Korea Advanced Institute of Science and Technology
School of Electrical Engineering
EE488 Intro. to Machine Learning, Spring 2018

Issued: Apr. 30, 2018
Due: May. 23, 2018

Assignment III - part I

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. Anyone suspected of copying others will be penalized. The homework will take considerable amount of time so start early.

1. **K-NN** For a given test sample $\mathbf{x}$ with nearest neighbor $\mathbf{x}_n$, error occurs when the label associated with $\mathbf{x}_n$ denoted as $\theta_n \in \Omega$ does not equal the true label of $\mathbf{x}$ denoted as $\theta \in \Omega$. Thus the error rate given both the $\mathbf{x}$ and its nearest neighbor $\mathbf{x}_n$ is $P(\theta \neq \theta_n | \mathbf{x}, \mathbf{x}_n)$. Assume finite label set of C classes such that $\Omega = \{\omega_1, \omega_2, \ldots, \omega_C\}$.

   (i) Given $\mathbf{x}$ and nearest neighbor $\mathbf{x}_n$, express error rate in terms of accuracy (when $\theta = \theta_n = \omega_l$ where $l \in \{1, \ldots, C\}$).
   [Solution] The error rate can be expressed as:

$$P(\theta \neq \theta_n | \mathbf{x}, \mathbf{x}_n) = 1 - \sum_{l=1}^{\mathcal{C}} P(\theta = \omega_l, \theta_n = \omega_l | x, x_n) = 1 - \sum_{i=1}^{C} P^2(\omega_i | \mathbf{x})$$

   (ii) When $P(\omega_m | \mathbf{x}) = S$, then $\sum_{i=1}^{C} P^2(\omega_i | \mathbf{x})$ is minimum when $P(\omega_i | \mathbf{x}) = \frac{1-S}{C-1}$ for $i = 1, \ldots, m-1, m+1, \ldots, C$. Explain why this might be so.
   [Solution] Minimization under the constraint $P(\omega_m | \mathbf{x}) - S = 0$

$$\begin{aligned} L &= \sum_{i=1}^{C} P^2(\omega_i | \mathbf{x}) + \lambda(P(\omega_m | \mathbf{x}) - S) \\ &= P^2(\omega_m | \mathbf{x}) + \sum_{i=1, i \neq m}^{C} P^2(\omega_i | \mathbf{x}) + \lambda(P(\omega_m | \mathbf{x}) - S) \\ &= P^2(\omega_m | \mathbf{x}) + \sum_{i=1, i \neq m}^{C} P^2(\omega_i | \mathbf{x}) + \lambda(1 - \sum_{i=1, i \neq m}^{C} P(\omega_i | \mathbf{x}) - S) \end{aligned}$$

   Differentiate with $P(\omega_i | \mathbf{x})$

$$2P(\omega_i | \mathbf{x}) - \lambda = 0$$
$$P(\omega_i | \mathbf{x}) = \frac{\lambda}{2}$$

   Therefore $\sum_{i=1}^{C} P^2(\omega_i | \mathbf{x})$ is minimum when $P(\omega_i | \mathbf{x}) = \frac{\lambda}{2}$ for $i = 1, ..., m-1, m+1, ..., C$. Using the constraint

$$\begin{aligned} S &= P(\omega_m|\mathbf{x}) \\ &= 1 - \sum_{i=1, i\neq m}^{C} P(\omega_i|\mathbf{x}) \\ &= 1 - (C-1)P(\omega_i|\mathbf{x}) \end{aligned}$$

Therefore $\sum_{i=1}^{C} P^2(\omega_i|\mathbf{x})$ is minimum when $P(\omega_i|\mathbf{x}) = \frac{1-S}{C-1}$.

(iii) Assume a dense training set in other words training data of infinite size, then the nearest neighbor of $\mathbf{x}$ will be itself such that $\mathbf{x}_n = \mathbf{x}$. Assume Bayes error rate defined as $P^*(e|\mathbf{x}) = 1 - \max_{\theta\in\Omega} P(\theta|\mathbf{x}) = 1 - P(\omega_m|\mathbf{x})$. Lower bound $\sum_{i=1}^{C} P^2(\omega_i|\mathbf{x})$ in terms of $P^*(e|\mathbf{x})$.

(iv) Assume a dense training set in other words training data of infinite size, then the nearest neighbor of $\mathbf{x}$ will be itself such that $\mathbf{x}_n = \mathbf{x}$. Assume Bayes error rate defined as $P^*(e|\mathbf{x}) = 1 - \max_{\theta\in\Omega} P(\theta|\mathbf{x}) = 1 - P(\omega_m|\mathbf{x})$. Lower bound $\sum_{i=1}^{C} P^2(\omega_i|\mathbf{x})$ in terms of $P^*(e|\mathbf{x})$.
[Solution] Using above

$$\begin{aligned} \sum_{i=1}^{C} P^2(\omega_i|\mathbf{x}) &= P^2(\omega_m|\mathbf{x}) + \sum_{i=1, i\neq m}^{C} P^2(\omega_i|\mathbf{x}) \\ &\geq (1 - P^*(e|\mathbf{x}))^2 + (\frac{P^*(e|\mathbf{x})}{C-1})^2(C-1) \\ &= (1 - P^*(e|\mathbf{x}))^2 + \frac{P^{*2}(e|\mathbf{x})}{C-1} \end{aligned}$$

(v) Upper bound error rate in terms of Bayes error rate.
[Solution] Using above

$$\begin{aligned} P(e|\mathbf{x}) &= 1 - \sum_{i=1}^{C} P^2(\omega_i|\mathbf{x}) \\ &\leq 1 - (1 - P^*(e|\mathbf{x}))^2 - \frac{P^{*2}(e|\mathbf{x})}{C-1} \\ &= 2P^*(e|\mathbf{x}) - \frac{C}{C-1} P^{*2}(e|\mathbf{x}) \end{aligned}$$

