Mathematics Notes

Huu Duc Nguyen M.Sc.

27 April 2022

Contents

Abbreviations

RL	Reinforcement Learning
prob.	probability
params.	parameters
func.	function
pdf.	Probability Density Function
SVD	Singular Value Decomposition
K-L	Kullback–Leibler
IG	Information Gain
i.f.f.	if and only if
LP	Linear Programming
QP	Quadratic Programming
TSP	Travelling Salesman Problem

1 Matrix

1.1 Singular Value Decomposition

Singular Value Decomposition (SVD)

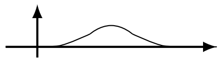
MLcoban.com

2 Probabilities

2.1 General

2.1.1 Basic Definitions

- If x is discrete: $\sum_x p(x) = 1$ with $\forall\ 0 \leq p(x) \leq 1$
- If x is continuous: $\int p(x)\, dx = 1 \Rightarrow \exists$ a **Probability Density Function** (**pdf.**)

  $p(x)$ can take any positive value, as long as $\int p(x)\, dx = 1$  **NOTE:**
  : theoretically $p(x) = 0, \forall x$
- Common types

Joint probability:	$p(x_i, y_i)$	$= p(X = x_i, Y = y_i)$
Marginal probability:	$p(x_i)$	$= p(X = x_i)$
Conditional probability:	$p(y_i \mid x_i)$	$= p(Y = y_i \mid X = x_i)$

- Sum rule: $\sum$ joint probability (prob.) = marginal prob.
  $\Rightarrow$ Marginalization
  - discrete variable: $p(x) = \sum_y p(x, y)$
  - continuous variable: $p(x) = \int p(x, y) dy$
- Product rule: Product of marginal prob. and conditional prob. = joint prob.

2.1.2 Expectation

For variable x:	$\mathbb{E}\left[x\right] = \sum_x x.p(x)$	$\left(= \int x.p(x)dx\right)$
For function $f(\cdot)$:	$\mathbb{E}\left[f(x)\right] = \sum_x f(x).p(x)$	$\left(= \int f(x).p(x)dx\right)$

2.1.3 Independence and Variability

- Independence. E.g.: x, y are independent, then

$$\begin{cases} p(x|y) = p(x) \\ p(y|x) = p(y) \end{cases} \Leftrightarrow p(x, y) = p(x).p(y)$$

- Variability

- variance: how much variability there is in $f(x)$ around its mean value $\mathbb{E}[f(x)]$
  $var[f] = \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2$
- covariance: for two random variables x, y
  $cov[x, y] = \mathbb{E}_{x,y}[xy] - \mathbb{E}[x].\mathbb{E}[y]$
- covariance matrix: if x, y are vectors

$$\begin{aligned} cov[\mathbf{x}, \mathbf{y}] &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\{\mathbf{x} - \mathbb{E}[\mathbf{x}]\}\left\{\mathbf{y}^T - \mathbb{E}\left[\mathbf{y}^T\right]\right\}\right] \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\mathbf{x}\mathbf{y}^T\right] - \mathbb{E}[\mathbf{x}].\mathbb{E}\left[\mathbf{y}^T\right] \end{aligned}$$

2.1.4 Bayes Rule

$$\begin{aligned} p(x_i|y_i).p(y_i) &= p(y_i|x_i).p(x_i) = p(x_i, y_i) \\ \Rightarrow p(y_i|x_i) &= \frac{p(x_i|y_i).p(y_i)}{p(x_i)} = \frac{p(x_i|y_i).p(y_i)}{\sum\limits_y p(x_i|y_i).p(y_i)} \end{aligned}$$

$\Rightarrow$ the ***Bayes equation***: $\boxed{\textbf{posterior} = \frac{\textbf{likelihood} \times \textbf{prior}}{\textbf{normalization factor}}}$

2.2 Types of Probability Distributions

Reference source: machinelearningcoban.com.

