Supplementary problems for CS229 public course

Xiaozhu Zhang

Fall 2020

Contents

1 Probability

1.1 Show that $\mathbb{E}(\mathbb{E}(g(X,Y)|X)) = \mathbb{E}(g(X,Y))$.

[The following is referred from Prof. Zhihua Zhang.]

The rule of iterated expectation. Assume X and Y's expectation exist, then for $g(x,y)$, show that $\mathbb{E}(\mathbb{E}(g(X,Y)|X)) = \mathbb{E}(g(X,Y))$.

Since

$$\mathbb{E}(g(X,Y)|X=x) = \int_{-\infty}^{+\infty} g(x,y)f(y|x)dy,$$

we have

$$\begin{aligned}
\mathbb{E}(\mathbb{E}(g(X,Y)|X)) &= \int_{-\infty}^{+\infty} \mathbb{E}(g(X,Y)|X=x)f(x)dx \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x,y)f(y|x)f(x)dydx \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(x,y)f(x,y)dxdy \\
&= \mathbb{E}(g(X,Y)).
\end{aligned} \tag{1}$$

Specifically, we could derive the corollary that

$$\mathbb{E}(\mathbb{E}(Y|X)) = \mathbb{E}(Y)$$

and

$$\mathbb{E}(\mathbb{E}(X|Y)) = \mathbb{E}(X).$$

1.2 Show that $\mathrm{Var}(Y) = \mathbb{E}(\mathrm{Var}(Y|X)) + \mathrm{Var}(\mathbb{E}(Y|X))$.

[The following is also referred from Prof. Zhihua Zhang.]

X and Y is random variable, show that $\mathrm{Var}(Y) = \mathbb{E}(\mathrm{Var}(Y|X)) + \mathrm{Var}(\mathbb{E}(Y|X))$.

We could rewrite the expectation of variance and the variance of expectation as

$$\mathbb{E}(\mathrm{Var}(Y|X)) = \mathbb{E}_X\left[\mathbb{E}_Y\left(Y|X - \mathbb{E}_Y(Y|X)\right)^2\right]$$

and

$$\mathrm{Var}(\mathbb{E}(Y|X)) = \mathbb{E}_X\left[\left(\mathbb{E}_Y(Y|X) - \mathbb{E}_Y(Y)\right)^2\right].$$

Thus,

$$\begin{aligned}
&\mathbb{E}(\mathrm{Var}(Y|X)) + \mathrm{Var}(\mathbb{E}(Y|X)) && (2)\\
&= \mathbb{E}_X\left[\mathbb{E}_Y\left(Y|X - \mathbb{E}_Y(Y|X)\right)^2\right] + \mathbb{E}_X\left[\left(\mathbb{E}_Y(Y|X) - \mathbb{E}_Y(Y)\right)^2\right] && (3)\\
&= \mathbb{E}_X\left[\mathbb{E}_Y\left(Y|X - \mathbb{E}_Y(Y|X)\right)^2 + \left(\mathbb{E}_Y(Y|X) - \mathbb{E}_Y(Y)\right)^2\right] && (4)\\
&= \mathbb{E}_X\left[\mathbb{E}_Y(Y^2|X) - 2\mathbb{E}_Y^2(Y|X) + \mathbb{E}_Y^2(Y|X) + \mathbb{E}_Y^2(Y|X) - 2\mathbb{E}_Y(Y|X)\mathbb{E}_Y(Y) + \mathbb{E}_Y^2(Y)\right] && (5)\\
&= \mathbb{E}_X\left[\mathbb{E}_Y(Y^2|X)\right] - 2\mathbb{E}_X\left[\mathbb{E}_Y(Y|X)\mathbb{E}_Y(Y)\right] + \mathbb{E}_Y^2(Y) && (6)\\
&= \mathbb{E}_Y(Y^2) - 2\mathbb{E}_Y^2(Y) + \mathbb{E}_Y^2(Y) && (7)\\
&= \mathbb{E}_Y(Y^2) - \mathbb{E}_Y^2(Y) && (8)\\
&= \mathrm{Var}(Y). && (9)
\end{aligned}$$

Furthermore, we could derive that

$$\mathrm{Var}(Y) \geq \mathrm{Var}(\mathbb{E}(Y|X)),$$

which is helpful when we deal with the future problems of bias-variance trade-off.

1.3 Show that the covariance matrix is singular when $n \leq p$.

Let us begin by clarifying the notation. The covariance matrix here is the sample covariance (an estimation of the population covariance):

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}\left(x^{(i)} - \bar{x}\right)\left(x^{(i)} - \bar{x}\right)^T$$

In order to show that $\hat{\Sigma}$ is singular when $n \leq p$, it suffices to show that $\mathrm{rank}(\hat{\Sigma}) \leq n-1$. Let $z_i = x^{(i)} - \bar{x}$ and note that $\sum_{i=1}^{n} z_i = 0$, we could rewrite $\hat{\Sigma}$ as

$$\begin{aligned}
\hat{\Sigma} &= \frac{1}{n-1}\sum_{i=1}^{n} z_i z_i^T \\
&= \frac{1}{n-1}\left[\sum_{i=1}^{n-1} z_i z_i^T + z_n z_n^T\right] \\
&= \frac{1}{n-1}\left[\sum_{i=1}^{n-1} z_i z_i^T + \left(-\sum_{i=1}^{n} z_i\right) z_n^T\right] \\
&= \frac{1}{n-1}\left[\sum_{i=1}^{n-1} z_i (z_i - z_n)^T\right].
\end{aligned} \tag{10}$$

Thus $\hat{\Sigma}$ is of the summation of $n-1$ independent rank-1 matrices, and as a result $\mathrm{rank}(\hat{\Sigma}) \leq n-1$.

Note that we use the fraction $\frac{1}{n-1}$ here instead of $\frac{1}{n}$, which could be enlightened by the rank of sample covariance matrix. Here we offer a different explanation which leads to the same result.

Consider the expectation of the summation:

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{n}\left(x^{(i)}-\bar{x}\right)\left(x^{(i)}-\bar{x}\right)^{T}\right] &= \mathbb{E}\left[\sum_{i=1}^{n}x^{(i)}x^{(i)T}-n\bar{x}\bar{x}^{T}\right] \\ &= \sum_{i=1}^{n}\mathbb{E}\left[x^{(i)}x^{(i)T}\right]-n\mathbb{E}\left[\bar{x}\bar{x}^{T}\right] \\ &= n\left(\Sigma+\mu\mu^{T}\right)-n\left(\frac{\Sigma}{n}+\mu\mu^{T}\right) \\ &= (n-1)\Sigma.\end{aligned} \tag{11}$$

Therefore, $\mathbb{E}(\hat{\Sigma}) = \Sigma$, in other words, $\hat{\Sigma}$ is an unbiased estimator of the population covariance matrix Σ. Both the upper bound of $\text{rank}(\hat{\Sigma})$ and the multiples of Σ in the expectation above, inspire a more essential idea, **the degree of freedom**.

2 Intro to supervised learning (GLM, GDA, Naive Bayes)

2.1 Show that Poisson is a member of the exponential family.

The PDF of Poisson distribution is:

$$\begin{aligned}P(y) &= \frac{e^{-\lambda}\lambda^{y}}{y!} \\ &= \frac{1}{y!}\exp(y\log\lambda-\lambda).\end{aligned} \tag{12}$$

Thus,

$$\begin{aligned}b(y) &= \frac{1}{y!} \\ T(y) &= y \\ \eta &= \log\lambda, (\lambda = e^{\eta}) \\ a(\eta) &= \lambda = e^{\eta},\end{aligned}$$

and Poisson distribution is a member of exponential family. If we want to construct a GLM whose y follows Poisson distribution, then let $\eta = \theta^{T}x$ and

$$g(\eta) = \frac{\partial a(\eta)}{\partial\eta} = e^{\eta},$$

and we got

$$h_{\theta}(x) = e^{\theta^{T}x}.$$

We could use the algorithm

$$\theta_{j} := \theta_{j} + \alpha(y^{(i)} - h_{\theta}(x^{(i)}))x_{j}^{(i)}$$

to find the θ that fits the model.

2.2 Show that $\frac{\partial a(\eta)}{\partial \eta} = \mathbb{E}[Y;\eta]$ and $\frac{\partial^2 a(\eta)}{\partial \eta^2} = \mathrm{Var}[Y;\eta]$.

[The following is referred from Berkeley.]

Given a measure ν, we define an exponential family of probability distributions as those distributions whose density (relative to η) have the following general form:

$$p(y;\eta) = b(y)\exp\left\{\eta^T T(y) - A(\eta)\right\}.$$

Intergrating the equation above with respect to the measure ν, we have:

$$A(\eta) = \log \int b(y)\exp\left\{\eta^T T(y)\right\}\nu(dy).$$

Let us consider computing the first derivative of $A(\eta)$ for a general exponential family distribution:

$$\begin{aligned}
\frac{\partial A}{\partial \eta^T} &= \frac{\partial}{\partial \eta^T}\left\{\log \int \exp\ \left\{\eta^T T(y)\right\} b(y)\nu(dy)\right\}. \\
&= \frac{\int T(y)\exp\left\{\eta^T T(y)\right\} b(y)\nu(dy)}{\int \exp\left\{\eta^T T(y)\right\} b(y)\nu(dy)} \\
&= \int T(y)\exp\left\{\eta^T T(y) - A(\eta)\right\} b(y)\nu(dy) \\
&= \mathbb{E}[Y;\eta].
\end{aligned} \tag{13}$$

Let us now take a second derivative:

$$\begin{aligned}
\frac{\partial^2 A}{\partial \eta \partial \eta^T} &= \int T(y)\left[T(y) - \frac{\partial A}{\partial \eta^T}\right]^T \exp\left\{\eta^T T(y) - A(\eta)\right\} b(y)\nu(dy) \\
&= \int T(y)\left[T(y) - \mathbb{E}[T(Y)]\right]^T \exp\left\{\eta^T T(y) - A(\eta)\right\} b(y)\nu(dy) \\
&= \mathbb{E}\left[T(Y)T(T)^T\right] - \mathbb{E}[T(Y)]\mathbb{E}[T(Y)]^T \\
&= \mathrm{Var}[Y;\eta].
\end{aligned} \tag{14}$$

2.3 Show that all iteration algorithms for GLM are of the form: $\theta_j := \theta_j + \left[y^{(i)} - h_\theta\left(x^{(i)}\right)\right] x_j^{(i)}$.

Here we only consider the case when $T(y) = y$. The likelihood of GLM is of the following form if we replace η with $\theta^T x$:

$$\mathcal{L}_i(\theta) = b\left(y^{(i)}\right)\exp\left[x^{(i)T}\theta y^{(i)} - a\left(\theta^T x^{(i)}\right)\right].$$

Take logarithm and we could get:

$$\ell_i(\theta) = \log b\left(y^{(i)}\right) + x^{(i)T}\theta y^{(i)} - a\left(\theta^T x^{(i)}\right).$$

Now let us take the first derivation of $\ell_i(\theta)$ with respect to θ:

$$\begin{aligned}
\nabla_\theta \ell_i(\theta) &= x^{(i)}y^{(i)} - h_\theta\left(x^{(i)}\right)x^{(i)} \\
&= \left[y^{(i)} - h_\theta\left(x^{(i)}\right)\right]x^{(i)},
\end{aligned} \tag{15}$$

and to be more specific,

$$\nabla_{\theta_j} \ell_i(\theta) = \left[y^{(i)} - h_\theta \left(x^{(i)} \right) \right] x_j^{(i)}.$$

Therefore, the iteration algorithms are of the form

$$\begin{aligned} \theta_j &:= \theta_j + \alpha \frac{\partial \ell_i}{\partial \theta_j} \\ &= \theta_j + \alpha \left[y^{(i)} - h_\theta \left(x^{(i)} \right) \right] x_j^{(i)}. \end{aligned} \tag{16}$$

2.4 Show how to perform Newton's algorithm when the Hessian matrix is singular.

Generally speaking, we will resort to MAP (Bayesian method) to perform Newton's algorithm when the Hessiain matrix is singular. The estimator θ is no longer a constant anymore; instead, we treat it as a random variable with a distribution. In other words, we use regularization method if we speak in terms of the log-likelihood (check 3.8 to learn the equivalence of regularization and Bayesian prior).

Without losing of generality, we would like to illustrate this by considering the *logistic regression* problem when H is non-invertible. With

$$h_\theta(x) = g\left(\theta^T x\right) = \frac{1}{1 + \exp\left(-\theta^T x\right)},$$

and

$$p(\theta|\lambda) = \exp\left(-\lambda ||\theta||_2^2\right), \quad \lambda > 0$$

the MAP is

$$\begin{aligned} \mathcal{P} &= \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)};\theta\right) p(\theta|\lambda) \\ &= \prod_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) \right)^{y^{(i)}} \left(1 - h_\theta \left(x^{(i)} \right) \right)^{1-y^{(i)}} \exp\left(-\lambda ||\theta||_2^2\right). \end{aligned} \tag{17}$$

Thus we have

$$\ell = \log \mathcal{P} = \sum_{i=1}^{m} \left[y^{(i)} \log h_\theta \left(x^{(i)} \right) + \left(1 - y^{(i)} \right) \log \left(1 - h_\theta \left(x^{(i)} \right) \right) \right] - \lambda ||\theta||_2^2.$$

Now let us take the first derivative of ℓ *w.r.t.* θ:

$$\begin{aligned} \nabla_\theta \ell =& \sum_{i=1}^{m} \left[y^{(i)} \frac{h_\theta \left(x^{(i)} \right) \left(1 - h_\theta \left(x^{(i)} \right) \right)}{h_\theta \left(x^{(i)} \right)} - \left(1 - y^{(i)} \right) \frac{h_\theta \left(x^{(i)} \right) \left(1 - h_\theta \left(x^{(i)} \right) \right)}{1 - h_\theta \left(x^{(i)} \right)} \right] x^{(i)} - 2\lambda\theta \\ =& \sum_{i=1}^{m} \left[y^{(i)} - h_\theta \left(x^{(i)} \right) \right] x^{(i)} - 2\lambda\theta \\ =& X^T \left(y - g(X^T\theta) \right) - 2\lambda\theta, \end{aligned} \tag{18}$$

where $X \in \mathbb{R}^{m \times n}$ is the design matrix, $y \in \mathbb{R}^m$ is the label vector, and $g(X^T\theta) \in \mathbb{R}^m$ is of the form $\left(g\left(\theta^T x^{(1)}\right), ..., g\left(\theta^T x^{(m)}\right) \right)^T$.