2. **Bayesian Statistics**

(i) Given observation data $D = \{0, 0, 1, 0, 1, 1, 1\}$, use beta prior distribution with parameter (α_0, β_0) to determine $P(x = 1|D)$ in terms of (α_0, β_0).
(solution):

$$\begin{aligned} p(\theta|D) &\propto p(D|\theta)p(\theta) \\ &\propto \begin{pmatrix} 7 \\ 4 \end{pmatrix} \theta^4(1-\theta)^{7-4} \frac{\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)} \theta^{\alpha_0-1}(1-\theta)^{\beta_0-1} \\ &\theta^{4-\alpha_0-1}(1-\theta)^{7-4-\beta_0-1} \\ &= \text{Beta}(\alpha_0+4, \beta_0+3) \end{aligned}$$

Thus,

$$\begin{aligned} p(x=1|D) &= \int_0^1 p(x=1|\theta, D)p(\theta|D)d\theta \\ &= \int_0^1 p(x=1|\theta)p(\theta|D)d\theta \\ &= \int_0^1 \theta p(\theta|D)d\theta \\ &= \mathbb{E}[\theta|D] \\ &= \frac{4+\alpha_0}{7+\alpha_0+\beta_0} \text{ where } \mathbb{E}[\theta] = \frac{\alpha}{\alpha+\beta} \text{ for } \theta \sim \text{Beta}(\alpha, \beta) \end{aligned}$$

(ii) Given observation data $D = \{x_1, x_2, \ldots, x_n\}$ from a Gaussian distribution with mean μ and known variance σ^2. Using prior distribution for μ to be a Gaussian with mean μ_0 and variance $\frac{1}{r_0}$, determine $P(\mu|D)$.
(solution):

a. Let $x_1, x_2, \cdots, x_n$ be an i.i.d. sample, the likelihood is given by

$$\begin{aligned} p(D|\mu, \sigma^2) &= \prod_{i=1}^{n} p(x_i|\mu, \sigma^2) \\ &= \frac{1}{(2\pi\sigma^2)^{-n/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right) \end{aligned}$$

Let's define the empirical mean and variance

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

Rewrite the term in the exponent as follows

$$\begin{aligned} \sum_i (x_i-\mu)^2 &= \sum_i [(x_i-\bar{x}) - (\mu-\bar{x})]^2 \\ &= \sum_i (x_i-\bar{x})^2 + \sum_i (\bar{x}-\mu)^2 - \sum_i (x_i-\bar{x})(\mu-\bar{x}) \\ &= ns^2 + n(\bar{x}-\mu)^2 \end{aligned}$$

$$\begin{aligned} \because \sum_i (x_i-\bar{x})(\mu-\bar{x}) &= (\mu-\bar{x})\left(\left(\sum_i x_i\right) - n\bar{x}\right) \\ &= (\mu-\bar{x})(n\bar{x}-n\bar{x}) = 0 \end{aligned}$$

Hence

$$\begin{aligned} p(D|\mu,\sigma^2) &= \frac{1}{(2\pi)^{n/2}}\frac{1}{\sigma^n}\exp\left(-\frac{1}{2\sigma^2}[ns^2+n(\bar{x}-\mu)^2]\right) \\ &\propto \left(\frac{1}{\sigma^2}\right)^{n/2}\exp\left(-\frac{n}{2\sigma^2}(\bar{x}-\mu)^2\right)\exp\left(-\frac{ns^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{n}{2\sigma^2}(\bar{x}-\mu)^2\right) \\ &\propto N(\bar{x}|\mu,\frac{\sigma^2}{n}) \end{aligned}$$

b. Prior: the natural conjugate prior has the form

$$\begin{aligned} p(\mu) &\propto \exp\left(-\frac{n}{2\sigma_0^2}(\mu-\mu_0)^2\right) \\ &\propto N(\mu|\mu_0,\sigma_0) \text{ where } \sigma_0 = \frac{1}{r_0} \end{aligned}$$

c. Posterior

$$\begin{aligned} p(\mu|D) &\propto p(D|\mu,\sigma^2)p(\mu|\mu_0,\sigma_0^2) \\ &\propto \exp\left(-\frac{n}{2\sigma^2}(\bar{x}-\mu)^2\right)\exp\left(-\frac{n}{2\sigma_0^2}(\mu-\mu_0)^2\right) \\ &= \exp\left(-\frac{n}{2\sigma^2}\sum_i(x_i^2+\mu^2-2x_i\mu)-\frac{n}{2\sigma_0^2}(\mu^2+\mu_0^2-2\mu_0\mu)\right) \\ &\propto \exp\left(-\frac{\mu^2}{2}\left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)+\mu\left(\frac{\mu_0}{\sigma_0^2}+\frac{\sum_i x_i}{\sigma^2}\right)-\left(\frac{\mu_0^2}{2\sigma_0^2}+\frac{\sum_i x_i^2}{2\sigma^2}\right)\right) \\ &\stackrel{\text{def}}{=} \exp\left(-\frac{1}{2\sigma_n^2}(\mu^2-2\mu\mu_n+\mu_n^2)\right) = \exp\left(-\frac{1}{2\sigma_n^2}(\mu-\mu_n)^2\right) \end{aligned}$$

where the product of two Gaussians is Gaussian.

