Study Notes

Yangyang Li
yangyang.li@northwestern.edu

Update on October 13, 2023

Contents

CONTENTS

List of Figures

List of Theorems

List of Definitions

LIST OF DEFINITIONS

Acronyms

MAP Maximum A Posteriori Estimation 14
MLE Maximum Likelihood Estimation 13, 14

PDF Probability Density Function 14

Acronyms

Preface

Contents

0.1 Features of this template

> *TeX, stylized within the system as LaTeX, is a typesetting system which was designed and written by Donald Knuth and first released in 1978. TeX is a popular means of typesetting complex mathematical formulae; it has been noted as one of the most sophisticated digital typographical systems.*
>
> - Wikipedia

0.1.1 crossref

different styles of clickable definitions and theorems

- nameref: Gaussian distribution
- autoref: Definition A.1, **??**
- cref: Definition A.1,
- hyperref: Gaussian,

0.1.2 ToC (Table of Content)

- mini toc of sections at the beginning of each chapter
- list of theorems, definitions, figures
- the chapter titles are bi-directional linked

0.1.3 header and footer

fancyhdr

- right header: section name and link to the beginning of the section
- left header: chapter title and link to the beginning of the chapter
- footer: page number linked to ToC of the whole document

0.1.4 bib

- titles of reference is linked to the publisher webpage e.g., [Kit+02]
- backref (go to the page where the reference is cited) e.g., [Chi09]
- customized video entry in reference like in [Bab16]

0.1.5 preface, index, quote (epigraph) and appendix

index page at the end of this document...

Part I

Machine Learning

Chapter 1

Probability

Contents

1.1 Basic Concepts

Permutation is an arrangement of objects in which the order is important.

$$P\left(n, r\right) = \frac{n!}{(n-r)!}$$

Combination is an arrangement of objects in which the order is not important.

$$C\left(n, r\right) = \frac{P\left(n, r\right)}{r!} = \frac{n!}{r!\left(n-r\right)!}$$

in which $0 \le r \le n$.

1.2 Maximum Likelihood Estimation

$$\boldsymbol{X} = \left(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\right)^{\mathrm{T}}, \boldsymbol{x}_i = \left(x_{i1}, x_{i2}, \ldots, x_{ip}\right)^{\mathrm{T}} \tag{1.1}$$

in which N is the number of samples, p is the number of features. The data is sampled from a distribution $p(\boldsymbol{x} \mid \theta)$, where θ is the parameter of the distribution.

For N i.i.d. samples, the likelihood function is $p(\boldsymbol{X} \mid \theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i \mid \theta))$

In order to get θ, we use Maximum Likelihood Estimation (MLE) to maximize the likelihood function.

$$\theta_{\mathtt{MLE}} = \underset{\theta}{\operatorname{argmax}} \log p(\boldsymbol{X} \mid \theta) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\boldsymbol{x}_i \mid \theta) \tag{1.2}$$

1.3 Maximum A Posteriori Estimation

In Bayes' theorem, the θ is not a constant value, but $\theta \sim p(\theta)$. Hence,

$$p(\theta \mid \boldsymbol{X}) = \frac{p(\boldsymbol{X} \mid \theta)p(\theta)}{p(\boldsymbol{X})} = \frac{p(\boldsymbol{X} \mid \theta)p(\theta)}{\int\limits_\theta p(\boldsymbol{X} \mid \theta)p(\theta)d\theta} \tag{1.3}$$

In order to get θ, we use Maximum A Posteriori Estimation (MAP) to maximize the posterior function.

$$\theta_{\texttt{MAP}} = \underset{\theta}{\operatorname{argmax}}\, p(\theta \mid \boldsymbol{X}) = \underset{\theta}{\operatorname{argmax}} \frac{p(\boldsymbol{X} \mid \theta)p(\theta)}{p(\boldsymbol{X})} \tag{1.4}$$

After θ is estimated, then calculating $\frac{p(\boldsymbol{X} \mid \theta) \cdot p(\theta)}{\int\limits_\theta p(\boldsymbol{X} \mid \theta)p(\theta)d\theta}$ to get the posterior distribution. We can use the posterior distribution to predict the probability of a new sample $\boldsymbol{x}$.

$$p(x_{\texttt{new}} \mid \boldsymbol{X}) = \int\limits_\theta p(x_{\texttt{new}} \mid \theta) \cdot p(\theta \mid \boldsymbol{X})d\theta \tag{1.5}$$

1.4 Gaussian Distribution

Gaussian distribution is also called normal distribution.

$$\theta = (\mu, \sigma^2), \quad \mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{1.6}$$

For MLE,

$$\theta = (\mu, \Sigma) = (\mu, \sigma^2), \quad \theta_{\texttt{MLE}} = \underset{\theta}{\operatorname{argmax}} \log p(\boldsymbol{X} \mid \theta) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\boldsymbol{x}_i \mid \theta) \tag{1.7}$$

Generally, the Probability Density Function (PDF) of a Gaussian distribution is:

$$p(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \mu)^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{x} - \mu)\right) \tag{1.8}$$

in which μ is the mean vector, Σ is the covariance matrix, $\det$ is the determinant of matrix. $\det$ is the product of all eigenvalues of a matrix.