Next let us take the second derivative of ℓ *w.r.t.* θ:

$$\begin{aligned}\nabla_{\theta\theta^T}\ell &= \nabla_\theta \left(\nabla_\theta \ell\right)^T \\ &= \nabla_\theta \left[y^T - g(X^T\theta)^T\right] X - 2\lambda\theta^T \\ &= -X^T D X - 2\lambda I_n \\ &= H,\end{aligned} \tag{19}$$

where D is a diagonal matrix such that

$$D_{ii} = \text{diag}\left[g\left(\theta^T x^{(1)}\right)\left(1 - g\left(\theta^T x^{(1)}\right)\right), ..., g\left(\theta^T x^{(m)}\right)\left(1 - g\left(\theta^T x^{(1)}\right)\right)\right].$$

Finally, we need to show that H is **non-sigular**. Since

$$X^T D X = X^T D^{-\frac{1}{2}} D^{-\frac{1}{2}} X = \left(D^{-\frac{1}{2}} X\right)^T \left(D^{-\frac{1}{2}} X\right),$$

we conclude that $X^T DX$ is positive semi-definite. As a result, if we perform spectral decomposition on $X^T DX = U^T \Lambda U$, then there must be that $\Lambda_{ii} \geq 0$ for all $i \in \{1, ..., n\}$. Now we can rewirte H as

$$H = -\left(X^T DX + 2\lambda I_n\right) = -\left(U^T \Lambda U + 2\lambda I_n\right) = -U^T\left(\Lambda + 2\lambda I_n\right) U.$$

Given that $\lambda > 0$ and $\Lambda_{ii} \geq 0$, we know that all eigenvalues of the Hessian matrix are nonzero; in other words, H is invertible.

2.5 Derive the gradient ascent algorithm and Newton's algorithm for softmax regression.

We begin by writing down the log-likelihood

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^m \log p\left(y^{(i)} | x^{(i)}; \theta\right) \\ &= \sum_{i=1}^m \log \phi_1^{1\left\{y^{(i)}=1\right\}} \phi_2^{1\left\{y^{(i)}=2\right\}} \ldots \phi_k^{1\left\{y^{(i)}=k\right\}} \\ &= \sum_{i=1}^m \log \prod_{l=1}^k \left(\frac{\exp\left(\theta_l^T x^{(i)}\right)}{\sum_{j=1}^k \exp\left(\theta_j^T x^{(i)}\right)}\right)^{1\left\{y^{(i)}=l\right\}} \\ &= \sum_{i=1}^m \sum_{l=1}^k 1\left\{y^{(i)} = l\right\} \left[\theta_l^T x^{(i)} - \log \sum_{j=1}^k \exp\left(\theta_j^T x^{(i)}\right)\right] \\ &= \sum_{i=1}^m \sum_{l=1}^k 1\left\{y^{(i)} = l\right\} \theta_l^T x^{(i)} - \sum_{i=1}^m \sum_{l=1}^k 1\left\{y^{(i)} = l\right\} \log \sum_{j=1}^k \exp\left(\theta_j^T x^{(i)}\right).\end{aligned} \tag{20}$$

Now let us take the first derivative of $\ell(\theta)$:

$$\begin{aligned}\nabla_{\theta_p}\ell(\theta) &= \sum_{i=1}^{m} 1\left\{y^{(i)}=p\right\}x^{(i)} - \sum_{i=1}^{m}\sum_{l=1}^{k} 1\left\{y^{(i)}=l\right\}\frac{\exp\left(\theta_p^T x^{(i)}\right)x^{(i)}}{\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)} \\ &= \sum_{i=1}^{m}\left[1\left\{y^{(i)}=p\right\} - \sum_{l=1}^{k} 1\left\{y^{(i)}=l\right\}\frac{\exp\left(\theta_p^T x^{(i)}\right)}{\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)}\right]x^{(i)},\end{aligned} \tag{21}$$

and this therefore gives the **gradient ascent** rule for any θ_p

$$\theta_p := \theta_p + \alpha\sum_{i=1}^{m}\left[1\left\{y^{(i)}=p\right\} - \sum_{l=1}^{k} 1\left\{y^{(i)}=l\right\}\frac{\exp\left(\theta_p^T x^{(i)}\right)}{\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)}\right]x^{(i)}.$$

Next, in order to solve for the Hessian matrix, let us take the second derivative of $\ell(\theta)$:

$$\begin{aligned}\nabla^2_{\theta_p,\theta_p^T}\ell(\theta) &= \sum_{i=1}^{m}\sum_{l=1}^{k} 1\left\{y^{(i)}=l\right\}\left\{\frac{\exp\left(2\theta_p^T x^{(i)}\right)}{\left[\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)\right]^2} - \frac{\exp\left(\theta_p^T x^{(i)}\right)}{\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)}\right\}x^{(i)}x^{(i)T} \\ &= X^T D X \\ &= H,\end{aligned} \tag{22}$$

where $D \in \mathbb{R}^{m\times m}$ is a diagonal matrix with

$$D_{ii} = \sum_{l=1}^{k} 1\left\{y^{(i)}=l\right\}\left\{\frac{\exp\left(2\theta_p^T x^{(i)}\right)}{\left[\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)\right]^2} - \frac{\exp\left(\theta_p^T x^{(i)}\right)}{\sum_{j=1}^{k}\exp\left(\theta_j^T x^{(i)}\right)}\right\}.$$

Since there are k canonical parameters $\phi_1, ..., \phi_k$ and the constraint $\sum_{p=1}^{k}\phi_p = 1$, we actually have $k-1$ such Hessian matrices.

Finally, for any θ_p, the **Newton's algorithm** can be expressed as

$$\theta_p := \theta_p - H^{-1}\nabla_{\theta_p}\ell(\theta).$$

2.6 Show how to find the maximum likelihood estimators for GDA.

Since

$$\begin{aligned}p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left[-\frac{(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}{2}\right] \\ p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left[-\frac{(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}{2}\right],\end{aligned}$$

the log-likelihood is

$$\begin{aligned}\ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\right) \\ &= \log \prod_{i=1}^{m} p\left(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma\right) p\left(y^{(i)}; \phi\right) \\ &= \sum_{i=1}^{m} \left[\log p\left(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma\right) + \log p\left(y^{(i)}; \phi\right)\right].\end{aligned} \tag{23}$$

1. To solve for ϕ,

$$\begin{aligned}\nabla_\phi \ell &= \nabla_\phi \sum_{i=1}^{m} \log p\left(y^{(i)}; \phi\right) \\ &= \nabla_\phi \sum_{i=1}^{m} \left[y^{(i)} \log \phi + \left(1 - y^{(i)}\right) \log(1 - \phi)\right] \\ &= \sum_{i=1}^{m} \left[y^{(i)} \frac{1}{\phi} - \left(1 - y^{(i)}\right) \frac{1}{1 - \phi}\right].\end{aligned} \tag{24}$$

   Setting $\nabla_\phi \ell = 0$ gives

$$\begin{aligned}0 &= \sum_{i=1}^{m} \left[y^{(i)}(1 - \phi) - \left(1 - y^{(i)}\right) \phi\right] \\ &= \sum_{i=1}^{m} \left[y^{(i)} - \phi\right] \\ &= \sum_{i=1}^{m} y^{(i)} - m\phi,\end{aligned} \tag{25}$$

   which is

$$\phi = \frac{\sum_{i=1}^{m} y^{(i)}}{m}.$$

2. To solve for μ_0,

$$\begin{aligned}\nabla_{\mu_0} \ell &= \nabla_{\mu_0} \sum_{i=1}^{m} \log p\left(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma\right) \\ &= \nabla_{\mu_0} \sum_{i=1}^{m} \left[-\frac{\left(x^{(i)} - \mu_0\right)^T \Sigma^{-1} \left(x^{(i)} - \mu_0\right)}{2}\right] 1\left\{y^{(i)} = 0\right\} \\ &= \nabla_{\mu_0} \sum_{i=1}^{m} \left[-\frac{2\mu_0^T \Sigma^{-1} x^{(i)} - \mu_0^T \Sigma^{-1} \mu_0 - x^{(i)T} \Sigma^{-1} x^{(i)}}{2}\right] 1\left\{y^{(i)} = 0\right\} \\ &= \sum_{i=1}^{m} \left(\Sigma^{-1} \mu_0 - \Sigma^{-1} x^{(i)}\right) 1\left\{y^{(i)} = 0\right\}\end{aligned} \tag{26}$$

Setting $\nabla_{\mu_0}\ell = 0$ gives

$$\begin{aligned} 0 &= \sum_{i=1}^{m} \left(\mu_0 - x^{(i)}\right) 1\left\{y^{(i)} = 0\right\} \\ &= \mu_0 \sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\} - \sum_{i=1}^{m} x^{(i)} 1\left\{y^{(i)} = 0\right\}, \end{aligned} \tag{27}$$

which is

$$\mu_0 = \frac{\sum_{i=1}^{m} x^{(i)} 1\left\{y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\}}.$$

3. The solution for μ_1 proceeds in the identical manner, which is

$$\mu_0 = \frac{\sum_{i=1}^{m} x^{(i)} 1\left\{y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}}.$$

4. To solve for Σ,

$$\begin{aligned} \nabla_\Sigma \ell &= \nabla_\Sigma \sum_{i=1}^{m} \log p\left(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma\right) \\ &= \nabla_\Sigma \left[-\frac{m}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{m} \left(x^{(i)} - \mu_{y^{(i)}}\right)^T \Sigma^{-1} \left(x^{(i)} - \mu_{y^{(i)}}\right) \right] \\ &= -\frac{m}{2}\frac{|\Sigma|\Sigma^{-1}}{|\Sigma|} + \frac{1}{2}\sum_{i=1}^{m} \Sigma^{-1}\left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T \Sigma^{-1}. \end{aligned} \tag{28}$$

The last step holds due to the general formula

$$\frac{\partial a^T X^{-1} b}{\partial X} = -X^{-T} a b^T X^{-T}.$$

Setting $\nabla_\Sigma \ell = 0$ gives

$$0 = -m\Sigma^{-1} + \sum_{i=1}^{m} \Sigma^{-1}\left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T \Sigma^{-1} \tag{29}$$

$$\iff m\Sigma^{-1} = \sum_{i=1}^{m} \Sigma^{-1}\left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T \Sigma^{-1} \tag{30}$$

$$\iff m\Sigma = \sum_{i=1}^{m} \left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T \tag{31}$$

$$\iff \Sigma = \frac{1}{m}\sum_{i=1}^{m} \left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T. \tag{32}$$

2.7 Show that the quantity $p(y = 1|x; \phi, \mu_0, \mu_1, \Sigma)$ from GDA can be expressed in the form of sigmoid function.

Want:

$$p(y = 1|x; \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-\theta^T x)}.$$

Since

$$\begin{aligned}
& p(y = 1|x; \phi, \mu_0, \mu_1, \Sigma) && (33)\\
= & \frac{p(x|y = 1; \mu_1, \Sigma)p(y = 1; \phi)}{p(x|y = 1; \mu_1, \Sigma)p(y = 1; \phi) + p(x|y = 0; \mu_0, \Sigma)p(y = 0; \phi)} && (34)\\
= & \frac{\exp\left[-\frac{1}{2}(x - u_1)^T\Sigma^{-1}(x - u_1)\right]\phi}{\exp\left[-\frac{1}{2}(x - u_1)^T\Sigma^{-1}(x - u_1)\right]\phi + \exp\left[-\frac{1}{2}(x - u_0)^T\Sigma^{-1}(x - u_0)\right](1 - \phi)} && (35)\\
= & \frac{1}{1 + \frac{1-\phi}{\phi}\exp\left[2\left(\mu_0^T - \mu_1^T\right)\Sigma^{-1}x + \left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0\right)\right]} && (36)\\
= & \frac{1}{1 + \exp\left[2\left(\mu_0^T - \mu_1^T\right)\Sigma^{-1}x + \left(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0\right) + \log\frac{1-\phi}{\phi}\right]}, && (37)
\end{aligned}$$

we can conclude that

$$\theta = \begin{bmatrix} \left(\mu_0^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1\right) + \log\frac{\phi}{1-\phi} \\ 2\Sigma^{-1}(\mu_1 - \mu_0) \end{bmatrix}$$

with $x_0 = 1$.