Matching coefficients of μ^2, we find σ_n^2 is given by

$$\begin{aligned} \frac{-\mu^2}{2\sigma_n^2} &= \frac{-\mu^2}{2}\left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right) \\ \frac{1}{\sigma_n^2} &= \frac{1}{\sigma_0^2}+\frac{n}{\sigma^2} \\ \sigma_n^2 &= \frac{\sigma^2\sigma_0^2}{n\sigma_0^2+\sigma^2} = \frac{1}{\frac{n}{\sigma^2}+\frac{1}{\sigma_0^2}} \end{aligned}$$

Matching coefficients of μ we get

$$\begin{aligned} \frac{2\mu\mu_n}{-2\sigma_n^2} &= \mu\left(\frac{\sum_i^n x_i}{\sigma^2}+\frac{\mu_0}{\sigma_0^2}\right) \\ \frac{\mu_n}{\sigma_n^2} &= \frac{\sum_{i=1}^n x_i}{\sigma^2}+\frac{\mu_0}{\sigma_0^2} = \frac{\sigma_0^2 n\bar{x}+\sigma^2\mu_0}{\sigma^2\sigma_0^2} \end{aligned}$$

Hence

$$\mu_n = \frac{\sigma^2}{n\sigma_0^2+\sigma^2}\mu_0+\frac{n\sigma_0^2}{n\sigma_0^2+\sigma^2}\bar{x} = \sigma_n^2\left(\frac{\mu_0}{\sigma_0^2}+\frac{n\bar{x}}{\sigma^2}\right)$$

This operation of matching first and second powers of μ is called completing the square. The precision of a Gaussian, which is 1/variance,

$$\lambda = 1/\sigma^2$$
$$\lambda_0 = 1/\sigma_0^2$$
$$\lambda_n = 1/\sigma_n^2$$

The posterior can be rewritten by

$$p(\mu|D,\lambda) = N(\mu|\mu_n, \lambda_n)$$
$$\lambda_n = \lambda_0 + n\lambda$$
$$\mu_n = \frac{\bar{x}n\lambda + \mu_0\lambda_0}{\lambda_n} = w\mu_{ML} + (1-w)\mu_0 \text{ where } w = \frac{n\lambda}{\lambda_n}$$

(iii) Given observation data $D = \{x_1, x_2, \ldots, x_n\}$ from a Gaussian distribution with mean μ and precision γ. Using prior distribution for (μ, γ) a Normal-gamma distribution, determine $P(\mu, \gamma|D)$.
(solution):

a. Let $x_1, x_2, \cdots, x_n$ be an i.i.d. sample, the likelihood is given by

$$\begin{aligned} p(D|\mu,\gamma) &= \prod_{i=1}^{n} p(x_i|\mu, 1/\gamma) \\ &= \frac{1}{(2\pi)^{n/2}}\gamma^{n/2}\exp\left(-\frac{\gamma}{2}\sum_{i=1}^{n}(x_i-\mu)^2\right) \\ &= \frac{1}{(2\pi)^{n/2}}\gamma^{n/2}\exp\left(-\frac{\gamma}{2}\left(n(\mu-\bar{x})^2 + \sum_{i=1}^{n}(x_i-\bar{x})^2\right)\right) \leftarrow \text{(ii.a)} \\ &\propto \gamma^{n/2}\exp\left(-\frac{n\gamma}{2}(\bar{x}-\mu)^2\right) \end{aligned}$$

b. Prior is the Normal-Gamma:

$$\begin{aligned} NG(\mu,\gamma|\mu_0,k_0,\alpha_0,\beta_0) &\stackrel{\text{def}}{=} N(\mu|\mu_0,(k_0\gamma)^{-1})Ga(\gamma|\alpha_0,\beta_0) \\ &= \frac{1}{Z_{NG}(\mu_0,k_0,\alpha_0,\beta_0)}\gamma^{1/2}\exp\left(-\frac{k_0\gamma}{2}(\mu-\mu_0)^2\right)\gamma^{\alpha_0-1}e^{-\gamma\beta_0} \\ &= \frac{1}{Z_{NG}}\gamma^{\alpha_0-\frac{1}{2}}\exp\left(-\frac{\gamma}{2}(k_0(\mu-\mu_0)^2)+2\beta_0)\right) \end{aligned}$$

$$\text{where } Z_{NG}(\mu_0,k_0,\alpha_0,\beta_0) = \frac{\Gamma(\alpha_0)}{\beta_0^{\alpha_0}}\left(\frac{2\pi}{k_0}\right)^{1/2}$$

c. Posterior can be derived as follows:

$$
\begin{aligned}
p(\mu,\gamma|D) &\propto NG(\mu|\gamma|\mu_0,k_0,\alpha_0,\beta_0)p(D|\mu,\gamma) \\
&\propto \gamma^{\frac{1}{2}}\exp\left(-\frac{k_0\gamma(\mu-\mu_0)^2}{2}\right)\gamma^{\alpha_0-1}\exp\left(-\beta_0\gamma\right)\times\gamma^{n/2}\exp\left(-\frac{\gamma}{2}\sum_{i=1}^{n}(x_i-\mu)^2\right) \\
&\propto \gamma^{\frac{1}{2}}\gamma^{\alpha_0+n/2-1}\exp\left(-\beta_0\gamma\right)\exp\left(-\frac{\gamma}{2}\left(k_0(\mu-\mu_0)^2+\sum_{i=1}^{n}(x_i-\mu)^2\right)\right) \\
&\propto \gamma^{\frac{1}{2}}\gamma^{\alpha_0+n/2-1}\exp\left(-\beta_0\gamma\right)\exp\left(-\frac{\gamma}{2}\left(k_0(\mu-\mu_0)^2+n(\mu-\bar{x})^2+\sum_{i=1}^{n}(x_i-\bar{x})^2\right)\right) \\
&\propto \gamma^{\frac{1}{2}}\exp\left(-\frac{\gamma}{2}(k_0+n)(\mu-\mu_n)^2\right)\times\gamma^{\alpha_0+n/2-1}\exp(-\beta_0\gamma)\exp\left(-\frac{\gamma}{2}\sum_{i}(x_i-\bar{x})^2\right) \\
&\exp\left(-\frac{\gamma}{2}\frac{k_0n(\bar{x}-\mu_0)^2}{k_0+n}\right) \\
&\propto N(\mu|\mu_n,((k+n)\gamma)^{-1})Ga(\gamma|\alpha_0+n/2,\beta_n)
\end{aligned}
$$

