THU-70250043, Pattern Recognition (Spring 2017) **Homework: 1**

Bayesian Methods

Lecturer: Changshui Zhang `zcs@mail.tsinghua.edu.cn`

Student: Xindi Lu `2016210557`

MLE and MAP

1. The Poisson distribution is useful for modeling the number of events occurring within a unit time, such as the number of packets arrived at some server per minute. The probability mass function of a Poisson distribution is as follows:

$$P(k|\lambda) := \frac{\lambda^k e^{-\lambda}}{k!}, \tag{1}$$

where $\lambda > 0$ is the parameter of the distribution and $k \in \{0, 1, 2, ...\}$ is the discrete random variable modeling the number of events encountered per unit time.

1.1. Let $\{k_1, k_2, ..., k_n\}$ be an i.i.d. sample drawn from a Poisson distribution with parameter λ. Derive the MLE estimate $\hat{\lambda}_{mle}$ of λ based on this sample.

1.2. Let K be a random variable following a Poisson distribution with parameter λ. Derive its mean E[K] and variance Var[K]. Since $\hat{\lambda}_{mle}$ depends on the sample used for estimation, it is also a random variable. Derive the mean and the variance of $\hat{\lambda}_{mle}$, and compare them with E[K] and Var[K]. What do you find?

1.3. Suppose you believe the Gamma distribution

$$p(\lambda) := \frac{\lambda^{\alpha-1} e^{-\lambda/\beta}}{\Gamma(\alpha)\beta^\alpha}, \tag{2}$$

is a good prior for λ, where $\Gamma(\cdot)$ is the Gamma function, and you also know the values of the two hyper-parameters $\alpha > 1$ and $\beta > 0$. Derive the MAP estimation $\hat{\lambda}_{map}$.

1.4. What happens to $\hat{\lambda}_{map}$ when the sample size n goes to zero or infinity? How do they relate to the prior distribution and $\hat{\lambda}_{mle}$?

Solution:
1.1. The likelihood function is given by

$$l(\lambda) = \prod_{i=1}^{n} \frac{\lambda^{k_i} e^{-\lambda}}{k_i!} = \frac{\lambda^{\sum\limits_{i=1}^{n} k_i} e^{-n\lambda}}{\prod_{i=1}^{n} k_i!}$$

$$\implies \quad \frac{\partial l(\lambda)}{\lambda} = \frac{1}{\prod_{i=1}^n k_i!}[(\sum_{i=1}^n k_i)\lambda^{\sum\limits_{i=1}^n k_i - 1} e^{-n\lambda} - n\lambda^{\sum\limits_{i=1}^n k_i} e^{-n\lambda}] = 0$$

$$\implies \quad \hat{\lambda}_{MLE} = \frac{\sum\limits_{i=1}^n k_i}{n}.$$

1.2.

$$E[K] = \sum_{k=1}^{\infty} k\frac{\lambda^k e^{-\lambda}}{k!} = \lambda e^{-\lambda}\sum_{k=1}^{\infty}\frac{\lambda^{k-1}}{(k-1)!} = \lambda e^{-\lambda}e^{\lambda} = \lambda$$

$$\begin{aligned} E[K^2] = \sum_{k=1}^{\infty} k^2\frac{\lambda^k e^{-\lambda}}{k!} &= \sum_{k=1}^{\infty} k(k-1)\frac{\lambda^k e^{-\lambda}}{k!} + E[K] \\ &= \lambda^2 e^{-\lambda}\sum_{k=2}^{\infty}\frac{\lambda^{k-2}}{(k-2)!} + \lambda \\ &= \lambda^2 + \lambda \end{aligned}$$

$$\implies \quad Var[K] = E[K^2] - (E[K])^2 = \lambda^2 + \lambda - \lambda^2 = \lambda$$

Since $\{k_1, k_2, \cdots, k_n\}$ are i.i.d. sample drawn from a Poisson distribution with parameter λ, we have $E[k_i] = E[K], Var[k_i] = Var[K]$.
Thus, for $\hat{\lambda}_{MLE}$, we have

$$E[\hat{\lambda}_{MLE}] = E[\frac{\sum_{i=1}^n k_i}{n}] = \frac{1}{n}\sum_{i=1}^n E[K] = \frac{1}{n}\times n\lambda = \lambda$$

$$Var[\hat{\lambda}_{MLE}] = Var[\frac{\sum_{i=1}^n k_i}{n}] = \frac{1}{n^2}\sum_{i=1}^n Var[K] = \frac{1}{n^2}\times n\lambda = \frac{\lambda}{n}$$

We can find that

$$E[\hat{\lambda}_{MLE}] = E[K], \quad Var[\hat{\lambda}_{MLE}] = \frac{1}{n}Var[K].$$