2.8 Derive the MLE for the Naive Bayes model.

The model is parameterized by $\phi_{j|y=1} = p(x_j = 1|y = 1)$, $\phi_{j|y=0} = p(x_j = 1|y = 0)$, and $\phi_y = p(y = 1)$. We can write down the joint likelihood of the data:

$$\begin{aligned}
\mathcal{L}(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) &= \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}\right) \\
&= \prod_{i=1}^{m}\left(\prod_{j=1}^{n} p\left(x_j^{(i)}|y^{(i)}; \phi_{j|y=0}, \phi_{j|y=1}\right)\right) p\left(y^{(i)}; \phi_y\right).
\end{aligned} \tag{38}$$

Thus, the log-likelihood is

$$\ell(\phi_y, \phi_{j|y=0}, \phi_{j|y=1}) = \sum_{i=1}^{m}\left[\sum_{j=1}^{n} \log p\left(x_j^{(i)}|y^{(i)}; \phi_{j|y=0}, \phi_{j|y=1}\right) + \log p\left(y^{(i)}; \phi_j\right)\right].$$

1. To solve for ϕ_y,

$$\begin{aligned}\nabla_{\phi_y}\ell(\phi_y,\phi_{j|y=0},\phi_{j|y=1}) &= \nabla_{\phi_y}\sum_{i=1}^{m}\log p\left(y^{(i)};\phi_y\right)\\ &= \nabla_{\phi_y}\sum_{i=1}^{m}\left[y^{(i)}\log\phi_y+\left(1-y^{(i)}\right)\log(1-\phi_y)\right] \qquad (39)\\ &= \sum_{i=1}^{m}\left[y^{(i)}\frac{1}{\phi_y}-\left(1-y^{(i)}\right)\frac{1}{1-\phi_y}\right].\end{aligned}$$

Setting $\nabla_{\phi_y}\ell(\phi_y,\phi_{j|y=0},\phi_{j|y=1})=0$ gives

$$\begin{aligned}0 &= \sum_{i=1}^{m}\left[y^{(i)}(1-\phi_y)-\left(1-y^{(i)}\right)\phi_y\right]\\ &= \sum_{i=1}^{m}\left[y^{(i)}-\phi_y\right] \qquad (40)\\ &= \sum_{i=1}^{m}y^{(i)}-m\phi_y,\end{aligned}$$

which is,

$$\phi_y=\frac{\sum_{i=1}^{m}y^{(i)}}{m}.$$

2. To solve for $\phi_{l|y=1}$,

$$\nabla_{\phi_{l|y=1}}\ell(\phi_y,\phi_{j|y=0},\phi_{j|y=1}) \qquad (41)$$

$$=\nabla_{\phi_{l|y=1}}\sum_{i=1}^{m}\sum_{j=1}^{n}\log p\left(x_j^{(i)}|y^{(i)};\phi_{j|y=1}\right)1\left\{y^{(i)}=1\right\} \qquad (42)$$

$$=\nabla_{\phi_{l|y=1}}\sum_{i=1}^{m}\log p\left(x_l^{(i)}|y^{(i)};\phi_{l|y=1}\right)1\left\{y^{(i)}=1\right\} \qquad (43)$$

$$=\nabla_{\phi_{l|y=1}}\sum_{i=1}^{m}\left[1\left\{x_l^{(i)}=1\right\}\log\phi_{l|y=1}+1\left\{x_l^{(i)}\neq 1\right\}\log\left(1-\phi_{l|y=1}\right)\right]1\left\{y^{(i)}=1\right\} \qquad (44)$$

$$=\sum_{i=1}^{m}\left[1\left\{x_l^{(i)}=1\right\}\frac{1}{\phi_{l|y=1}}-1\left\{x_l^{(i)}\neq 1\right\}\frac{1}{1-\phi_{l|y=1}}\right]1\left\{y^{(i)}=1\right\}. \qquad (45)$$

Setting $\nabla_{\phi_{l|y=1}}\ell(\phi_y,\phi_{j|y=0},\phi_{j|y=1})$ gives

$$\begin{aligned}0 &= \sum_{i=1}^{m}\left[1\left\{x_l^{(i)}=1\right\}\left(1-\phi_{l|y=1}\right)-1\left\{x_l^{(i)}\neq 1\right\}\phi_{l|y=1}\right]1\left\{y^{(i)}=1\right\}\\ &= \sum_{i=1}^{m}\left[1\left\{x_l^{(i)}=1\right\}-\phi_{l|y=1}\right]1\left\{y^{(i)}=1\right\} \qquad (46)\\ &= \sum_{i=1}^{m}1\left\{x_l^{(i)}=1\wedge y^{(i)}=1\right\}-\phi_{l|y=1}\sum_{i=1}^{m}1\left\{y^{(i)}=1\right\},\end{aligned}$$

which is

$$\phi_{l|y=1} = \frac{\sum_{i=1}^{m} 1\left\{x_l^{(i)} = 1 \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 1\right\}}.$$

3. The solution for $\phi_{l|y=0}$ proceeds in the identical manner, which is

$$\phi_{l|y=0} = \frac{\sum_{i=1}^{m} 1\left\{x_l^{(i)} = 1 \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = 0\right\}}.$$

2.9 Derive the MLE for the multinomial event model.

The model is parameterized by $\phi_{k|y=0} = p(x_j = k|y = 1)$, $\phi_{k|y=1} = p(x_j = k|y = 1)$, and $\phi_j = p(y = 1)$. Specifically, $\phi_{k|y=0}$ means the chance of word j being k if $y = 0$, and it does not depend on j; $\phi_{k|y=0}$ is similar. From the expression of the likelihood

$$\begin{aligned}\mathcal{L}(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) &= \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}\right) \\ &= \prod_{i=1}^{m} \left(\prod_{j=1}^{n_i} p\left(x_j^{(i)}|y^{(i)}; \phi_{k|y=0}, \phi_{k|y=1}\right)\right) p\left(y^{(i)}; \phi_y\right),\end{aligned} \tag{47}$$

we can easily get the log-likelihood

$$\ell(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = \sum_{i=1}^{m} \left[\sum_{j=1}^{n_i} \log p\left(x_j^{(i)}|y^{(i)}; \phi_{k|y=0}, \phi_{k|y=1}\right) + \log p\left(y^{(i)}; \phi_y\right)\right].$$

1. To solve for ϕ_y,

$$\begin{aligned}\nabla_{\phi_y} \ell(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) &= \nabla_{\phi_y} \sum_{i=1}^{m} \log p\left(y^{(i)}; \phi_y\right) \\ &= \nabla_{\phi_y} \sum_{i=1}^{m} \left[y^{(i)} \log \phi_y + \left(1 - y^{(i)}\right) \log(1 - \phi_y)\right] \\ &= \sum_{i=1}^{m} \left[y^{(i)} \frac{1}{\phi_y} - \left(1 - y^{(i)}\right) \frac{1}{1 - \phi_y}\right].\end{aligned} \tag{48}$$

Setting $\nabla_{\phi_y} \ell(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = 0$ gives

$$\begin{aligned}0 &= \sum_{i=1}^{m} \left[y^{(i)}(1 - \phi_y) - \left(1 - y^{(i)}\right) \phi_y\right] \\ &= \sum_{i=1}^{m} \left[y^{(i)} - \phi_y\right] \\ &= \sum_{i=1}^{m} y^{(i)} - m\phi_y,\end{aligned} \tag{49}$$

which is,

$$\phi_y = \frac{\sum_{i=1}^{m} y^{(i)}}{m}.$$

2. To solve for $\phi_{k|y=1}$,

$$\nabla_{\phi_{k|y=1}} \ell(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) \tag{50}$$

$$= \nabla_{\phi_{k|y=1}} \sum_{i=1}^{m} \sum_{j=1}^{n_i} \log p\left(x_j^{(i)} | y^{(i)}; \phi_{k|y=0}, \phi_{k|y=1}\right) 1\left\{y^{(i)} = 1\right\} \tag{51}$$

$$= \nabla_{\phi_{k|y=1}} \sum_{i=1}^{m} \sum_{j=1}^{n_i} \left[1\left\{x_j^{(i)} = k\right\} \log \phi_{k|y=1} + 1\left\{x_j^{(i)} \neq k\right\} \log\left(1 - \phi_{k|y=1}\right)\right] 1\left\{y^{(i)} = 1\right\} \tag{52}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n_i} \left[1\left\{x_j^{(i)} = k\right\} \frac{1}{\phi_{k|y=1}} - 1\left\{x_j^{(i)} \neq k\right\} \frac{1}{1 - \phi_{k|y=1}}\right] 1\left\{y^{(i)} = 1\right\}. \tag{53}$$

Setting $\nabla_{\phi_{k|y=1}} \ell(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = 0$ gives

$$\begin{aligned} 0 &= \sum_{i=1}^{m} \sum_{j=1}^{n_i} \left[1\left\{x_j^{(i)} = k\right\} \left(1 - \phi_{k|y=1}\right) - 1\left\{x_j^{(i)} \neq k\right\} \phi_{k|y=1}\right] 1\left\{y^{(i)} = 1\right\} \\ &= \sum_{i=1}^{m} \sum_{j=1}^{n_i} \left[1\left\{x_j^{(i)} = k\right\} - \phi_{k|y=1}\right] 1\left\{y^{(i)} = 1\right\} \\ &= \sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 1\right\} - \phi_{k|y=1} \sum_{i=1}^{m} n_i 1\left\{y^{(i)} = 1\right\}, \end{aligned} \tag{54}$$

which is

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 1\right\}}{\sum_{i=1}^{m} n_i 1\left\{y^{(i)} = 1\right\}}.$$

3. The solution for $\phi_{k|y=0}$ proceeds in the identical manner, which is

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\left\{x_j^{(i)} = k \wedge y^{(i)} = 0\right\}}{\sum_{i=1}^{m} n_i 1\left\{y^{(i)} = 0\right\}}.$$

Remark. Contrast the multinomial event model with the Naive Bayes model. Their differences come from the ways in representing data. For the Naive Bayes model, $x^{(i)} \in \mathbb{R}^n$ is a vector of dictionary with n common words, each element of which is 1 if the corresponding word appears, while 0 otherwise. Therefore, $x_j^{(i)}$ is binomial. However, for the multinomial event model, though we also have a dictionary of n common words, here $x^{(i)} \in \mathbb{R}^{n_i}$ is a vector composed of all words in the i-th example, assuming the i-th example is of length n_i. As a result, $x_j^{(i)}$ complies to a multinomial distribtuion (n-nomial in specific). Note that the representation of multinomial event model conveys more information, since it tells us how many times a word appears rather than simply whether it appears or not.

3 Kernels, SVM and learning theory

3.1 Show how to apply the kernel trick to GLM.

Our goal here is to use kernels to implicitly represent the attributes in a high-dimensional space without explicitly computing $\phi(x)$. We assume that the parameter θ can be expressed as a linear combination of $x^{(i)}$ [and think about the gradient ascent algorithm to justify this]:

$$\theta = \sum_{i=1}^{m} \alpha_i x^{(i)},$$

and therefore the mapping from y to x, $h_\theta(x)$, is

$$h_\theta(x) = g(\theta^T x) = g\left(\sum_{i=1}^{m} \alpha_i x^{(i)T} x\right) = g\left(\sum_{i=1}^{m} \alpha_i \left\langle x^{(i)}, x \right\rangle\right).$$

Now that we have the inner product of two examples, we could replace $\left\langle x^{(i)}, x \right\rangle$ with any kernel function we want to do the learning in a high-dimensional space, and solve for the parameters $\alpha_i, i = 1, \ldots, m$, instead of θ. Feasible optimizaiton algorithms include gradient ascent, coordinate ascent, etc.

3.2 Show that the polynomial kernel $K(x, z) = (x^T z + c)^d$ maps p-dimensional input to an $C(n + d, d)$ feature space.

Assume that x and z are two n-dimensional vectors so $x^T z + c$ is a summation of $n+1$ items. We intend to expand $(x^T z + c)^d$ to the form of polynomial, each item of which is a product of d elements. Counting the number of such items is equivalent to the following combination problem: how many ways are there to select d items from a total of **unordered** $n + 1$ items **with replacement**? Classical probability theory tells us the answer is

$$C((n + 1) + d - 1, (n + 1) - 1) = C(n + d, d).$$

3.3 Show that Gaussian kernel is valid.

[Most proofs of the properties and lemmas below can be found at this website.]

The Gaussian kernel is defined as

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right), \sigma > 0.$$

Before the formal explanation of why this kernel is valid, let us first view some closure properties of valid kernels.

Suppose $K_1(x, y)$ and $K_2(x, y)$ are kernels over $\mathcal{X} \times \mathcal{X}$, $a \in \mathbb{R}^+$, and power series $\sum_{n=0}^{\infty} a_n x^n < \infty$ with $a_n \geq 0$ for all $n \in \mathbb{N}$. Then the following functions are kernels:

1. $K(x, y) = K_1(x, y) + K_2(x, y)$,
2. $K(x, y) = aK_1(x, y)$,
3. $K_1(x, y) + K_2(x, y)$,

4. $K(x,y) = \sum_{n=0}^{\infty} a_n K_1^n(x,y)$.

Lemma 1. *Kernel $K(x,y) = ||x-y||_2^2$ is negative definite.*

Proof. If $\sum_{i=1}^{n} \alpha_i = 0$, then

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j \left(x^{(i)} - x^{(j)}\right)^T \left(x^{(i)} - x^{(j)}\right) \tag{55}$$

$$= \sum_{i,j=1}^{n} \alpha_i \alpha_j \left(x^{(i)T} x^{(i)} - 2x^{(i)T} x^{(j)} + x^{(j)T} x^{(j)}\right) \tag{56}$$

$$= -2 \sum_{i,j=1}^{n} \left(\alpha_i x^{(i)}\right)^T \left(\alpha_j x^{(j)}\right) \leq 0. \tag{57}$$

Lemma 2. *Let $\mathcal{X}$ be a nonempty set and $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric kernel. Then ϕ is n.d **iff** $K = \exp(-t\phi)$ is p.d. for all $t > 0$.*

Now we can move on to the formal proofs.

[1st solution.]
According to lemma 1 and the property 2 we conclude that $||x-y||^2/2\sigma^2$ is a *n.d.* kernel. Then by lemma 2 and let $t = 1$, the proof completes.

[2nd solution.]
We rewrite the Gaussian kernel as

$$\begin{aligned} K(x,y) &= \exp\left(-\frac{(x-y)^T(x-y)}{2\sigma^2}\right) \\ &= \exp\left(-\frac{x^Tx + y^Ty - 2x^Ty}{2\sigma^2}\right) \\ &= \frac{\exp\left(\frac{x^Ty}{2\sigma^2}\right)}{\exp\left(\frac{x^Tx}{2\sigma^2}\right)\exp\left(\frac{y^Ty}{2\sigma^2}\right)}. \end{aligned} \tag{58}$$

Note that the denominator of the last equation is a positive constant, which do not hurt the validity of $K(x,y)$ as a kernel. When we check the numerator, x^Ty is already a valid kernel with the feature mapping being $\phi(x) = x/\sqrt{2}\sigma$. Therefore, if we consider the Taylor expansion:

$$\exp\left(\frac{x^Ty}{2\sigma^2}\right) = \sum_{n=0}^{\infty} \frac{1}{n!}\left(\frac{x^Ty}{2\sigma^2}\right)^n,$$

then by property 4, the proof completes.

We can be enlightened with further insights by such an factorization of $K(x,y)$. First, this factorization inspires a sense of normalization. Second, the Taylor expansion reveals that the feature mapping of the Gaussian kernel, is such a ϕ that expands the input to an infinite feature space. This gives one of the reasons why Gaussian kernel is such a popular kernel.

[3rd solution.]

Lemma 3. *If the random variable $Z \sim \mathcal{N}(0,1)$, then its characteristic function is*

$$\mathbb{E}\left[e^{itZ}\right] = \exp\left(-\frac{t^2}{2}\right).$$

For all $a_i \neq 0$,

$$\begin{aligned}
\sum_{i,j=1}^{n} a_i a_j \exp\left(-\frac{||x^{(i)} - x^{(j)}||_2^2}{2\sigma^2}\right) &= \sum_{i,j=1}^{n} a_i a_j \prod_k \exp\left(-\frac{\left(x_k^{(i)} - x_k^{(j)}\right)^2}{2\sigma^2}\right) \\
&= \prod_k \sum_{i,j=1}^{n} a_i a_j \mathbb{E}\left[e^{i\left(x_k^{(i)} - x_k^{(j)}\right) Z/\sigma}\right] \\
&= \prod_k \mathbb{E}\left[\sum_{i,j=1}^{n} a_i e^{i x_k^{(i)} Z/\sigma} a_j e^{-i x_k^{(j)} Z/\sigma}\right] \\
&= \prod_k \mathbb{E}\left[\left|\sum_{i=1}^{n} a_i e^{i x_k^{(i)} Z/\sigma}\right|^2\right] \geq 0.
\end{aligned} \tag{59}$$

As a result the kernel matrix is *p.s.d.* The proof completes.