where

$$
\begin{aligned}
\mu_n &= \frac{k_0\mu_0+n\bar{x}}{k_0+n} \\
\beta_n &= \beta_0+\frac{1}{2}\sum_{i=1}^{n}(x_i-\bar{x})^2+\frac{k_0n(\bar{x}-\mu_0)^2}{2(k_0+n)} \\
k_n &= k_0+n \\
\alpha_n &= \alpha_0+n/2
\end{aligned}
$$

(iv) Express poisson, multinomial distribution, Laplace distribution, Dirichlet distribution, and gamma distribution in exponential form. Determine the conjugate prior for each of the distribution.

(solution):

a. Exponentail from of Possion distribution:

$$p(x|\lambda)=\frac{\lambda^x e^{-\lambda}}{x!}=\frac{1}{x!}e^{-\lambda}e^{x\ln\lambda}$$

where

A. $\eta=\ln\lambda, h(x)=\frac{1}{x!}$
B. $g(\eta)=e^{-\lambda}=e^{-e^{\eta}}$
C. $u(x)=x$
D. $p(x|\eta)=\frac{1}{x!}e^{-e^{\eta}}e^{\eta x}$

$\rightarrow$ The conjugate prior distribution of Possion distribution is Gamma distribution.

b. Exponential form of Multinomial distiribution.

$$p(x|\mu) = \prod_{k=1}^{K} \mu_k^{x_k} = \exp\left(\sum_{k=1}^{K} x_k \ln(\mu_k)\right)$$

$$\text{where } \sum_{k=1}^{K} \mu_k = 1, \mathbf{x} = (x_1, x_2, \cdots, x_k), \mu = (\mu_1, \mu_2, \cdots, \mu_k)$$

we can rewrite:

$$= \exp\left(\sum_{k=1}^{K-1} x_k \ln \mu_k + \left(1 - \sum_{k=1}^{K-1} x_k\right) \ln\left(1 - \sum_{k=1}^{K-1} \mu_k\right)\right)$$

$$= \exp\left(\sum_{k=1}^{K-1} \ln\left(\frac{\mu_k}{1 - \sum_{k=1}^{K-1} \mu_k}\right) x_k + \ln\left(1 - \sum_{k=1}^{K-1} \mu_k\right)\right)$$

$$= \left(1 - \sum_{k=1}^{K-1} \mu_k\right) \exp\left(\sum_{k=1}^{K-1} \ln\left(\frac{\mu_k}{1 - \sum_{k=1}^{K-1} \mu_k}\right) x_k\right)$$

$$\text{where } \mu_K = 1 - \sum_{k=1}^{K-1} \mu_k.$$

A. $\eta_k = \ln\left(\frac{\mu_k}{1-\sum_{k=1}^{K-1} \mu_k}\right) = \ln\left(\frac{\mu_k}{\mu_K}\right); \mu_k = \frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}} = softmax(k, \eta)$

B. $h(x) = 1$

C. $g(\eta) = 1 - \sum_{k=1}^{K-1} \mu_k = \mu_K = \frac{1}{\sum_{k=1}^{K} e^{\eta_k}}$

D. $u(x) = x_k \in \{x_1, x_2, \cdots, x_K\}$

E. $p(x|\eta) = \frac{e^{\eta_k x_k}}{\sum_{k=1}^{K} e^{\eta_k}}; \eta = (\eta_1, \eta_2, \cdots, \eta_K)^T$

→ The conjugate prior distribution of Multinomial distribution is Dirichlet distribution.

c. Exponential form of Laplace distribution:
The Laplace distribution can be described by parameters $\theta = (\mu, \lambda)$ and pdf

$$p(x|\mu, \lambda) = \frac{\lambda}{2} \exp\left(-\lambda|\mathbf{x} - \mu|\right).$$

Let $p \in (0, 1), \lambda \in (0, \infty)$ and $\mu \in \mathbb{R}$ be given. Then the pdf given by:

$$\psi(x|\mu, \lambda, p) = \begin{cases} \beta \exp(-\lambda\alpha(x - \mu)), & x \geq \mu \\ \beta \exp(\lambda\alpha^{-1}(x - \mu)), & x < \mu, \end{cases}$$

where $\alpha = \sqrt{\frac{p}{1-p}}$ and $\beta = \frac{\lambda\alpha}{\alpha^2+1}$. The exponential family with the following funtions:

