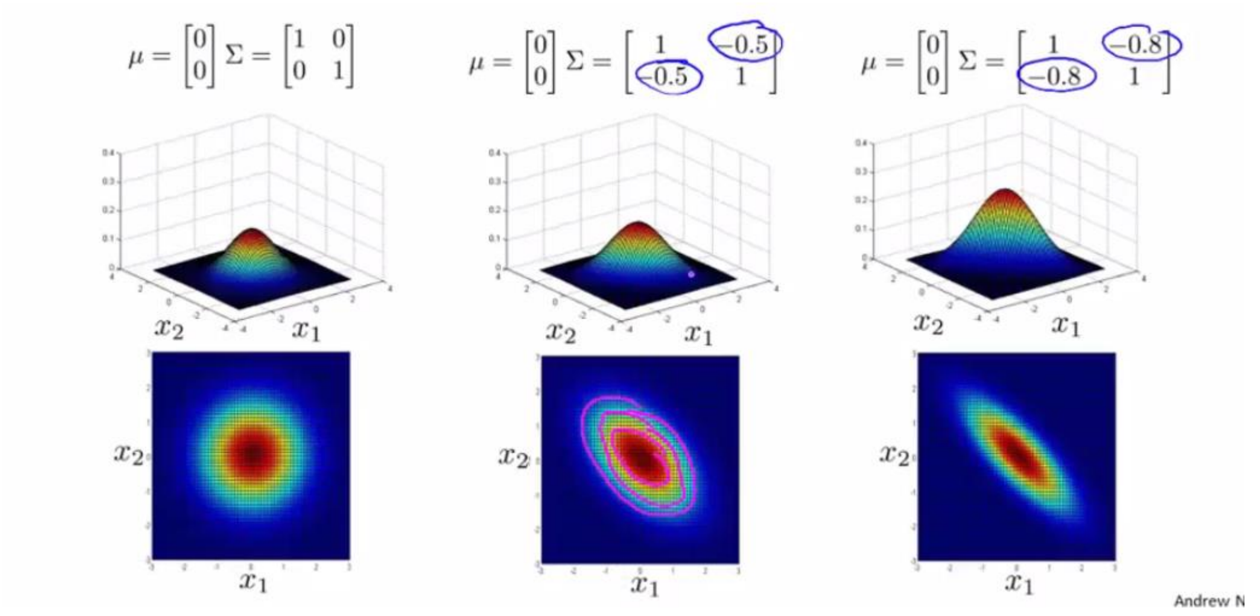
Figure 3.1

**The central limit theorem:**
The sum of many independent random variables is approximately normally distributed

# Common Probability Distribution (2)

- Multivariate normal distribution

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



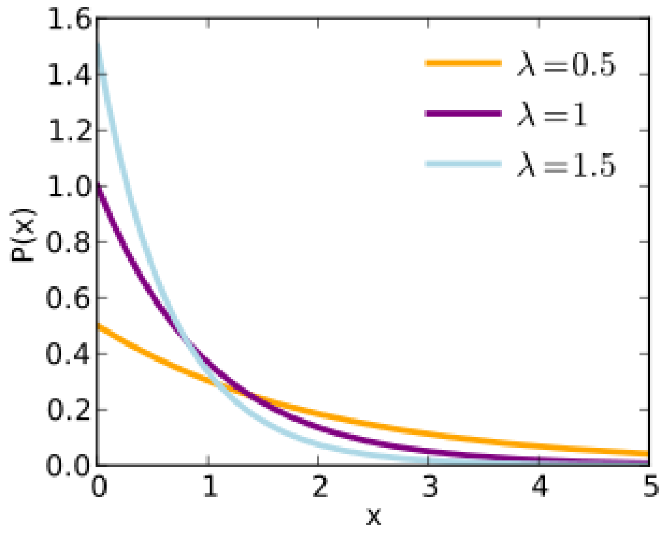
https://notesonml.wordpress.com/2015/06/30/chapter-14-anomaly-detection-part-2-multivariate-gaussian-distribution/

# Common Probability Distribution (3)

- Exponential distribution

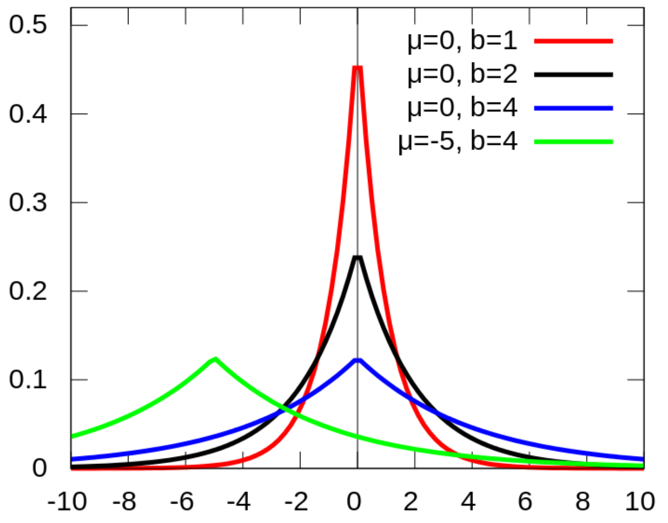
$$p(x;\lambda) = \lambda \mathbf{1}_{x\geq 0} \exp(-\lambda x)$$

$\mathbf{1}_{x\geq 0}$ assign zero to negative values of x



- Laplace distribution

$$\text{Laplace}(x;\mu,\gamma) = \frac{1}{2\lambda}\exp(-\frac{|x-\mu|}{\gamma})$$



# Mixtures of Distributions

- Empirical Distribution

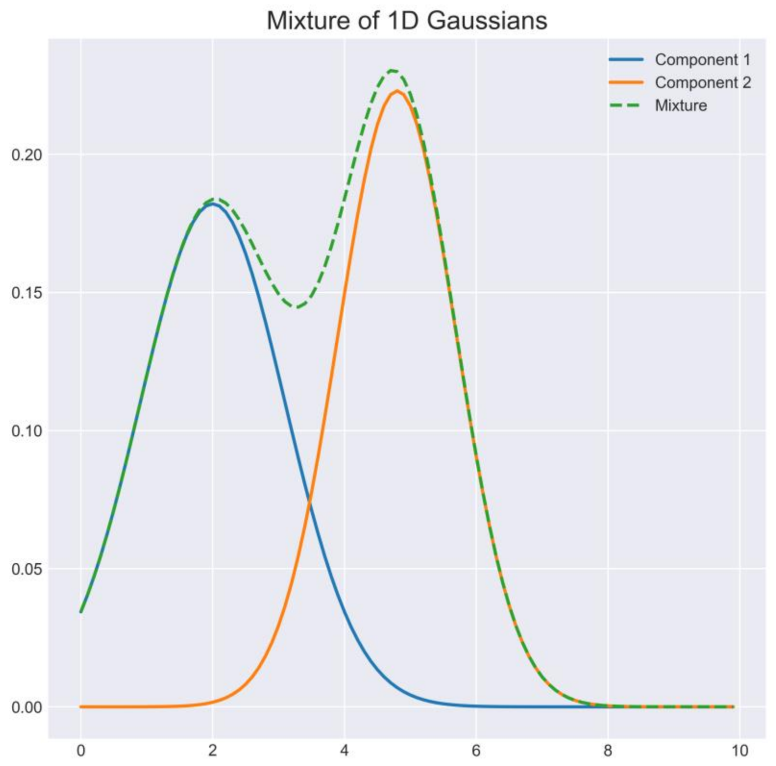
$$p(x) = \frac{1}{m}\sum \delta\left(x - x^{(i)}\right)$$

