P1.

a) $Y = AX + V$. $V \sim N(V|0, Q)$

~~map $p(A, X|Y) \propto p(Y|A, X)p(X)p(A)$~~

let $Q(X) = (Y - AX)'(Y - AX)$

$X_{LSE} = \min(Q(X))$.

$\frac{\partial Q(X)}{\partial (X)} = -2A'Y + 2A'AX = 0 \Rightarrow A'AX = A'Y. \Rightarrow X_{LSE} = (A'A)^{-1}A'Y.$

b) $Y = AX + V$. $X \sim N(x|m_0, \Sigma_0)$, $V \sim N(V|0, \beta^{-1}I)$

$\begin{bmatrix} x \\ y \end{bmatrix} \sim G\left(\begin{bmatrix} x \\ y \end{bmatrix} \middle| \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} \begin{bmatrix} \Sigma_{xx} \Sigma_{xy} \\ \Sigma_{yx} \Sigma_{yy} \end{bmatrix}\right)$ $\bar{y} = A\bar{x} + V$

$\hat{y} = A\hat{x} + V.$

$\Sigma_{\bar{x}\bar{x}} = \Sigma_{\bar{x}}$ $\Sigma_{\bar{y}\bar{y}} = \Sigma_{\bar{y}} = A\Sigma_{\bar{x}}A^T + \beta^{-1}I.$ $\mu_{\bar{y}} = A\mu_{\bar{x}}$

$\Sigma_{\bar{x}\bar{y}} = cov\{\bar{x}\bar{y}^T\} = cov\{\bar{x}\hat{y}^T\} = \Sigma_{\bar{x}}A^T.$ $A^T = \Sigma_{\bar{x}\bar{x}}^{-1}\Sigma_{\bar{x}\bar{y}}$ $A = \Sigma_{\bar{y}\bar{x}}\Sigma_{\bar{x}\bar{x}}^{-1}$

$Q = \beta^{-1}I = \Sigma_{\bar{y}\bar{y}} - \Sigma_{\bar{y}\bar{x}}\Sigma_{\bar{x}\bar{x}}^{-1}\Sigma_{\bar{x}\bar{y}}$

$\begin{cases} \Sigma_{\bar{y}|\bar{x}} = cov\{A\bar{x} + V|\bar{x}\} = cov\{V\} = \beta^{-1}I \\ \mu_{\bar{y}|\bar{x}} = E\{A\bar{x} + N|\bar{x}\} = A\bar{x} = \mu_{\bar{y}} + A(x - \mu_{\bar{x}}) = \mu_{\bar{y}} + \Sigma_{\bar{y}\bar{x}}\Sigma_{\bar{x}\bar{x}}^{-1}(x - \mu_{\bar{x}}). \end{cases}$

~~$P(y)$~~ $= P(Y = y_j) = \sum_{i=1}^{\infty} P\{X = x_i, Y = y_j\} = \sum_{i=1}^{\infty} P_{ij}\Delta P_{ij}.$

$\bar{x} = \underbrace{(A^TQ^{-1}A)^{-1}A^TQ^{-1}}_{\bar{A}}\bar{y} - \underbrace{(A^TQ^{-1}A)^{-1}A^TQ^{-1}}_{\bar{A}}N = \bar{A}\bar{y} - \bar{A}N.$

$\begin{cases} \Sigma_{\bar{x}|\bar{y}} = \bar{A}Q\bar{A}^T = (A^TQ^{-1}A)^{-1} \\ \mu_{\bar{x}|\bar{y}} = \bar{A}y. \end{cases}$

$\Sigma_{\bar{x}|\bar{y}}^{-1} = \Sigma_{\bar{x}|\bar{y}}^{-1} + \Sigma_{\bar{x}}^{-1} = A^TQ^{-1}A + \Sigma_{\bar{x}}^{-1}$

$\Sigma_{\bar{x}}^{-1}|\bar{y}\mu_{\bar{x}|\bar{y}} = \Sigma_{\bar{x}|\bar{y}}^{-1}\mu_{\bar{x}|\bar{y}} + \Sigma_{\bar{x}}^{-1}\mu_{\bar{x}}$

$= A^TQ^{-1}y + \Sigma_{\bar{x}}^{-1}\mu_{\bar{x}}$

$\mu_{\bar{x}|y} = \Sigma_{\bar{x}|\bar{y}}(A^TQ^{-1}y + \Sigma_{\bar{x}}^{-1}\mu_{\bar{x}})$

$p(\hat{y}) = \sum_h p(y = 1|\hat{x}, h)p(h|D) = \mu_{\hat{y}} = A\mu_{\hat{x}}$ $x \sim N(x|m_0, \Sigma_0).$

2. $p(t|t, \alpha, \beta) = \int p(t|x, \beta)\, p(x|t, \alpha, \beta)\, dx.$

$p(t|A, t, \alpha, \beta) = N(t|m_N^T\phi(A), \sigma_N^2(A))$ $\sigma_N^2(x) = \frac{1}{\beta} + \phi(x)^T S_N \phi(x)$

P2.

a). $p(w) = \mathcal{N}(w|m_0, S_0)$

$p(t|w) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T\phi(x_n), \beta^{-1})$.