2.2.1 Bernoulli Distribution

Bernoulli Distribution is a distribution to describe binary discrete variables. It's the case that the variable can only take value in 2 classes $x \in \{0, 1\}$. E.g., the probability of throwing a coin. The Bernoulli distribution is defined with parameter $\lambda \in [0, 1]$:

$$p(x) = \text{Bern}_x[\lambda] = \begin{cases} p(x = 1) = \lambda \\ p(x = 0) = 1 - \lambda \end{cases} \tag{2.1}$$

In short form, the above equation can be combined into one:

$$p(x) = \lambda^x (1 - \lambda)^{(1-x)} \Rightarrow \begin{cases} p(0) = \lambda^0 (1 - \lambda)^1 = 1 - \lambda \\ p(1) = \lambda^1 (1 - \lambda)^0 = \lambda \end{cases} \tag{2.2}$$

2.2.2 Categorical Distribution

Categorical Distribution is the generalization of *Bernoulli Distribution* for K classes of the discrete variable $x \in \{1, 2, \ldots, K\}$. Accordingly, there will be K parameters to describe this pdf.: $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_K]$, with $\lambda_k \geq 0$ and $\sum \lambda_k = 1$. Each λ_k represents the probability to take the output k: $p(x = k) = \lambda_k$. In short: $p(x) = \text{Cat}_x[\lambda]$.

Another common way to represent the output is the one-hot vector, $\mathbf{x} \in \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_K\}$ with $\mathbf{e}_k$ is the k-unit vector, which has all 0-element, except the k-element equal to 1. E.g., given 3 classes: $\mathbf{e}_1 = [1, 0, 0]^T, \mathbf{e}_2 = [0, 1, 0]^T, \mathbf{e}_3 = [0, 0, 1]^T$. We will then have:

$$p(\mathbf{x} = \mathbf{e}_k) = \prod_{j=1}^{K} \lambda_j^{x_j} = \lambda_k \tag{2.3}$$

because for $\mathbf{x} = \mathbf{e}_k$, only $x_k = 1$, while $x_j = 0, \forall j \neq k$.

2.2.3 Univariate Normal Distribution

Univariate Normal Distribution is also known as the Gaussian distribution. For single dimension data (in 1D): $x \in (-\infty, \infty)$, the mean $\mu \in \mathbb{R}$, and the variance σ^2 with $\sigma \in \mathbb{R}$.

$$p(x) = \text{Norm}_x \left[\mu, \sigma^2\right] = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}.\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.4}$$

<u>**NOTE:**</u>

- **Marginals prob. of Gaussian are again Gaussian.**
- When estimating the parameters (params.) of a Gaussian, beware the underestimation problem.

$$\begin{aligned}
\mathbb{E}\left[\mu_{ML}\right] &= \mu \\
\mathbb{E}\left[\sigma^2_{ML}\right] &= \left(\frac{N-1}{N}\right)\sigma^2 \\
\Rightarrow \tilde{\sigma}^2 &= \left(\frac{N}{N-1}\right)\sigma^2_{ML} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \hat{\mu})^2
\end{aligned}$$

2.2.4 Multivariate Normal Distribution

Multivariate Normal Distribution is the extension of *Univariate Normal Distribution* to multi-dimensional data: $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^D, \sigma^2 \Rightarrow \Sigma \in \mathbb{S}^D_{++}$ ($\mathbb{S}^D_{++}$ is the set of positive definite symmetric matrix)

$$p(x) = \text{Norm}_x[\boldsymbol{\mu}, \Sigma] = \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{2\pi^{D/2}|\Sigma|^{\frac{1}{2}}}.\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.5}$$