A. $\eta(\lambda, p) = \begin{bmatrix} -\lambda\alpha \\ -\lambda\alpha^{-1} \end{bmatrix}$

B. $g(\lambda, p) = \beta$

C. $h(x) = 1, u(x) = \begin{bmatrix} |x - \mu|\mathbb{I}[x \geq \mu] \\ |x - \mu|\mathbb{I}[x < \mu] \end{bmatrix}$

D. $p(x|\eta) = \beta \exp(\eta(\lambda, p)u(x))$

→ The conjugate prior of Laplace distiribution is Gamma-Gamma-Beta:

$$p(p, \lambda|\nu, \chi) = Gamma(\lambda\alpha|\nu, \chi_1)Gamma(\lambda\alpha^{-1}|\nu, \chi_2)Beta(p|\nu')$$

d. Exponential form of Dirichlet distiribution:

$$\begin{aligned} p(x_1, \cdots, x_K|\alpha) &= Dir(x_1, \cdots, x_K|\alpha) \\ &= \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, x_i > 0, \sum_i x_i = 1; \alpha_i > 0 \\ &= \exp(\ln \Gamma\left(\sum_i^K \alpha_i\right) - \sum_{i=1}^{K} \Gamma(\alpha_i)) \exp\left(\begin{bmatrix} \alpha_i - 1 \\ \vdots \\ \alpha_K - 1 \end{bmatrix}^T \begin{bmatrix} \ln x_1 \\ \vdots \\ \ln x_K \end{bmatrix}\right) \end{aligned}$$

A. $\eta = \begin{bmatrix} \alpha_i - 1 \\ \vdots \\ \alpha_K - 1 \end{bmatrix} = \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_K \end{bmatrix}$

B. $u(x) = \begin{bmatrix} \ln x_1 \\ \vdots \\ \ln x_K \end{bmatrix}$

C. $h(x) = 1$

D. $g(\eta) = \exp\left(\ln \Gamma\left(\sum_{i=1}^{K} \eta_i\right) - \sum_{i=1}^{K} \Gamma(\eta_i)\right)$

E. $p(x|\eta) = g(\eta) \exp(\eta^T u(x))$

→ The conjugate prior of Dirichlet distribution is itself since Dirichlet distribution belongs to the exponential family.

e. Exponential form of Gamma distribution:

$$p(\mathbf{x}|\eta) = h(\mathbf{x}) g(\eta) e^{\eta^T u(\mathbf{x})}$$

$$\begin{aligned} Gamma(\mathbf{x}|\alpha, \beta) &= p(\mathbf{x}|\alpha, \beta) \\ &= \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, x \geq 0; \alpha, \beta > 0 \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \exp(-\beta x + (\alpha - 1) \ln(x)) \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \exp\left(\begin{bmatrix} -\beta \\ \alpha - 1 \end{bmatrix}^T \begin{bmatrix} x \\ \ln(x) \end{bmatrix}\right) \\ &= \frac{(-(\eta_1)^{\eta_2+1})}{\Gamma(\eta_2 + 1)} \exp\left(\eta^T \begin{bmatrix} x \\ \ln(x) \end{bmatrix}\right) \end{aligned}$$

where

A. $h(\mathbf{x}) = 1$

B. $g(\eta) = \frac{\beta^\alpha}{\Gamma(\alpha)} = \frac{(-(\eta_1)^{\eta_2+1})}{\Gamma(\eta_2+1)}$

C. $\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} -\beta \\ \alpha - 1 \end{bmatrix} \rightarrow \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} -\eta_1 \\ \eta_2 + 1 \end{bmatrix}$

D. $u(x) = \begin{bmatrix} x \\ \ln(x) \end{bmatrix}$

E. $p(x|\eta) = g(\eta) \exp(\eta^T u(x))$

→ The conjugate prior of Gamma distribution is Gamma distiribution.

3. **Decision Theory**: Consider a binary classification problem where binary data x is generated from two different Bernoulli distribution where each is choosen depending on the binary value of y which follows a Bernoulli distribution, $y \sim Bernoulli(\frac{1}{2})$. If $y = 1$, then $x \sim Bernoulli(p)$ and otherwise, $x \sim Bernoulli(q)$. Assume that $p > q$. What is the Bayes optimal classifier and what is its expect loss or risk based on 0-1 loss?
[solution] Notice that when $y \sim Bernoulli(\frac{1}{2})$, we have $P(y = 1) = P(y = 0) = 1/2$.

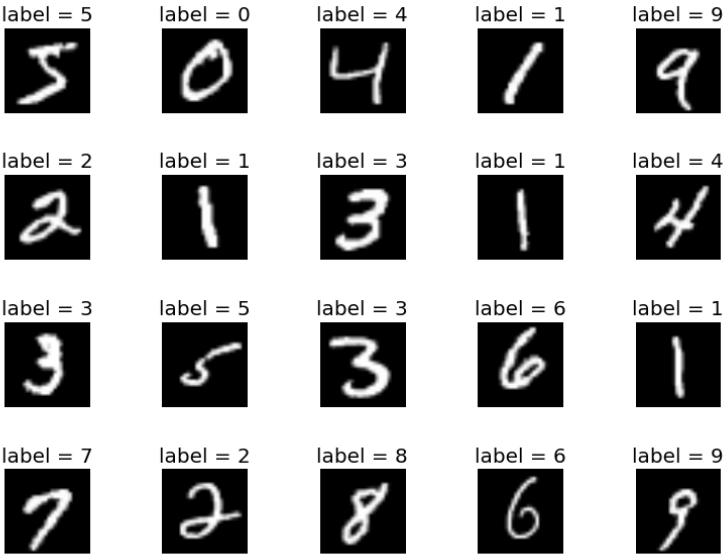
$$\begin{aligned} f^*(x') = \arg\max_{y'} P(y = y'|x = x') &= \arg\max_{y'} P(x = x'|y = y')P(y = y') \\ &= \arg\max_{y'} P(x = x'|y = y') \end{aligned}$$

Therefore, $f^*(1) = 1$ since $p = P(x = 1|y = 1) < P(x = 1|y = 0) = q$, and $f^*(0) = 0$ since $1 - p = P(x = 0|y = 1) > P(x = 0|y = 0) = 1 - q$. Hence $f^*(x) = x$.
The risk is $R^* = P(f^*(x) \neq y) = P(x \neq y)$. Therefore, the Bayes risk is

$$R^* = P(y = 1)P(x = 0|y = 1) + P(y = 0)P(x = 1|y = 0) = \frac{1}{2} \cdot (1 - p) + \frac{1}{2} \cdot q.$$

