CS329 Machine Learning

Midterm Exam - 2023 Fall

Fan Site
fanst2021@mail.sustech.edu.cn

This is an unofficial answer, probably with mistakes. Please avoid copying.
Feel free to contact me to point out any mistakes, your corrections are highly appreciated.

Question 1 Least Square

a) Consider $Y = AX + V$ and $V \sim \mathcal{N}(\mathbf{v}|\mathbf{0}, Q)$, what is the least square solution of X?

b) If there is a constraint $b^{\mathrm{T}}X = c$, what is the optimal solution of X?

c) If there is an *additional* constraint of $X^{\mathrm{T}}X = d$, in addition to the constraint in b), what is the optimal solution of X?

d) If both A and X are unknown, how to solve A and X alternatively by using two constraints of $X^{\mathrm{T}}X = d$ and $\mathrm{Trace}(A^{\mathrm{T}}A) = e$?

Solution (a)

The least square error function is

$$E(X) = \frac{1}{2}(Y - AX)^{\mathrm{T}}Q^{-1}(Y - AX)$$

Let the gradient w.r.t. X be 0

$$\frac{\partial E(X)}{\partial X} = -2A^{\mathrm{T}}Q^{-1}(Y - AX) = 0$$

$$A^{\mathrm{T}}Q^{-1}AX = A^{\mathrm{T}}Q^{-1}Y$$

So we have the least square solution of X:

$$\widehat{X} = \left(A^{\mathrm{T}}Q^{-1}A\right)^{-1}A^{\mathrm{T}}Q^{-1}Y$$

Solution (b)

The Lagrange function is

$$L(X, \lambda) = (Y - AX)^{\mathrm{T}}Q^{-1}(Y - AX) + \lambda(b^{\mathrm{T}}X - c)$$

Let the gradient w.r.t. X and λ be 0

$$\frac{\partial L}{\partial X} = -2A^{\mathrm{T}}Q^{-1}(Y - AX) + \lambda b = 0$$

$$\frac{\partial L}{\partial \lambda} = b^{\mathrm{T}}X - c = 0$$

Solve the two equations above and we obtain $\widehat{X}$.

Solution (c)

The Lagrange function is

$$L(X, \lambda_1, \lambda_2) = (Y - AX)^{\mathrm{T}}Q^{-1}(Y - AX) + \lambda_1(b^{\mathrm{T}}X - c) + \lambda_2(X^{\mathrm{T}}X - d)$$

Let the gradient w.r.t. X, λ_1 and λ_2 be 0

$$\frac{\partial L}{\partial X} = -2A^{\mathrm{T}}Q^{-1}(Y - AX) + \lambda_1 b + 2\lambda_2 X = 0$$

$$\frac{\partial L}{\partial \lambda_1} = b^{\mathrm{T}}X - c = 0$$

$$\frac{\partial L}{\partial \lambda_2} = X^{\mathrm{T}}X - d = 0$$

Solve the three equations above and we obtain $\widehat{X}$.

Solution (d)

Since A and X is both unknown, to solve A and X, we should follow the steps below:

1. Fix A, solve for X.

Given a fixed A, optimize X w.r.t. the constraints on X:

$$L(X, \lambda) = (Y - AX)^{\mathrm{T}} Q^{-1} (Y - AX) + \lambda (X^{\mathrm{T}} X - d)$$

$$\frac{\partial L}{\partial X} = -2A^{\mathrm{T}} Q^{-1} (Y - AX) + 2\lambda X = 0$$

$$\frac{\partial L}{\partial \lambda} = X^{\mathrm{T}} X - d = 0$$

Solve the two equations above and we obtain X.

2. Fix X, solve for A.

Given a fixed X, optimize A w.r.t. the constraints on A:

$$L(A, \gamma) = (Y - AX)^{\mathrm{T}} Q^{-1} (Y - AX) + \gamma (\mathrm{Trace}(A^{\mathrm{T}} A) - e)$$

$$\frac{\partial L}{\partial A} = -2Q^{-1} (Y - AX) X^{\mathrm{T}} + 2\gamma A$$

$$\frac{\partial L}{\partial \gamma} = \mathrm{Trace}(A^{\mathrm{T}} A) - e$$

Solve the two equations above and we obtain A.

3. Repeat Step 1-2 to optimize A and X alternately until convergence.

Question 2 Linear Gaussian System

Consider $Y = AX + V$, where X and V are Gaussian, $X \sim \mathcal{N}(\mathbf{x}|\mathbf{m}_0, \mathbf{\Sigma}_0)$, $V \sim \mathcal{N}(\mathbf{v}|\mathbf{0}, \beta^{-1}\mathbf{I})$.