  - $\delta$ is a ***dirac delta function***

- Gaussian Mixture Model

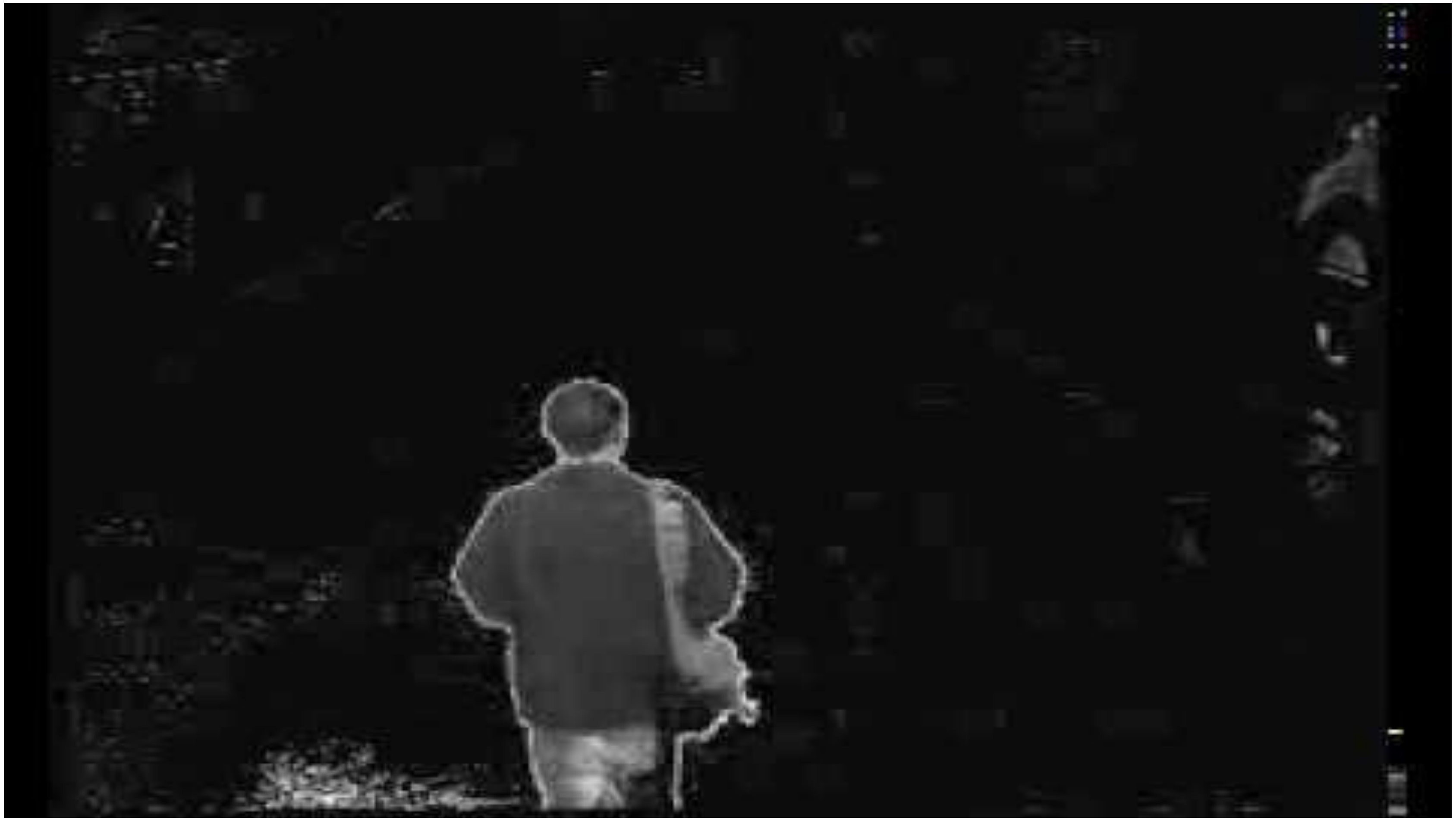
$$p(x) = \sum_i \phi_i N\left(x|\mu_i, \sigma_i^2\right)$$

$\phi_i$: latent variable (weight of gaussian)



  - GMM is a universal approximator of densities of a distribution

# Application of GMM in Computer Vision



Background subtraction by GMM

https://www.youtube.com/watch?v=KGal_NvwI7A

# Useful Properties of Common Functions

- Logistic Sigmoid

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

  - Good to produce [0,1] random values

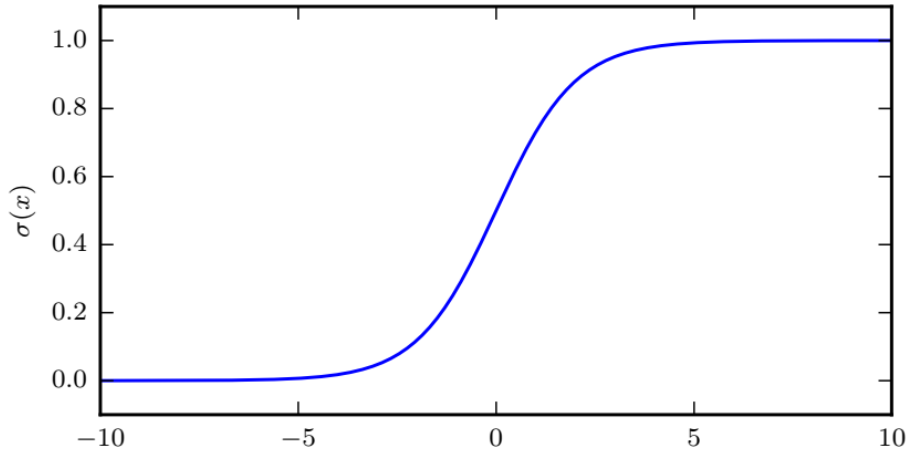


Figure 3.3: The logistic sigmoid function.

- Softplus function

$$\varsigma(x) = \log(1 + \exp(-x))$$

  - Good to prodce [0,∞] random values

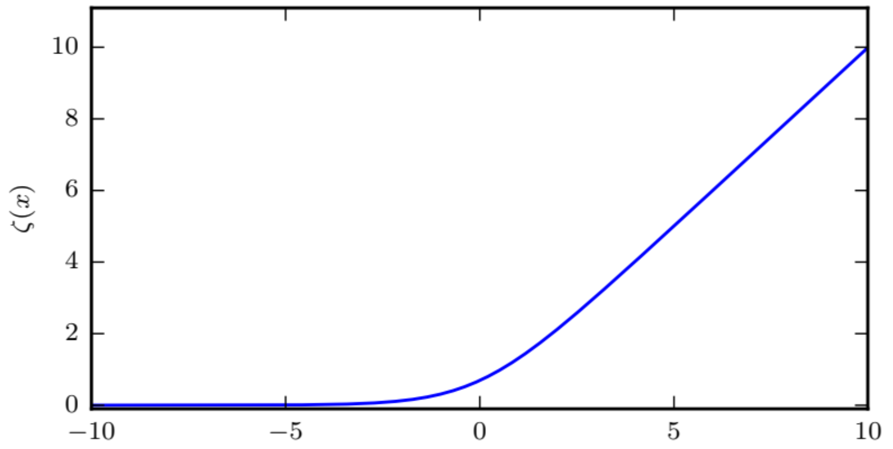


Figure 3.4: The softplus function.

# Bayes' Rule

***Prior probability*** ***likelihood***

$$P(x|y) = \frac{P(x)P(y|x)}{P(y)} = \frac{P(x,y)}{P(y)}$$

***Posterior probability***

***Prior probability***

## Factory Problem

The entire output of a factory is produced on three machines. The three machines account for 20%, 30%, and 50% of the factory output. The fraction of defective items produced is 5% for the first machine; 3% for the second machine; and 1% for the third machine. If an item is chosen at random from the total output and is found to be defective, what is the probability that it was produced by the third machine?

# Factory Problem

The entire output of a factory is produced on three machines. The three machines account for 20%, 30%, and 50% of the factory output. The fraction of defective items produced is 5% for the first machine; 3% for the second machine; and 1% for the third machine. If an item is chosen at random from the total output and is found to be defective, what is the probability that it was produced by the third machine?

$$P(X_A) = 0.2, P(X_B) = 0.3, P(X_C) = 0.5$$

$$P(Y|X_A) = 0.05, P(Y|X_B) = 0.03, P(Y|X_C) = 0.01$$

$$P(\mathrm{Y}) = P(Y|X_A)P(X_A) + P(Y|X_B)P(X_B) + P(Y|X_C)P(X_C)$$

$$P(X_C|\mathrm{Y}) = \frac{P(X_C)P(Y|X_C)}{P(Y)} = 5/24$$

# Fundamentals of Media Processing

Lecturer:
池畑　諭（Prof. IKEHATA　Satoshi）
児玉　和也（Prof. KODAMA Kazuya）

Support:
佐藤　真一（Prof. SATO　Shinichi）
孟　洋（Prof. MO　Hiroshi）

10/16 (Today) Introduction Chap. 1

## Basic of Machine Learning (Maybe for beginners)

10/23 Basic mathematics (1) (Linear algebra, probability, numerical computation) Chap. 2,3,4

10/30 Basic mathematics (2) (Linear algebra, probability, numerical computation) Chap. 2,3,4

11/6 Machine Learning Basics (1) Chap. 5

11/13 Machine Learning Basics (2) Chap. 5

## Basic of Deep Learning

11/20 Deep Feedforward Networks Chap. 6

11/27 Regularization and Deep Learning Chap. 7

12/4 Optimization for Training Deep Models Chap. 8

## CNN and its Application

12/11 Convolutional Neural Networks and Its Application (1) Chap. 9 and more

12/18 Convolutional Neural Networks and Its Application (2) Chap. 9 and more

## Last Week

- Linear Algebra
  - Matrix Decomposition (eigendecomposition, SVD)
  - Solving linear systems

- Probability
  - Random variable $x$, probability mass function $P(x)$, probability density function $p(x)$
  - Marginal probability, conditional probability
  - Expectation, variance, covariance
  - Common probabilistic distribution (e.g., Normal distribution, Laplace distribution)
  - Mixture of Gaussian
  - Bays' rule

# Information Theory

Information theory studies the quantification, storage, and communication of information. It was originally proposed by Claude E. Shannon in 1948 to find fundamental limits on signal processing and communication operations such as data compression, in a landmark paper entitled "A Mathematical Theory of Communication".