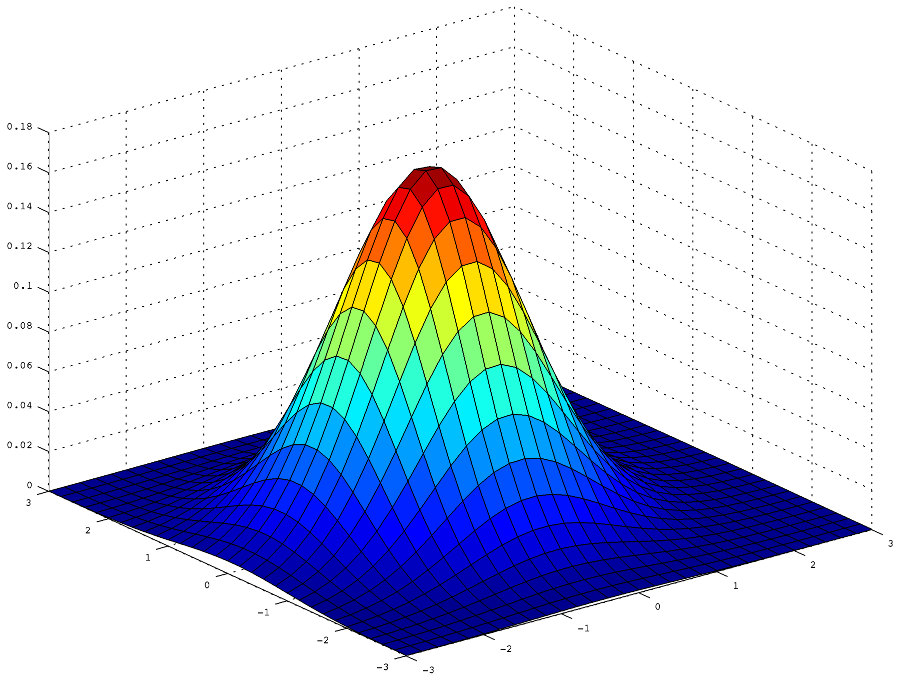


Figure 2.1: Bivariate Gaussian distribution (src).

2.2.5 Beta Distribution

This distribution describes the parameter for another distributions. E.g., Dirichlet pdf. describes Categorical Distribution (Subsec. 2.2.2)

3 Information Theory

[TODO: intro]

3.1 Entropy

$$
\begin{aligned}
& p(\mathbf{x}) && \text{– distribution (e.g. over observation } \mathbf{x}) && (3.1)\\
\mathcal{H}(p) &= -\,\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log p(\mathbf{x})] && \text{– entropy - how "broad" } p(\mathbf{x}) \text{ is} && (3.2)\\
&= -\sum_{i=1}^{N} p(\mathbf{x}_i)\log p(\mathbf{x}_i) && && (3.3)
\end{aligned}
$$

Intuition:

- How *random* is the variable?
  The more random the variable, the higher the entropy (Fig. 3.1)
- How large is the log prob. in expectation *under* itself?
  - If you mostly see low log prob., then there are many places with similar prob., and the entropy, as negative log, would be high (Fig. 3.1a).
  - If you mostly see high log prob., then the variable focuses around only a few place, thus, the entropy, as negative log, would be low (Fig. 3.1b).

(a) Example of $p(x)$ with high entropy. **(b)** Example of $p(x)$ with low entropy.

Figure 3.1: Examples of $p(x)$ with different entropy.

3.2 Cross Entropy

The cross entropy between two given prob. distributions p and q is defined as:

$$H(p,q) = \mathbb{E}_p[-\log q] \tag{3.4}$$

With **p, q** as discrete variables:

$$H(\mathbf{p},\ \mathbf{q}) = \sum_{i=1}^{C} p_i \log q_i \tag{3.5}$$

NOTE: $\nexists \log(0) \Rightarrow$ condition: $\mathbf{q} > 0$

3.3 Kullback - Leiber Divergence

The Kullback–Leibler (K-L)-divergence tells:

- How *different* are two distributions?
- How small is the expected log prob. of one distribution under another, **minus entropy**?