Calculate the followings:

- conditional distribution $p(Y|X)$
- joint distribution $p(Y, X)$
- marginal distribution $p(Y)$
- posterior distribution $p(X|Y = \mathbf{y}, \beta, \mathbf{m}_0, \mathbf{\Sigma}_0)$
- posterior predictive distribution $p\left(\hat{Y}|Y = \mathbf{y}, \beta, \mathbf{m}_0, \mathbf{\Sigma}_0\right)$
- prior predictive distribution $p(Y|\beta, \mathbf{m}_0, \mathbf{\Sigma}_0)$

Solution

1. **Conditional distribution**

$$p(Y|X) = \mathcal{N}(Y|AX, \beta^{-1} I)$$

2. **Joint distribution**

Let $Z = (X, Y)$, and $p(Y, X) = \mathcal{N}(\mu_Z, \Sigma_Z) = \mathcal{N}(Y|AX, \beta^{-1} I)\mathcal{N}(X|\mathbf{m}_0, \mathbf{\Sigma}_0)$, where

$$\mu_Z = (\mathbf{m}_0, A\mathbf{m}_0) \quad \Sigma_Z = \begin{pmatrix} \Sigma_{\mathrm{XX}} & \Sigma_{\mathrm{XY}} \\ \Sigma_{\mathrm{YX}} & \Sigma_{\mathrm{YY}} \end{pmatrix} = \begin{pmatrix} \mathbf{\Sigma}_0 & \mathbf{\Sigma}_0 A^{\mathrm{T}} \\ A\mathbf{\Sigma}_0 & A\mathbf{\Sigma}_0 A^{\mathrm{T}} + \beta^{-1} I \end{pmatrix}$$

3. **Marginal distribution**

$$p(Y) = \int \mathcal{N}(AX, \beta^{-1} I)\mathcal{N}(\mathbf{m}_0, \mathbf{\Sigma}_0)\ \mathrm{d}\ X = \mathcal{N}(Y|A\mathbf{m}_0, A^{\mathrm{T}}\mathbf{\Sigma}_0 A + \beta^{-1} I)$$

4. **Posterior distribution**

Assume $x = Hy + u$,

$$P(x|y) = \mathcal{N}(x|Hy, L),\ p(u) = \mathcal{N}(u|0, L)$$

$$p(x|y) \propto p(y|x)p(x) = \mathcal{N}(y|Ax, Q)\mathcal{N}(x|\mu, \Sigma)$$

$$-\frac{1}{2}(x - Hy)^{\mathrm{T}} L^{-1}(x - Hy) \propto -\frac{1}{2}(y - Ax)^{\mathrm{T}}\beta(y - Ax) - \frac{1}{2}(x - \mathbf{m}_0)^{\mathrm{T}}\mathbf{\Sigma}_0^{-1}(x - \mathbf{m}_0)$$

Align the quadratic and first order terms

$$L^{-1} = A^{\mathrm{T}}\beta A + \mathbf{\Sigma}_0^{-1}$$

$$L^{-1}Hy = A^{\mathrm{T}}\beta y + \mathbf{\Sigma}_0^{-1}\mathbf{m}_0$$

Therefore $p(X|Y = \mathbf{y}, \beta, \mathbf{m}_0, \mathbf{\Sigma}_0) = \mathcal{N}(\mu_{\mathrm{MAP}}, \Sigma_{\mathrm{MAP}})$, where

$$\Sigma_{\mathrm{MAP}} = \left(A^{\mathrm{T}}\beta A + \mathbf{\Sigma}_0^{-1}\right)^{-1}$$

$$\mu_{\mathrm{MAP}} = \Sigma_{\mathrm{MAP}}(A^{\mathrm{T}}\beta y + \mathbf{\Sigma}_0^{-1}\mathbf{m}_0)$$

5. **Posterior predictive distribution**

$$p\left(\hat{Y}|Y = \mathbf{y}, \beta, \mathbf{m}_0, \mathbf{\Sigma}_0\right) = \int p(Y|X)p(X|Y = \mathbf{y}, \beta, \mathbf{m}_0, \mathbf{\Sigma}_0)\ \mathrm{d}X = \mathcal{N}(\mu, \Sigma)$$

where

$$\mu = A\mu_{\mathrm{MAP}}$$

$$\Sigma = A^{\mathrm{T}}\Sigma_{\mathrm{MAP}}A + \beta^{-1} I$$

6. **Prior predictive distribution**

$$p(Y|\beta, \mathbf{m}_0, \mathbf{\Sigma}_0) = \int p(Y|X, \beta_0)p(X, \mathbf{m}_0, \mathbf{\Sigma}_0)\mathrm{d}X = \mathcal{N}(Y|A\mathbf{m}_0, A\mathbf{\Sigma}_0 A^{\mathrm{T}} + \beta^{-1}\mathbf{I})$$

Question 3 Linear Regression

Consider $y = \mathbf{w}\boldsymbol{\phi}(x) + v$, where v is Gaussian, *i.e.*, $v \sim \mathcal{N}(v|0, \beta^{-1})$, and $\mathbf{w}$ has a Gaussian *priori*, *i.e.*, $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \alpha^{-1}\mathbf{I})$.