1.3 Note that

$$\begin{aligned} \hat{\lambda}_{map} &= arg\max_{\lambda}\prod_{i=1}^n P(k_i|\lambda)p(\lambda) \\ &= arg\max_{\lambda}\prod_{i=1}^n \frac{\lambda^{k_i}e^{-\lambda}}{k_i!}p(\lambda) \\ &= arg\max_{\lambda}\frac{\lambda^{\sum\limits_{i=1}^n k_i}e^{-n\lambda}}{\prod_{i=1}^n k_i!}\frac{\lambda^{\alpha-1}e^{-\lambda/\beta}}{\Gamma(\alpha)\beta^{\alpha}} \end{aligned}$$

Let the the partial derivatives equal to zero and we have

$$(\sum_{i=1}^n k_i + \alpha - 1) - (n + \frac{1}{\beta})\lambda = 0$$

$$\implies \quad \hat{\lambda}_{map} = \frac{\sum_{i=1}^n k_i + \alpha - 1}{n + \frac{1}{\beta}}$$

1.4 Note that

$$\lim_{n\to\infty} \hat{\lambda}_{map} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} k_i + \alpha - 1}{n + \frac{1}{\beta}} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} k_i}{n} \equiv \lim_{n\to\infty} \hat{\lambda}_{mle}$$

$$\lim_{n\to 0} \hat{\lambda}_{map} = \lim_{n\to 0} \frac{\sum_{i=1}^{n} k_i + \alpha - 1}{n + \frac{1}{\beta}} = \frac{\alpha - 1}{\frac{1}{\beta}} = \beta(\alpha - 1).$$

We can see that when the number of samples goes to infinity, MLE and MAP almost obtain the same estimation; when the sample size goes to zero, λ_{mle} doesn't exist while MAP can still have a estimation of λ_{map} based on the prior distribution $p(\lambda)$ ($\frac{\partial p(\lambda)}{\partial \lambda} = 0 \longrightarrow \lambda = \beta(\alpha - 1)$).

2. The density function of a $p-$dimensional Gaussian distribution is as follows,

$$N(x|\mu, \Lambda^{-1}) := \frac{\exp(-\frac{1}{2})(x-\mu)^T \Lambda (x-\mu)}{(2\pi)^{p/2}\sqrt{|\Lambda^{-1}|}}, \tag{3}$$

where Λ is the inverse of the covariance matrix, or the so-called precision matrix. Let $\{x_1, x_2, ..., x_n\}$ be an i.i.d. sample from a $p-$dimensional Gaussian distribution.

2.1. Suppose that $n \gg p$. Derive the MLE estimates $\hat{\mu}_{mle}$ and $\hat{\Lambda}_{mle}$.

2.2. Suppose you believe the Gaussian-Wishart prior defined as

$$gw(\mu, \Lambda) := \ N(\mu|\mu_0, (s\Lambda)^{-1}) W(\Lambda|V, v) \tag{4}$$

is a good prior for μ and Λ, where

$$W(\Lambda|V, v) := \frac{|\Lambda|^{(v-p-1)/2}}{Z(V, v)} \exp\left(-\frac{tr(V^{-1}\Lambda)}{2}\right) \tag{5}$$

with $tr(\cdot)$ being the trace of a square matrix and $Z(V, v)$ the normalization term. You also know the values of the hyper-parameters $\mu_0 \in \mathbb{R}^p, s > 0, v > p+1$, and $V \in \mathbb{R}^{p\times p}$ being positive definite. Derive the MAP estimates $\hat{\mu}_{map}$ and $\hat{\Lambda}_{map}$.

2.3. Again, what happens to $\hat{\mu}_{map}$ and $\hat{\Lambda}_{map}$ when n goes to zero or infinity? How do they relate to the prior distribution and the MLE estimates?

Solution:
2.1 The likelihood function is given by

$$L(\boldsymbol{\mu}, \Lambda) = \prod_{i=1}^{n} N(\mathbf{x}_i|\boldsymbol{\mu}, \Lambda^{-1}) = \frac{1}{(2\pi)^{pn/2}|\Lambda^{-1}|^{n/2}} exp\{-\frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})^T \Lambda (\mathbf{x}_i - \boldsymbol{\mu})\}.$$

$$\Longrightarrow \quad lnL(\boldsymbol{\mu}, \Lambda) = -\frac{pn}{2} ln(2\pi) + \frac{n}{2} ln|\Lambda| - \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})^T \Lambda (\mathbf{x}_i - \boldsymbol{\mu})$$

Then, we take the partial derivatives with regard to $\boldsymbol{\mu}$ and Λ respectively:

$$\begin{aligned}\frac{\partial lnL(\boldsymbol{\mu},\Lambda)}{\partial\boldsymbol{\mu}} &= \sum_{i=1}^{n}\Lambda(\mathbf{x}_i-\boldsymbol{\mu}) = 0\\ \frac{\partial lnL(\boldsymbol{\mu},\Lambda)}{\partial\Lambda} &= \frac{n}{2}|\Lambda|^{-1} - \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^T = 0\end{aligned}$$

Then, we can obtain

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{mle} &= \frac{\sum_{i=1}^{n}\mathbf{x}_i}{n} = \bar{\mathbf{x}}_i,\\ \hat{\Lambda}_{mle} &= n[\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}}_i)(\mathbf{x}_i-\bar{\mathbf{x}}_i)^T]^{-1} = n\boldsymbol{S}^{-1}.\end{aligned}$$