3.4 Show that ℓ_1-norm kernel is valid.

[The following is referred from Prof. Zhihua Zhang.]

Lemma 4.

$$\exp(-tz^{1/2}) = \int \exp(-tsz) dF(s),$$

where

$$f(s) = \sqrt{\frac{t}{2\pi}} s^{-3/2} \exp\left(-\frac{t}{2s}\right).$$

Lemma 5. *Let F be a probability measure on the half line of $\mathbb{R}^+$ such that $0 < \int_0^\infty s dF(s) < \infty$ and $\mathcal{L}_F(u) = \int_0^\infty \exp(-su) dF(s), u \in \mathbb{R}^+$, then $\phi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is n.d. **iff** $\int_0^\infty \exp(-ts\phi) dF(s)$ is p.d. for all $t > 0$.*

Note that the ℓ_1-norm kernel is of the form

$$K(x,y) = \exp\left(-||x-y||_1\right),$$

where $||x||_1$ is the ℓ_1-norm of x. According to lemma 4, take $t = 1$ and $z = ||x_i - y_i||_2^2$, then we have

$$\exp\left(-||x_i - y_i||_2^1\right) = \int \exp(-s||x_i - y_i||_2^2) dF(s).$$

According to lemma 1, $||x_i - y_i||_2^2$ is *n.d*; then by lemma 5, $\exp\left(-||x_i - y_i||_2^1\right)$ is *p.d.* Since

$$K(x,y) = \exp\left(-\sum_{i=1}^{n} |x_i - y_i|\right) = \prod_{i=1}^{n} \exp\left(-|x_i - y_i|\right) = \prod_{i=1}^{n} \exp\left(-||x_i - y_i||_2^1\right),$$

we conclude that ℓ_1-norm kernel is *p.d*, in other words, a valid kernel.

3.5 Derive the objective and constraints of ℓ_1-norm soft margin SVM.

The ℓ_1-norm soft margin SVM is of the following form:

$$\begin{aligned} \min_{\xi,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{60}$$

First let us construct the Lagrangian of the problem:

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \frac{1}{2}w^T w + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(w^T x^{(i)} + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{m}\beta_i\xi_i.$$

Second, in order to find the dual form of the problem, we need to minimize $\mathcal{L}(w, b, \xi, \alpha, \beta)$ respect to w, b and ξ.

$$\nabla_w \mathcal{L}(w, b, \xi, \alpha, \beta) = w - \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)} = 0$$

gives

$$w = \sum_{i=1}^{m}\alpha_i y^{(i)} x^{(i)}.$$

$$\frac{\partial}{\partial b}\mathcal{L}(w, b, \xi, \alpha, \beta) = \sum_{i=1}^{m}\alpha_i y^{(i)} = 0$$

gives

$$0 = \sum_{i=1}^{m}\alpha_i y^{(i)}.$$

$$\nabla_{\xi_i}\mathcal{L}(w, b, \xi, \alpha, \beta) = C - \alpha_i - \beta_i = 0$$

gives

$$C = \alpha_i + \beta_i.$$

If we plug w, b, β_i back to the Lagrangian then we could get

$$\mathcal{L}(w, b, \xi, \alpha, \beta) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\alpha_i\alpha_j y^{(i)} y^{(j)} x^{(i)T} x^{(j)},$$

which is exactly the objective of the dual problem we want. Moreover, we must have $\alpha_i \leq C$ since the Lagrange multipliers are required to be greater than or equal to 0.

Next, we derived the complete dual form of the problem:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\alpha_i\alpha_j y^{(i)} y^{(j)}\left\langle x^{(i)}, x^{(j)}\right\rangle \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m}\alpha_i y^{(i)} = 0. \end{aligned} \tag{61}$$

Notice that we have to justify that the optima values of w, b, α for the dual form are the same as those for the primal problem. In the primal problem, f and all the g_i's are convex and the constraints g_i are (strictly) feasible, indicating that there must exist w^*, α^* and β^* so that w^* is the solution to the primal problem while α^*, β^* are the solution to the dual problem. What's more, w^*, α^* and β^* satisfy the KKT conditions, and thus it is a solution to both the primal and dual problems.

At the end, we need to figure out how to update b [Platt's paper].

3.6 Explain why SVM avoids the problem of overfitting although kernels expand the attributes into infinite-dimensional feature spaces.

Recall the ℓ_1-norm SVM:

$$\begin{aligned} \min_{\xi,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{62}$$

In fact, we can rewrite the optimization problem as

$$\min_{w,b} \quad \sum_{i=1}^{m}\left(1 - y^{(i)}\left(w^T x^{(i)} + b\right)\right)_+ + \lambda\|w\|^2,$$

where $(1-z)_+ = \max\{0, 1-z\}$ and it is called the *hinge loss*. Therefore, from the perspective of regularization, the first term of the expression above can be regarded as the loss funtion, while the second term as the penalty function, which is one of the most helpful tools to deal with the problem of overfitting.

As an extention of the problem, we can also talk more about the loss function. In fact, one of the standard loss functions on classification problem is the *0-1 loss*:

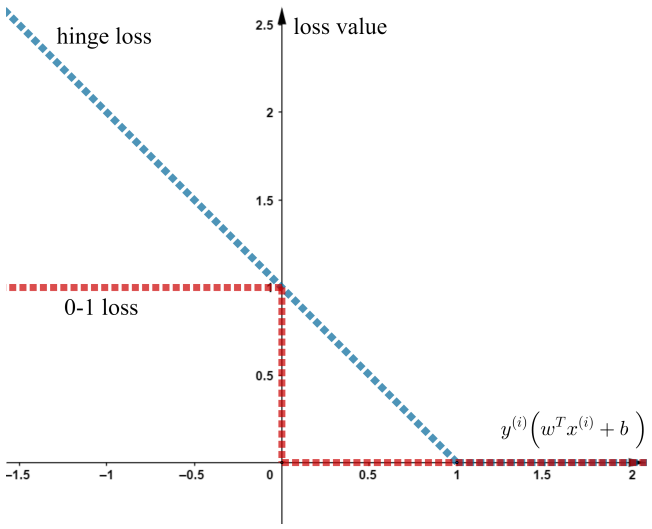
$$1\left(h(x^{(i)}) \neq y^{(i)}\right),$$

where

$$h(x^{(i)}) = \begin{cases} 1, & w^T x^{(i)} + b \geq 0, \\ 0, & w^T x^{(i)} + b < 0. \end{cases} \tag{63}$$

However, the 0-1 loss bears a serious disadvantage: such an optimization problem can only be solved by enumeration, forming a NP-hard problem. The hinge loss, as a result, for the sake of computational efficiency, is used to estimate and replace the 0-1 loss here.

The graph below contrasts the hinge loss with the 0-1 loss. As a function composited by the signal function, the 0-1 loss is either continuous or differentiable at the point 0; while the hinge loss, a convex upper bound of the 0-1 loss, is continuous but not differentiable at the point 1. In fact, the continuity here guarantees the existence of at least sub-derivative; on the other hand, the non-differentiability, or a cusp in other words, helps the performance of SVM, forcing many α_i's to 0 and picking up only the nearest support vectors.



In conclusion, the factors that help SVM avoid overfitting come from not only $||w||^2$ as the penalty function, but also the delicate choice of the hinge loss function. The former factor picks up significant variables while the latter factor help to choose appropriate support vectors. They work together to guarantee the excellent performance of SVM and make it one of the most popular classifiers these days.

3.7 Describe the choice of the heuristics used to select the next α_i, α_j to update (in Platt's paper).

3.8 Show the relationship between regularization and Bayesian prior.

Regularization is a term sort of used by **frequentists** while the prior is of the **Bayesian** view of the world.

In frequentist statistics, we treat the parameter θ as an unknown constant value, and add regularization penalty to a loss function to estimate θ. Generally we could choose $-\sum_i \log \mathcal{L}_i$ as the loss function, which originates from the idea that for a Gaussian distribution $\mathcal{N}(\theta, 1)$,

$$\begin{aligned}
\theta_{\text{MLE}} &= \arg\min_\theta \left[-\sum_i \log \mathcal{L}_i \right] \\
&= \arg\min_\theta \left[-\sum_{i=1}^{m} \log \left(\frac{1}{\sqrt{2\pi}} \exp\left\{ -\frac{\left(x^{(i)}-\theta\right)^2}{2} \right\} \right) \right] \\
&= \arg\min_\theta \left[\frac{m}{2}\log(2\pi) + \frac{1}{2}\sum_{i=1}^{m}\left(x^{(i)}-\theta\right)^2 \right] \\
&= \arg\min_\theta \frac{1}{2}\sum_{i=1}^{m}\left(x^{(i)}-\theta\right)^2 \qquad (\textit{the least square loss}).
\end{aligned} \tag{64}$$

Therefore the whole objective funtion with regularization we use to estimate θ is:

$$\theta_{\text{Reg}} = \arg\min_\theta \left[-\sum_i \log \mathcal{L}_i + \lambda P(\theta) \right]$$

where λ is the tuning parameter.

However, from Bayesian view we treat the parameter θ as a random variable with a prior distribution $p(\theta|\lambda)$, where λ is the canonical parameter of the distribution. A common statistical procedure to estimate θ is the MAP (maximum a posteriori):

$$\begin{aligned}
\theta_{\text{MAP}} &= \arg\max_{\theta} p(\theta|S) \\
&= \arg\max_{\theta} \frac{\prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)},\theta\right) p(\theta|\lambda)}{p(S)} \\
&= \arg\max_{\theta} \prod_{i=1}^{m} p\left(y^{(i)}|x^{(i)},\theta\right) p(\theta|\lambda) \\
&= \arg\min_{\theta} \left[-\sum_{i=1}^{m} \log p\left(y^{(i)}|x^{(i)},\theta\right) - \log p(\theta|\lambda) \right] \\
&= \arg\min_{\theta} \left[-\sum_{i} \log \mathcal{L}_i - \log p(\theta|\lambda) \right]
\end{aligned} \tag{65}$$

If we contrast θ_{Reg} with θ_{MAP}, we can conclude that the regularization penalty is at the same position as the negative log-prior; in other words, regularization is a **dual form** of the Bayesian prior. In a sense, we have

$$\lambda P(\theta) \approx -\log p(\theta|\lambda).$$

Specifically, if we use ℓ_2-norm penality, then the prior is

$$p(\theta|\lambda) \approx \exp\left(-\lambda||\theta||_2^2\right),$$

which is of the form of Gaussian distribution.

4 Trees, ensemble methods and neural networks

4.1 Show the connection between Adaboost and Forward Stagewise Additive Modeling.

Boosting is one of the central algorithms of the ensambling methods, adding a batch of *weak classifiers* (whose prevalence $> 50\%$) together and transforming them as a whole into a *strong classifier*. The **forward stagewise additive modeling** algorithm is a general framework for ensembling, as showed in algorithm 1.

Note that the framework is quite flexible and of few assumptions. We only need to guarantee the additive nature of those ensemblings (or weak classifiers in our case) if we plan to derive specific algorithms based on this framework. In fact, using the framework, we can get one of the most popular boosting algorithms called **Adaboost**, as showed in algorithm 2.

We will spend the remaining page to show how to derive Adaboost from the general framework.

First of all, we need to specify the notations. For a binary classification problem, we code $y_i \in \{-1, 1\}$, and the desired set of weak classifiers $\{G_m(x)|G_m(x) \in \{-1, 1\}, m = 1, ..., M\}$ in the function subspace spanned by $G(x; \gamma)$.

Algorithm 1: Forward stagewise additive modeling

Input: Labeled training data $(x_1, y_1), ..., (x_N, y_N)$

Output: Ensemble classifer $f(x) = f_m(x)$

Initialization $f_0(x) = 0$

for $m = 0$ *to* M **do**

  Compute $(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta G(x_i; \gamma))$

  Set $f_m(x) = f_{m-1}(x) + \beta_m G(x; y_i)$

end

$f(x) = f_m(x)$

Algorithm 2: Adaboost

Input: Labeled training data $(x_1, y_1), ..., (x_N, y_N)$

Output: Ensemble classifer $f(x)$

$w_i \leftarrow \frac{1}{N}$ for $i = 1, ..., N$

for $m = 0$ *to* M **do**

  Fit weak classifier $G_m(x)$ to training data weighted by w_i

  Compute weighted error $err_m = \frac{\sum_i w_i 1(y_i \neq G_m(x_i))}{\sum_i w_i}$

  Compute weight $\alpha_m = \log\left(\frac{1-err_m}{err_m}\right)$

  $w_i \leftarrow w_i * \exp(\alpha_m 1(y_i \neq G_m(x_i)))$

end

$f(x) = \text{sign}(\sum_m \alpha_m G_m(x))$

Next, we choose the loss function as

$$L = \exp\left(-yf(x)\right),$$

which is a convex upper bound of the 0-1 loss and enjoys good properties such as the Bayesian consistency. The pros and cons about several loss functions will be illustrated in 4.2 in detail.

Now we need to figure out a way to compute

$$\arg\min_{\beta,G} \sum_{i=1}^{N} \exp\left[-y_i \left(f_{m-1}(x_i) + \beta G(x_i)\right)\right] \tag{66}$$

$$= \arg\min_{\beta,G} \sum_{i=1}^{N} \exp\left[-y_i f_{m-1}(x_i)\right] \exp\left[-y_i \beta G(x_i)\right] \tag{67}$$

$$= \arg\min_{\beta,G} \sum_{i=1}^{N} w_i^{(m)} \exp\left[-y_i \beta G(x_i)\right], \tag{68}$$

where $w_i^{(m)} = \exp\left[-y_i f_{m-1}(x_i)\right]$. Due to the difficulty of numerical optimization, we need to solve the problem by 2 steps.