Hence,

$$\log p(\boldsymbol{X} \mid \theta) = \sum_{i=1}^{N} \log p(x_i \mid \theta) = \sum_{i=1}^{N} \log \frac{1}{\sqrt{(2\pi)^p \det(\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \mu)^{\mathrm{T}}\Sigma^{-1}(\boldsymbol{x} - \mu)\right) \tag{1.9}$$

Let's only consider 1 dimension case for brevity, then

$$\log p(\boldsymbol{X} \mid \theta) = \sum_{i=1}^{N} \log p(x_i \mid \theta) = \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x - \mu)^2}{\sigma^2}\right) \tag{1.10}$$

Let's get the optimal value for μ,

$$\mu_{\texttt{MLE}} = \underset{\mu}{\operatorname{argmax}} \log p(\boldsymbol{X} \mid \theta) = \underset{\mu}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{2} (x_i - \mu)^2 \tag{1.11}$$

So,

$$\frac{\partial \log p(\boldsymbol{X} \mid \theta)}{\partial \mu} = \sum_{i=1}^{N} (\mu - x_i) = 0 \to \mu_{\texttt{MLE}} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1.12}$$

Let's get the optimal value for σ^2,

$$\begin{aligned}
\sigma_{\texttt{MLE}} &= \underset{\sigma}{\operatorname{argmax}} \log p(\boldsymbol{X} \mid \theta) \\
&= \underset{\sigma}{\operatorname{argmax}} \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \\
&= \underset{\sigma}{\operatorname{argmax}} \sum_{i=1}^{N} \left[-\log\sqrt{2\pi\sigma^2} - \frac{(x-\mu)^2}{2\sigma^2}\right] \\
&= \underset{\sigma}{\operatorname{argmin}} \sum_{i=1}^{N} \left[\log\sigma + \frac{(x-\mu)^2}{2\sigma^2}\right]
\end{aligned}$$

Hence,

$$\frac{\partial}{\partial \sigma} \sum_{i=1}^{N} \left[\log\sigma + \frac{(x_i-\mu)^2}{2\sigma^2}\right] = 0 \to \sigma^2_{\texttt{MLE}} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2 \tag{1.13}$$

$\mathbb{E}_D\left[\mu_{\texttt{MLE}}\right]$ is unbaised.

$$\mathbb{E}_D\left[\mu_{\texttt{MLE}}\right] = \mathbb{E}_D\left[\frac{1}{N}\sum_{i=1}^{N} x_i\right] = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_D\left[x_i\right] = \frac{1}{N}\sum_{i=1}^{N} \mu = \mu \tag{1.14}$$

However, $\mathbb{E}_D\left[\sigma^2_{\texttt{MLE}}\right]$ is biased.

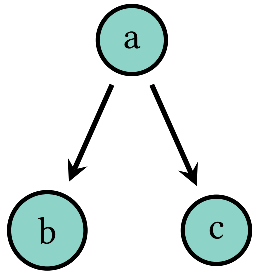

Figure 1.1: A simple Bayesian network.

$$\mathbb{E}_D\left[\sigma^2_{\texttt{MLE}}\right] = \mathbb{E}_D\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\texttt{MLE}})^2\right] \tag{1.15}$$

$$= \mathbb{E}_D\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_{\texttt{MLE}})^2\right] \tag{1.16}$$

$$= \mathbb{E}_D\left[\frac{1}{N}\sum_{i=1}^{N}\left(x_i^2 - 2x_i\mu_{\texttt{MLE}} + \mu^2_{\texttt{MLE}}\right)\right] = \mathbb{E}_D\left[\sum_{i=1}^{N}x_i^2 - 2\frac{1}{N}\sum_{i=1}^{N}x_i\mu_{\texttt{MLE}} + \mu^2_{\texttt{MLE}}\right] \tag{1.17}$$

$$= \mathbb{E}_D\left[\frac{1}{N}\sum_{i=1}^{N}\left(x_i^2 - \mu^2\right) + \mu^2 - \mu^2_{\texttt{MLE}}\right] \tag{1.18}$$

$$= \sigma^2 - \mathbb{E}_D\left[\mu^2_{\texttt{MLE}} - \mu^2\right] \tag{1.19}$$

$$= \sigma^2 - \left(\mathbb{E}_D\left[\mu^2_{\texttt{MLE}}\right] - \mathbb{E}_D\left[\mu^2_{\texttt{MLE}}\right]\right) \tag{1.20}$$

$$= \sigma^2 - \texttt{Var}\left[\mu_{\texttt{MLE}}\right] = \sigma^2 - \texttt{Var}\left[\frac{1}{N}\sum_{i=1}^{N}x_i\right] \tag{1.21}$$

$$= \sigma^2 - \frac{1}{N^2}\sum_{i=1}^{N}\texttt{Var}\left[x_i\right] = \frac{N-1}{N}\sigma^2 \tag{1.22}$$

$$\tag{1.23}$$

1.5 Bayesian Network

1.6 Probability Graph

this section is not finished yet. Need to be reviewed p54

1.6.1 Variables Elimination

1.6.2 Belief propagation

Belief propagation is mainly used for tree data structure, and equals Section 1.6.1 with caching.