2.2

$$h(\boldsymbol{\mu},\Lambda) = lnL(\boldsymbol{\mu},\Lambda) + lngw(\boldsymbol{\mu},\Lambda) = \sum_{i=1}^{n} lnN(\mathbf{x}_i|\boldsymbol{\mu},\Lambda^{-1}) + lnN(\boldsymbol{\mu}|\boldsymbol{\mu}_0,(s\Lambda)^{-1}) + lnW(\Lambda|V,v)$$

Note that

$$\begin{aligned}lnN(\boldsymbol{\mu}|\boldsymbol{\mu}_0,(s\Lambda)^{-1}) &= ln\frac{\exp(-\frac{1}{2})(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T(s\Lambda)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)}{(2\pi)^{p/2}\sqrt{|s\Lambda|^{-1}}}\\ &= -\frac{p}{2}ln(2\pi) + \frac{1}{2}ln|s\Lambda| - \frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T(s\Lambda)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)\\ \Longrightarrow \frac{\partial lnN(\boldsymbol{\mu}|\boldsymbol{\mu}_0,(s\Lambda)^{-1})}{\partial\boldsymbol{\mu}} &= -(s\Lambda)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)\\ \frac{\partial lnN(\boldsymbol{\mu}|\boldsymbol{\mu}_0,(s\Lambda)^{-1})}{\partial\Lambda} &= \frac{1}{2s}|\Lambda|^{-1} - \frac{s}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T.\end{aligned}$$

Also, we have

$$\begin{aligned}lnW(\Lambda|\boldsymbol{V},v) &= ln\frac{|\Lambda|^{(v-p-1)/2}}{Z(\boldsymbol{V},v)}\exp\left(-\frac{tr(\boldsymbol{V}^{-1}\Lambda)}{2}\right)\\ &= \frac{v-p-1}{2}ln|\Lambda| - \frac{tr(\boldsymbol{V}^{-1}\Lambda)}{2} - lnZ(\boldsymbol{V},v)\\ \Longrightarrow \frac{\partial lnW(\Lambda|\boldsymbol{V},v)}{\partial\Lambda} &= \frac{v-p-1}{2}|\Lambda|^{-1} - \frac{1}{2}(\boldsymbol{V}^{-1})^T\end{aligned}$$

Therefore, we can obtain

$$\begin{aligned}&\frac{\partial h(\boldsymbol{\mu},\Lambda)}{\partial\boldsymbol{\mu}} = \sum_{i=1}^{n}\Lambda(\mathbf{x}_i-\boldsymbol{\mu}) - (s\Lambda)(\boldsymbol{\mu}-\boldsymbol{\mu}_0) = 0\\ &\frac{\partial h(\boldsymbol{\mu},\Lambda)}{\partial\Lambda} = \frac{n}{2}|\Lambda|^{-1} - \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^T + \frac{1}{2s}|\Lambda|^{-1} - \frac{s}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T + \frac{v-p-1}{2}|\Lambda|^{-1} - \frac{1}{2}(\boldsymbol{V}^{-1})^T = 0\\ &\Longrightarrow \quad \hat{\boldsymbol{\mu}}_{map} = \frac{\sum_{i=1}^{n}\mathbf{x}_i + s\boldsymbol{\mu}_0}{n+s},\\ &\hat{\Lambda}_{map} = (n+\frac{1}{s}+v-p-1)[\sum_{i=1}^{n}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_{map})(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_{map})^T + s(\hat{\boldsymbol{\mu}}_{map}-\boldsymbol{\mu}_0)(\hat{\boldsymbol{\mu}}_{map}-\boldsymbol{\mu}_0)^T + (\boldsymbol{V^{-1}})^T]^{-1}.\end{aligned}$$

2.4 Note that
when n goes to infinity,

$$\lim_{n\to\infty} \hat{\boldsymbol{\mu}}_{map} = \frac{\sum_{i=1}^{n} \mathbf{x}_i}{n} \equiv \hat{\boldsymbol{\mu}}_{mle},$$
$$\lim_{n\to\infty} \hat{\Lambda}_{map} = n[\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)^T]^{-1} = \hat{\Lambda}_{mle};$$

when n goes to zero

$$\lim_{n\to 0} \hat{\boldsymbol{\mu}}_{map} = \boldsymbol{\mu}_0,$$
$$\lim_{n\to 0} \hat{\Lambda}_{map} = (\frac{1}{s} + v - p - 1)[s(\hat{\boldsymbol{\mu}}_{map} - \boldsymbol{\mu}_0)(\hat{\boldsymbol{\mu}}_{map} - \boldsymbol{\mu}_0)^T + (\boldsymbol{V^{-1}})^T]^{-1}.$$

The conclusion is the same as what we have drawn in the previous part: when the number of samples goes to infinity, MLE and MAP almost obtain the same estimation; when the sample size goes to zero, MLE estimation may don't exist while MAP can still have a estimation of parameters based on the prior distribution $gw(\mu, \Lambda)$ (take the partial derivatives of the prior distribution and let them equal zero).