Step 1. Fix the parameter $\beta > 0$, and solve for G:

$$\begin{aligned} G_m(x_i) &= \arg\min_G \sum_{i=1}^N w_i^{(m)} \exp\left[-y_i \beta G(x_i)\right] \\ &= \arg\min_G \sum_{i=1}^N w_i^{(m)} \exp\left[\beta\left(2 \cdot 1(y_i \neq G(x_i)) - 1\right)\right] \\ &= \arg\min_G \sum_{i=1}^N w_i^{(m)} 1(y_i \neq G(x_i)). \end{aligned} \tag{69}$$

Step 2. Fix the $G_m(x_i)$ we get in step 1, and solve for β_m:

$$\begin{aligned} \beta_m &= \arg\min_\beta \sum_{i=1}^N w_i^{(m)} \exp\left[-y_i \beta G(x_i)\right] \\ &= \arg\min_\beta \left[\sum_{y_i = G(x_i)} w_i^{(m)} \exp(-\beta) + \sum_{y_i \neq G(x_i)} w_i^{(m)} \exp(\beta)\right] \\ &= \arg\min_\beta \left[\exp(-\beta) \sum_{i=1}^N w_i^{(m)} - \exp(-\beta) \sum_{y_i \neq G(x_i)} w_i^{(m)} + \exp(\beta) \sum_{y_i \neq G(x_i)} w_i^{(m)}\right] \\ &= \arg\min_\beta \left[\exp(-\beta) \sum_{i=1}^N w_i^{(m)} + (\exp(\beta) - \exp(-\beta)) \sum_{i=1}^N w_i^{(m)} 1(y_i \neq G(x_i))\right]. \end{aligned} \tag{70}$$

Take derivative of the last expression and set it as 0:

$$0 = -\exp(-\beta) \sum_{i=1}^N w_i^{(m)} + (\exp(\beta) + \exp(-\beta)) \sum_{i=1}^N w_i^{(m)} 1(y_i \neq G(x_i)),$$

so we can get

$$\exp(-\beta) = (\exp(\beta) + \exp(-\beta)) \frac{\sum_{i=1}^N w_i^{(m)} 1(y_i \neq G(x_i))}{\sum_{i=1}^N w_i^{(m)}}.$$

Let

$$err_m = \frac{\sum_{i=1}^N w_i^{(m)} 1(y_i \neq G(x_i))}{\sum_{i=1}^N w_i^{(m)}}$$

and through simple algebra, then we can finally have

$$\beta_m = \beta^* = \frac{1}{2} \log\left(\frac{1 - err_m}{err_m}\right).$$

Now that we've already had the expression for G_m and β_m, we can move on to the formula for updating w_i. Note that while $w_i^{(m)} = \exp\left[-y_i f_{m-1}(x_i)\right]$,

$$\begin{aligned} w_i^{(m+1)} &= \exp\left[-y_i f_m(x_i)\right] \\ &= \exp\left[-y_i \left(f_{m-1}(x_i) + \beta_m G_m(x_i)\right)\right] \\ &= w_i^{(m)} \exp\left[-y_i \beta_m G_m(x_i)\right] \\ &= w_i^{(m)} \exp\left[2\beta_m \cdot 1(y_i \neq G(x_i))\right] \exp(-\beta_m). \end{aligned} \tag{71}$$

In fact, the term $\exp(\beta_m)$ can be ignored since it is a constant *w.r.t.* all examples. Let

$$\alpha_m = 2\beta_m = \log\left(\frac{1 - err_m}{err_m}\right),$$

then the updating rule for w_i is justified.

Before we close we would like to talk more about the output $f(x) = \text{sign}(\sum_m \alpha_m G_m(x))$. In fact, by the recursive relation in algorithm 1(4), the standard output should be of the form $f(x) = \sum_m \beta_m G_m(x)$ with $\beta_m \in \mathbb{R}^+$. However, on the one hand, what we ultimately desire is not a real number but a discrete value (-1 or 1) instead, so it is natural (see 4.2) to take the sign of $f(x)$ and write the output as $f(x) = \text{sign}(\sum_m \beta_m G_m(x))$; on the other hand, since

$$\text{sign}\left(\sum_m \alpha_m G_m(x)\right) = \text{sign}\left(2\sum_m \beta_m G_m(x)\right) = \text{sign}\left(\sum_m \beta_m G_m(x)\right),$$

it is equivalent to write the output as $f(x) = \text{sign}(\sum_m \alpha_m G_m(x))$.

4.2 Contrast the loss functions of Adaboost, of SVM and of logistic regression.

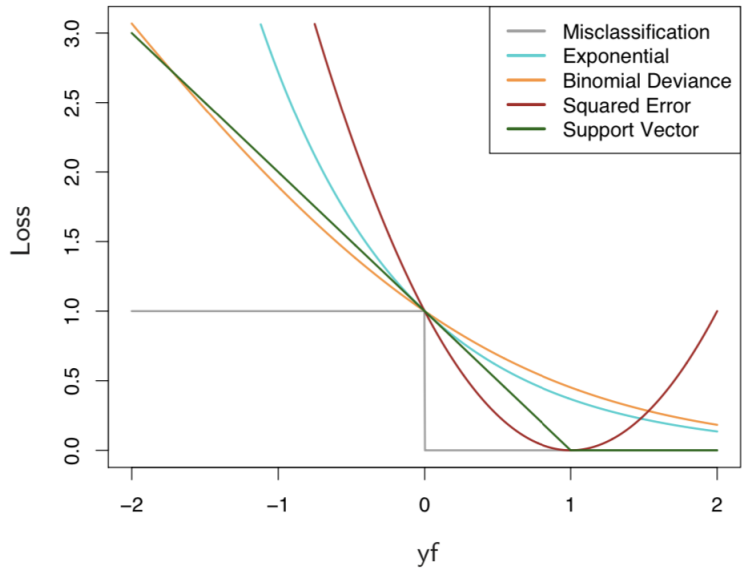


FIGURE 10.4. *Loss functions for two-class classification. The response is* $y = \pm 1$*; the prediction is* f*, with class prediction* $\text{sign}(f)$*. The losses are misclassification:* $I(\text{sign}(f) \neq y)$*; exponential:* $\exp(-yf)$*; binomial deviance:* $\log(1 + \exp(-2yf))$*; squared error:* $(y - f)^2$*; and support vector:* $(1 - yf)_+$ *(see Section 12.3). Each function has been scaled so that it passes through the point* $(0, 1)$*.*

The loss function of Adaboost is the exponential loss: $L = \exp(-yf)$, while the loss function of SVM is the hinge loss: $L = (1 - yf)_+$. Now we will show that the loss function of logistic regression is the binomial deviance: $L = \log(1 + \exp(-2yf))$.

Let the linear function be $f = \frac{1}{2}\log\left(\frac{p}{1-p}\right)$, and let the logistic loss function using label $y' \in \{0,1\}$ be

$$L = -\left[y' \log p + (1-y')\log(1-p)\right].$$

Then we have

$$\begin{aligned} L &= -\left[y' \log \frac{p}{1-p} + \log(1-p)\right] \\ &= -\left[2y'f + \log\left(1 - \frac{\exp(2f)}{1+\exp(2f)}\right)\right] \\ &= -\left[2y'f - \log\left(1+\exp(2f)\right)\right] \\ &= -(y+1)f + \log\left(1+\exp(2f)\right). \end{aligned} \tag{72}$$

When $y = -1$, $L = \log(1+\exp(2f))$; when $y = 1$,

$$L = -2f + \log(1+\exp(2f)) = \log(\exp(-2f)) + \log(1+\exp(2f)) = \log(1+\exp(-2f)).$$

So we can conclude that $L = \log(1+\exp(-2yf))$.

The loss functions of Adaboost, SVM, and logistic regression are all smooth proxies for the discrete misclassification loss. They are chosen due to the convenience of optimization computation. It is not hard to see that the hinge loss and binomial deviance are similar to each other, while the exponential loss place much more penalty on the misclassification side (when $yf < 0$). Therefore, the exponential loss is less robust in noisy settings, especially when there are many mis-specified class labels in the training data set.

4.3 Derive a boosting algorithm using the loss function of logistic regression.

The algorithm has the following form:

We assume that the labels of the classification problems are $y_i \in \{0,1\}$. We should follow the framework of Algorithm 1. Let $G(x_i;\gamma)$ be the weighted linear regression. Since the log-likelihood of the logistic regression is

$$\begin{aligned} \ell &= \sum_{i=1}^{N} \left[y_i \log p(x_i) + (1-y_i)\log(1-p(x_i))\right] \\ &= \sum_{i=1}^{N} \left[y_i \log \frac{p(x_i)}{1-p(x_i)} + \log(1-p(x_i))\right] \\ &= \sum_{i=1}^{N} \left[y_i f(x_i) - \log(1+e^{f(x_i)})\right], \end{aligned} \tag{73}$$

where $f(x_i) = f_{m-1}(x_i) + \beta^T x_i$. In order to find the β that minimizes the loss function $L = -2\ell$, we need to maximize the log-likelihood. Consider the first derivative of ℓ w.r.t. β:

$$\begin{aligned} \nabla_\beta \ell &= \sum_{i=1}^{N} \left[y_i x_i - \frac{e^{f(x_i)}}{1+e^{f(x_i)}} x_i\right] \\ &= X^T(y-p) \end{aligned} \tag{74}$$

Algorithm 3: Logit boosting

Input: Labeled training data $(x_1, y_1), ..., (x_N, y_N)$

Output: Ensemble classifer $G(x)$

$w_i \leftarrow \frac{1}{N}$, $f_0(x_i) \leftarrow 0$, $p(x_i) \leftarrow \frac{1}{2}$, for $i = 1, ..., N$

for $m = 0$ *to* M **do**

(a) Compute the working response

$$z_i = \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))}$$

and weights $w_i = p(x_i)(1 - p(x_i))$

(b) Fit the function by $f_m(x)$ by a weighted least-squares regression (fitting z_i by x_i with weights w_i)

(c) Update $f(x) \leftarrow f(x) + f_m(x)$ and

$$p(x) \leftarrow \frac{e^{f(x)}}{1 + e^{f(x)}}$$

end

$G(x) = \text{sgn}(f(x))$

where $y \in \mathbb{R}^N$, $p \in \mathbb{R}^N$, and $p_i = e^{f(x_i)}/(1 + e^{f(x_i)})$; and the second derivative of ℓ:

$$\begin{aligned} \nabla_{\beta\beta^T}\ell &= \nabla_{\beta^T} X^T(y - p) \\ &= -X^T W X \end{aligned} \tag{75}$$

where W is diagonal and $W_{ii} = p_i(1 - p_i)$. Consequently, we can use the Newton-Raphson iteration:

$$\begin{aligned} \beta^{\text{new}} &= \beta^{\text{old}} - (\nabla_{\beta\beta^T}\ell)^{-1}\nabla_\beta \ell \\ &= \beta^{\text{old}} + (X^T W X)^{-1} X^T(y - p) \\ &= (X^T W X)^{-1} X^T W(X\beta^{\text{old}} + W^{-1}(y - p)) \\ &= (X^T W X)^{-1} X^T W z, \end{aligned} \tag{76}$$

where z is the response

$$z = X\beta^{\text{old}} + W^{-1}(y - p) = f_{m-1}(X) + W^{-1}(y - p),$$

and the β^{new} can be written as the solution to the following weighted least squares problem:

$$\beta^{\text{new}} \leftarrow \arg\min_\beta (z - X\beta)^T W(z - X\beta).$$

As a result, in accordance with the algorithm 1 framework, we can regard the $W^{-1}(y - p)$ as the working response, each entry of which is exactly $(y_i - p_i)/(p_i(1 - p_i))$; and at the same time, the weight for each observation is exactly $p_i(1 - p_i)$. It can be easily shown that the followings are equivalent:

Logistic regression:

1. $z = X\beta^{\text{old}} + W^{-1}(y - p)$

2. $\beta^{\text{new}} \leftarrow (X^T W X)^{-1} X^T W z$

3. $f(x) \leftarrow X\beta^{\text{new}}$

and **Logit Boosting**:

1. $z = W^{-1}(y - p)$

2. $\beta^{\text{new}} \leftarrow (X^T W X)^{-1} X^T W z$

3. $f(x) \leftarrow X\beta^{\text{old}} + X\beta^{\text{new}}$

As an adoption of the logistic regression problem, the Logit boostint algorithm can be justified. □

5 Unsupervised learning (clustering, EM algorithm, PCA, ICA)

5.1 Derive the EM algorithm for mixtures of Gaussians.

The objective function is defined as

$$J(Q;\phi,\mu,\Sigma) \tag{77}$$

$$= \sum_{i=1}^{m}\sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p\left(x^{(i)}, z^{(i)}; \mu, \Sigma, \phi\right)}{Q_i(z^{(i)})} \tag{78}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{k} Q_i(z^{(i)}=j) \log \frac{p\left(x^{(i)}|z^{(i)}=j; \mu, \Sigma\right) p\left(z^{(i)}=j;\phi\right)}{Q_i(z^{(i)}=j)} \tag{79}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \left[-\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_j| - \frac{1}{2}\left(x^{(i)}-\mu_j\right)^T \Sigma_j^{-1}\left(x^{(i)}-\mu_j\right) + \log\phi_j - \log w_j^{(i)}\right]. \tag{80}$$

Take the derivative of $J(Q;\phi,\mu,\Sigma)$ *w.r.t.* μ_l, we have

$$\begin{aligned}
\nabla_{\mu_l} J(Q;\phi,\mu,\Sigma) &= \nabla_{\mu_l} \sum_{i=1}^{m}\sum_{j=1}^{k} w_j^{(i)} \left[-\frac{1}{2}\left(x^{(i)}-\mu_j\right)^T \Sigma_j^{-1}\left(x^{(i)}-\mu_j\right)\right] \\
&= \nabla_{\mu_l} \sum_{i=1}^{m} w_j^{(i)} \left[-\frac{1}{2}\left(-x^{(i)T}\Sigma_l^{-1}u_l - u_l^T\Sigma_l^{-1}x^{(i)} + \mu_l^T\Sigma_l^{-1}\mu_l\right)\right] \\
&= -\frac{1}{2}\sum_{i=1}^{m} w_l^{(i)}\left(2\Sigma_l^{-1}\mu_l - 2\Sigma_l^{-1}x^{(i)}\right) \\
&= \sum_{i=1}^{m} w_l^{(i)}\left(\Sigma_l^{-1}x^{(i)} - \Sigma_l^{-1}\mu_l\right).
\end{aligned} \tag{81}$$