A key measure in information theory is ***entropy***. Entropy quantifies the amount of uncertainty involved in the value of a random variable or the outcome of a random process. Some other important measures in information theory are mutual information, channel capacity, error exponents, and relative entropy.

The field is at the intersection of mathematics, statistics, computer science, physics, neurobiology, information engineering, and electrical engineering. The theory has also found applications in other areas, including statistical inference, natural language processing, cryptography, neurobiology, human vision, the evolution and function of molecular codes (bioinformatics), model selection in statistics, thermal physics, quantum computing, linguistics, plagiarism detection, pattern recognition, and anomaly detection.

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

## Information Theory

- ***Self-information*** (for single outcome)
  - Likely event has low information, less likely event has higher information

$$I(x) = -\log P(x)$$

In units of ***nats*** or ***bits***: amount of information gained by observing an event of probability 1/e or 1/2

For example, identifying the outcome of a fair coin flip (with two equally likely outcomes) provides less information (lower entropy) than specifying the outcome from a roll of a die (with six equally likely outcomes).

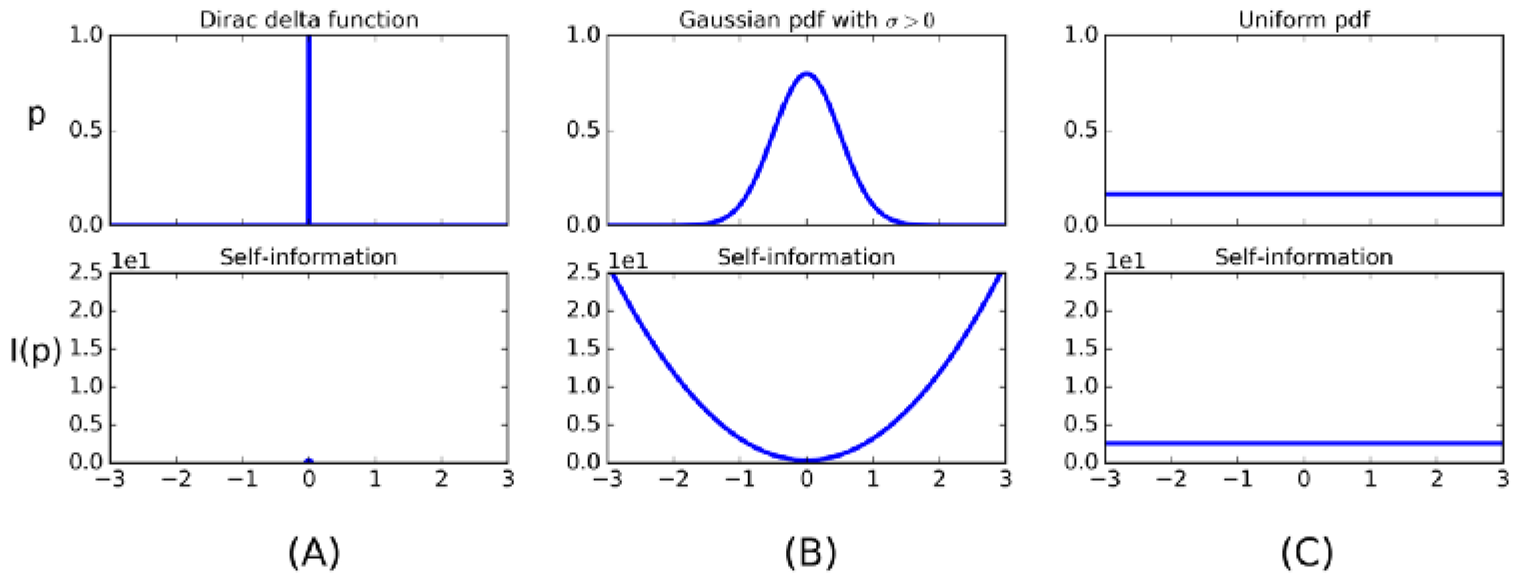
- ***Shanon entropy*** (amount of uncertainty in an entire probability distribution)

$$H(x) = E_{x\sim P}[I(x)] = -E_{x\sim P}[\log P(x)]$$

  - Known as differential entropy for p(x)

# Example

$$H(x) = E_{x\sim P}[I(x)] = -\sum_{i=1}^{m} p_i \log p_i$$



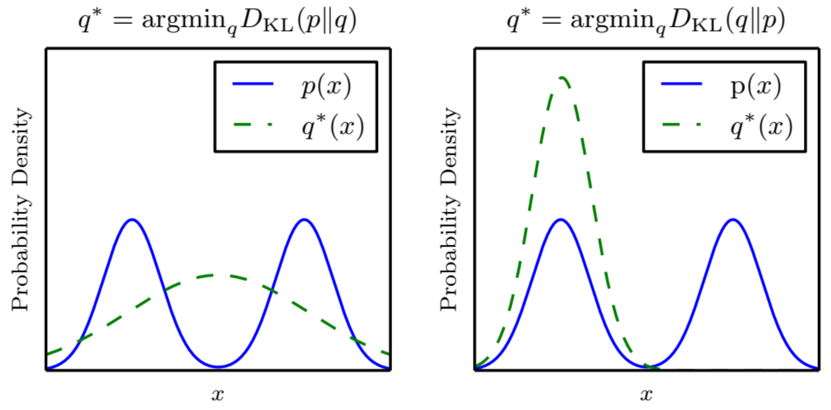
0 (Dirac delta), 174 (Gaussian), and 431 (uniform).

https://medium.com/swlh/shannon-entropy-in-the-context-of-machine-learning-and-ai-24aee2709e32

# Kullback-Leibler (KL) divergence

$$D_{KL}(P||Q) = E_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = E_{x\sim P}[\log P(x) - \log Q(x)]$$

- The difference of two distributions (higher is different)
  - KL divergence is positive or zero only when P and Q are the same distribution
  - Often used for model fitting (e.g., fitting GMM (Q(x)) on P(x)
  - Asymmetric measure ($D_{KL}(P||Q) \neq D_{KL}(Q||P)$)



$\min \int p(\log p(x) - \log q(x))$ Figure 3.6 $\min \int q(\log p(x) - \log q(x))$

Suppose we have a distribution $p(x)$ and want to approximate it with $q(x)$:

$D_{KL}(P||Q)$; $p$: high, $q$: high
$D_{KL}(Q||P)$; $p$:low, $q$:low

# Cross-entropy

- $H(P,Q) = H(P) + D_{KL}(P||Q) = -E_{x\sim P} \log Q(x)$

- The average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an "artificial" probability distribution Q, not true distribution P

- Minimizing the cross-entropy with respect to Q is equivalent to minimizing the KL divergence

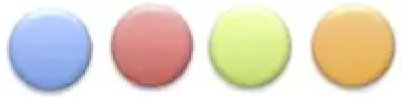
- In classification problems, the commonly used cross entropy loss, measures the cross entropy between the empirical distribution of the labels (given the inputs) and the distribution predicted by the classifier

# Example

Correct probability distribution (P(x))

50% 25% 12.5% 12.5%

Incorrect probability distribution (Q(x))



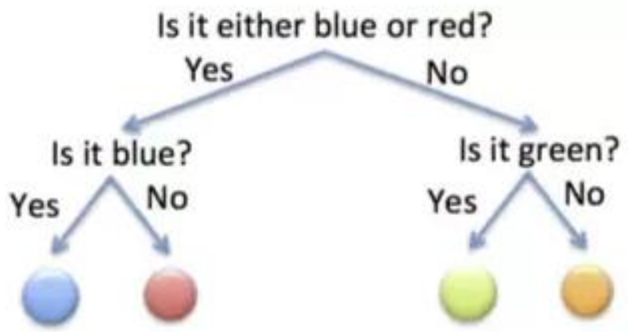
25% 25% 25% 25%

$$-E_{x\sim P}\log Q(x) = -\sum P\log Q =$$

Thus, cross entropy for a given strategy is simply the expected number of questions to guess the color under that strategy. For a given setup, the better the strategy is, the lower the cross entropy is. The lowest cross entropy is that of the optimal strategy, which is just the entropy defined above.