Example reference: countbayesie

$$H = -\sum_{i=1}^{N} p(x_i) \log p(x_i) \tag{3.6}$$

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \left[\log p(x_i) - \log q(x_i)\right] \tag{3.7}$$

$$= \sum_{i=1}^{N} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{3.8}$$

$$= \mathbb{E}_{x \sim p(x)} \left[\log p(x) - \log q(x)\right] \tag{3.9}$$

$\Rightarrow$ How many bits of info we expect to loose
$\Rightarrow$ ***A function*** that we can ***optimized***

cross entropy = entropy + KL Divergence

$$H(p,q) = H(p) + D_{KL}(p||q) \tag{3.10}$$

E.g.: KL divergence between two normal distributions $\mathcal{N}(\mu_1, \sigma_1)$ and $\mathcal{N}(\mu_2, \sigma_2)$:

$$D_{KL}(p,q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{3.11}$$

PSEUDO CODE: with $\mu_2 = 0, \sigma_2 = 1$

$$\mu, \sigma = \texttt{encoder}(\hat{x}) \tag{3.12}$$

$$\texttt{z} = \mu + \sigma * \texttt{random\_normal(0, 1)} \tag{3.13}$$

$$\texttt{y} = \texttt{decoder(z)} \tag{3.14}$$

$$\texttt{recon\_loss} = \texttt{x.log(y) + (1-x)log(1-y)} \tag{3.15}$$

$$\texttt{KL\_loss} = \frac{1}{2}[\mu^2 + \sigma^2 - \texttt{log}(\sigma^2 + 1e^{-8}) - 1] \tag{3.16}$$

$$\texttt{ELBO} = \texttt{recon\_loss - KL\_loss} \tag{3.17}$$

$$\texttt{loss} = \texttt{-ELBO} \tag{3.18}$$

3.4 Mutual Information

$$p(x) \qquad \text{– distribution (e.g. over observation } \mathbf{x}) \tag{3.19}$$

$$\mathcal{H}(p(\mathbf{x})) = -\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[\log p(\mathbf{x})] \qquad \text{– entropy - how "broad" } p(\mathbf{x}) \text{ is} \tag{3.20}$$

$$\mathcal{I}(\mathbf{x};\mathbf{y}) = D_{KL}\big(p(\mathbf{x},\mathbf{y})||p(\mathbf{x})p(\mathbf{y})\big) \qquad \text{– mutual information} \tag{3.21}$$

$$= \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p(\mathbf{x},\mathbf{y})}\left[\log \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})p(\mathbf{y})}\right] \tag{3.22}$$

$$= \mathcal{H}(p(\mathbf{y})) - \mathcal{H}(p(\mathbf{y}|\mathbf{x})) \qquad \text{– relates to IG} \tag{3.23}$$

The last equation implies that we can interpret mutual information $\mathcal{I}(\mathbf{x};\mathbf{y})$ as IG, how much more do we know about $\mathbf{y}$ after receiving observation about $\mathbf{x}$.

E.g.:

- If $\mathbf{x}$ and $\mathbf{y}$ are independent of each others, then $p(\mathbf{x},\mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$, thus $\mathcal{I}(\mathbf{x};\mathbf{y}) = 0$
- If $\mathbf{x}$ and $\mathbf{y}$ depends on each others more and more, the different between the joint prob. $p(\mathbf{x},\mathbf{y})$ and the product of marginal prob. $p(\mathbf{x})p(\mathbf{y})$ grows, thus the $\mathcal{I}(\mathbf{x};\mathbf{y})$ is larger.

4 Convexity

4.1 Convex Sets

- Definition 1: line connects 2 points of a convex set lies within the set
- Definition 2: $\mathcal{C}$ is a convex set if for $\forall x_1, x_2 \in \mathcal{C}$:

$$x_\theta = \theta x_1 + (1-\theta)x_2 \in \mathcal{C}, \quad \forall\, 0 \leq \theta \leq 1$$

- Hyperplane is a ***convex set***

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = \mathbf{a}^T\mathbf{x} = b, \quad b, a_i \in \mathbb{R}, \quad i \in \mathbb{N}$$

- Half-space is a ***convex set***

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = \mathbf{a}^T\mathbf{x} \leq b, \quad b, a_i \in \mathbb{R}, \quad i \in \mathbb{N}$$

- Matrix A is ***positive definite*** if:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0, \quad \forall\mathbf{x} \in \mathbb{R}^n \iff \boxed{A \succ 0}$$

  because $\exists A^{-1} \Rightarrow \forall\lambda \neq 0$ (eigenvalues)
- Intersection of convex sets ia a ***convex set***
  $\Rightarrow$ Polyhedra, which is the intersection of halfspaces and hyperplanes ***is convex***.
- x is said to be a ***convex combination*** of $x_1, x_2, \ldots, x_k$ if

$$x = \theta_1x_1 + \theta_2x_2 + \cdots + \theta_kx_k \quad \text{with} \quad \theta_1 + \theta_2 + \cdots + \theta_k = 1$$