3. It is known that MLEs do not always exist. Even if they do, they may not be unique.

3.1. Give an example where MLEs do not exist.

3.2. Give an example where MLEs exist but are not unique. Please specify the family of distributions being considered, and the kind of samples from which multiple MLEs can be found.

3.3. By finding the two examples as described above, hopefully you have gained some intuition on the properties of the log-likelihood that are crucial to the existence and uniqueness of MLE. What are those properties?

Solution:
3.1 Assume that $X \sim N(\mu, \sigma^2)$, $\mu \in [a, b]$ and a, b are known number.
The log-likelihood function is given by

$$\begin{aligned} L(\mu, \sigma^2) &= \frac{(x-\mu^2)}{2\sigma^2} - \frac{1}{2} ln(2\pi\sigma^2) \\ \Longrightarrow \quad \frac{\partial L(\mu, \sigma^2)}{\partial \mu} &= \frac{1}{\sigma^2}(x - \mu) \\ \frac{\partial L(\mu, \sigma^2)}{\partial \sigma^2} &= \frac{(x-\mu)^2}{2\sigma^4} - \frac{1}{2\sigma^2} \end{aligned}$$

Note that when $x \in [a, b]$, $L(\mu, \sigma^2)$ peaks at $\mu = x$ for given σ. However, in this case, $L(\mu, \sigma^2) = -\frac{1}{2} ln(2\pi\sigma^2)$ will go to $+\infty$ with $\sigma \to 0$. Thus, $\hat{\sigma}_{mle}$ doesn't exist.

3.2 Assume that $X \sim U(\theta - \frac{1}{2}, \theta + \frac{1}{2})$. Let $\{x_1, x_2, \cdots, x_n\}$ be samples of X. Then we can obtain the likelihood function:

$$l(\theta) = \begin{cases} 1 & \theta - \frac{1}{2} \leqslant x_i \leqslant \theta + \frac{1}{2}, i = 1, 2, \cdots, n \\ 0 & \text{otherwise.} \end{cases}$$

Note that when $\theta - \frac{1}{2} \leqslant x_{min} \leqslant x_{max} \leqslant \theta + \frac{1}{2}$, $l(\theta)$ is maximized at 1. Thus,

$$x_{max} - \frac{1}{2} \leqslant \hat{\theta}_{mle} \leqslant x_{min} + \frac{1}{2},$$

that is, MLEs are not unique.

3.3 For the existence and uniqueness of MLE, the log-likelihood should not be a constant and it should be monotonous with regard to each parameter.

4. Consider a training data of N i.i.d. (independently and identically distribute) observations, $\boldsymbol{X} = \{x_1, x_2, ..., x_N\}$ with corresponding N target values $\boldsymbol{T} = \{t_1, t_2, ..., t_N\}$.

We want to fit these observations into some model

$$t = y(x, \boldsymbol{w}) + \epsilon \tag{6}$$

where $\boldsymbol{w}$ is the model parameters and ϵ is some error term.

4.1 To find $\boldsymbol{w}$, we can minimize the sum of square error

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 \tag{7}$$

Now suppose we believe that the distribution of error term ϵ is gaussian

$$p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1}) \tag{8}$$

where $\beta = \frac{1}{\sigma^2}$ is the inverse of variance. Using the property of gaussian distribution, we have

$$p(t|x, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(x, \boldsymbol{w}), \beta^{-1}) \tag{9}$$

Under this assumption, the likelihood function is given by

$$p(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N}\mathcal{N}(t_n|y(x_n, \boldsymbol{w}), \beta^{-1}) \tag{10}$$

Show that the problem of finding the maximum likelihood (ML) solution for $\boldsymbol{w}$ is equivalent to the problem of minimizing the sum of square error (7).

4.2 In order to avoid overfitting, we often add a weight decay term to (7)

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2 + \frac{\lambda}{2}||\boldsymbol{w}||^2 \tag{11}$$

On the other hand, we believe that $\boldsymbol{w}$ has a prior distribution of

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1}\boldsymbol{I}) \tag{12}$$

Using Bayes theorem, the posterior distribution for $\boldsymbol{w}$ is proportional to the product of the prior distribution and the likelihood function

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{T}, \alpha, \beta) \propto p(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta)p(\boldsymbol{w}|\alpha) \tag{13}$$

Show that the problem of finding the maximum of the posterior (MAP) solution for $\boldsymbol{w}$ is equivalent to the problem of minimizing (11).