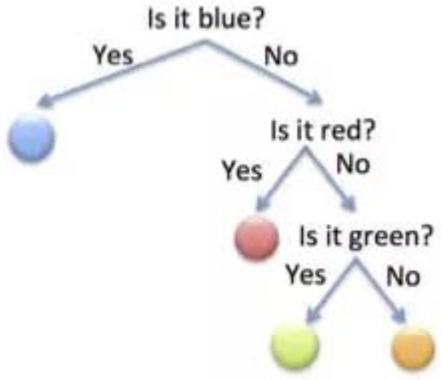
https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

Strategy 1



expected number of questions to guess the coin is 2.

Strategy 2

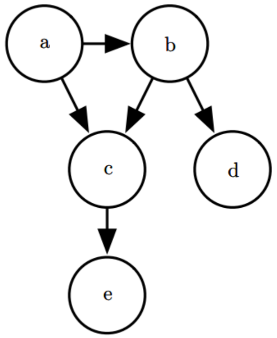


expected number of questions to guess the coin is 1.75<2.

# Structured Probabilistic Models

- In machine learning, it is inefficient to represent an entire probabilistic distribution in a single function. To reduce the parameter, the function is often factorized
- Suppose $a$ influence $b$, and $b$ influence $c$, but $a$ and $c$ are independent; $p(a,b,c) = p(a)p(b|a)p(c|b)$
- We can describe these factorizations using graphs (graphical model)

Figure 3.7



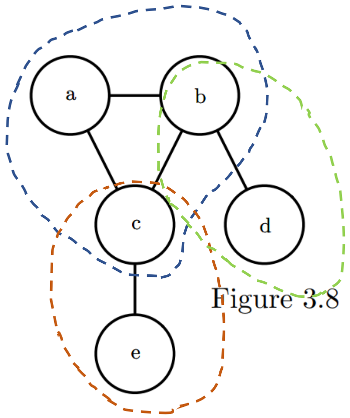


Figure 3.8

$\mathcal{G}$: graph

$\mathcal{C}^i$: clique

$\phi^{(i)}(\mathcal{C}^i)$: factor

- **Directed** model

$$p(a,b,c,d,e) = p(a)p(b|a)p(c|a,b)p(d|b)p(e|c)$$

- **Undirected** model (factor graph)

$$p(a,b,c,d,e) = \frac{1}{Z}\phi^{(1)}(\mathrm{a,b,c})\phi^{(2)}(b,d)\phi^{(3)}(c,e)$$

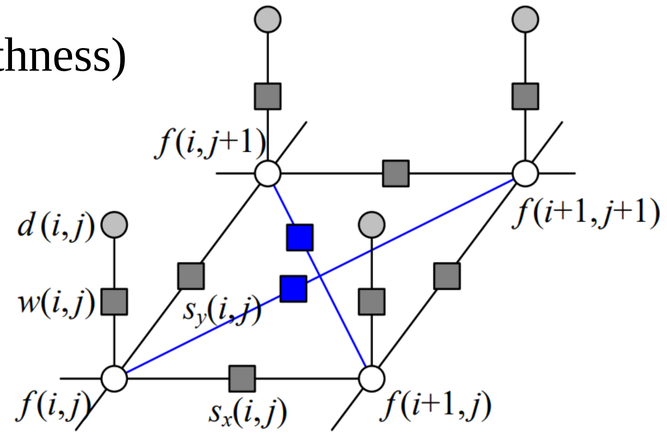
# Markov Random Field (MRF)

- In computer vision algorithm, the most common graphical model may be ***Markov Random Filed*** (MRF), whose log-likelihood can be described using local neighborhood interaction (or penalty) terms.
- MRF models can be defined over discrete variables, such as image labels (e.g., image restoration)

Likelihood term penalty term (pairwise smoothness)

$$E(\boldsymbol{x},\boldsymbol{y}) = E_d(\boldsymbol{x},\boldsymbol{y}) + E_p(\boldsymbol{x})$$

$$E_p(\boldsymbol{x}) = \sum_{\{(i,j),(k,l)\}\in\mathcal{N}} V_{i,j,k,l}(f(i,j), j(k,l))$$



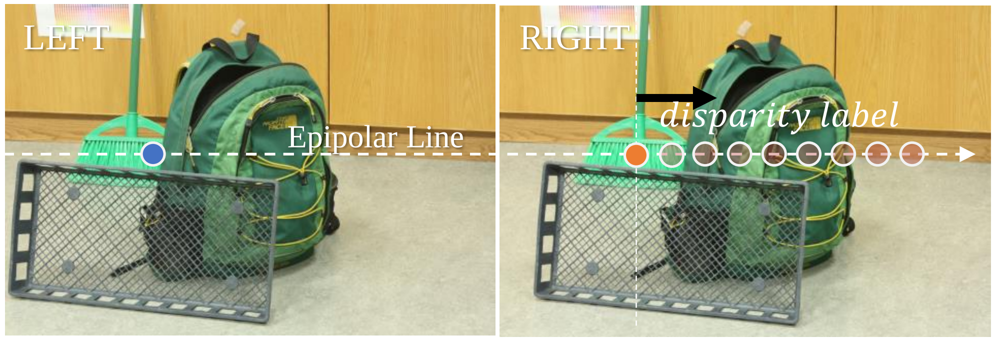
$\mathcal{N}_4$ and $\mathcal{N}_8$ neighborhood system

# STEREO MATCHING

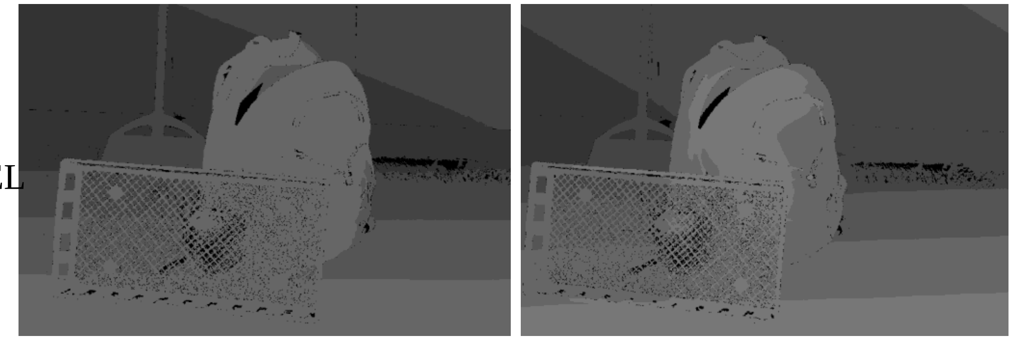
## STEREO as PIXEL-LABELING PROBLEM

Assign a disparity label to each pixel

INPUT

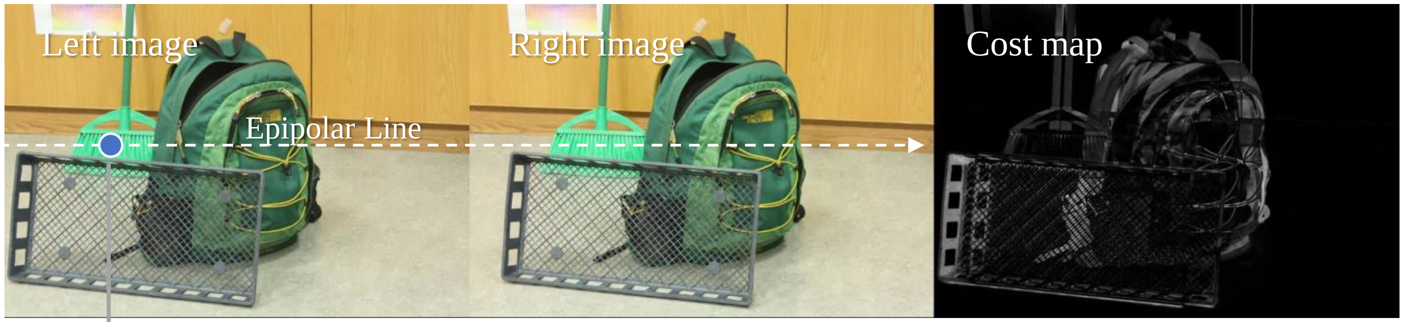


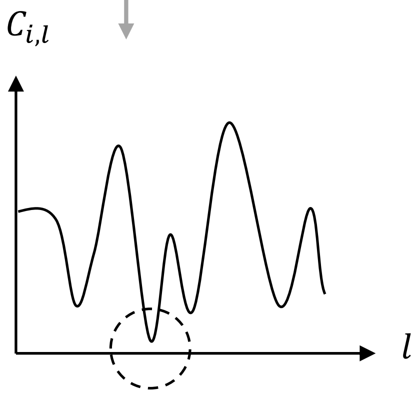
LABEL



# CLASSIC: WINNER-TAKES-ALL

Assign labels with lowest stereo matching cost





$$C_{i,l} = \|I_{i+l} - I_i\|$$

- Left and right color difference in RGB space
- Other matching cost: SSD, SAD, NCC,…