- ***Convex hull*** of a set $(x_1, x_2, \ldots, x_k)$ is a set of all possible convex combination of that set.
  Convex hull of a set is the smallest convex set that contains that set.
- Two convex sets $\mathcal{C}$ and $\mathcal{D}$ are ***disjoint*** then exist a, b such:

$$\begin{cases} \mathbf{a}^T\mathbf{x} \leq b & \forall\mathbf{x} \in \mathcal{C} \\ \mathbf{a}^T\mathbf{x} \geq b & \forall\mathbf{x} \in \mathcal{D} \end{cases}$$

  Set of all $\mathbf{x}$ that $\mathbf{a}^T\mathbf{x} - b = 0$ is a hyperplane that separate $\mathcal{C}$ and $\mathcal{D}$
  $\Rightarrow$ ***separating hyperplane***

4.2 Convex function

Definition: $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function if the domain $\text{dom}(f)$ is a convex set and:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \forall x, y \in \text{dom}(f), \quad 0 \leq \theta \leq 1$$

- A function f is ***strictly convex*** if:

  $$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y)$$

  If there is a minimum point that it is the only minimum point and a global minimum
- ***Affine function:*** $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b$ is both convex and concave
  If the variable is a matrix $\mathbf{X}$: $f(\mathbf{X}) = trace(\mathbf{A}^T\mathbf{X}) + b$
- ***Quadratic form:*** $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\ \mathbf{x} + \mathbf{b}^T\mathbf{x} + c$ is
  - convex if $A \succeq 0$
  - concave if $-A \succeq 0$
- A function satisfies 3 norm conditions $\Rightarrow$ convex
- α-subset level of $f : \mathbb{R}^n \to \mathbb{R}$: $\mathcal{C}_\alpha = \{\mathbf{x} \in \mathrm{dom} f | f(\mathbf{x}) \leq \alpha\}$

Checking if f is convex:

- First order condition if and only if (i.f.f.): $\begin{cases} \text{differentiable with convex domain} \\ f(x) \geq f(x_0) + \nabla f(x_0)^T(x - x_0), \quad \forall x, x_0 \in \mathrm{dom} f \end{cases}$
- Second order condition: the Hessian $\nabla^2 f(x) \succeq 0$

5 Optimization

Problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f_0(\mathbf{x}) \qquad \text{subject to} \begin{cases} f_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m & \text{(inequality constraints)} \\ h_j(\mathbf{x}) = 0, j = 1, 2, \ldots, p & \text{(equality constraints)} \end{cases} \tag{5.1}$$

$$\text{feasible set } \mathcal{D} = \bigcap_{i=0}^{m} \text{dom} f_i \cap \bigcap_{j=0}^{p} \text{dom} h_j \Rightarrow \text{set of all } \mathbf{x} \text{ satisfying all constraints} \tag{5.2}$$

5.1 Convex Optimization Problem

Problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f_0(\mathbf{x}) \tag{5.3}$$

$$\text{subject to} \begin{cases} f_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, m & \text{(inequality constraints)} \\ \mathbf{a}_j^T \mathbf{x} - b_j = 0, j = 1, 2, \ldots, p & \text{(equality constraints)} \\ f_0 \text{ is convex func.} \\ f_i \text{ is convex func.} \\ h_j \text{ is affine func.} \end{cases} \tag{5.4}$$

$$\Rightarrow \begin{cases} f_i(x) \leq 0 \Rightarrow \text{0-sublevel set of } f_i \\ h_j(x) = 0, \quad \forall x \Rightarrow \text{hyperplane} \end{cases} \tag{5.5}$$

⇒ we optimize a convex function in a convex set domain.