Assume that $\boldsymbol{\phi}(x)$ is known, please derive

- posterior distribution $p(\mathbf{w}|D, \beta, \mathbf{m}_0, \alpha)$
- posterior predictive distribution $p(\hat{y}|\hat{x}, D, \beta, \mathbf{m}_0, \alpha)$
- prior predictive distribution $p(D|\beta, \mathbf{m}_0, \alpha)$

where $D = \{\phi_n, y_n\}, n = 1, ..., N$ is the training dataset and $\phi_n = \phi(x_n)$

Solution

1. Posterior distribution

$$p(\mathbf{w}|D,\beta,\mathbf{m}_0,\alpha) \propto p(D|w,\beta)p(\mathbf{w}|\alpha,\mathbf{m}_0) = \mathcal{N}(\mathbf{m}_0,\alpha^{-1}\mathbf{I})\prod_{n=1}^{N}\mathcal{N}(w^{\mathrm{T}}\phi_n,\beta^{-1}) = \mathcal{N}(\mu,\Sigma)$$

where

$$\Sigma^{-1} = \beta\sum_{n=1}^{N}\phi_n^{\mathrm{T}}\phi_n + \alpha\ \mathbf{I}$$

$$\mu = \Sigma\left(\sum_{n=1}^{N}\beta\phi_n y_n + \alpha\mathbf{I}\ \mathbf{m}_0\right)$$

2. **Posterior predictive distribution**

$$\begin{aligned} p(\hat{y}|\hat{x},D,\beta,\mathbf{m}_0,\alpha) &= \int p(\hat{y}|\mathbf{w},\beta)p(\mathbf{w}|D,\beta,\mathbf{m}_0)\mathrm{d}\mathbf{w} \\ &= \mathcal{N}(\hat{y}|\mu^{\mathrm{T}}\phi,\phi^{\mathrm{T}}\Sigma\phi+\beta^{-1}I) \end{aligned}$$

μ and Σ are obtained in the posterior distribution above.

3. **Prior predictive distribution**

$$\begin{aligned} p(D|\beta,\mathbf{m}_0,\alpha) &= \int p(D|\mathbf{w},\beta)p(\mathbf{w}|\mathbf{m}_0,\alpha)\mathrm{d}\mathbf{w} \\ &= \left(\frac{1}{\sqrt{2\pi\beta}}\right)^N \frac{1}{\sqrt{2\pi\alpha}}\exp\left(-\frac{\beta}{2}\sum_{n=1}^{N}y_n^2 - \frac{\alpha}{2}\mathbf{m}_0^{\mathrm{T}}\mathbf{m}_0 + \frac{1}{2}\mu^{\mathrm{T}}\Sigma\mu\right) \end{aligned}$$

where

$$\Sigma = \beta\sum_{n=1}^{N}(\phi\phi^{\mathrm{T}}) + \alpha$$

$$\mu = \Sigma\left(\beta\sum_{n=1}^{N}y_n\phi_n^{\mathrm{T}} + \alpha\mathbf{m}_0\right)$$

Question 4 Logistic Regression

Consider a two-class classification problem with the logistic sigmoid function, $y = \sigma(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}))$, for a given dataset $D = \{\phi_n, t_n\}$, where $t_n \in \{0,1\}$, $\phi_n = \phi(\mathbf{x}_n), n = 1,...,N$.

The likelihood function is given by

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

where $\mathbf{w}$ has a Gaussian *priori, i.e.*, $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{m}_0,\alpha^{-1}\mathbf{I})$.

Please derive the followings:

- posterior distribution $p(\mathbf{w}|D,\mathbf{m}_0,\alpha)$
- posterior predictive distribution $p(t|x,D,\mathbf{m}_0,\alpha)$
- prior predictive distribution $p(D|\mathbf{m}_0,\alpha)$

Hint: use Delta approximation and Laplace approximation properly.

Solution

1. **Posterior distribution**

$$p(\mathbf{w}|D,\mathbf{m}_0,\alpha) \propto p(D|\mathbf{w})p(\mathbf{w}|\mathbf{m}_0,\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0,\alpha^{-1}\mathbf{I}) \prod_{n=1}^{N} \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)^{t_n}(1-\sigma(\mathbf{w}^{\mathrm{T}}\phi_n))^{1-t_n}$$

2. **Posterior predictive distribution**

$$p(t|x,D,\mathbf{m}_0,\alpha) = \int p(t|\mathbf{w},\mathbf{x})p(\mathbf{w}|D,\mathbf{m}_0,\alpha)\mathrm{d}\mathbf{w}$$

By Laplace Approximation,

$$b = \nabla_{\mathbf{w}} - \ln p(\mathbf{w}|D,\mathbf{m}_0,\alpha) = \sum_{n=1}^{N}(t_n - \sigma(\mathbf{w}^{\mathrm{T}}\phi_n))\phi_n \alpha(\mathbf{w} - \mathbf{m}_0)$$

$$H = \nabla^2_{\mathbf{w}} - \ln p(\mathbf{w}|D,\mathbf{m}_0,\alpha) = \sum_{n=1}^{N}\sigma(\mathbf{w}^{\mathrm{T}}\phi_n)(1-\sigma(\mathbf{w}^{\mathrm{T}}\phi_n))\phi_n^{\mathrm{T}}\phi_n + \alpha$$

Iterating updating rule to obtain $\mathbf{w}^*$

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{old}} - H^{-1}b$$

Therefore we have

$$p(t|x,D,\mathbf{m}_0,\alpha) = (y^*)^t(1-y^*)^{1-t}\mathcal{N}(\mathbf{w}^*,H^{-1})$$

where $y^* = \sigma(\mathbf{w}^{*\mathrm{T}}\phi(\mathbf{x}))$.