$p(w|t) = p(t|w)p(w)$. We only focus on α in the resulting exponential $\exp(\frac{\alpha}{2})$

$\sum_{n=1}^{N} (t_n - w^T\phi(x_n))^T\beta(t_n - w^T\phi(x_n)) + (w-m_0)^T S_0^{-1}(w-m_0)$. (1)

$f(w) = (w-m_N)^T S_N^{-1}(w-m_N) = w^T S_N^{-1} w - 2m_N^T S_N^{-1} w + \text{const}$. (2)

$\sum_{n=1}^{N} (w^T\phi(x_n))^T\beta w^T\phi(x_n)) + w^T S_0^{-1} w = w^T[\sum_{n=1}^{N} \phi(x_n)\beta\phi(x_n)^T + S_0^{-1}]w$.

match $\Rightarrow$ $S_N^{-1} = \sum_{n=1}^{N} \phi(x_n)\beta\phi(x_n)^T + S_0^{-1} = \beta\Phi^T\Phi + S_0^{-1}$.

since $(\Phi^T\Phi)_{ij} = \sum_k \Phi_{ki}\Phi_{kj} = \sum_k \phi_i(x_k)\phi_j(x_k)$.

The linear term in w ~~the right hand side of (1)~~

$2\sum_{n=1}^{N} (w^T\phi(x_n))^T\beta t_n + 2m_0^T S_0^{-1} w = 2[\beta\sum_{n=1}^{N} t_n\phi(x_n)^T + m_0^T S_0^{-1}]w$

$= 2[\beta t^T\Phi + m_0^T S_0^{-1}]w$.

$\Rightarrow m_N^T S_N^{-1} = \beta t^T\Phi + m_0^T S_0^{-1}$.

Right multiplying with S_N, taking the transpose and using the symmetry of S_N

b2. Assume apply Laplace approximation to the problem of Bayesian logistic regression the Laplace approximation is obtained by finding the mode of the posterior distribution and then fitting a Gaussian centered at that mode. This requires evaluation of the second evaluation of the second derivatives of the log posterior, which is equivalent to finding the Hessian matrix.

$p(w) = \mathcal{N}(w|m_0, S_0)$. m_0 and S_0 are fixed hyperparameters.

$p(w|t) \propto p(w)p(t|w)$. $t = (t_1, \dots, t_N)^T$. Taking the log of both sides.

$\ln p(w|t) = -\frac{1}{2}(w-m_0)^T S_0^{-1}(w-m_0) + \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + C$.

$y(n) = \sigma(w^T\phi_n)$. To obtain a Gaussian approximation to the posterior distribution, we first maximize the posterior distribution to give the MAP solution w_{MAP} which defines the mean of the Gaussian. The covariance is then given by the inverse of the matrix of second derivatives of negative log likelihood, which takes the form

$S_N = -\nabla\nabla \ln p(w|t) = S_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T$.

$q(w) = \mathcal{N}(w|w_{MAP}, S_N)$.

a) $y_k = \sigma(a_k) = \frac{1}{1+\exp(-a_k)}$ $\quad a_j = \sum_{i=0}^{D} w_{ji} x_i$.

$$\frac{\partial y}{\partial w^{(1)}} = \frac{\partial y}{\partial a_2} \cdot \frac{\partial a_2}{\partial z} \cdot \frac{\partial z}{\partial a_1} \frac{\partial a_1}{\partial w^{(1)}}$$

$$= \frac{\partial}{\partial a_2}\left(\frac{1}{1+e^{-a_2}}\right) w^{(2)} \cdot \frac{\partial h(a_1)}{\partial a_1} D = 0.$$

$$\frac{\partial y}{\partial w^{(2)}} = \frac{\partial y}{\partial a_2} \frac{\partial a_2}{\partial w^{(2)}} = 0.$$

$$\frac{\partial y}{\partial a_1} = \frac{\partial y}{\partial a_2} \cdot \frac{\partial a_2}{\partial z} \cdot \frac{\partial z}{\partial a_1} = \frac{a_2}{e^{a_2}+e^{-a_2}+2} \cdot \cancel{w^{(2)}} \cdot \frac{\partial h(a_1)}{\partial a_1}$$

$$\frac{\partial y}{\partial a_2} = \frac{\partial y}{\partial a_2}\left(\frac{1}{1+e^{-a_2}}\right) = \frac{a_2 e^{-a_2}}{(1+e^{-a_2})^2} = \frac{a_2}{e^{a_2}+e^{-a_2}+2}$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial a_2} \cdot \frac{\partial a_2}{x} = 0.$$

b). ~~$E(w) = \frac{1}{2}\sum_{n=1}^{N} \|y(x_n, w) - t_n\|^2$.~~ $\quad (p(t|x,w) = \mathcal{N}(t|y(x,w), \beta^{-1})$.