To solve for μ_l, setting $\nabla_{\mu_l} J(Q;\phi,\mu,\Sigma) = 0$ gives

$$\mu_l = \frac{\sum_{i=1}^{m} w_l^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_l^{(i)}}.$$

Next, let us solve for ϕ_j with the restriction $\sum_j \phi_j = 1$. We construct the Lagrangian

$$\mathcal{L} = J(Q; \phi, \mu, \Sigma) + \beta \left(\sum_{j=1}^{k} \phi_j - 1 \right).$$

Take derivatives of $\mathcal{L}$ *w.r.t.* ϕ_l and β. Setting them equal to 0 yields:

$$\nabla_{\phi_l} \mathcal{L} = \nabla_{\phi_l} \left[\sum_{i=1}^{m} w_l^{(i)} \log \phi_l + \beta \left(\sum_{j=1}^{k} \phi_j - 1 \right) \right] = \sum_{i=1}^{m} \frac{w_l^{(i)}}{\phi_l} + \beta = 0,$$

and

$$\frac{\partial \mathcal{L}}{\partial \beta} = \left(\sum_{j=1}^{k} \phi_j - 1 \right) = 0.$$

Combine the two expressions above we have

$$\phi_l = \frac{1}{m} \sum_{i=1}^{m} w_l^{(i)} \qquad \text{(since } \beta = -m\text{)}.$$

Finally, let us consider the derivative of $J(Q; \phi, \mu, \Sigma)$ *w.r.t.* Σ_l and set it as 0:

$$\begin{aligned}
& \nabla_{\Sigma_l} J(Q; \phi, \mu, \Sigma) && (82) \\
&= \nabla_{\Sigma_l} \sum_{i=1}^{m} w_l^{(i)} \left[-\frac{1}{2} \log |\Sigma_l| - \frac{1}{2} \left(x^{(i)} - \mu_l \right)^T \Sigma_l^{-1} \left(x^{(i)} - \mu_l \right) \right] && (83) \\
&= -\frac{1}{2} \sum_{i=1}^{m} w_l^{(i)} \left[\Sigma_l^{-1} - \Sigma_l^{-1} \left(x^{(i)} - \mu_l \right) \left(x^{(i)} - \mu_l \right)^T \Sigma_l^{-1} \right] = 0. && (84)
\end{aligned}$$

This yields

$$\Sigma_l = \frac{\sum_{i=1}^{m} w_l^{(i)} \left(x^{(i)} - \mu_l \right) \left(x^{(i)} - \mu_l \right)^T}{\sum_{i=1}^{m} w_l^{(i)}}.$$

In summary, the whole algorithm is:

Repeat until convergence: {
(E-step) For evert i, j, set

$$\begin{aligned}
w_j^{(i)} &= p\left(z^{(i)} = j | x^{(i)}; \phi^{[t]}, \mu^{[t]}, \Sigma^{[t]} \right) \\
&= \frac{p\left(x^{(i)} | z^{(i)} = j; \mu^{[t]}, \Sigma^{[t]} \right) p\left(z^{(i)} = j; \phi^{[t]} \right)}{\sum_j p\left(x^{(i)} | z^{(i)} = j; \mu^{[t]}, \Sigma^{[t]} \right) p\left(z^{(i)} = j; \phi^{[t]} \right)} \\
&= \frac{\frac{1}{(2\pi)^{n/2} \left| \Sigma_j^{[t]} \right|^{1/2}} \exp\left[-\frac{1}{2} \left(x^{(i)} - \mu_j^{[t]} \right)^T \Sigma_j^{[t]-1} \left(x^{(i)} - \mu_j^{[t]} \right) \right] \phi_j^{[t]}}{\sum_j \frac{1}{(2\pi)^{n/2} \left| \Sigma_j^{[t]} \right|^{1/2}} \exp\left[-\frac{1}{2} \left(x^{(i)} - \mu_j^{[t]} \right)^T \Sigma_j^{[t]-1} \left(x^{(i)} - \mu_j^{[t]} \right) \right] \phi_j^{[t]}}.
\end{aligned} \tag{85}$$

Note that $w_j^{(i)}$ is a function of just parameters computed from t-th iteration, while having nothing to do with parameters at the $(t+1)$-th iteration.

(M-step) Update the parameters:

$$\mu_j^{[t+1]} := \frac{\sum_{i=1}^m w_j^{(i)} x^{(i)}}{\sum_{i=1}^m w_j^{(i)}},$$

$$\phi_j^{[t+1]} := \frac{1}{m} \sum_{i=1}^m w_j^{(i)},$$

$$\Sigma_j^{[t+1]} := \frac{\sum_{i=1}^m w_j^{(i)} \left(x^{(i)} - \mu_j\right) \left(x^{(i)} - \mu_j\right)^T}{\sum_{i=1}^m w_j^{(i)}}.$$

}

5.2 Derive the conditional multivariate Gaussian distribution.

Suppose we partition a vector-valued random variable

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

where $x_1 \in \mathbb{R}^p$, $x_2 \in \mathbb{R}^q$ and $x \in \mathbb{R}^m$ $(m = p + q)$. Suppose $x \sim \mathcal{N}_m(\mu, \Sigma)$, where

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}.$$

We want to derive the distribution of $x_1|x_2$, but before we start let us spend time on some lemmas.

Lemma 6.

$$(A + CBD)^{-1} = A^{-1} - A^{-1}C \left(B^{-1} + DA^{-1}C\right)^{-1} DA^{-1}.$$

Lemma 7.

$$\begin{aligned} |A| &= \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \\ &= |A_{11}| \left|A_{22} - A_{21}A_{11}^{-1}A_{12}\right| \\ &= |A_{22}| \left|A_{11} - A_{12}A_{22}^{-1}A_{21}\right|. \end{aligned} \tag{86}$$

Proofs of the two lemmas above can be found at this website.

Lemma 8.

$$\Theta = \Sigma^{-1} = \begin{bmatrix} \left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1} & -\left(\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)^{-1}\Sigma_{12}\Sigma_{22}^{-1} \\ -\left(\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)^{-1}\Sigma_{21}\Sigma_{11}^{-1} & \left(\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)^{-1} \end{bmatrix}.$$

Proof can be found at this website using the method of LDU decomposition.

Note that

$$f(x_1, x_2) = \frac{1}{(2\pi)^{\frac{m}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \right\}$$

and

$$f(x_2) = \frac{1}{(2\pi)^{\frac{q}{2}} |\Sigma_{22}|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2}(x_2-\mu_2)^T \Sigma_{22}^{-1}(x_2-\mu_2) \right\}.$$

Thus

$$\begin{aligned} f(x_1|x_2) &= \frac{f(x_1, x_2)}{f(x_2)} \\ &= \frac{1}{(2\pi)^{\frac{p}{2}} \left|\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2}Q \right\}, \end{aligned} \tag{87}$$

where

$$\begin{aligned} Q &= \begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}^T \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}^{-1} \begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix} - (x_2-\mu_2)^T \Sigma_{22}^{-1}(x_2-\mu_2) \\ &\overset{\substack{x_1:=x_1+\mu_1 \\ x_2:=x_2+\mu_2}}{=\!=\!=} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} \Theta_{11} & \Theta_{12} \\ \Theta_{21} & \Theta_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - x_2^T \Sigma_{22}^{-1} x_2 \\ &= x_1^T \Theta_{11} x_1 + x_1^T \Theta_{12} x_2 + x_2^T \Theta_{21} x_1 + x_2^T \Theta_{22} x_2 - x_2^T \Sigma_{22}^{-1} x_2 \\ &= \left(x1 - \Sigma_{12}\Sigma_{22}^{-1} x_2\right)^T \Theta_{11} \left(x1 - \Sigma_{12}\Sigma_{22}^{-1} x_2\right) + x_2^T \left(\Theta_{22} - \Sigma_{22}^{-1}\Sigma_{12}^T \Theta_{11} \Sigma_{12}\Sigma_{22}^{-1} - \Sigma_{22}^{-1}\right) x_2 \\ &= \left(x1 - \Sigma_{12}\Sigma_{22}^{-1} x_2\right)^T \Theta_{11} \left(x1 - \Sigma_{12}\Sigma_{22}^{-1} x_2\right) \\ &\overset{\substack{x_1:=x_1-\mu_1 \\ x_2:=x_2-\mu_2}}{=\!=\!=} \left[x_1 - \mu_1 - \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2)\right]^T \Theta_{11} \left[x_1 - \mu_1 - \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2)\right]. \end{aligned} \tag{88}$$

The last equation holds because $\Theta_{22} - \Sigma_{22}^{-1}\Sigma_{12}^T \Theta_{11} \Sigma_{12}\Sigma_{22}^{-1} - \Sigma_{22}^{-1} = 0$ according to Lemma 6.

Therefore we conclude that $x_1|x_2 \sim \mathcal{N}(\mu_{1|2}, \Sigma_{1|2})$, where

$$\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2),$$

$$\Sigma_{1|2} = \Theta_{11}^{-1} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}.$$

[Another solution.]
Consider the linear transformation:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} I_p & 0 \\ -\Sigma_{21}\Sigma_{11}^{-1} & I_q \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

Then

$$\mathrm{Cov}(z_1, z_2) = \begin{bmatrix} I_p & 0 \\ -\Sigma_{21}\Sigma_{11}^{-1} & I_q \end{bmatrix} \Sigma \begin{bmatrix} I_p & -\Sigma_{11}^{-1}\Sigma_{12} \\ 0 & I_q \end{bmatrix} = \begin{bmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} \end{bmatrix}.$$

Thus

$$\mathbb{E}(z_2) = \mathbb{E}\left(x_2 - \Sigma_{21}\Sigma_{11}^{-1}x_1\right) = \mu_2 - \Sigma_{21}\Sigma_{11}^{-1}\mu_1,$$
$$\mathrm{Var}(z_2) = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12},$$
$$z_1 \perp z_2 \quad (x_1 \perp z_2).$$

Since $z_2 = x_2 - \Sigma_{21}\Sigma_{11}^{-1}x_1$, we have $x_2 = z_2 + \Sigma_{21}\Sigma_{11}^{-1}x_1$, and furthermore,

$$\mathbb{E}(x_2|x_1) = \mathbb{E}(z_2) + \Sigma_{11}^{-1}x_1 = \mu_2 + \Sigma_{11}^{-1}(x_1 - \mu_1),$$
$$\mathrm{Var}(x_2|x_1) = \mathrm{Var}(z_2) = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}.$$

Note that the linear tranformation of a multivariate Gaussian distribution is still Gaussian, indicating that $x_2|x_1 \sim \mathcal{N}\left(\mathbb{E}(x_2|x_1), \mathrm{Var}(x_2|x_1)\right)$.

5.3 Derive the EM algorithm for factor analysis.

The factor analysis model is defined as

$$x = \mu + \Lambda z + \epsilon$$
$$z \sim \mathcal{N}(0, I), \quad \epsilon \sim \mathcal{L}(0, \Psi), \quad z \perp \epsilon$$

where $\mu \in \mathbb{R}^n$, the matrix $\Lambda \in \mathbb{R}^{n\times k}(k < n)$, and the diagonal matrix $\Psi \in \mathbb{R}^{n\times n}$. Note that in this model z is an unobserved "latent" variable while we only have access to the dataset $\{x^{(i)}\}, i = 1, ..., n$.

First we try to estimate the parameters by MLE. According to the model and distributions above, we know $x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$, so

$$\begin{aligned}\ell(x;\mu,\Lambda,\epsilon) =& \log\prod_{i=1}^{m}\frac{1}{(2\pi)^{n/2}\left|\Lambda\Lambda^T+\Psi\right|^{1/2}}\exp\left[-\frac{1}{2}\left(x^{(i)}-\mu\right)^T\left(\Lambda\Lambda^T+\Psi\right)^{-1}\left(x^{(i)}-\mu\right)\right]\\ =& \sum_{i=1}^{m}\left[-\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left|\Lambda\Lambda^T+\Psi\right| - \frac{1}{2}\left(x^{(i)}-\mu\right)^T\left(\Lambda\Lambda^T+\Psi\right)^{-1}\left(x^{(i)}-\mu\right)\right].\end{aligned} \tag{89}$$

Take derivative *w.r.t.* μ and set it as 0, we have

$$\begin{aligned}\nabla_\mu\ell(x;\mu,\Lambda,\epsilon) =& -\frac{1}{2}\nabla_\mu\sum_{i=1}^{m}\left(x^{(i)}-\mu\right)^T\left(\Lambda\Lambda^T+\Psi\right)^{-1}\left(x^{(i)}-\mu\right)\\ =& \sum_{i=1}^{m}\left[\left(\Lambda\Lambda^T+\Psi\right)^{-1}x^{(i)} - \left(\Lambda\Lambda^T+\Psi\right)^{-1}\mu\right] = 0.\end{aligned} \tag{90}$$

Thus

$$\mu = \frac{1}{m}\sum_{i=1}^{m}x^{(i)}.$$

However, if we plug μ in $\ell(x;\mu,\Lambda,\epsilon)$ and take derivatives *w.r.t.* Λ and Ψ, we can only get

$$\Lambda = \hat{\Sigma}\left(\Lambda\Lambda^T+\Psi\right)^{-1}\Lambda,$$

$$\left(\Lambda\Lambda^T+\Psi\right)^{-1}-\left(\Lambda\Lambda^T+\Psi\right)^{-1}\hat{\Sigma}\left(\Lambda\Lambda^T+\Psi\right)^{-1}=0,$$

where $\hat{\Sigma}$ is the sample covariance matrix. Literally it is not easy to derive the explicit expressions for Λ and Ψ, so we resort to EM algorithm to solve this problem.

Remark 1. If we add a restriction $\Psi=\sigma^2 I_n$ (each dimension enjoys the same variance) to the model, then using the trick $\left(\Lambda\Lambda^T+\sigma^2 I_n\right)^{-1}\Lambda=\Lambda\left(\Lambda^T\Lambda+\sigma^2 I_k\right)^{-1}$ and further spectral decomposition of $\Lambda^T\Lambda$, we can actually get a pair of fairly neat MLE estimators. The details can be found at this website, and the restricted model is also called **probabilistic PCA**.