3. **Prior predictive distribution**

$$\begin{aligned} p(D|\mathbf{m}_0,\alpha) &= \int p(D|\mathbf{w})p(\mathbf{w}|\mathbf{m}_0,\alpha)\mathrm{d}\mathbf{w} \\ &= \int \mathcal{N}(\mathbf{w}|\mathbf{m}_0,\alpha^{-1}\mathbf{I}) \prod_{n=1}^{N} \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)^{t_n}(1-\sigma(\mathbf{w}^{\mathrm{T}}\phi_n))^{1-t_n}\mathrm{d}\mathbf{w} \end{aligned}$$

By approximation,

$$\ln p(D|\mathbf{m}_0,\alpha) = \ln p(D|\mathbf{w}^*) - \frac{1}{2}M\ln N = \sum_{n=1}^{N} t_n \ln y_n^* + (1-t_n)\ln(1-y_n^*) - \frac{1}{2}N\ln M$$

Question 5 Neural Network

Consider a two-layer neural network described by the following equations:

$$\begin{aligned} a_1 = w^{(1)}x, \quad a_1 &= w^{(2)}z \\ z = h(a_1), \quad y &= \sigma(a_2) \end{aligned}$$

where $\mathbf{x}$ and y are the input and output of the neural network, $h(\cdot)$ is a nonlinear function, and $\sigma(\cdot)$ is the sigmoid function.

1. Please derive the following gradients: $\dfrac{\partial y}{\partial w^{(1)}}$, $\dfrac{\partial y}{\partial w^{(2)}}$, $\dfrac{\partial y}{\partial a_1}$, $\dfrac{\partial y}{\partial a_2}$ and $\dfrac{\partial y}{\partial x}$.
2. Please detive the updating rules for $w^{(1)}$ and $w^{(2)}$ given the classification errors between y and t, where t is the ground truth of the output y.

Solution (1)

$$\frac{\partial y}{\partial w^{(1)}} = \frac{\partial y}{\partial a_2}\frac{\partial a_2}{\partial z}\frac{\partial z}{\partial a_1}\frac{\partial a_1}{\partial w^{(1)}} = y(1-y)w^{(2)}h'(a_1)x$$

$$\frac{\partial y}{\partial w^{(2)}} = \frac{\partial y}{\partial a_2}\frac{\partial a_2}{\partial w^{(2)}} = y(1-y)z$$

$$\frac{\partial y}{\partial a_1} = \frac{\partial y}{\partial a_2}\frac{\partial a_2}{\partial z}\frac{\partial z}{\partial a_1} = y(1-y)w^{(2)}h'(a_1)$$

$$\frac{\partial y}{\partial a_2} = \frac{\partial \sigma(a_2)}{\partial a_2} = y(1-y)$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial a_2}\frac{\partial a_2}{\partial z}\frac{\partial z}{\partial a_1}\frac{\partial a_1}{\partial x} = y(1-y)w^{(2)}h'(a_1)w^{(1)}$$

Solution (2)

The classification error:

$$E = -t\ln y - (1-t)\ln(1-y)$$

$$\frac{\partial E}{\partial y} = -\frac{t}{y} + \frac{1-t}{1-y} = \frac{y-t}{y(1-y)}$$

$$\frac{\partial E}{\partial w^{(1)}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w^{(1)}} = (y-t)w^{(2)}h'(a_1)x$$

$$\frac{\partial E}{\partial w^{(2)}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial w^{(2)}} = (y-t)z$$

The updating rule:

$$w^{(1)} \leftarrow w^{(1)} - \eta\frac{\partial E}{\partial w^{(1)}} = w^{(1)} - \eta(y-t)w^{(2)}h'(a_1)x$$

$$w^{(2)} \leftarrow w^{(2)} - \eta\frac{\partial E}{\partial w^{(2)}} = w^{(2)} - \eta(y-t)z$$

Question 6 Bayesian Neural Network

a) Consider a neural network for regression, $t = y(\mathbf{w}, \mathbf{x}) + v$, where v is Gaussian, *i.e.*, $v \sim \mathcal{N}(v|0, \beta^{-1})$, and $\mathbf{w}$ has a Gaussian *priori*, *i.e.*, $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \alpha^{-1}\mathbf{I})$.

Assume that $y(\mathbf{w}, \mathbf{x})$ is the neural network output please derive the followings:

- posterior distribution $p(\mathbf{w}|D, \beta, \mathbf{m}_0, \alpha)$,
- posterior predictive distribution $p(t|x, D, \beta, \mathbf{m}_0, \alpha)$
- prior predictive distribution $p(D|\beta, \mathbf{m}_0, \alpha)$

where $D = \{x_n, t_n\}, n = 1, ..., N$ is the training dataset.

b) Consider a neural network for two-class classification, $y = \sigma(a(\mathbf{w},\mathbf{x}))$ and a dataset $D = \{x_n, t_n\}$, where $t_n \in \{0, 1\}$, $\mathbf{w}$ has a Gaussian *priori*, *i.e.*, $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$, and $a(\mathbf{w},\mathbf{x})$ is the neural network model.