4. **Implementing Backpropagation algorithm(Matlab Programming Assignment)**

Write a matlab code for classifying the handwritten digits included in the MNIST database. Examples of hand written digits are shown below. The MNIST dataset includes 60,000 training and 10,000 testing. The size of each image is 28×28, and the each image can be classified into 10 different classes (0,1,...,9)



For implementation, a skeleton code is provided in the "nn" folder. The main function `main_nn.m` calls the following 5 functions:(1) `[x_train,y_train,x_test,y_test] = createDataset()`, (2) `net = initialize_network(num_neuron, init)`, (3) `[net] = feed_forward(data_input, net)`,(4) `net_update = back_propagation(net, data_output)`, (5) `net = weight_update(net, net_update, training)`.

(i) `[x_train,y_train,x_test,y_test] = createDataset()` loads both the training and test dataset respectively.

```
function [x_train,y_train,x_test,y_test] = createDataset()
%% Codes %%
addpath ./data;
x_train = loadMNISTImages('./data/train-images-idx3-ubyte')';
y_train = loadMNISTLabels('./data/train-labels-idx1-ubyte');
y_train(y_train==0) = 10; % Remap 0 to 10
tt = zeros(size(y_train,1),10);
tt(sub2ind(size(tt),1:size(y_train,1),y_train'))=1;
y_train = tt;
%load test data
x_test = loadMNISTImages('./data/t10k-images-idx3-ubyte')';
y_test = loadMNISTLabels('./data/t10k-labels-idx1-ubyte');
y_test(y_test==0) = 10; % Remap 0 to 10
tt = zeros(size(y_test,1),10);
tt(sub2ind(size(tt),1:size(y_test,1),y_test'))=1;
y_test = tt;
end
```

(ii) **`net = initialize_network(num_neuron, init)`** takes vector **`num_neuron`** which specifies how many hidden nodes to use per each layer and **`init`** which has mean and stdev of weight parameters while doing the initialization. The output of this function gives struct architecture **`net`** that saves weight parameters and biases.

```
function net = initialize_network(num_neuron, init)

%% initialize structure
net.layer_num = length(num_neuron);
net.num_neuron = num_neuron;

%% initialize each layer
net.layer = cell(net.layer_num,1);
for layer_index = 1 : net.layer_num
net.layer{layer_index, 1} = zeros(net.num_neuron(layer_index, 1), 1);
end
%% initialize weight
net.weight = cell(net.layer_num,1);
for layer_index = 2 : net.layer_num
net.weight{layer_index, 1} = init.weight_std * randn(net.num_neuron(layer_index, 1),...
 net.num_neuron(layer_index-1, 1));
end
%% initialize bias
net.bias = cell(net.layer_num, 1);
for layer_index = 2 : net.layer_num
net.bias{layer_index, 1} = init.bias_std * randn(net.num_neuron(layer_index, 1), 1);
end

end
```

(iii) **`[net, pred] = feed_forward(data_input, net)`** takes mini-batch of training set {**`x_train`**,**`y_train`**} where **`x_train`** is the data array while **`y_train`** is the corresponding label array of training dataset and mini-batch of training set is represented as {**`data_input, data_output`**}. The outputs of this function are **`net`** which saves all the forward propagation informations and **`pred`** which estimates the label of corresponding inputs.

```
function [net, pred] = feed_foward(input, net)

%% Your code here %%
% Make activation function
activation_function = @(x) (1./(1+exp(-1*x)));
```

```
%% Codes %%
net.layer{1,1} = input;
for index_layer = 2 : net.layer_num
net.layer{index_layer, 1} = net.weight{index_layer, 1} * net.layer{index_layer-1, 1} ...
+ net.bias{index_layer, 1};
net.layer{index_layer, 1} = activation_function(net.layer{index_layer, 1});
end

[~,ind] = max(net.layer{index_layer, 1});
pred = zeros(size(net.layer{index_layer, 1}));
for i=1:size(ind,2)
pred(ind(i),i) = 1;
end
end
```

(iv) `net_update = back_propagation(net, data_output)` takes `net` and `data_output`. The output of this function is `net_update` which holds the gradients of each parameters.

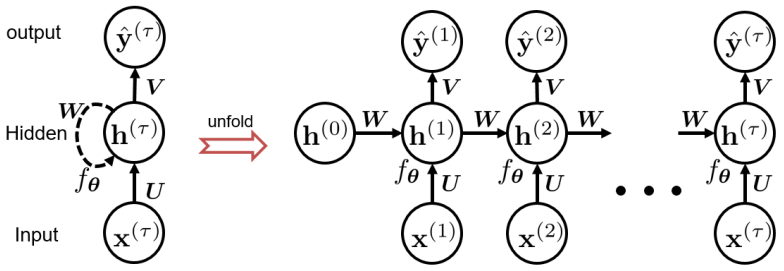
```
function net_update = back_propagation(net, data_output)
%% Your code here %%
J = net.layer{end,1} - data_output;
a = size(net.layer,1);
diffsigmoid = @(x) x.*(1-x);
net_update.layer{a,1} = J.*diffsigmoid(net.layer{a,1});
net_update.weight{a,1} = net_update.layer{a,1}*net.layer{a-1,1}';
for i = a-1:-1:2
net_update.layer{i,1} = diffsigmoid(net.layer{i,1}).*(net.weight{i+1,1}'*net_update.layer{i+1,1});
net_update.weight{i,1} = net_update.layer{i,1}*net.layer{i-1,1}';
end

end
```