Now let us move on to the **EM algorithm** of factor analysis. Since the loss function $J(Q;\Lambda,\Psi)$ depends on the posterior distribution $Q(z^{(i)})=p(z^{(i)}|x^{(i)};\Lambda,\Psi)$, we first have to derive a way to compute this posterior. Consider the joint distribution

$$\begin{bmatrix}z\\x\end{bmatrix}\sim\mathcal{N}(\mu_{zx},\Sigma^*),$$

then

$$\mu_{zx}=\begin{bmatrix}\mu_z\\\mu_x\end{bmatrix}=\begin{bmatrix}0\\\mu\end{bmatrix},$$

$$\Sigma^*=\begin{bmatrix}\Sigma_{zz}&\Sigma_{zx}\\\Sigma_{xz}&\Sigma_{xx}\end{bmatrix}$$

where $\Sigma_{zz}=I_k$, $\Sigma_{xx}=\Lambda\Lambda^T+\Psi$,

$$\Sigma_{zx}=\mathrm{Cov}(z,x)=\mathrm{Cov}(z,\mu+\Lambda z+\epsilon)=\mathrm{Cov}(z,\Lambda z)=\Lambda^T,$$

and $\Sigma_{xz}=\Sigma_{zx}^T=\Lambda$. Therefore, by the formula of conditional multivariate Gaussian distribution, we conclude that $z|x\sim\mathcal{N}(\mu_{z|x},\Sigma_{z|x})$ where

$$\mu_{z|x}=\Lambda^T\left(\Lambda\Lambda^T+\Psi\right)^{-1}(x-\mu),$$

$$\Sigma_{z|x}=I_k-\Lambda^T\left(\Lambda\Lambda^T+\Psi\right)^{-1}\Lambda.$$

This is exactly the E-step.

Next let us move on to the M-step. Since $x|z\sim\mathcal{N}(\mu+\Lambda z,\Psi)$, consider

$$\arg\max_{\Lambda,\Psi}J(Q;\Lambda,\Psi) \tag{91}$$

$$=\arg\max_{\Lambda,\Psi}\sum_{i=1}^m\int_{z^{(i)}}Q_i(z^{(i)})\log\frac{p(x^{(i)},z^{(i)};\Lambda,\Psi)}{Q_i(z^{(i)})}dz^{(i)} \tag{92}$$

$$=\arg\max_{\Lambda,\Psi}\sum_{i=1}^m\int_{z^{(i)}}Q_i(z^{(i)})\log\frac{p(x^{(i)}|z^{(i)};\Lambda,\Psi)p(z^{(i)})}{Q_i(z^{(i)})}dz^{(i)} \tag{93}$$

$$=\arg\max_{\Lambda,\Psi}\sum_{i=1}^m\int_{z^{(i)}}Q_i(z^{(i)})\log p(x^{(i)}|z^{(i)};\Lambda,\Psi)dz^{(i)} \tag{94}$$

$$=\arg\max_{\Lambda,\Psi}\sum_{i=1}^m\mathbb{E}\left[\log p(x^{(i)}|z^{(i)};\Lambda,\Psi)\right] \tag{95}$$

$$=\arg\min_{\Lambda,\Psi}\sum_{i=1}^m\mathbb{E}\left[\log|\Psi|+\left(x^{(i)}-\mu-\Lambda z^{(i)}\right)^T\Psi^{-1}\left(x^{(i)}-\mu-\Lambda z^{(i)}\right)\right], \tag{96}$$

where the expectation here is *w.r.t.* $z^{(i)} \sim Q_i$.

In order to solve for Λ, take derivative of 96 *w.r.t.* Λ and set it as 0,

$$\nabla_\Lambda \sum_{i=1}^{m} \mathbb{E}\left[\log|\Psi| + \left(x^{(i)} - \mu - \Lambda z^{(i)}\right)^T \Psi^{-1}\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)\right] \tag{97}$$

$$= \nabla_\Lambda \sum_{i=1}^{m} \mathbb{E}\left[\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)^T \Psi^{-1}\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)\right] \tag{98}$$

$$= \nabla_\Lambda \sum_{i=1}^{m} \mathbb{E}\left[\mathrm{tr}\left(z^{(i)T}\Lambda^T\Psi^{-1}\Lambda z^{(i)}\right) - 2\mathrm{tr}\left(z^{(i)T}\Lambda^T\Psi^{-1}\left(x^{(i)} - \mu\right)\right)\right] \tag{99}$$

$$= \nabla_\Lambda \sum_{i=1}^{m} \mathbb{E}\left[\mathrm{tr}\left(\Psi^{-1}\Lambda z^{(i)}z^{(i)T}\Lambda^T\right) - 2\mathrm{tr}\left(\Psi^{-1}\left(x^{(i)} - \mu\right)z^{(i)T}\Lambda^T\right)\right] \tag{100}$$

$$= \sum_{i=1}^{m} \mathbb{E}\left[2\Psi^{-1}\Lambda z^{(i)}z^{(i)T} - 2\Psi^{-1}\left(x^{(i)} - \mu\right)z^{(i)T}\right] = 0. \tag{101}$$

Then we can get

$$\Lambda = \left(\sum_{i=1}^{m}\left(x^{(i)} - \mu\right)\mathbb{E}\left[z^{(i)}\right]^T\right)\left(\sum_{i=1}^{m}\mathbb{E}\left[z^{(i)}z^{(i)T}\right]\right)^{-1}.$$

In order to solve for Ψ, take derivative of equation 96 *w.r.t.* Ψ and set it as 0,

$$\nabla_\Psi \sum_{i=1}^{m} \mathbb{E}\left[\log|\Psi| + \left(x^{(i)} - \mu - \Lambda z^{(i)}\right)^T \Psi^{-1}\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)\right] \tag{102}$$

$$= \sum_{i=1}^{m} \mathbb{E}\left[\Psi^{-1} - \Psi^{-1}\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)^T\Psi^{-1}\right] = 0. \tag{103}$$

Then we can get

$$\begin{aligned}
\Psi &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)\left(x^{(i)} - \mu - \Lambda z^{(i)}\right)^T\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\Lambda z^{(i)}z^{(i)T}\Lambda^T + \Lambda z^{(i)}\left(\mu - x^{(i)}\right)^T + \left(\mu - x^{(i)}\right)z^{(i)T}\Lambda^T + \left(\mu - x^{(i)}\right)\left(\mu - x^{(i)}\right)^T\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\left\{\Lambda\mathbb{E}\left[z^{(i)}z^{(i)T}\right]\Lambda^T + \Lambda\mathbb{E}\left[z^{(i)}\right]\left(\mu - x^{(i)}\right)^T + \left(\mu - x^{(i)}\right)\mathbb{E}\left[z^{(i)}\right]^T\Lambda^T\right\} + \hat{\Sigma} \\
&= \hat{\Sigma} - \frac{1}{m}\sum_{i=1}^{m}\left\{\Lambda\mathbb{E}\left[z^{(i)}z^{(i)T}\right]\Lambda^T\right\}.
\end{aligned} \tag{104}$$

The last equation holds because

$$\left(\sum_{i=1}^{m}\left(x^{(i)} - \mu\right)\mathbb{E}\left[z^{(i)}\right]^T\right) = \Lambda\left(\sum_{i=1}^{m}\mathbb{E}\left[z^{(i)}z^{(i)T}\right]\right).$$

Note that the explicit expression of μ does not depend on any other parameters, the explicit expression of Λ only depends on the parameter μ, while the explicit expression of

Ψ depends on both μ and Λ. This implies that we should compute μ, Λ and Ψ in order, using the parameters computed just before.

In summary, the whole algorithm is

Set

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

for all iterations.

Repeat until convergence: {
(E-step) For every i, set

$$\mathbb{E}\left[z^{(i)}\right] = \Lambda^T\left(\Lambda\Lambda^T + \Psi\right)^{-1}\left(x^{(i)} - \mu\right),$$

$$\mathbb{E}\left[z^{(i)}z^{(i)T}\right] = I_k - \Lambda^T\left(\Lambda\Lambda^T + \Psi\right)^{-1}\Lambda + \mathbb{E}\left[z^{(i)}\right]\mathbb{E}\left[z^{(i)}\right]^T.$$

Note that the Λ and Ψ at the E-step are parameters computed from the t-th iteration.
(M-step) Update the parameters:

$$\Lambda^{[t+1]} := \left(\sum_{i=1}^{m}\left(x^{(i)} - \mu\right)\mathbb{E}\left[z^{(i)}\right]^T\right)\left(\sum_{i=1}^{m}\mathbb{E}\left[z^{(i)}z^{(i)T}\right]\right)^{-1},$$

$$\Psi^{[t+1]} := \hat{\Sigma} - \frac{1}{m}\sum_{i=1}^{m}\left\{\Lambda^{[t+1]}\mathbb{E}\left[z^{(i)}z^{(i)T}\right]\Lambda^{[t+1]T}\right\}.$$

}

Remark 2. One important thing to consider is the validity of EM algorithm, or in other words, do the EM algorithm results converge to the MLE estimators? Recall that in remark 1, we mentioned a pair of nice MLE estimators if we add a restriction $\Psi = \sigma^2 I_n$ to the model (and the resulting model is called PPCA). According to the same reference material, the MLE estimators are equivalent to the EM algorithm results.

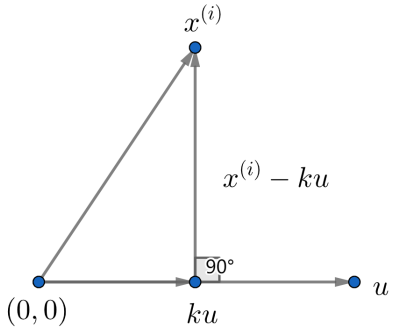
Remark 3. Still consider the PPCA model. If we add another restriction $\Lambda^T\Lambda = I$ so Λ is an orthogonal matrix, then through a similar derivation as the EM algorithm, we can conclude that Λ to PPCA is as U to PCA. In other words,

$$x - \mu = \Lambda z + \epsilon \qquad (in\ PPCA)$$

is in a way equivalent to

$$x = Uy \qquad (in\ PCA).$$

5.4 Derive PCA by picking the basis that minimizes the approximation error.



In order to minimize the approximation error, first we should derive the distance between each example point $x^{(i)}$ and the direction vector u where $||u|| = 1$. In the image shown above, ku is the projection $x^{(i)}$ onto u, and thus $\left(x^{(i)} - ku\right) \perp u$. Then we have:

$$\left(x^{(i)} - ku\right)^T u = 0 \iff k = x^{(i)T} u = u^T x^{(i)}.$$

As a result the distance we want can be expressed as:

$$\left(x^{(i)} - ku\right)^T \left(x^{(i)} - ku\right) \tag{105}$$
$$= x^{(i)T} x^{(i)} - k x^{(i)T} u - k u^T x^{(i)} + k^2 u^T u \tag{106}$$
$$= x^{(i)T} x^{(i)} - k^2 \tag{107}$$
$$= x^{(i)T} x^{(i)} - u^T x^{(i)} x^{(i)T} u \tag{108}$$

Next, let us minimize this distance over all example points,

$$\arg \min_{u,||u||=1} \frac{1}{m} \sum_{i=1}^{m} \left(x^{(i)} - ku\right)^T \left(x^{(i)} - ku\right) \tag{109}$$
$$= \arg \min_{u,||u||=1} \frac{1}{m} \sum_{i=1}^{m} \left(x^{(i)T} x^{(i)} - u^T x^{(i)} x^{(i)T} u\right) \tag{110}$$
$$= \arg \max_{u,||u||=1} \frac{1}{m} \sum_{i=1}^{m} u^T x^{(i)} x^{(i)T} u \tag{111}$$
$$= \arg \max_{u,||u||=1} u^T \left(\sum_{i=1}^{m} x^{(i)} x^{(i)T}\right) u \tag{112}$$
$$= \arg \max_{u,||u||=1} u^T \Sigma u. \tag{113}$$

This is equivalent to the objective function that we derive by maximizing the variance of projections.

Remark 1. The *Eckart-Young Theorem* is a more rigorous version of our statement above.

Remark 2. Sometimes people may wonder why among all n eigenvalues of Σ, only 3 or 4 of the greatest ones combined are large enough to represent over 85% or even over 90% of the total variance $\sum_i \lambda_i$. In other words, people suspect why spectral decomposition can ever achieve such an effect. The derivation gives us an insight: we motive PCA with an objective to minimize the approximation error or to maximize the variance; more specifically, we intend to find the direction u on which projections enjoy the greatest variance. Geometrically speaking, the eigenvectors u's are the axes of the contours from the distribution of x, if we assume x is Gaussian. Generally, in the case when PCA is used, the contour from data is not as perfect as a globe but more or less an ellipsoid (otherwise why bother to use PCA on uncorrelated data!). Thus we can always find several long axes that mainly represent the whole distribution.

5.5 Show that Λ is the covariance matrix of $y = U^T x$.

In order to solve the optimization problem 113, we use the method of Lagrange multipliers:

$$\mathcal{L} = u^T \Sigma u - \lambda \left(u^T u - 1 \right).$$

Take derivative *w.r.t.* u and λ, we have

$$\nabla_u \mathcal{L} = 2\Sigma u - 2\lambda u = 0,$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = u^T u - 1 = 0.$$

The equation $\Sigma u = \lambda u$ tells us that u is an eigenvector of the covariance matrix Σ while $u^T u - 1 = 0$ shows that the eigenvectors should be normalized such that U is an orthogonal matrix. By spectral decomposition we have $\Sigma = U \Lambda U^T$, therefore

$$\mathrm{Var}(y) = \mathrm{Cov}\left(U^T x, U^T x\right) = U^T \Sigma U = U^T U \Lambda U^T U = \Lambda.$$

This result shows that the variance of each dimension, y_k, is exactly the eigenvalue λ_k; and different dimensions are uncorrelated since $\Lambda_{ij} = 0$. This is why PCA is such a nice and neat dimensionality reduction method, and is quite fit for linear regression or other statistical procedures that requires independent input features.

5.6 Show that the eigenvectors u_i's form a new basis of the data.

In PCA we construct new data y from the raw data x by multiplying the orthogonal direction matrix

$$U = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_n \\ | & & | \end{bmatrix}_{n \times n};$$

specifically, the transformation is $y = U^T x$, and we can further obtain the equation $x = Uy$. According to simple algebra knowledge, x is in the column space of matrix U, or in other words, x is a linear combination of the eigenvectors u_i's. On the hand, since U is an orthogonal matrix, $\{u_1, ..., u_n\}$ forms a n-dim subspace of $\mathbb{R}^n$ (which is exactly the whole space!).