Please derive the followings:

- posterior distribution $p(\mathbf{w}|D,\alpha)$
- posterior predictive distribution $p(t|\mathbf{x},D,\alpha)$
- prior predictive distribution $p(D|\alpha)$

Solution (a)

1. **Posterior distribution**

$$p(\mathbf{w}|D,\beta,\mathbf{m}_0,\alpha) \propto p(D|\mathbf{w},\beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0,\alpha^{-1}I)\prod_{n=1}^{N}\mathcal{N}(t_n|y(\mathbf{w},\mathbf{x}_n),\beta^{-1})$$

By Laplace approximation,

$$b = \nabla_{\mathbf{w}} -\ln p(\mathbf{w}|D,\beta,\mathbf{m}_0,\alpha) = \sum_{n=1}^{N} -\beta(t_n-y_n)\frac{\partial y}{\partial\ \mathbf{w}} + \alpha(\mathbf{w}-\mathbf{m}_0)$$

$$H = \nabla^2_{\mathbf{w}} -\ln p(\mathbf{w}|D,\beta,\mathbf{m}_0,\alpha) = \sum_{n=1}^{N} -\beta(t_n-y_n)\frac{\partial^2 y}{\partial \mathbf{w}^2} + \beta\left[\frac{\partial y}{\partial\ \mathbf{w}}\right]^{\mathrm{T}}\left[\frac{\partial y}{\partial\ \mathbf{w}}\right] + \alpha$$

Iterating the updating rule to obtain $\mathbf{w}^*$

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{old}} - H^{-1}b$$

Therefore the posterior distribution is approximated as

$$p(\mathbf{w}|D,\beta,\mathbf{m}_0,\alpha) = \mathcal{N}(\mathbf{w}^*,H^{-1})$$

2. **Posterior predictive distribution**

Substitute the $\mathbf{w}^*$ obtained in Q1 above,

$$p(t|\mathbf{x},D,\beta,\mathbf{m}_0,\alpha) = \int p(t|\mathbf{w},\beta)p(\mathbf{w}|D,\beta,\mathbf{m}_0,\alpha)\mathrm{d}\mathbf{w} = \mathcal{N}(t|y(\mathbf{w}^*,\mathbf{x}),\beta^{-1})\mathcal{N}(\mathbf{w}^*,H^{-1})$$

3. **Prior predictive distribution**

Substitute the $\mathbf{w}^*$ obtained in Q1 above,

$$p(D|\beta,\mathbf{m}_0,\alpha) = \int p(D|\mathbf{w},\beta)p(\mathbf{w}|\mathbf{m}_0,\alpha)\mathrm{d}\mathbf{w} = \int \mathcal{N}(\mathbf{w}|\mathbf{m}_0,\alpha^{-1}I)\prod_{n=1}^{N}\mathcal{N}(t_n|y(\mathbf{w},\mathbf{x}_n),\beta^{-1})\mathrm{d}\mathbf{w}$$

By approximation,

$$\begin{aligned}\ln p(D|\beta,\mathbf{m}_0,\alpha) &= \ln p(D|\mathbf{w}^*,\beta) - \frac{1}{2}M\ln N \\ &= \sum_{n=1}^{N} -\frac{1}{2}(t_n - y(\mathbf{w}^*,\mathbf{x}_n))^{\mathrm{T}}\beta(t_n-y(\mathbf{w}^*,\mathbf{x}_n)) - \frac{1}{2}M\ln N\end{aligned}$$

Solution (b)

1. **Posterior distribution**

$$p(\mathbf{w}|D,\alpha) \propto p(D|\mathbf{w})p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|0,\alpha^{-1}I)\prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

By Laplace Approximation,

$$b = \nabla_{\mathbf{w}} - \ln p(\mathbf{w}|D, \alpha) = -\sum_{n=1}^{N}(t_n - y_n) - \mathbf{w}^{\mathrm{T}}\alpha\ \mathbf{w}$$

$$H = \nabla^2_{\mathbf{w}} - \ln p(\mathbf{w}|D, \alpha) = \sum_{n=1}^{N} y_{n(1-y_n)} - 2\alpha\ \mathbf{w}$$

Iterating the updating rule to obtain $\mathbf{w}^*$:

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{old}} - H^{-1}b$$

Therefore we have

$$p(\mathbf{w}|D, \alpha) = \mathcal{N}(\mathbf{w}^*, H^{-1})$$

2. **Posterior predictive distribution**

Substitute the $\mathbf{w}^*$ obtained in Q1 above,

$$p(t|\mathbf{x}, D, \alpha) = \int p(t|\mathbf{x}, \mathbf{w})p(\mathbf{w}|D, \alpha)\mathrm{d}\mathbf{w} = (y^*)^t(1-y^*)^{1-t}\mathcal{N}(\mathbf{w}^*, H^{-1})$$

3. **Prior predictive distribution**

Substitute the $\mathbf{w}^*$ obtained in Q1 above,

$$p(D|\alpha) = \int p(D|\mathbf{w})p(\mathbf{w}|\alpha)\mathrm{d}\mathbf{w} = \int \mathcal{N}(\mathbf{w}|\ 0, \alpha^{-1}I)\prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}\mathrm{d}\mathbf{w}$$

By approximation,

$$\begin{aligned}\ln p(D|\alpha) &= \ln p(D|\mathbf{w}^*) - \frac{1}{2}M\ln N \\ &= \sum_{n=1}^{N}(t_n \ln y_n^* + (1-t_n)\ln(1-y_n^*)) - \frac{1}{2}M\ln N\end{aligned}$$

Question 7 Critical Analyses

1. Please explain why the dual problem formulation is used to solve the SVM machine learning problem

Primal problem:

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

$$\text{Subject to } y_i(w_i x + b) \geq 1 - \xi_i$$

Dual problem:

$$\text{Maximize} \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{i=1}^{N}\sum_{m=1}^{N} a_n a_n t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\text{Subject to } 0 \leq a_n \leq C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

1. Solving the dual problem can be more computationally efficient, especially when the number of features is large.

2. The dual formulation naturally accommodates the use of the kernel trick, allowing SVM to handle non-linearly separable data by implicitly mapping it into a higher-dimensional space, using a series of KKT condition.