(v) `net = weight_update(net, net_update, l_rate)` takes `net`, `net_update` and learning rate `l_rate`. The output of this functions is the updated weight parameters `net`.

```
function net = weight_update(net, net_update, l_rate)
%% Your code here %%
for i = 2:net.layer_num
net.weight{i,1} = net.weight{i,1} - l_rate.learning_rate*net_update.weight{i,1};
net.bias{i,1} = net.bias{i,1} - l_rate.learning_rate*mean(net_update.layer{i,1},2);
end
end
```

5. **Recurrent Neural Networks**:



An RNN is shown in the above figure. Input sequence $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(\tau)}$ is fed to the hidden units $\mathbf{h}^{(\tau)} = f_{\boldsymbol{\theta}}(\mathbf{h}^{(\tau-1)}, \mathbf{x}^{(\tau)}) = \tanh(\boldsymbol{U}\mathbf{x}^{\tau} + \boldsymbol{W}\mathbf{h}^{(\tau-1)} + \boldsymbol{b})$ which hold information of past inputs. The output of the hidden units are fed to the output unit $\hat{\mathbf{y}}^{(\tau)} = softmax(\boldsymbol{V}\mathbf{h}^{(\tau)} + \boldsymbol{c})$. The

RNN is trained to minimize the cross entropy given below for a set of given input/target sequence pair $(\mathbf{x}^{(\tau)}, \mathbf{y}^{(\tau)})$. The cross entropy is defined as $L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{\tau}\sum_{\imath} y_i^{(\tau)} \log \hat{y}_i^{(\tau)}$.

(i) Express $\boldsymbol{h}^{(\tau)}$ in terms of $f_{\boldsymbol{\theta}}, \boldsymbol{h}^{(0)}$ and $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(\tau)}$. (hint: unfolding $f_{\boldsymbol{\theta}}$ from $\mathbf{x}^{(\tau)}$ to $\mathbf{x}^{(1)}$)

$$
\begin{aligned}
\boldsymbol{h}^{(\tau)} &= f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(\tau-1)}, \mathbf{x}^{(\tau)}) \\
&= f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(\tau-2)}, \mathbf{x}^{(\tau-1)}), \mathbf{x}^{(\tau)}) \\
&= f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(\tau-3)}, \mathbf{x}^{(\tau-2)}), \mathbf{x}^{\tau-1}), \mathbf{x}^{(\tau)}) \\
&= f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(\tau-4)}, \mathbf{x}^{(\tau-3)}), \mathbf{x}^{(\tau-2)}), \mathbf{x}^{\tau-1}), \mathbf{x}^{(\tau)}) \\
&= f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(\tau-5)}, \mathbf{x}^{(\tau-4)}), \mathbf{x}^{(\tau-3)}), \mathbf{x}^{(\tau-2)}), \mathbf{x}^{\tau-1}), \mathbf{x}^{(\tau)}) \\
&= \cdots (\text{unfolding until it reaches } \mathbf{x}^{(1)}) \cdots \\
&= f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(f_{\boldsymbol{\theta}}(\cdots f_{\boldsymbol{\theta}}(\boldsymbol{h}^{(0)}, \mathbf{x}^{(1)}), \cdots, \mathbf{x}^{(\tau-4)}), \mathbf{x}^{(\tau-3)}), \mathbf{x}^{(\tau-2)}), \mathbf{x}^{\tau-1}), \mathbf{x}^{(\tau)})
\end{aligned}
$$

(ii) Please provide expression for $\frac{\partial L}{\partial \boldsymbol{V}}$ and $\frac{\partial L}{\partial \boldsymbol{W}}$ in terms of values $\mathbf{o}^{(\tau)} = \boldsymbol{V}\mathbf{h}^{(\tau)} + \boldsymbol{c}$, $\hat{\mathbf{y}}$ and $\mathbf{h}^{(\tau)}$.

a. The total loss L

$$= \sum_t L^{(t)}(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = -\sum_t \sum_i y_i^{(t)} \log \hat{y}_i^{(t)}$$

$$
\begin{aligned}
\text{where } \hat{\mathbf{y}}^{(t)} &= \text{softmax}(\mathbf{o}^{(t)}) = \frac{1}{\sum_k e^{o_k^{(t)}}} \left[\cdots, e^{o_i^{(t)}}, \cdots\right] \\
\mathbf{o}^{(t)} &= \mathbf{V}\mathbf{h}^{(t)} + c \\
\mathbf{h}^{(t)} &= \tanh(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + b)
\end{aligned}
$$

b. $\nabla_{\mathbf{o}^{(\tau)}} L$

$$= \sum_t \sum_i \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial \hat{y}_i^{(t)}} \frac{\partial \hat{y}_i^{(t)}}{\partial o_i^{(t)}} = y_i^{(t)}(\hat{y}_i^{(t)} - 1)$$

$$
\begin{aligned}
\text{where } \frac{\partial L}{\partial L^{(t)}} &= 1 \\
\frac{\partial L^{(t)}}{\partial \hat{y}_i^{(t)}} &= -\frac{y_i^{(t)}}{\hat{y}_i^{(t)}} \leftarrow L^{(t)}(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = -\sum_i y_i^{(t)} \log \hat{y}_i^{(t)} \\
\frac{\partial \hat{y}_i^{(t)}}{\partial o_j^{(t)}} &= \begin{cases} \hat{y}_i^{(t)}(1 - \hat{y}_i^{(t)}) & \text{if } i = j \\ -\hat{y}_i^{(t)} \hat{y}_j^{(t)} & \text{if } i \neq j \end{cases}
\end{aligned}
$$