Solution:
4.1 Note that

$$\mathcal{N}(t|y(x, \boldsymbol{w}), \beta^{-1}) = \frac{1}{\sqrt{2\pi}\sigma} exp\{-\frac{(t - y(x, \boldsymbol{w}))^2}{2\sigma^2}\},$$

where $\beta^{-1} = \sigma^2$.
Then, the likelihood function is given by

$$l(\boldsymbol{w}) = p(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \frac{1}{(2\pi)^{N/2}\sigma^N} exp\{-\sum_{n=1}^{N} \frac{(t_n - y(x_n, \boldsymbol{w}))^2}{2\sigma^2}\}$$

$$\begin{aligned}\Longrightarrow \quad lnl(\boldsymbol{w}) &= -\frac{N}{2}ln(2\pi) - Nln\sigma - \sum_{n=1}^{N} \frac{(t_n - y(x_n, \boldsymbol{w}))^2}{2\sigma^2} \\ &= -\frac{N}{2}ln(2\pi) - Nln\sigma - \frac{1}{\sigma^2}\frac{1}{2}\sum_{n=1}^{N}(y(x_n, \boldsymbol{w}) - t_n)^2 \\ &= -\frac{N}{2}ln(2\pi) - Nln\sigma - \frac{1}{\sigma^2}E(\boldsymbol{w})\end{aligned}$$

The previous equations infer that to find the maximum likelihood (ML) solution for $\boldsymbol{w}$ is to maximize $l(\boldsymbol{w})$, that is, to maximize $ln(\boldsymbol{w})$, which is equivalent to the problem of minimizing the sum of square error (7).

4.2 Assume that $\boldsymbol{w} \in R^p$ and we have

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1}\boldsymbol{I}) = \frac{\exp(-\frac{1}{2})(\boldsymbol{w} - \boldsymbol{0})^T \alpha \boldsymbol{I}^{-1}(\boldsymbol{w} - \boldsymbol{0})}{(2\pi)^{p/2}\sqrt{|\alpha^{-1}\boldsymbol{I}|}}.$$

Then,

$$\begin{aligned} lnp(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta)p(\boldsymbol{w}|\alpha) &= lnp(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta) + lnp(\boldsymbol{w}|\alpha) \\ &= -\frac{N}{2}ln(2\pi) - Nln\sigma - \frac{1}{\sigma^2}\frac{1}{2}\sum_{n=1}^{N}(y(x_n, \boldsymbol{w}) - t_n)^2 - \frac{\alpha}{2}\boldsymbol{w}^T\boldsymbol{I}\boldsymbol{w} - \frac{p}{2}ln(2\pi) - \frac{1}{2}ln|\alpha^{-1}\boldsymbol{I}| \\ &= -\frac{1}{\sigma^2}[\frac{1}{2}\sum_{n=1}^{N}(y(x_n, \boldsymbol{w}) - t_n)^2 + \frac{\alpha/\sigma^2}{2}||\boldsymbol{w}||^2] - \frac{N}{2}ln(2\pi) - Nln\sigma - \frac{p}{2}ln(2\pi) - \frac{1}{2}ln|\alpha^{-1}\boldsymbol{I}| \end{aligned}$$

If we let $\lambda = \alpha/\sigma^2$, we can prove that the problem of finding the maximum of the posterior (MAP) solution for $\boldsymbol{w}$ is equivalent to the problem of minimizing (11).

Naive Bayes

1. Considers the learning function $X \to Y$, where class label $Y \in \{T, F\}$, $X = \langle X_1, X_2, ..., X_n \rangle$ where X_1 is a boolean variable and $\{X_2, ..., X_n\}$ are continuous variables. Assume that for each continuous X_i, $P(X_i|Y = y)$ follows a Gaussian distribution. List and give the total number of the parameters that you would need to estimate in order to classify a future example using a Naive Bayes classifier. Give the formula for computing $P(Y|X)$ in terms of these parameters and feature variables X_i.

Solution: Naive Bayes method assumes the independence of each feature for given class. Thus,

$$P(Y = y_i|\boldsymbol{X}) = \frac{\prod\limits_{k=1}^{n} P(X_k|y_i)P(y_i)}{\sum\limits_{j=1}^{2}\prod\limits_{k=1}^{n} P(X_k|y_j)P(y_j)} = \frac{P(X_1|y_i)\prod\limits_{k=2}^{n} f(X_k|y_i)P(y_i)}{\sum\limits_{j=1}^{2} P(X_1|y_j)\prod\limits_{k=2}^{n} f(X_k|y_j)P(y_j)}$$

where $f(X_k|y_i), k = 2, 3, \cdots, n$ denote the conditional probability distribution functions for feature $X_k, k = 2, 3, \cdots, n$, y_i denotes the class label.
We have known that

$$f(X_k|y_i) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}, \qquad k = 2, 3, \cdots, n.$$

That is, for each $X_k (k \geqslant 2)$, we have to estimate two parameters σ_k and μ_k. Therefore, there are $2(n-1)$ parameters to be estimated in total. Then, the classifier can be written as

$$P(Y = y_i|\boldsymbol{X}) = \frac{P(X_1|y_i)\prod\limits_{k=2}^{n} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}} P(y_i)}{\sum\limits_{j=1}^{2} P(X_1|y_j)\prod\limits_{k=2}^{n} \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}} P(y_j)}$$

The remaining terms $P(X_1|y_i)$, $P(y_i)$ can be obtained by computing frequency.