A series of “block matching”

# MRF Modeling For Stereo Matching

■ Markov Random Field (MRF)

- Many computer vision problems were formulated on the "graph"
- MRF: the graph structure where each node is only affected by its "neighbor"

## Pairwise term for "smoothness prior"

Unary

Pairwise

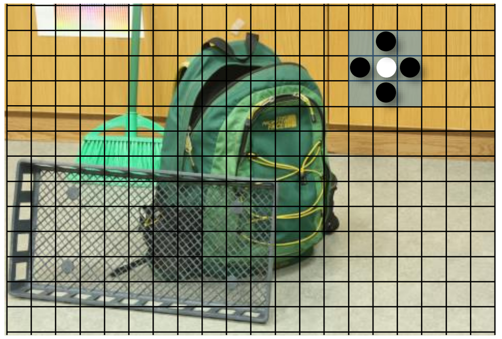
$$\min_{l} \sum_{i} C_{i,l} + \sum_{i} \sum_{j \in N(i)} S_{l_i, l_j}$$

$$C_{i,l} = \|I_{i+l} - I_i\| \qquad S_{l_i,l_j} = w_{ij}\left\|l_i - l_j\right\|$$

Left-right consistency

Label smoothness

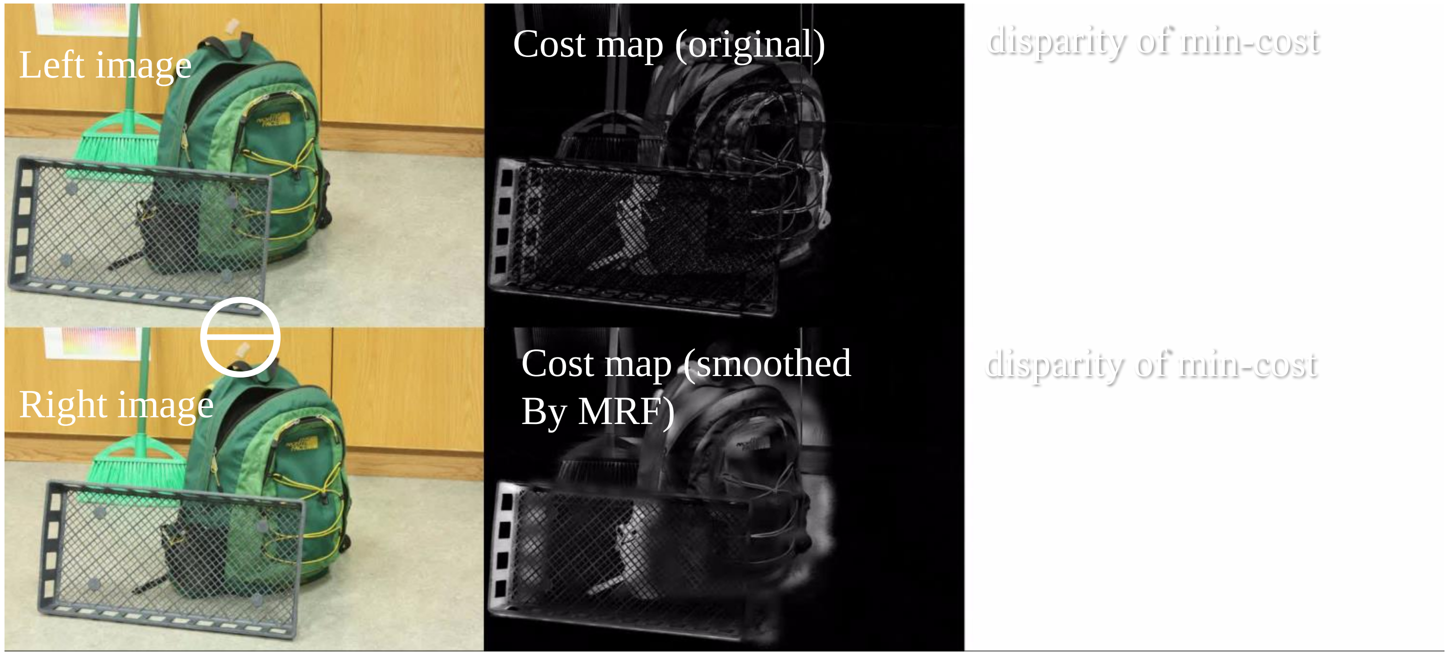
Image is a graph of uniform grid nodes



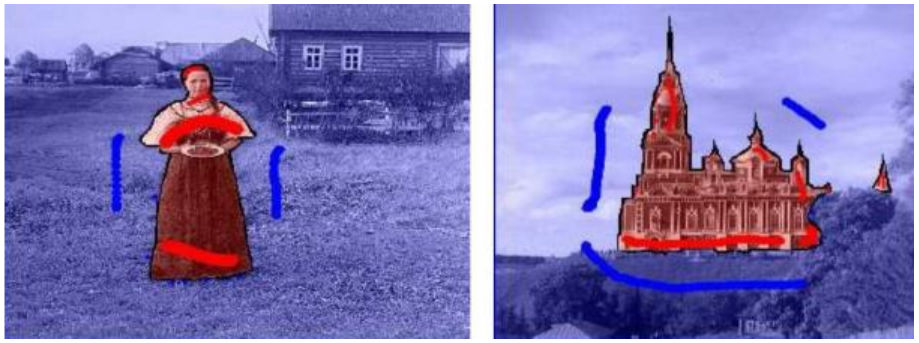
## Why MRF?: Convenient optimization methods are available

- Belief Propagation (BP), local optimum (Freeman2000, Sun2003)
- Graph Cuts (GC), global optimum (Kolmogorov and Zabih2001)

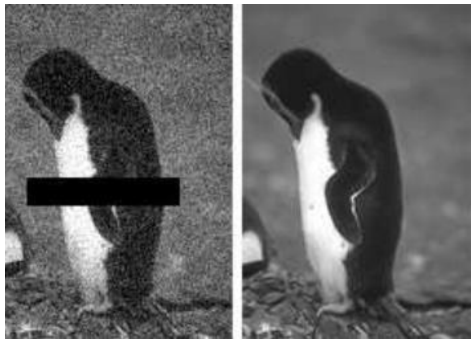
## WTA vs. MRF


Left image
Cost map (original)
disparity of min-cost
Right image
Cost map (smoothed By MRF)
disparity of min-cost

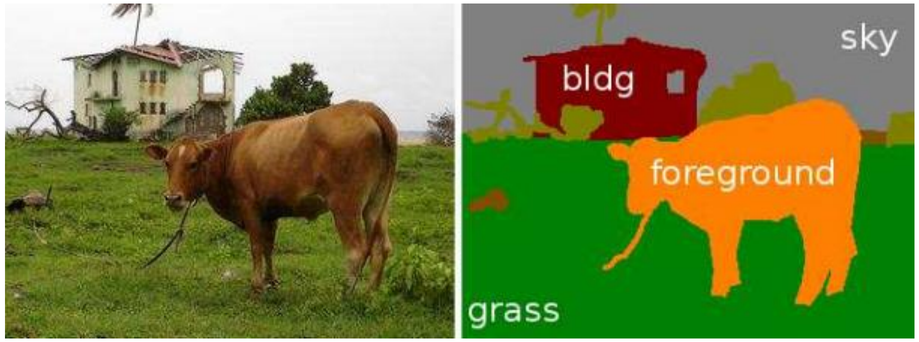
# Computer Vision LOVES MRF



Foreground / Background segmentation
(Boykov2006)



Denoising (0-255)
(Szeliski2008)



Semantic segmentation (He2004)

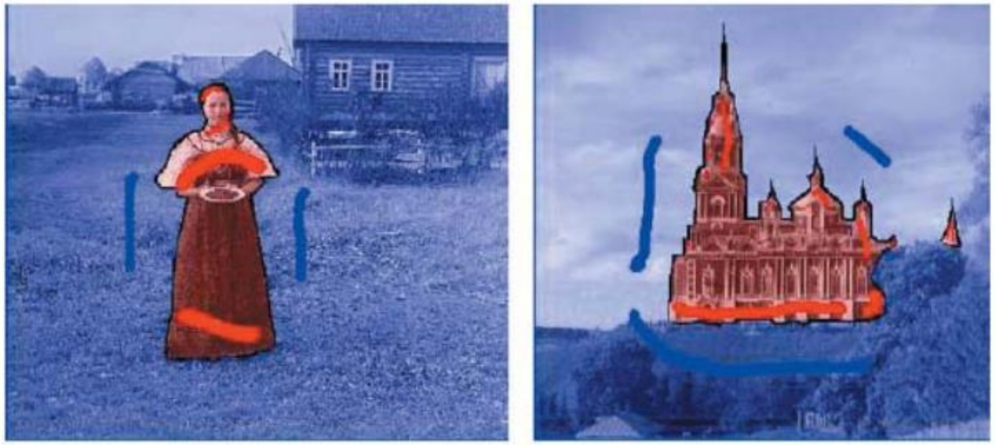
Accuracy is most important!
Better cost functions and
optimization techniques!