$$p(t|X,w,\beta) = \prod_{n=1}^{N} p(t_n|x_n,w,\beta). \Rightarrow E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w)-t_n\}^2.)$$

$$E(w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} \qquad y_n \to y(x_n,w)$$

$$p(t|x,w) = \prod_{k=1}^{K} y_k(x,w)^{t_k}[1-y_k(x,w)]^{1-t_k}$$

$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\{t_{nk}\ln y_{nk} + (1-t_{nk})\ln(1-y_{nk})\} \Rightarrow E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn}\ln y_k(x_n,w)$$

$$\Rightarrow E(w) = \sum_{n=1}^{N} E_n(w) \qquad y_k = \sum_i w_{ki} x_i.$$

$$E_n = \frac{1}{2}\sum_k (y_{nk}-t_{nk})^2. \qquad \frac{\partial E_n}{\partial w_{ji}} = (y_{nj}-t_{nj})x_{ni} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}.$$

$\delta_j = \frac{\partial E_n}{\partial a_j}$ $\quad$ δ's are often referred to as errors for reasons we shall see shortly.

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \quad \gg \quad \frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i. \qquad \delta_k = y_k - t_k.$$

for updating $w^{(1)}$: $E(w_{\tau+1}^{(1)}) = w^{(1)} - \eta\nabla E(w^{(1)}) = w_\tau^{(1)} + \eta\sum_{n=1}\{\frac{t_n}{y_n}\frac{\partial y}{\partial w^{(1)}} + \frac{1-t_n}{1-y_n}\frac{\partial y_n}{\partial w^{(1)}}\}$.

$$= w_\tau^{(2)} + \eta\sum_{n=1}\{\frac{t_n}{y_n}\frac{\partial y}{\partial w^{(1)}} + \frac{1-t_n}{1-y_n}\frac{\partial y_n}{\partial w^{(1)}}\}$$

$$= w_\tau^{(2)} + \eta\sum_{n=1}\{\frac{t_n}{y_n} y_n(1-y_n) w^{(2)} h'(a_1) x + \frac{1-t_n}{1-y_n} y_n(1-y_n) w^{(2)} h'(a_1) x\}$$

$$= w_\tau^{(2)} + \eta\sum_{n=1}\{t_n - 2t_n y_n w^{(2)} h'(a_1) x + y_n w^{(2)} h'(a_1) x\}$$

for updating $w^{(2)}$: $E(w_{\tau+1}^{(2)}) = w^{(2)} - \eta\nabla E(w^{(2)}) = w_\tau^{(2)} + \eta\sum_{n=1}\{\frac{t_n}{y_n}\frac{\partial y}{\partial w^{(2)}} + \frac{1-t_n}{1-y_n}\frac{\partial y_n}{\partial w^{(2)}}\}$

$$= w_\tau^{(2)} + \eta\sum_{n=1}\{\frac{t_n}{y_n} y_n(1-y_n) z + \frac{1-t_n}{1-y_n} y_n(1-y_n) z\}$$

$$= w_\tau^{(2)} + \eta\sum_{n=1}\{t_n - t_n y_n z + (1-t_n) y_n z\}$$

$$= w_\tau^{(2)} + \eta\sum_{n=1}\{t_n - 2t_n y_n z + y_n z\}$$

P4.

a) $t = y(w, x) + v$. $v \sim \mathcal{N}(v|0, \beta^{-1})$. $w \sim \mathcal{N}(w|0, \alpha^{-1}I)$.

=> $p(t|x)$ is Gaussian. with x-dependent mean given by the output of a neural network model $y(x, w)$. and with precision β.

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

we choose a prior distribution over the weights w that is Gaussian of the form.

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

with a corresponding set of target value $D = \{t_1, \cdots, t_N\}$, the likelihood function is given by

$$p(D|w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, w), \beta^{-1}).$$

and so the resulting posterior distribution is then

$p(w|D, \alpha, \beta) \propto p(w|\alpha)p(D|w, \beta)$. → as a consequence of the nonlinear dependence of $y(x, w)$ on w, will be non-Gaussian.