However, such U above actually cannot achieve the effect of dimensionality reduction since U forms a n-dim subspace rather a k-dim subspace ($k < n$). In practice we only take k main columns out of U and pile them up as a new matrix

$$U^* = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{bmatrix}_{n \times k}.$$

In this case, x is no longer in the column space of U^*. Instead, $U^* y^*$ is actually the projection of x onto the k-dim subspaces of $\mathbb{R}^n$ formed by the basis $\{u_1, ..., u_k\}$, and the projection matrix is $P_{n \times n} = U^* \left(U^{*T} U^*\right)^{-1} U^{*T}$.

Based on this fact, it's not hard to explain why our raw data x could be reversely estimated by

$$x \approx \hat{x} = y_1 u_1 + ... + y_k u_k,$$

and $\hat{x}$ will just lie on the new basis, removing noises from the the raw data x.

Remark. Many people may make a mistake by stating that u_i's form a basis of the new data y. This is false because y is never in the column space of any forms of U, but instead y is the coordinates on the new basis.

5.7 Explain the relationship between Y in PCA and Z in PCO.

PCO is short for "principal coordinate analysis" which can be regarded as the twin of PCA. In (population) PCA we consider the spectral decomposition of $\Sigma = U \Lambda U^T$ and get the new data y by a linear transformation $y = U^T x$; while in (mostly sample) PCO we consider the spectral decomposition of a newly contructed matrix related to $\hat{\Sigma}$, and get the new data directly.

Note that the sample covariance matrix can be rewritten as

$$\begin{aligned} \hat{\Sigma} &= \sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right)^T \left(x^{(i)} - \bar{x}\right) \\ &= \left(X - \bar{X}\right)^T \left(X - \bar{X}\right) \\ &= \left(X - \frac{1}{n} 1_n 1_n^T X\right)^T \left(X - \frac{1}{n} 1_n 1_n^T X\right) \\ &= X^T \left(I_n - \frac{1}{n} 1_n 1_n^T\right)^T \left(I_n - \frac{1}{n} 1_n 1_n^T\right) X \\ &= X^T H_n^T H_n X \qquad \left(let\ H_n = I_n - \frac{1}{n} 1_n 1_n^T\right) \\ &= X^T H_n X = X^T H_n H_n X. \end{aligned} \tag{114}$$

By spectral decomposition we have $\hat{\Sigma} = GLG$; therefore, piling up each $y^{(i)} = G_k^T \left(x^{(i)} - \bar{x}\right)$ row by row, we have

$$Y = \begin{bmatrix} — & y^{(1)T} & — \\ & \vdots & \\ — & y^{(n)T} & — \end{bmatrix} = (X - \bar{X}) G = H_n X G.$$

Let $T = H_n X X^T H_n$. Consider the spectral decomposition $TZ = Z\Lambda$, and normalize the eigenvectors such that $z_i^T z_i = \lambda_i$ for all i. If we compute the SVD decomposition of $H_n X$ as $H_n X = U\Gamma V^T$, then we have:

$$\hat{\Sigma} = (H_n X)^T (H_n X) = V\Gamma^2 V^T \quad (= GLG^T),$$
$$T = (H_n X)(H_n X)^T = U\Gamma^2 U^T \quad (= Z\Lambda Z^{-1}),$$
$$Y = (H_n X)G = U\Gamma V^T G = U\Gamma V^T V = U\Gamma.$$

The last equation holds since $V = G$, and $L = \Lambda = \Gamma^2$.

Next, let us proof that $Y = Z$.

Proof. It suffices to show that $U\Gamma = Z$. Since $TU = U\Gamma^2$, we have $TU\Gamma = U\Gamma\Gamma^2 = U\Gamma\Lambda$. Contrast the two equations

$$T(U\Gamma) = (U\Gamma)\Lambda$$

and

$$TZ = Z\Lambda,$$

we conclude that both $U\Gamma$ and Z are matrices of T's eigenvectors, maybe with minor difference of scaling. Therefore we need to check the length of eigenvectors from each matrix:

$$(U\Gamma)^T(U\Gamma) = \Gamma^T U^T U\Gamma = \Gamma^T\Gamma = \Lambda,$$

while

$$Z^T Z = \Lambda.$$

As a result, we conclude that $U\Gamma = Z$. □

Remark. PCO is just another version of PCA that ends up with the same result. Why and when PCO? Note that PCA computes the spectral decomposition of $\hat{\Sigma}$, a $p \times p$ matrix; while PCO computes the spectral decomposition of T, a $n \times n$ matrix. Therefore, if our data has the property $n \geq p$ (in most cases), then there's no doubt that we should use PCA. However, there are some cases that we have $n < p$ when we work on face recognition ($p = 3 \cdot 256^2$), gene analysis ($n < 100, p > 10^4$) or other things high-dimensional enough.

5.8 Show that MDS is equivalent to PCO.

MDS is short for "multidimensional scaling" method, which helps to find the exact points if we are given the distances between them. Let D be a distance matrix and define $B = H_n A H_n$, where $A_{ij} = -\frac{1}{2}D_{ij}$. Then D is Euclidean iff B is *p.s.d.*

Specifically, for a dataset $\{x^{(i)}\}, i = 1, ..., n$, if we construct its Euclidean distance matrix D and further construct the matrix B, then B's normalized eigenvectors z_i's (such that $z_i^T z_i = \lambda_i$ where λ_i is the corresponding eigenvalue) are exactly the vectors from Z in PCO.

Recall that Z in PCO comes from the spectral decomposition of matrix $T = H_n X X^T H_n$, while Z in MDS comes from the exactly same decomposition procedure of matrix $B = H_n A H_n$. Therefore, in order to show the equivalence of the Z in PCO to the Z in MDS, it suffices to show that $T = B$.

Proof. We start by constructing the Euclidean distance matrix D from the design matrix X. Since for all $i, j \in \{1, ..., n\}$,

$$D_{ij} = ||x^{(i)} - x^{(j)}||_2^2 = \left(x^{(i)} - x^{(j)}\right)^T \left(x^{(i)} - x^{(j)}\right) = x^{(i)T}x^{(i)} + x^{(j)T}x^{(j)} - 2x^{(i)T}x^{(j)},$$

we have

$$D = \text{diag}\left(XX^T\right) 1_n^T + 1_n \text{diag}\left(XX^T\right)^T - 2XX^T,$$

where $\text{diag}(\cdot)$ is a vector such that $\text{diag}(\cdot)_i = (\cdot)_{ii}$.

Therefore,

$$A = -\frac{1}{2}D = -\frac{1}{2}\text{diag}\left(XX^T\right) 1_n^T - \frac{1}{2}1_n \text{diag}\left(XX^T\right)^T + XX^T,$$

and

$$B = H_n A H_n = H_n X X^T H_n.$$

The last equation holds because both $H_n 1_n = 0$ and $1_n^T H_n = 0$.

This completes the proof. □

5.9 Describe the spectral clustering method.

Recall that in k-means algorithm we use the *distortion function* as a metric of error:

$$J(c, \mu) = \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2.$$

Now we would like to change some notations and consider

$$\hat{\Sigma}_w = \sum_{j=1}^{c} \sum_{i \in V_j} \left(x^{(i)} - \bar{x}_j\right) \left(x^{(i)} - \bar{x}_j\right)^T,$$

indicating that we now treat the metric of error as the **with-in class covariance matrix**. Particularly, we know that $J(c, \mu) = \text{tr}(\hat{\Sigma}_w)$ and therefore our goal here is to minimize $\text{tr}(\hat{\Sigma}_w)$.

However, before we manage to express $\text{tr}(\hat{\Sigma}_w)$ in the form of matrices, let us first consider the **total covariance matrix** $\hat{\Sigma}$ and the **between-class covariance matrix** $\hat{\Sigma}_w$, since it is acknowledgely known that

$$\hat{\Sigma} = \hat{\Sigma}_w + \hat{\Sigma}_b,$$

which is the fundamental principle of covariance decomposition and variance analysis.

Specifically,

$$\begin{aligned}
\hat{\Sigma}_b &= \sum_{j=1}^{c} n_j(\bar{x}_j - \bar{x})(\bar{x}_j - \bar{x})^T \\
&= \begin{bmatrix} | & & | \\ \bar{x}_1 - \bar{x} & \cdots & \bar{x}_c - \bar{x} \\ | & & | \end{bmatrix} \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_c \end{bmatrix} \begin{bmatrix} — & (\bar{x}_1 - \bar{x})^T & — \\ & \vdots & \\ — & (\bar{x}_c - \bar{x})^T & — \end{bmatrix} \\
&= \left(M - \frac{1}{n} 1_c 1_n^T X \right)^T D \left(M - \frac{1}{n} 1_c 1_n^T X \right) \\
&= \left(D^{-1} E^T X - \frac{1}{n} D^{-1} E^T 1_n 1_n^T X \right)^T D \left(D^{-1} E^T X - \frac{1}{n} D^{-1} E^T 1_n 1_n^T X \right) \\
&= \left(X^T H E D^{-1} \right) D \left(D^{-1} E^T H X \right) \\
&= X^T H E D^{-1} E^T H X,
\end{aligned} \tag{115}$$

where $E \in \mathbb{R}^{n \times c}$ and each row of E is of the form $\{0,1\}^c$ *s.t.* $E1_c = 1_n$. E represents the class that each example will be assigned to. As a result, we conclude that

$$\hat{\Sigma}_w = \hat{\Sigma} - \hat{\Sigma}_b = X^T H X - X^T H E D^{-1} E^T H X.$$

Now that we have the matrix-form $\hat{\Sigma}_w$, next we need to find the E that minimize the trace of $\hat{\Sigma}_w$, or in other words, we want

$$\begin{aligned}
\arg\min_E \operatorname{tr}\left(\hat{\Sigma}_w \right) &= \arg\min_E \operatorname{tr}\left(X^T H X - X^T H E D^{-1} E^T H X \right) \\
&= \arg\max_E \operatorname{tr}\left(X^T H E D^{-1} E^T H X \right) \\
&= \arg\max_E \operatorname{tr}\left(E^T H X X^T H E D^{-1} \right) \\
&= \arg\max_E \operatorname{tr}\left(D^{-\frac{1}{2}} E^T H X X^T H E D^{-\frac{1}{2}} \right).
\end{aligned} \tag{116}$$

Let us view some lemmas before moving on.

Lemma 9. *Let $A \in \mathbb{R}^{p \times p}$ be a symmetric matrix with eigenvalues $\alpha_1 > \cdots > \alpha_p$, then*

$$\max_{R^T R = I_k,\ k \le p} \operatorname{tr}\left(R^T A R \right) = \sum_{i=1}^{k} \alpha_i,$$

and

$$\arg \max_{R^T R = I_k,\ k \le p} \operatorname{tr}\left(R^T A R \right) = U Q^T,$$

where U is composed of the first k eigenvectors corresponding to $\alpha_1, ..., \alpha_k$, and $Q \in \mathbb{R}^{k \times k}$ is an arbitrary orthogonal matrix.

Proof. The Lagrangian is

$$\mathcal{L} = \operatorname{tr}\left(R^T A R - B \left(R^T R - I_k \right) \right).$$

Take derivative *w.r.t.* R and set it as 0, we have

$$\nabla_R \mathcal{L} = 2(AR - RB) = 0,$$

so

$$AR = RB.$$

Since B is a symmetric matrix, we perform spectral decomposition on $B = Q\Lambda Q^T$, and we get

$$AR = RQ\Lambda Q^T \Longleftrightarrow A(RQ) = (RQ)\Lambda,$$

indicating that (Λ, RQ) is the eigenpair of A.

Since

$$\text{tr}\left(R^T AR\right) = \text{tr}\left(Q^T R^T ARQ\right) = \text{tr}(\Lambda),$$

we conclude that the maximum value of the objective function is exactly the sum of eigenvalues. If we constrain $R \in \mathbb{R}^{p\times k}$, then we have to pick k eigenvectors corresponding to the k biggest eigenvalues, and pile them up as U_k. As a result,

$$U_k = R_{p\times k} Q_{k\times k} \Longleftrightarrow R_{p\times k} = \begin{bmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{bmatrix} Q_{k\times k}^T,$$

and

$$\text{tr}\left(Q_{k\times k} U_k^T A U_k Q_{k\times k}^T\right) = \text{tr}\left(U_k^T A U_k\right) = \sum_{i=1}^{k} \alpha_i.$$

□

Lemma 10 (Orthogonal Procrustes Problem)**.** *Suppose $X, Y \in \mathbb{R}^{n\times p}$, and $U \in \mathbb{R}^{p\times p}$ be an orthogonal matrix which minimizes $||Y - XU^T||_F^2$ s.t. $U^T U = I_p$. Then $U = QR^T$, where Q and R are from the SVD on $Y^T X = Q\Lambda R^T$.*

The proof can be found on this website.

Now let us go back to the clustering problem and focus on the equation 116. Without losing of generality, let $R = ED^{-\frac{1}{2}}$ and $A = HXX^T H$; this is valid since $E^T E = D$ and thus $(ED^{-\frac{1}{2}})^T ED^{-\frac{1}{2}} = I_c$. Then according to lemma 9, we conclude that the maxima can be expressed as $R^* = (ED^{-\frac{1}{2}})^* = UQ^T$, where $U \in \mathbb{R}^{n\times c}$ is the matrix of c eigenvectors corresponding to the first c biggest eigenvalues, while $Q \in \mathbb{R}^{c\times c}$ is an arbitrary orthogonal matrix.

However, we still have two problems unsolved: how to determine Q, and moreover, how to estimate E given the expression of $ED^{-\frac{1}{2}}$. By lemma 10, we consider the optimization problem

$$\min_{Q^T Q = I_c} ||ED^{-\frac{1}{2}} - UQ^T||_F^2,$$

and we can come up with the following algorithm to complete the description of the spectral clustering method.

Algorithm 4: Spectral clustering method with Procrustean rounding

Input: The design matrix $X \in \mathbb{R}^{n \times p}$

Output: The class matrix E

Decomposition: perform the spectral decompostion on $HX^TXH = U\Lambda U^T$

Initialization: choose the initial class matrix E

repeat

  Compute $D = E^TE$

  Perform SVD on $D^{-\frac{1}{2}}E^TU_c = \Theta\Gamma V^T$ where U_c is the matrix of eigenvectors corresponding to the first c biggest eigenvalues, and compute $Q = \Theta V^T$

  Update $E \leftarrow U_cQ^TD^{-\frac{1}{2}}$

until convergence;