Multi-label MRF
High-order regularization

# Conditional Random Field (CRF)

- In classical Bayes model, prior p(x) is independent of the observation y. $p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$
- However, it is often helpful to update the prior probability based on the observation; the pairwise term depends on the y as well as x

$$E(\boldsymbol{x}|\boldsymbol{y}) = E_d(\boldsymbol{x},\boldsymbol{y}) + E_s(\boldsymbol{x},\boldsymbol{y}) = \sum_p V_p(\boldsymbol{x}_p,\boldsymbol{y}) + \sum_{p,q} V_{p,q}(\boldsymbol{x}_p,\boldsymbol{x}_q,\boldsymbol{y})$$

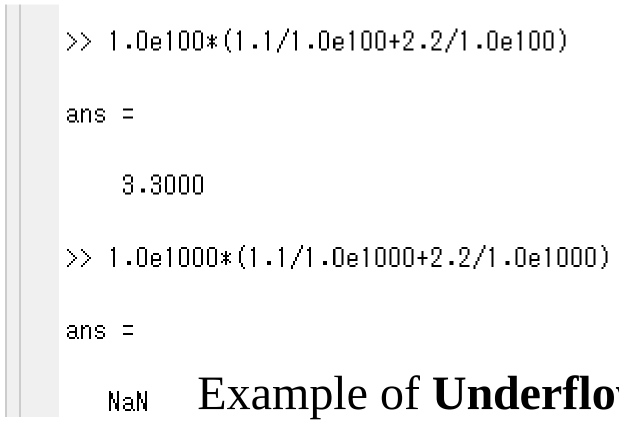


## Numerical Computation

# Numerical Concerns for Implementations of Deep Learning Algorithms

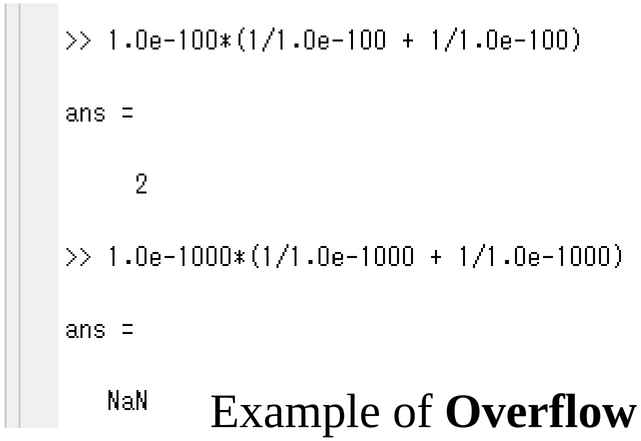
- Algorithms are often specified in terms of real numbers; real numbers cannot be implemented in a finite computer
  - Does the algorithm work when implemented with a finite number of bits?

- Do small changes in the input to a function cause large changes to an output?
  - Rounding errors, noise, measurement errors can cause large changes
  - Iterative search for best input is difficult



```
>> 1.0e100*(1.1/1.0e100+2.2/1.0e100)

ans =

    3.3000

>> 1.0e1000*(1.1/1.0e1000+2.2/1.0e1000)

ans =

   NaN
```

Example of **Underflow**



```
>> 1.0e-100*(1/1.0e-100 + 1/1.0e-100)

ans =

     2

>> 1.0e-1000*(1/1.0e-1000 + 1/1.0e-1000)

ans =

   NaN
```

Example of **Overflow**

## Poor Conditioning

- Conditioning refers to how rapidly a function changes with respect to small changes in its inputs

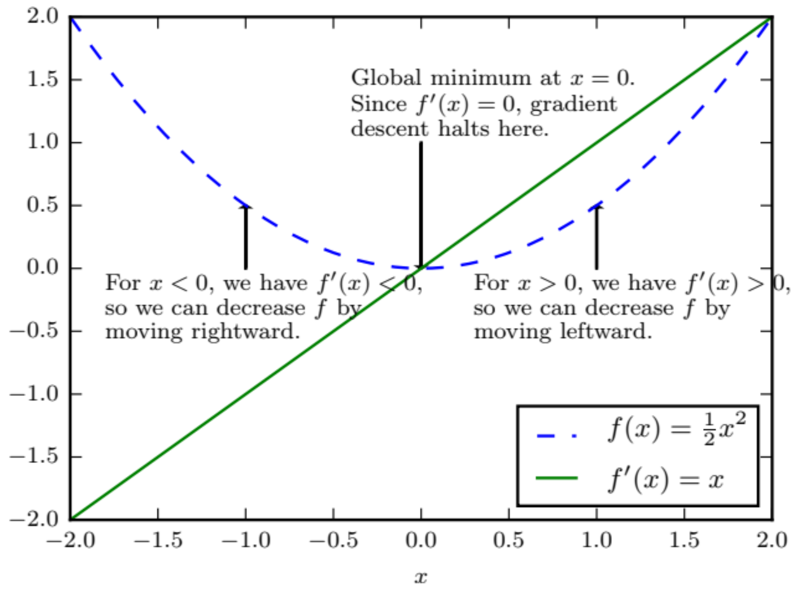
- We can evaluate the conditioning by a ***conditioning number***
  - The sensitivity is an intrinsic property of a function, not of computational error
  - For example, condition number for $f(\boldsymbol{x}) = A^{-1}\boldsymbol{x}$, where A is a positive semidefinite matrix, is

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right| ;\ \text{where } \lambda_s \text{ are eigenvalue of A}$$

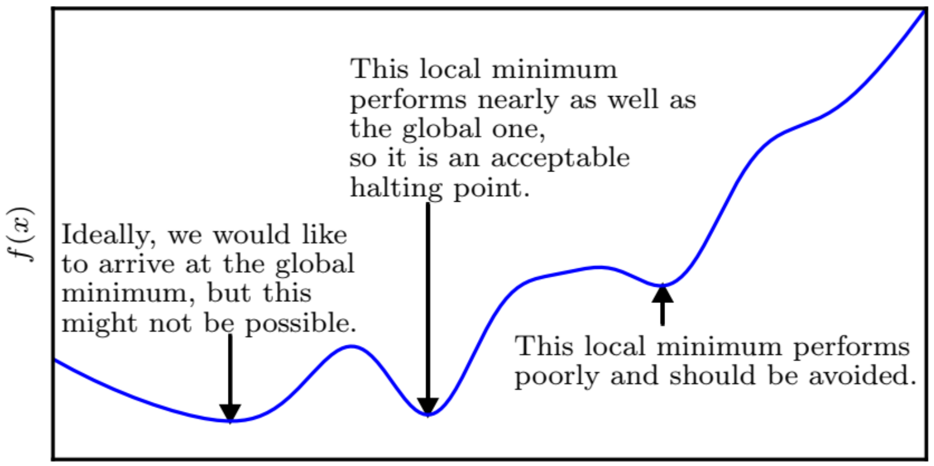
# Gradient-Based Optimization

- ***Objective function*:** the function we want to minimize
- May also call it criterion, cost function, loss function, error function
- $\boldsymbol{x}^* = \text{argmin} f(\boldsymbol{x})$
- The derivative of $f(x)$ is denoted as $f'(x)$ or $df/dx$

- The ***gradient descent*** is the technique to reduce $f(x)$ by moving $x$ in small steps with the opposite sign of the derivative
- Stationary points: local minima or maxima $f'(x) = 0$



Gradient descent

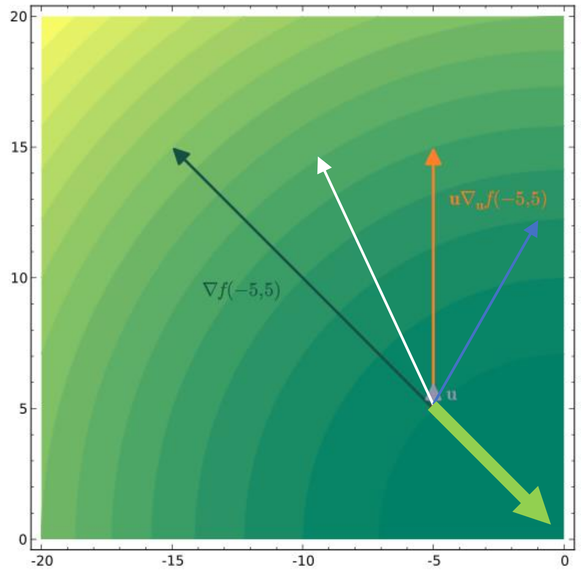


Local minima and global minimum

# Partial/Directional Derivatives for multiple inputs

$z = f(\boldsymbol{x})$ $\quad \dfrac{\partial f}{\partial x_i}$ $\quad \nabla_x f(\boldsymbol{x}) = \left[\dfrac{\partial f}{\partial x_1}, \cdots, \dfrac{\partial f}{\partial x_m}\right]$ Partial Derivatives