2. Please explain, in terms of cost functions, constraints and predictions:
   1. what are the differences between SVM classification and logistic regression
   2. what are the differences between ν-SVM regression and least square regression.

SVM Classification vs. Logistic Regression

	SVM Classification	**Logistic Regression**
Cost functions	$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$	$-\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$
Constraints	$t_n(\mathbf{w}^{\mathrm{T}}\phi(x_n)+b) \geq 1$	/
Predictions	$\begin{cases}\mathbf{w}^{\mathrm{T}}\phi(x_n)+b \geq 1 & \text{, positive}\\ \mathbf{w}^{\mathrm{T}}\phi(x_n)+b \leq -1 & \text{, negative}\end{cases}$	Probability of positive: $p = \sigma(y(\mathbf{w},\mathbf{x}))$

ν-SVM Regression vs. Least Square Regression

	ν-SVM Regression	**Least Square Regression**
Cost functions	$\tilde{L}(a) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n \mathbf{x}_m)$	$\frac{1}{2}(\mathbf{y} - (\mathbf{w}^{\mathrm{T}}\mathbf{x}+b))^2$
Constraints	$0 \leq a_n \leq \frac{1}{N}$, $\sum_{n=1}^{N} a_n t_n = 0$, $\sum_{n=1}^{N} a_n \geq \nu$	/
Predictions	$\hat{y} = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$	$\hat{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$

3. Please explain why neural network (NN) based machine learning algorithms use logistic activation functions?

1. The output can be interpreted as the probability of the input belonging to the positive class, and a threshold can be applied to make a binary decision. Making it capable for 2-class classification problems.
2. The logistic functions has a smooth derivative, which is important for gradient-based optimization algorithms like BP.
3. The logistic functions have non-zero gradients in the center of domains, preventing gradient-vanishing issues.

4. The derivation is trivial $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
5. The logistic function squashes its input into a range between 0 and 1, which is useful for binary classification problems and can manage the outliers.

4. Please explain:
    1. what are the differences between the logistic activation function and other activation functions (e.g., relu, tanh)
    2. when these activation functions should be used.

1. **Logistic, ReLU and tanh**

For the output range,

- Logistic activation function squashes input $[-\infty, \infty]$ to $[0, 1]$, and the output range is $(0, 1)$
- ReLU outputs the input directly if the input is positive otherwise 0, the output range is $[0, \infty)$
- tanh is similar to logistic function but the output range is $(-1, 1)$.

Logistic function and tanh can both meet vanishing gradient problem, while ReLU suffers from dying ReLU (inactive neurons) problem.

2. **Use cases**

- The logistic function is commonly used in the output layer for binary classification problems.
- The ReLU is used in hidden layers of deep neural networks due to its simplicity and effectiveness.
- The tanh is used for problems where data may have negative values (or where output is zero-centered), and it's used in contexts similar to sigmoid.

5. Please explain why Jacobian and Hessian matrices are useful for machine learning algorithms.

1. **Jacobian**

The Jacobian matrix represents the partial derivatives of a vector-valued function. In the context of machine learning, this function typically represents the mapping from input features to output predictions.

In optimization problems, such as training a machine learning model, the Jacobian matrix is used to find the direction and magnitude of the steepest ascent or descent of the objective function. This information is crucial for iterative optimization algorithms, like gradient descent, as it guides the updates to the model parameters.

In the case of neural networks, the Jacobian is often used to compute gradients during the back-propagation process, which is fundamental for updating the weights and biases of the network.

2. **Hessian**

The Hessian matrix is an extension of the Jacobian, representing the second-order partial derivatives of a scalar-valued function. It provides information about the curvature of the objective function.

In optimization, the Hessian matrix is particularly useful for understanding the local geometry of the objective function. It helps distinguish between different types of critical points, such as minima, maxima, or saddle points.

For machine learning algorithms, the Hessian matrix is employed in second-order optimization methods, like Newton's method. These methods use both the first-order information (gradients) and second-order information (Hessian) to make more informed updates to the model parameters, potentially converging faster than first-order methods alone.

6. Please explain why exponential family distributions are so common in engineering practice.

   Please give some examples which are **NOT** exponential family distributions.

1. **Mathematical Simplicity:**

   Exponential family distributions have simple mathematical forms, which makes them analytically tractable, making mathematical analysis, optimization, and statistical inference easier.

2. **Properties**

   Exponential family distributions have properties that make statistical estimation easier. Like the *memoryless property* of exponential distributions:

   $$p(X > s + t \mid X > t) = p(X > s)$$

   which provides i.i.d. data and is suitable for MLE.

3. **Conjugate Priors:**

   Exponential family distributions often have conjugate prior distributions. This property simplifies MAP and Bayesian analysis.

Cauchy Distribution and **Multimodal Gaussian distribution** are **NOT** members of the exponential family.

7. Please explain why KL divergence is useful for machine learning? Please provide two examples of using KL divergence in machine learning.