2. Consider a simple learning problem of determining whether Alice and Bob from CA will go to hiking or not $Y : \mathit{Hike} \in \{T, F\}$ given the weather conditions $X_1 : \mathit{Sunny} \in \{T, F\}$ and $X_2 : \mathit{Windy} \in \{T, F\}$ by a Naive Bayes classifier. Using training data, we estimated the parameters $P(\mathit{Hike}) = 0.5$, $P(\mathit{Sunny}|\mathit{Hike}) = 0.9$, $P(\mathit{Windy}|\neg\mathit{Hike}) = 0.8$, $P(\mathit{Windy}|\mathit{Hike}) = 0.3$ and $P(\mathit{Sunny}|\neg\mathit{Hike}) = 0.4$. Assume that the true distribution of X_1, X_2, and Y satisfies the Naive Bayes assumption of conditional independence with the above parameters.

2.1. Assume Sunny and Windy are truly independent given Hike. Write down the Naive Bayes decision rule for this problem using both attributes Sunny and Windy.

2.2. Given the decision rule above, what is the expected error rate of the Naive Bayes classifier? (The expected error rate is the probability that each class generates an observation where the decision rule is incorrect.)

2.3. What is the joint probability that Alice and Bob go to hiking and the weather is sunny and windy, that is $P(Sunny, Windy, Hike)$?

2.4. Next, suppose that we gather more information about weather conditions and introduce a new feature denoting whether the weather is X_3: Rainy or not. Assume that each day the weather in CA can be either Rainy or Sunny. That is, it can not be both Sunny and Rainy (similarly, it can not be $\neg Sunny$ and $\neg Rainy$). In the above new case, are any of the Naive Bayes assumptions violated? Why (not)? What is the joint probability that Alice and Bob go to hiking and the weather is sunny, windy and not rainy, that is $P(Sunny, Windy, \neg Rainy, Hike)$?

2.5. What is the expected error rate when the Naive Bayes classifier uses all three attributes? Does the performance of Naive Bayes improve by observing the new attribute Rainy? Explain why.

Solution:
2.1 Note that

$$l(\boldsymbol{X}) = \frac{P(\boldsymbol{X}|Hike)}{P(\boldsymbol{X}|\neg Hike)} = \frac{P(X_1|Hike)P(X_2|Hike)}{P(X_1|\neg Hike)P(X_2|\neg Hike)}$$
$$\frac{P(\neg Hike)}{P(Hike)} = \frac{0.5}{0.5} = 1.$$

Thus, the Naive Bayes decision rule is given by

$$\begin{aligned} if \quad & l(\boldsymbol{X}) > 1, Y = T; \\ if \quad & l(\boldsymbol{X}) < 1, Y = F. \end{aligned}$$

Since

$$\begin{aligned} l((Sunny, Windy)) &= \frac{0.9 \times 0.3}{0.4 \times 0.8} < 1 \\ l((Sunny, \neg Windy)) &= \frac{0.9 \times 0.7}{0.4 \times 0.2} > 1 \\ l((\neg Sunny, Windy)) &= \frac{0.1 \times 0.3}{0.6 \times 0.8} < 1 \\ l((\neg Sunny, \neg Windy)) &= \frac{0.1 \times 0.7}{0.6 \times 0.2} < 1 \end{aligned}$$

we can infer that Alice and Bob will go to hiking only when it's sunny and not windy. otherwise, they will not go to hiking.

2.2 The expected error rate is given by

$$\begin{aligned} P(e) = {} & P(Hike)[P((Sunny, Windy)|Hike) + P((\neg Sunny, \neg Windy)|Hike) \\ & + P((\neg Sunny, Windy)|Hike)] + P(\neg Hike)P((Sunny, \neg Windy)|\neg Hike) \\ & = 0.5 \times 0.37 + 0.5 \times 0.08 = 0.225. \end{aligned}$$

2.3 The point probability is given by

$$\begin{aligned} P(Sunny, Windy, Hike) &= P(Sunny, Windy|Hike)P(Hike) \\ &= P(Sunny|Hike)P(Windy|Hike)P(Hike) \\ &= 0.9 \times 0.3 \times 0.5 = 0.135. \end{aligned}$$

2.4 The introduction of the new feature X_3 denoting whether the weather is rainy or not violets Naive Bayes assumption of conditional independence of each feature. It's easy to see that Rainy and Sunny are mutual exclusive and further, they are complementary events.

Since $X3 = \neg Rainy \longrightarrow X_1 = Sunny$, the feature X_3 is redundant and the point probability $P(Sunny, Windy, \neg Rainy, Hike)$ is given by

$$P(Sunny, Windy, \neg Rainy, Hike) = P(Sunny, Windy, Hike) = 0.135.$$

2.5 When the naive classifier uses all three attributes, we can obtain the decision rule and the corresponding error rate in the same way as we have done in the previous part. Here we omit the derivation process and give the error rate directly:

$$P(e) = P(Hike)P_{Hike}(e) + P(\neg Hike)P_{\neg Hike}(e) = 0.5 \times 0.064 + 0.5 \times 0.256 = 0.16 < 0.225.$$

We can see that the error rate becomes smaller when introducing the new feature X_3: Rainy or not. Adding feature X_3 can be viewed as using feature X_1 twice when training the classifier, that is, we give more weight to the feature: Sunny or not and this feature gives us more information to decide whether to go to hiking. Thus, the corresponding classifier appears to have stronger distinguishing ability.