- The **directional derivatives** in direction $u$ is the slope of the function $f$ in direction $u$



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

To find the "steepest" direction,

$$\min_{\boldsymbol{u}} u^T \nabla_x f(\boldsymbol{x}) = \min_{\boldsymbol{u}} \|\boldsymbol{u}\|_2 \|\nabla_x f(\boldsymbol{x})\|_2 \cos\theta$$
$$\cong \min_{\theta} \cos\theta$$

$$\boldsymbol{u} \xleftrightarrow{\text{Opposite direction}} \nabla_x f(\boldsymbol{x})$$

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \epsilon \nabla_x f(\boldsymbol{x}^t)$$

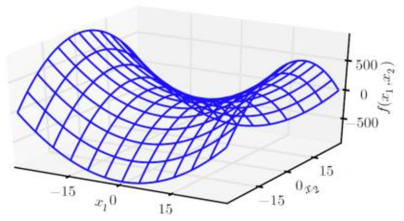
- Gradient descent for multiple inputs
- $\epsilon$ (learning rate) is fixed or adaptively selected (line search)

# Beyond the Gradient: Jacobian and Hessian Matrices

$\boldsymbol{f}: \mathbb{R}^m \rightarrow \mathbb{R}^n$ $$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}, \cdots, \frac{\partial f_1}{\partial x_m} \\ \vdots \\ \frac{\partial f_n}{\partial x_1}, \cdots, \frac{\partial f_n}{\partial x_m} \end{bmatrix}$$ Jacobian matrix

$$H(f)(\boldsymbol{x})_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$$ Hessian matrix

- When the function is continuous, $H(f)(\boldsymbol{x})_{ij} = H(f)(\boldsymbol{x})_{ji}$
- A real symmetric Hessian matrix has Eigendecomposition
  - When the Hessian is positive semidefinite, the point is local minimum
  - When the Hessian is negative semidefinite, the point is local maximum
  - Otherwise, the point is a **saddle** point



# Beyond the Gradient: Jacobian and Hessian Matrices

- The second derivative in a specific direction represented by a unit vector $\boldsymbol{d}$ is $\boldsymbol{d}^T H \boldsymbol{d}$

$$f(\boldsymbol{x}) \approx f\left(\boldsymbol{x}^{(0)}\right) + \left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)^T \boldsymbol{g} + \frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)^T H \left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right)$$

  - $\boldsymbol{x}^{(0)}$ is the current point, $\boldsymbol{g}$ is the gradient and $H$ is the Hessian at $\boldsymbol{x}^{(0)}$

- Then new point $\boldsymbol{x}$ will be given by $\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}$

$$f\left(\boldsymbol{x}^{(\mathbf{0})} - \epsilon \boldsymbol{g}\right) \approx f\left(\boldsymbol{x}^{(0)}\right) - \epsilon \boldsymbol{g}^T \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^T H \boldsymbol{g}$$

- When $\boldsymbol{g}^T H \boldsymbol{g}$ is positive, solving for the optimal learning rate that decreases the function is

$$\epsilon^* = \frac{g^T g}{g^T H g}$$

## Newton’s Method (Second-Order Algorithm)

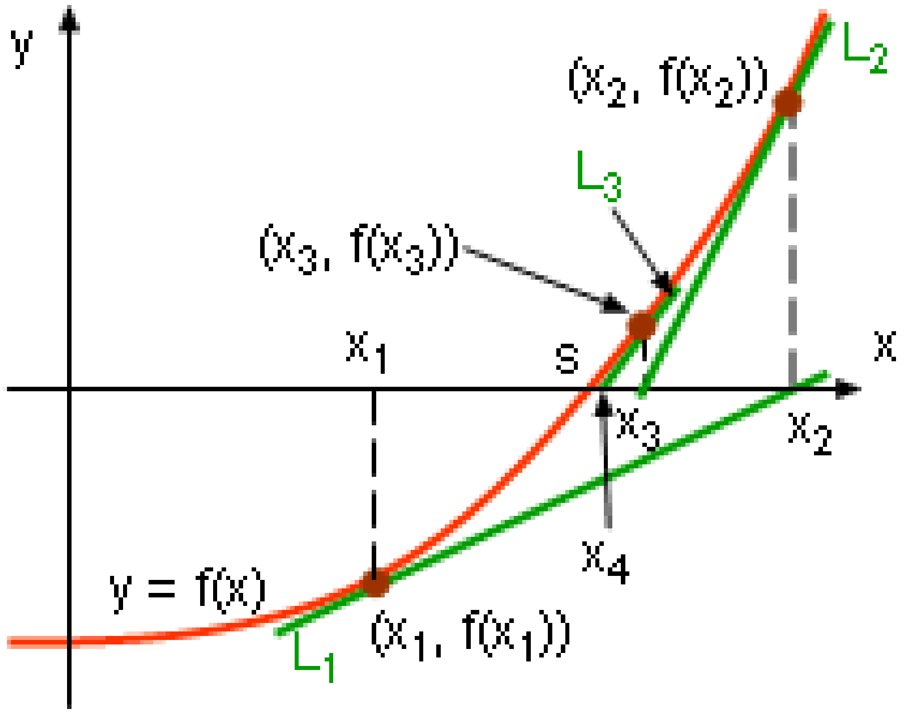
- In Gradient descent, the step size must be small enough
- ***Newton’s method*** is based on using 1st-order or 2nd-order Tyler Expansion

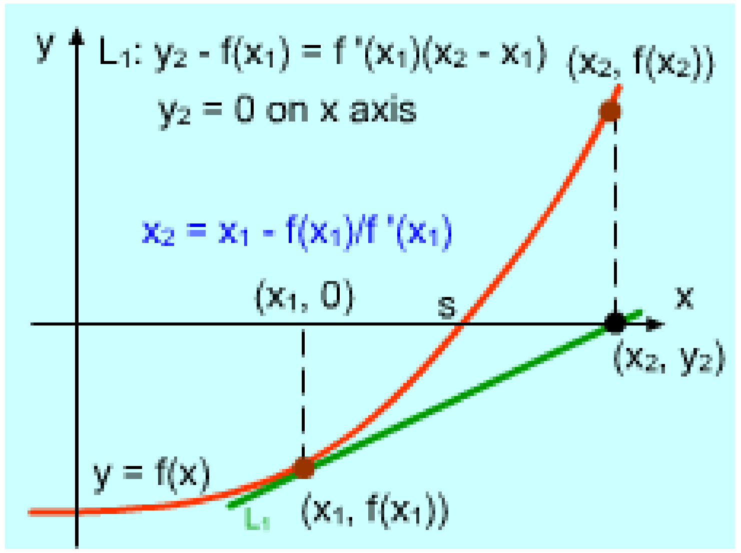
$$f(\boldsymbol{x}) \approx f\left(x^{(0)}\right) + \left(x - x^{(0)}\right)^T g + \frac{1}{2}\left(x - x^{(0)}\right)^T H\left(x - x^{(0)}\right)$$

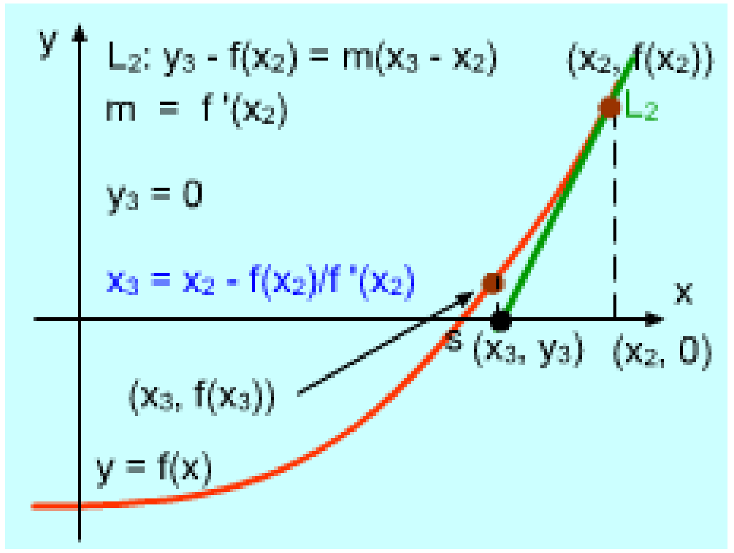
- The critical point ($\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$) is $\boldsymbol{x}^* = x^{(0)} - g^{-1}f \ (1^{st})$ or $x^{(0)} - H^{-1}g(2^{nd})$

- When *f* is a positive definite quadratic function, Newton’s method once to jump to the minimum of the function directly.
- When *f* is not truly quadratic but can be locally approximated as a positive definite quadratic, Newton’s method consists of applying multiple jumping
- Jumping to the minimum of the approximation can reach the critical point much faster than gradient descent would.