Machine learning is to approximate p_{model} to p_{data}.

The Kullback-Leibler divergence can be interpreted as a measure of the dissimilarity of the two distributions p(x) and q(x).

If we use a distribution that is different from the true one, then we must necessarily have a less efficient coding, and on average the additional information that must be transmitted is (at least) equal to the KL divergence between the two distributions.

1. When modeling a distribution $p(\boldsymbol{x})$ using $q(\mathbf{x}|\boldsymbol{\theta})$, minimizing the KL divergence between the two distributions is equivalent to maximizing the Likelihood function w.r.t. $\boldsymbol{\theta}$.

   $$\mathrm{KL}(p\|q) = \sum_{n=1}^{N}\{-\ln q(\mathbf{x}_n|\boldsymbol{\theta}) + \ln p(\mathbf{x}_n)\}$$

2. The KL divergence is used to calculate the mutual information:

   $$\mathrm{I}[\mathbf{x},\mathbf{y}] \equiv \mathrm{KL}(p(\mathbf{x},\mathbf{y})\|p(\mathbf{x})p(\mathbf{y}))$$

8. Please explain why data augmentation techniques are a kind of regularization skills for NNs.

Data augmentation acts as a form of **implicit regularization**. The increased diversity in the training dataset encourages the model to learn more general features rather than memorizing specific instances.

In terms of preventing overfitting, data augmentation introduces variability during training and has the same effect of regularization.

By exposing the neural network to various augmented versions of the training data, the model learns to recognize the patterns in different orientations, scales, and lighting conditions, making it less sensitive to variations in the input data.

9. Please explain why Gaussian distributions are preferred over other distributions for many machine learning models?

1. **Central Limit Theorem**

   The Central Limit Theorem states that the sum of a large number of i.i.d. random variables will be approximately Gaussian. This makes the Gaussian distribution a natural choice for modeling the sum of random variables, which is often the case in ML models.

2. **Simplicity**

   Gaussian distributions are completely characterized by their mean and variance, which makes them easy to work with mathematically. This simplicity facilitates analytical solutions and computations in ML models.

3. **Mathematical Properties**

   Gaussian distributions have many nice properties. For example, the linear transformation of a Gaussian distribution is also Gaussian, the joint of two independent Gaussian distribution is also Gaussian, product of Gaussian distributions is still Gaussian…

   These properties make Gaussian distributions useful in ML models.

4. **Robustness**

   Gaussian distributions are relatively robust in the presence of noise. The normal distribution tends to be less sensitive to outliers compared to some other distributions.

5. **Linear Model Usages**

   Gaussian distributions play a key role in linear models, such as linear regression. In these models, the assumption of normally distributed errors allows for the application of the least squares method, which leads to closed-form solutions and efficient computations.

7. **Bayesian Inference**

   Gaussian distributions have convenient properties in the context of Bayesian inference. In many cases, the conjugate priors for Gaussian likelihoods lead to analytical solutions, making Bayesian updating and posterior inference more tractable.

10. Please explain why Laplace approximation can be used for many cases?

The Laplace approximation aims to find a Gaussian approximation to a probability density defined over a set of continuous variables.

As a result of the central limit theorem, the posterior distribution for a model becomes increasingly approximated by a Gaussian as the number of observed data points is increased, and so the Laplace approximation is useful if the number of data points is relatively large.

By Laplace approximation, the posterior distribution is estimated to a Gaussian distribution, then we can use Gaussian properties or mathematical analyzing methods over the distribution.

11. What are the fundamental principles for model selection (degree of complexity) in machine learning?

1. Trade-off between bias and variance
2. Model complexity (number and scales of parameters)
3. Minimizing the loss function while preventing overfitting

12. How to choose a new data sample (feature) for regression and classification model training, respectively? How to choose it for testing? Please provide some examples.

Training Regression Models

1. **Choose relevant features.** Choose features that greately influence the target variable continuously according to the prior knowledge. For example, when predicting the overall score of CS329, we should take the grades of the exams into consideration since they have a large impact on the target(overall score).
2. **Avoid highly correlated features.** Features selected should have low correlation relatively, to reduce redundancy in training model and avoid the impact of this feature becomes too large. For example, if we select the age of a student as a feature, we should avoid selecting the grade of the student as a feature since the two features are highly correlated.
3. **Balance the training set.** If the dataset retrieved are unbalanced, with a large bias, the regression model will behave badly when predicting. For example, if a traffic sign dataset contains 1000 30km speed-limit signs, while only 100 15km speed-limit signs, then the input of a 15km speed-limit sign will probably be recognized as a 30km one. This can be solved by re-sampling techniques.

Training Classification Models

1. **Choose relevant features.** Same as above. For example, when training models for spam email classification, selecting the sender, subject and some keywords is helpful.
2. **Feature importance techniques.** When using models like tree models, we can assign importance(influence factor) to different features according to the prior knowledge. For example, when classifying the spam emails, we may lower the importance of frequent words like “is”, “dear” etc.
3. **Dimension reduction.** Due to the existence of the curse of dimensionality, we may preprocess the training set to reduce the number of dimensions by removing some useless features to boost the training process. For example, in facial recognition, reduce the number of facial features while retaining key information.

Selecting Testing dataset

This part is shared for both kinds of models.