Programming

In this problem you will implement Naive Bayes and Logistic Regression, then compare their performance on a document classification task. The data for this task is taken from the 20 Newsgroups data set, and is available from the attached zip file. The included README.txt describes the data set and file format.

Our Naive Bayes model will use the bag-of-words assumption. This model assumes that each word in a document is drawn independently from a multinomial distribution over possible words. (A multinomial distribution is a generalization of a Bernoulli distribution to multiple values.) Although this model ignores the ordering of words in a document, it works surprisingly well for a number of tasks. We number the words in our vocabulary from 1 to m, where m is the total number of distinct words in all of the documents. Documents from class y are drawn from a class-specific multinomial distribution parameterized by θ_y. θ_y is a vector, where $\theta_{y,i}$ is the probability of drawing word i and $\sum_{i=1}^{m} \theta_{y,i} = 1$. Therefore, the class-conditional probability of drawing document x from our Naive Bayes model is $P(X = x|Y = y) = \prod_{i=1}^{m} (\theta_{y,i})^{count_i(x)}$, where $count_i(x)$ is the number of times word i appears in x.

1. Provide high-level descriptions of the Naive Bayes and Logistic Regression algorithms. Be sure to describe how to estimate the model parameters and how to classify a new example.

2. Imagine that a certain word is never observed in the training data, but occurs in a test instance. What will happen when our Naive Bayes classifier predicts the probability of the this test instance? Explain why this situation is undesirable. How to avoid this problem? Will logistic regression have a similar problem? Why or why not?

3. Implement Logistic Regression and Naive Bayes. Use add-one smoothing when estimating the parameters of your Naive Bayes classifier. For logistic regression, we found that a step size around 0.0001 worked well. Train both models on the provided training data and predict the labels of the test data. Report the training and test error of both models. Submit your code along with your homework.

4. Which model performs better on this task? Why do you think this is the case?

Solution:
1. **For Naive Bayes model:**

1) Compute the total number of distinct words in the dictionary, denoted as m.

2) For each class, there are m parameters to be estimated, i.e., $\theta_{y,i}, i = 1, 2, \cdots, m$. The parameters of our multinomial Naive Bayes model can be estimated by

$$\hat{\theta}_{y,i} = P(x_k = i|Y = y) = \frac{\sum_{j=1}^{N} \sum_{k=1}^{n_j} l(Y^{(j)} = y \text{ and } x_k^{(j)} = i)}{\sum_{j=1}^{N} l(Y^{(j)} = y) n_j}$$

   $\hat{\theta}_{y,i}$ estimates the probability that a particular word in a a class-y document will be the i-th in the dictionary.
   N denotes the total number of documents in our training set.
   The $j-th$ document contains n_j words.

3) Compute class prior probability $P(Y = y)$ using frequency.

$$P(Y = y) = \frac{\sum_{j=1}^{N} l(Y^{(j)} = y)}{N}.$$

4) For a new test instance $\mathbf{x}^*$, we can decide its class by

$$y = arg \max_y ln \prod_{i=1}^{m} (\hat{\theta}_{y,i})^{count_i(\mathbf{x}^*)} P(Y = y) = arg \max_y \sum_{i=1}^{m} count_i(\mathbf{x}^*) ln \hat{\theta}_{y,i} + ln P(Y = y)$$

For Logistic Regression algorithms:
This is a multi-class classification problem. Therefore, we will use softmax regression, which is a generalization of logistic regression for binary classification.
In softmax regression, the probability that the i-th document belongs to the m-th class is given by

$$p(Y^{(i)} = m|\mathbf{x}^{(i)}; \theta) = \frac{e^{\boldsymbol{\theta}_m^T \mathbf{x}^{(i)}}}{\sum\limits_{j=1}^{k} e^{\boldsymbol{\theta}_j^T \mathbf{x}^{(i)}}},$$

where k denotes the total number of classes.
The cost function is given by

$$J(\theta) = -\frac{1}{N} [\sum_{i=1}^{N} \sum_{j=1}^{k} l(y^{(i)} = j) log \frac{e^{\boldsymbol{\theta}_j^T \mathbf{x}^{(i)}}}{\sum\limits_{l=1}^{k} e^{\boldsymbol{\theta}_l^T \mathbf{x}^{(i)}}}]$$

1) We could use gradient-descent algorithm to minimize $J(\theta)$ and to obtain corresponding parameters $\hat{\boldsymbol{\theta}}_k$.