c. $\nabla_{\mathbf{h}^{(\tau)}} L$ at $t = \tau$

$$= \nabla_{\mathbf{h}^{(\tau)}} \mathbf{o}^{(\tau)} \nabla_{\mathbf{o}^{(\tau)}} L = \mathbf{V}^T \nabla_{\mathbf{o}^{(\tau)}} L$$

d. $\nabla_{\mathbf{h}^{(t)}} L$ at $t < \tau$

$$= \left(\frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}}\right)^T (\nabla_{\mathbf{h}^{(t+1)}} L) + \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}}\right)^T (\nabla_{\mathbf{o}^{(t)}} L)$$
$$= \mathbf{W}^T (\nabla_{\mathbf{h}^{(t+1)}} L) \text{diag}\left(\mathbb{I} - (\mathbf{h}^{(t+1)})(\mathbf{h}^{(t+1)})^T\right) + \mathbf{V}^T (\nabla_{\mathbf{o}^{(t)}} L)$$

where $\dfrac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} = \begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0n} \\ w_{10} & w_{11} & \cdots & w_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n0} & w_{n1} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} 1 - h_0^{(t+1)^2} & 0 & \cdots & 0 \\ 0 & 1 - h_1^{(t+1)^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 - h_n^{(t+1)^2} \end{bmatrix}$

with an assumption of $\mathbf{h} \in \mathbb{R}^n$ and $\mathbf{W} \in \mathbb{R}^{n \times n}$

e. $\nabla_{\mathbf{V}} L = \frac{\partial L}{\partial \boldsymbol{o}} \frac{\partial \mathbf{o}}{\partial \boldsymbol{V}}$

$$= \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_{\mathbf{V}} o_i^{(t)}$$
$$= \sum_t (\nabla_{\mathbf{o}^{(t)}} L) \mathbf{h}^{(t)^T}$$

f. $\nabla_{\mathbf{W}} L = \frac{\partial L}{\partial \boldsymbol{h}} \frac{\partial \mathbf{h}}{\partial \boldsymbol{W}}$

$$= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\mathbf{W}}^{(t)} \mathbf{h}_i^{(t)}$$
$$= \sum_t \text{diag}\left(\mathbb{I} - (\mathbf{h}^{(t)})(\mathbf{h}^{(t)})^T\right) (\nabla_{\mathbf{h}^{(t)}} L) \mathbf{h}^{(t-1)^T}$$

g. $\nabla_{\mathbf{U}} L = \frac{\partial L}{\partial \boldsymbol{h}} \frac{\partial \mathbf{h}}{\partial \boldsymbol{U}}$

$$= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{\mathbf{U}}^{(t)} h_i^{(t)}$$
$$= \sum_t \text{diag}\left(\mathbb{I} - (\mathbf{h}^{(t)})(\mathbf{h}^{(t)})^T\right) (\nabla_{\mathbf{h}^{(t)}} L) \mathbf{x}^{(t)^T}$$

(iii) Express $f_{\boldsymbol{\theta}}^{(\tau)}(\cdot, \cdot)$ in terms of $\mathbf{U}, \mathbf{W}, \mathbf{h}^{(t)}, \mathbf{x}^{(t)}, \mathbf{b}$ for $t = 1, \ldots, \tau$ (more in detail than (i)).

$$
\begin{aligned}
f_{\boldsymbol{\theta}}^{(\tau)}(\mathbf{x}^{(\tau)}, \boldsymbol{h}^{(\tau-1)}) &= \tanh(\boldsymbol{U}\mathbf{x}^{(\tau)} + \boldsymbol{W}\mathbf{h}^{(\tau-1)} + \boldsymbol{b}) \\
&= \tanh(\boldsymbol{U}\mathbf{x}^{(\tau)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-1)} + \boldsymbol{W}\mathbf{h}^{(\tau-2)} + \boldsymbol{b})) + \boldsymbol{b}) \\
&= \tanh(\boldsymbol{U}\mathbf{x}^{(\tau)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-1)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-2)} + \boldsymbol{W}\mathbf{h}^{(\tau-3)} + \boldsymbol{b})) + \boldsymbol{b})) + \boldsymbol{b}) \\
&= \tanh(\boldsymbol{U}\mathbf{x}^{(\tau)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-1)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-2)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-3)} + \boldsymbol{W}\mathbf{h}^{(\tau-4)} + \boldsymbol{b})) \\
&\quad + \boldsymbol{b})) + \boldsymbol{b})) + \boldsymbol{b}) \\
&= \cdots (\text{unfolding until it reaches } \mathbf{x}^{(1)}) \cdots \\
&= \tanh(\boldsymbol{U}\mathbf{x}^{(\tau)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-1)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-2)} + \boldsymbol{W}(\tanh(\boldsymbol{U}\mathbf{x}^{(\tau-3)} + \\
&\quad \boldsymbol{W}(\cdots \tanh(\boldsymbol{U}\mathbf{x}^{(1)} + \boldsymbol{W}\mathbf{h}^{(0)} + \boldsymbol{b}) \cdots) + \boldsymbol{b})) + \boldsymbol{b})) + \boldsymbol{b})) + \boldsymbol{b})
\end{aligned}
$$

Submit Instructions for Programming Assignment

Please submit in .zip file to KLMS named "ee488_assignment3_student#.zip" , for example, "ee488_assignment3_20181234.zip".
This file should contain two folders and one document file for plotted result and explanation of the result.
In matlab code, the comment explaining your code **must be** included, or you will not get a full grade even if your code works fine. Please also include all the files that are required to run the code in the zip file. Do not change the name of the folder and comments should be written in English. Additionally submitting unexecutable code will receive no points. Using other libraries such as 'scipy' are not allowed.