5.2 Linear Programming

(Vietnamese: Quy hoach tuyen tinh)

Problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} + d \qquad \text{subject to} \begin{cases} \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ \mathbf{A}\mathbf{x} = \mathbf{b} \end{cases} \tag{5.6}$$

A standard form of Linear Programming (LP):

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \qquad \text{subject to} \begin{cases} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \leq \mathbf{0} \end{cases} \tag{5.7}$$

Python: `cvxopt.solvers.lp`

5.3 Quadratic Programming

Problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} + r \qquad \text{subject to} \begin{cases} \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ \mathbf{A}\mathbf{x} = \mathbf{b} \\ P \succeq 0 \quad (P \text{ is semi-definite}) \end{cases} \tag{5.8}$$

NOTE: When $P = 0$, Quadratic Programming (QP) is LP

Python: `cvxopt.solvers.qp`

5.4 Geometric Programming

- Function $f : \mathbb{R}^n \to \mathbb{R}$ with $\text{dom} f = \mathbb{R}^n_{++}$ (all element > 0) is a ***monomial function*** if:

$$f(x) = c\ x_1^{a_1}\ x_2^{a_2} \ldots x_n^{a_n}, \qquad c > 0, \quad a_i \in \mathbb{R} \tag{5.9}$$

- Function f is a ***posynomial function*** if:

$$f(x) = \sum_{k=1}^{K} c_k\ x_1^{a_{1k}}\ x_2^{a_{2k}} \ldots x_n^{a_{nk}}, \qquad c_k > 0 \tag{5.10}$$

Problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f_0(\mathbf{x}) \qquad \text{subject to} \begin{cases} f_i(\mathbf{x}) \leq 1, \quad i = 1, 2 \ldots m \\ h_j(\mathbf{x}) = 1, \quad j = 1, 2 \ldots p \\ f_0, f_i \text{ are posynomial func.} \\ h_j \text{ are monomial func.} \end{cases} \tag{5.11}$$

$\Rightarrow$ geometric programming ($x > 0$ is hidden)

Set: $\begin{cases} x_i = e^{y_i} \\ y_i = \log(x_i) \end{cases} \Rightarrow f_0(\mathbf{x}) = \exp(\mathbf{a}^T\mathbf{y} + b)$

Python: `cvxopt.solvers.gp`

6 Search Algorithms

Search problem is one common type of problem which has numerous presences in our lives. The well-known Travelling Salesman Problem (TSP) and its variants are search problems, in which the salesman have to find the shortest route that visit every cities. Many Reinforcement Learning (RL) problems can also be viewed as search problems, in which the machine find the most optimal plan to reach the goal.

A search problem consists of: an agent in a state space, a successor function, a start state and a goal state. The **agent** is the one taking the **action**, e.g., in TSP, the agent is the salesman, and the action is to travel; in RL, the agent is the robot or the machine, and the action could be to move to a different position. The **search state** represents the current situation that the agent is in, which would not necessary equivalent to the world state, which includes every possible details about the environment. E.g., in TSP, the state is the current city, every time the action traveling is taken, the agent moves from one city to another (one state to another). The **successor function** describes the transition from one state to another. This function usually comes with the transition action and costs.

A **solution of search problem** is a sequence of actions (a plan) which transform the start state to a goal state. **Search algorithms** find search solutions, which can be optimal, but in many practical cases, close to optimal within time limitation.

This chapter structures as follows:

- The first section describes how a search problem is formulated mathematically as a graph.
- The second section presents some well-known search algorithms.

6.1 Graph

A graph is the mathematical representation of a search problem. A graph consists of nodes and edges.

6.1.1 Undirected Graph

6.1.2 Directed Graph

6.1.3 Adjacency Matrix

6.1.4 Incidence Matrix

6.1.5 Trees and Forest

6.2 Search Algorithms

6.2.1 Properties

6.2.2 Depth-first search

6.2.3 Breadth-first search

6.2.4 Prim's Algorithm

6.2.5 Kruskal's Algorithm

6.2.6 Dijkstra's Algorithm

6.2.7 Bellman and Ford's Algorithm

6.2.8 A* Algorithm

[**TODO:**]