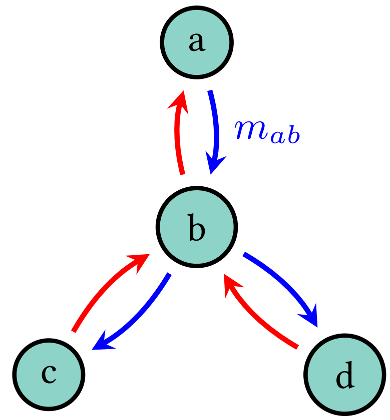

Figure 1.2: Belief propagation.

1.6.3 Max-product Algorithm

1.6.4 Factor Graph

1.7 Expectation Maximum

$$\Theta^{(t+1)} = \underset{\Theta}{\operatorname{argmax}} \int_Z \log P(x, z \mid \theta) \cdot P(z \mid x, \Theta^{(t)})\, dz$$

continue on p60

1.8 Gaussian Mixture Model

$$Q\left(\Theta, \Theta^{(t)}\right) = \int_Z \log P(x, z \mid \theta) \cdot P(z \mid x, \Theta^{(t)})\, dz$$

1.9 Hidden Markov Model

Chapter 2

Notes

2.1 Complexity of matrix multiply

Assume A is $m \times n$ and B is $n \times p$. The naive algorithm takes $O(mnp)$ time.

2.2 Representation of matrix

row major, col major, stride

Part II

Algorithm and Data Structure

Chapter 3

Algorithm

Contents

3.1 Graph

Part III

Programming

Chapter 4

C++

Contents

4.1 Code Snippets

4.1.1 Random Number Generation

```cpp
#include <algorithm>
#include <iostream>
#include <iterator>
#include <random>

int main() {
  std::random_device rd;
  std::mt19937 rng(rd());
  std::uniform_int_distribution<int> dist6(1, 6);
  std::generate_n(std::ostream_iterator<int>(std::cout, " "), 10,
                  [&dist6, &rng]() { return dist6(rng); });
  return 0;
}
```

4.1.2 Quick Sort

```cpp
#include <vector>
using std::vector;

class Solution {
public:
    void sort(vector<int>& nums, int low, int high){
        if (low >= high)  return;

        int p = partition(nums, low, high);

        sort(nums, low, p - 1); // Changed p to p-1
```

```
        sort(nums, p + 1, high);
    }

    int partition(vector<int>& nums, int low, int high){
        int pivot  = low, l = pivot + 1, r = high;

        while(l <= r) {
            if (nums[l] < nums[pivot]) ++l;
            else if (nums[r] >= nums[pivot]) --r;
            else std::swap(nums[l], nums[r]);
        }
        std::swap(nums[pivot], nums[r]);
        return r;
    }

    void shuffle(vector<int>& nums){
        std::srand((unsigned) time(nullptr));
        int n = nums.size();

        for (int i = 0; i < n; ++i){
            int r = i + rand() % (n - i);
            std::swap(nums[i], nums[r]);
        }
    }

    vector<int> sortArray(vector<int>& nums) {
        shuffle(nums); // Shuffle the array before sorting
        sort(nums, 0, nums.size() - 1); // Changed nums.size() to nums.size() - 1
        return nums;
    }
};
```

On average, the algorithm takes $O\left(n \log n\right)$ time. In the worst case, it takes $O\left(n^2\right)$ time. We shuffle the array before sorting to avoid the worst case.

Chapter 5

Rust

Part IV

Research

Chapter 6

Paper Reading

Appendix A

Formulas

A.1 Gaussian distribution

Definition A.1 (Gaussian distribution)**.** *Gaussian distribution*

Theorem A.1 (Central limit theorem)**.**

Bibliography

[Bab16] László Babai. “Graph Isomorphism in Quasipolynomial Time”. Jan. 19, 2016. arXiv: `1512.03547 [cs, math]` (cit. on p. 10). Online video

[Chi09] Andrew M. Childs. *Universal Computation by Quantum Walk*. Physical Review Letters 102.18 (May 4, 2009), p. 180501. arXiv: `0806.1972` (cit. on p. 10).

[Kit+02] Alexei Yu Kitaev et al. *Classical and quantum computation*. 47. American Mathematical Soc., 2002 (cit. on p. 10).

BIBLIOGRAPHY

Alphabetical Index