# Example (For univariate function: 1st order case)







# Constrained Optimization

- Constraint optimization problem is generally written as

$$\min_{\boldsymbol{x}} f \quad \text{s.t.} \quad \underbrace{g_i(\boldsymbol{x}) = 0,}_{\text{equality constraint}} \quad \underbrace{h_i(\boldsymbol{x}) \leq 0}_{\text{inequality constraint}}$$

- Karush-Kuhn-Tucker (KKT) Multiplier (Generalization of the Lagrange Multipliers)

$$\min_{x} \max_{\lambda_i} \max_{\mu_i \geq 0} L\left(x, \lambda_j, \mu_j\right)$$
$$= \min_{x} \max_{\lambda_i} \max_{\alpha \geq 0} \underbrace{f(x) + \sum \lambda_i g_i(x) + \sum \mu_i h_i(x)}_{\text{Generalized Lagrangian}}$$

# Karush-Kuhn-Tucker Condition

- ***KKT Conditions***: For a point to be optimal,

1. The gradient of the generalized Lagrangian is zero
2. All constraints on both *x* and the KKT multipliers are satisfied
3. The inequality constraints exhibit “complementary slackness”: $\alpha \odot h(x) = 0$

$$\begin{cases} \nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla g_i(\bar{x}) + \sum_{j=1}^{l} \bar{\mu}_j \nabla h_j(\bar{x}) = 0 \\ h_j(\bar{x}) = 0 \ \ (j = 1, \ldots, l) \\ \bar{\lambda}_i \geq 0, \ g_i(\bar{x}) \leq 0, \ \bar{\lambda}_i g_i(\bar{x}) = 0 \ \ (i = 1, \ldots, m) \end{cases}$$

## Example: Linear Least Squares

- Consider an unconstrained problem of:

$$f(\boldsymbol{x}) = \frac{1}{2}||A\boldsymbol{x} - \boldsymbol{b}||_2^2$$

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = A^T(A\boldsymbol{x} - \boldsymbol{b}) = A^T A\boldsymbol{x} - A^T\boldsymbol{b}$$

$$\boldsymbol{x}^{n+1} \leftarrow \boldsymbol{x}^n - \epsilon(A^T A\boldsymbol{x}^n - A^T\boldsymbol{b})$$ (Gradient Decent)

- If subjected to $\boldsymbol{x}^T\boldsymbol{x} \leq \mathbf{1}$

$$\min_{x,\lambda\geq 0} \mathrm{L}(\boldsymbol{x},\lambda) \quad \mathrm{L}(\boldsymbol{x},\lambda) = f(\boldsymbol{x}) - \lambda(\boldsymbol{x}^T\boldsymbol{x} - 1)$$

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x},\lambda) = A^T A\boldsymbol{x} - A^T\boldsymbol{b} + 2\lambda\boldsymbol{x} = 0 \quad \boldsymbol{x} = (A^T A + 2\lambda I)^{-1}A^T\boldsymbol{b}$$

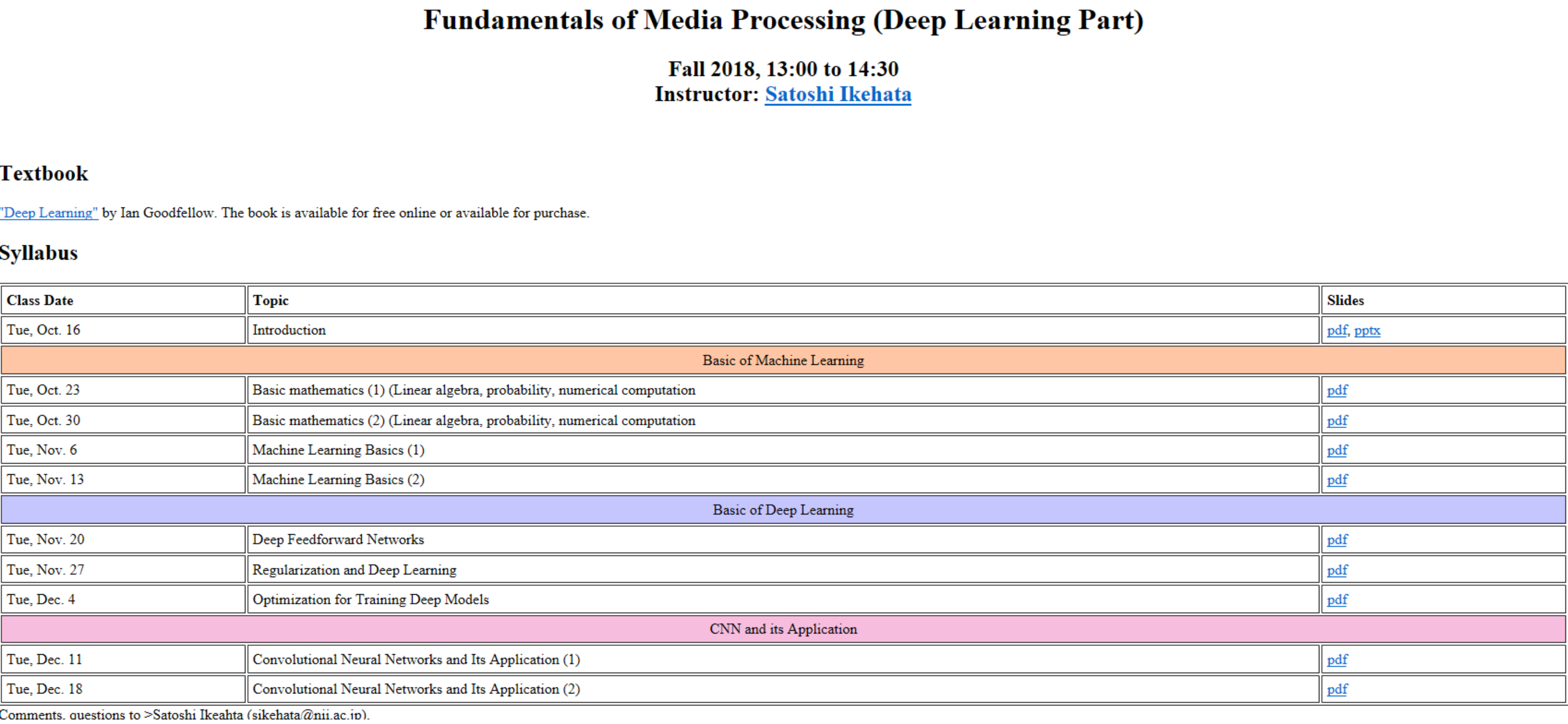
- The magnitude of $\lambda$ must obey the constraint:

  - Update $\lambda$ ($\geq 0$) until $\nabla_\lambda L(\boldsymbol{x},\lambda)$ becomes zero

$$\nabla_\lambda L(\boldsymbol{x},\lambda) = \boldsymbol{x}^T\boldsymbol{x} - 1$$

Class material is available at

https://satoshi-ikehata.github.io/mediaprocessing.html



## Fundamentals of Media Processing (Deep Learning Part)

**Fall 2018, 13:00 to 14:30**
**Instructor: Satoshi Ikehata**

### Textbook

"Deep Learning" by Ian Goodfellow. The book is available for free online or available for purchase.

### Syllabus

| Class Date | Topic | Slides |
|---|---|---|
| Tue, Oct. 16 | Introduction | pdf, pptx |
| | Basic of Machine Learning | |
| Tue, Oct. 23 | Basic mathematics (1) (Linear algebra, probability, numerical computation | pdf |
| Tue, Oct. 30 | Basic mathematics (2) (Linear algebra, probability, numerical computation | pdf |
| Tue, Nov. 6 | Machine Learning Basics (1) | pdf |
| Tue, Nov. 13 | Machine Learning Basics (2) | pdf |
| | Basic of Deep Learning | |
| Tue, Nov. 20 | Deep Feedforward Networks | pdf |
| Tue, Nov. 27 | Regularization and Deep Learning | pdf |
| Tue, Dec. 4 | Optimization for Training Deep Models | pdf |
| | CNN and its Application | |
| Tue, Dec. 11 | Convolutional Neural Networks and Its Application (1) | pdf |
| Tue, Dec. 18 | Convolutional Neural Networks and Its Application (2) | pdf |

Comments, questions to >Satoshi Ikeahta (sikehata@nii.ac.jp).