1. **Random sampling.** The testing data should be sufficiently random to avoid the impact of bias, and the test should be balanced. For example, if a model is trained for both male and female, if the test set only contains one of them, the test bias may be large.
2. **Presentative testcases.** The test set should include a representative set of features that the model is likely to encounter in real-world scenarios. And the feature distribution for test is similar to the training set. For example, the test cases for traffic sign detection should contain mosaic pictures in different weather and light condition.
3. **Better evaluation methods.** Consider using techniques like k-fold cross-validation to get a more robust estimate of the model's performance.
4. **Separated with test cases.** The testing set should be clearly separated with training set to avoid overfitting to the training(testing) set. For example, if the test set and training set are the same, the model will probably behave very well on this testing set, while badly on any other testing set.

13. Please explain why the MAP model is usually more preferred than the ML model?

1. **Prior Knowledge**

   MAP incorporates prior knowledge about the parameters by using a prior distribution. ML assumes a uniform prior, which means it gives equal weight to all possible parameter values.

2. **Small Sample Sizes**

   If the sample size is small, the likelihood function might not provide a precise estimate of the parameters. MAP can help by incorporating a prior distribution, which becomes particularly useful when there is limited data.

3. **Regularization**

   The prior distribution acts as a regularization term, penalizing extreme parameter values. This can help prevent overfitting and put a penalty when paramaters are too large.

4. **Bayesian Inference Framework**

   MAP is a Bayesian approach, which allows for a coherent and consistent framework for updating beliefs as more data becomes available. It provides a way to update prior knowledge based on new evidence, leading to more robust and adaptive modeling.

And we can still use MAP even if we know little about the prior by introducing a non-informative prior.

Question 8 Discussions

1. What are the generative and discriminative approaches to machine learning, respectively?

   Can you explain the advantages and disadvantages of these two approaches and provide a detailed example to illustrate your points?

Definitions

Approaches that explicitly or implicitly model the distribution of inputs and outputs are **generative models**.

$$\text{Model } p(\mathcal{C}_k, \mathbf{x}) = p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$$

$$\text{Use Bayes' theorem } p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

Approaches that model the posterior probabilities directly are called **discriminative models**.

$$\text{Model } p(\mathcal{C}_k|\boldsymbol{x}) \text{ directly}$$

Advantages of Generative Approches

- **Data Generation**: Since generative models capture the underlying distribution of the data, they can generate new samples that resemble the training data. This can be useful for data augmentation.
- **Handling Missing Data**: Generative models can handle missing or incomplete data more effectively as they model the entire distribution.

Disadvantages of Generative Approches

- **Complexity**: Generative models are more complex than discriminative ones, requiring estimation of a full joint probability distribution.
- **Performance**: In some cases, generative models may not perform as well as discriminative models in classification tasks, especially when the distribution of the data is complex.

Example of Generative Approaches

Naive Bayes Classifier: Model the joint probability distribution of features and labels using Bayes' theorem and assume independence among features given the class.

Despite its simplicity, Naive Bayes is effective in many text classification tasks.

Take spam email classification for example, by generative approaches, we calculate the frequency of N words in the vocabulary $p(X_i|\ Y = \text{spam/not spam})$ where X_i represents a word. Then we model the class prior $p(Y = \text{spam/not spam}) = \frac{\#\text{ of spam/not spam emails}}{\#\text{ of emails}}$.

To predict, we calculate the probability of the two models using Bayesian's theorem:

$$P(Y = \text{spam} \mid X_1, ..., X_n) \propto P(Y = \text{spam}) \prod_{n=1}^{N} P(X_i|Y = \text{spam})$$

where X_i are the words in the email.

Advantages of Discriminative Approches

- **Simplicity**: Discriminative models are often simpler and computationally less demanding than generative models, especially in high-dimensional spaces.
- **Better Performance**: Discriminative models perform better in classification tasks, particularly when the decision boundary is complex and the distribution of the data is not essential for the task.

Disadvantages of Discriminative Approches

- **Limited Data**: Discriminative models may perform badly when the amount of training data is limited, as they only focus on the decision boundary and do not model the underlying data distribution.
- **Handling Missing Data**: Discriminative models struggle with missing or incomplete data, as they rely on the information relevant to the decision boundary.

Example of Discriminative Approches

Support Vector Machine Classifier: A discriminative model that aims to find the hyperplane that maximally separates different classes.

It is widely used in binary and multi-class classification tasks and is known for its effectiveness in high-dimensional spaces.

Also, take spam email classification for example. Assume the size of the dictionary is N, then transform each email into an N-feature vector of high dimension.

Train the model (can be very slow since the dimension is too high) to find the optimal parameter set $\mathbf{w}$ hyperplane that separates spam emails and non-spam emails.

To predict if the input email is a spam one, calculate

$$\text{Prediction} = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$$

And we can directly classify if this email is a spam one.

2. How do you analyze the GAN model from the generative and discriminative perspectives?

GANs are composed of two neural networks, each serving a distinct role: a generator and a discriminator.

The **generator** is a neural network that takes random noise as input and transforms it into data samples. It learns to generate samples from a distribution similar to the one observed in the training data.

The **discriminator** is another neural network, often implemented as a binary classifier. It takes in both real and generated samples as input and outputs a probability indicating whether the sample is real or fake.

When training, the generator improves itself to create increasingly robust samples, while the discriminator aims to classify more accurately.