2) For a test instance, we could decide its class m by

$$m = arg \max_m \frac{e^{\boldsymbol{\theta}_m^T \mathbf{x}^{(i)}}}{\sum\limits_{j=1}^{k} e^{\boldsymbol{\theta}_j^T \mathbf{x}^{(i)}}},$$

2. If a certain word i^* is never observed in the training data, but occurs in a test in a test instance $\mathbf{x}^*$, our Naive Bayes classifier will predict a zero probability of this test instance for all classes. The reason for the undesirability of a new word i^* is that $\hat{\theta}_{y,i^*}$ will be zero for all classes y if the word i^* isn't observed when training the model and $P(\mathbf{x}^*|y)P(y)$ in the form of product will be zero.

There are many smoothing techniques that have been proposed to address the problem, such as, add-one smoothing(assuming each word in the universal set appear at least once), good-turing discounting and so on.

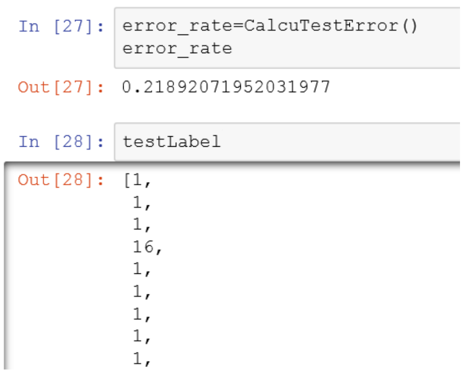
New words in the test instance is a problem for Naive Bayes classifier, but not for Logistic regression, because the logistic function is not in the form of product and one parameter equaling zero will not result in the zero of logistic function.

3. **Naive Bayes Classifier:**
If we use add-one smoothing, the parameters of our multinomial Naive Bayes model becomes

$$\hat{\theta}_{y,i} = P(x_k = i|Y = y) = \frac{\sum_{j=1}^{N}\sum_{k=1}^{n_j} l(Y^{(j)} = y \text{ and } x_k^{(j)} = i) + 1}{\sum_{j=1}^{N} l(Y^{(j)} = y)n_j + m},$$

where m denotes the total number of distinct words in our dictionary.

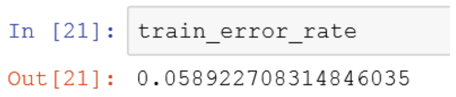
As the following picture indicates, the test error rate is 21.89% and the train error rate is 5.89%.



```
In [27]: error_rate=CalcuTestError()
         error_rate

Out[27]: 0.21892071952031977

In [28]: testLabel

Out[28]: [1,
          1,
          1,
          16,
          1,
          1,
          1,
          1,
          1,
```

Figure 1: Error rate for Naive Bayes

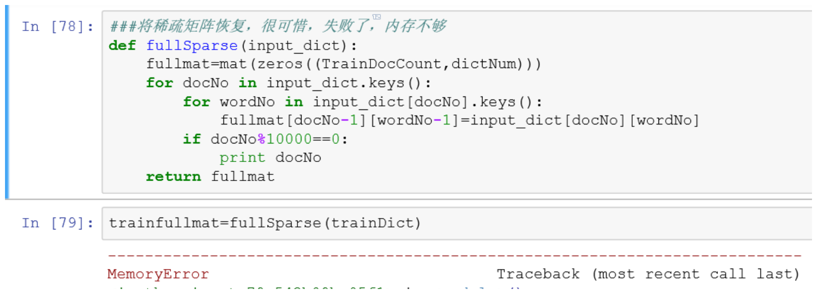
Softmax Classifier:



```
In [21]: train_error_rate

Out[21]: 0.058922708314846035
```

Figure 2: Error rate for Naive Bayes

Softmax Classifier:
I am sorry. I have tried my best but I didn't make it. I thought I could convert the original file in the form of docNo-wordNo-wordCount to a sparse matrix so that it could be easily addressed in the following softmax training process. However, the matrix is so large (11269×61188) that it exceeds my computer's processing capacity. Then I thought I could use a dictionary to store it, a data structure that can be indexed through class-label and documentNo as a matrix can do. Then new problems arose: dictionary is not convenient for the parameters estimation process since the number of distinct words that each class used in the training set is different. I didn't complete the softmax regression coding finally. The attachment of "Logistic Regression" is not complete. It's a pity.



```
In [78]: ###将稀疏矩阵恢复，很可惜，失败了，内存不够
         def fullSparse(input_dict):
             fullmat=mat(zeros((TrainDocCount,dictNum)))
             for docNo in input_dict.keys():
                 for wordNo in input_dict[docNo].keys():
                     fullmat[docNo-1][wordNo-1]=input_dict[docNo][wordNo]
                 if docNo%10000==0:
                     print docNo
             return fullmat

In [79]: trainfullmat=fullSparse(trainDict)

         ---------------------------------------------------------------------------
         MemoryError                               Traceback (most recent call last)
```

Figure 3: Memory error Bayes

4. Although I didn't complete the logistic regression, I predict that logistic will perform better. Because Compared with Naive Bayes, logistic doesn't assume that features are independent given class.