We find a Gaussian approximation to the posterior distribution by using the Laplace approximation. To do this, we must first find a (local) maximum of the posterior, and this must be done using iterative numerical optimization. As usual, it is convenient to maximize the logarithm of the posterior, which can be written in the form

$$\ln p(w|D) = -\frac{\alpha}{2}w^T w - \frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + C.$$

Assuming for the moment that α and β are fixed, we can find a maximum, w_{MAP}.

$$A = -\nabla\nabla \ln p(w|D, \alpha, \beta) = \alpha I + \beta H.$$

where H is the Hessian matrix comprising the second derivatives of sum-of-squares error function with respect to the components of w.

$$q(w|D) = \mathcal{N}(w|w_{MAP}, A^{-1}).$$

$$p(t|x, D) = \int p(t|x, w)\, q(w|D)\, dw.$$

now we assume the ~~pos~~ posterior distribution has small variance compared with the characteristic scales of w over which $y(x, w)$ is varying. This allow us to make a Taylor series expansion of the network function around w_{MAP}

$$y(x, w) \simeq y(x, w_{MAP}) + g^T(w - w_{MAP}) \qquad g = \nabla_w y(x, w)|_{w=w_{MAP}}$$

$$\Rightarrow p(t|x, w, \beta) \simeq \mathcal{N}(t|y(x, w_{MAP}) + g^T(w - w_{MAP}), \beta^{-1}).$$

$$p(t|x, D, \alpha, \beta) = \mathcal{N}(t|y(x, w_{MAP}), \sigma^2(x)) \qquad \sigma^2(x) = \beta^{-1} + g^T A^{-1} g.$$

P4

b)

log likelihood:

$$\ln p(\mathcal{D}|w) = \sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

$t_n \in \{0,1\}$ $y_n = y(x_n, w)$.

applying the Laplace framework to initialize the hyperparameter α and then to determine the parameter vector w by maximizing the log posterior distribution

$$E(w) = -\ln p(\mathcal{D}|w) + \frac{\alpha}{2} w^T w.$$

$$\ln p(\mathcal{D}|\alpha) \simeq -E(w_{MAP}) - \frac{1}{2}\ln|A| + \frac{W}{2}\ln\alpha + C.$$

$$E(w_{MAP}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \frac{\alpha}{2} w_{MAP}^T w_{MAP}.$$

in which $y_n = y(x_n, w_{MAP})$ $\Rightarrow$ $p(t|x, \mathcal{D}) \simeq p(t|x, w_{MAP})$.

Make a linear approximation for the output unit activation in the form

$$a(x, w) \simeq a_{MAP}(x) + b^T(w - w_{MAP})$$

where $a_{MAP}(x) = a(x, w_{MAP})$, and the vector $b \equiv \nabla a(x, w_{MAP})$ can be found by backpropagation. We have now a Gaussian approximation for the posterior distribution over w, and a model for a that is a linear function of w. ~~We can now appeal to the results of section~~

$$p(a|x, \mathcal{D}) = \int \delta(a - a_{MAP}(x) - b^T(x)(w - w_{MAP}))\, q(w|\mathcal{D})\, dw$$

$q(w|\mathcal{D})$ is the gaussian approximation to the ~~pre~~ posterior distribution given by before

$$\sigma_a^2(x) = b^T(x) A^{-1} b(x)$$

$$p(t=1|x, \mathcal{D}) = \int \sigma(a)\, p(a|x, \mathcal{D})\, da$$

$$p(t=1|x, \mathcal{D}) = \sigma(\kappa(\sigma_a^2) b^T w_{MAP}) \quad \text{where } \kappa(\cdot) \text{ is define as } \kappa(\sigma^2) = (1 + \pi\sigma^2/8)^{-\frac{1}{2}}$$

P5

a). Solving the primal problem, we obtain the optimal w, but know nothing about the α_i. In order to classify a query point x we need to explicitly compute the scalar product $w^T x$, which may be expensive if d is large.

Solving the dual problem, we obtain the α_i (where $\alpha_i = 0$ for all about but a few points - the support vectors). In order to classify a query point x, we calculate

$$w^T x + w_0 = \left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)^T x + w_0 = \sum_{i=1}^{n} \alpha_i y_i (x_i, x) + w_0$$

This term is very efficiently calculated if there are only few support vectors. Further, since we now have a scalar product only involving data vectors, we may apply the kernel trick.

b) i). ~~Lean~~Linear SVMs and logistic regression generally perform comparably in practice. Use SVM with a nonlinear kernel if you have reason to believe your data won't be linearly separable (or you need to be more robust to outliers than LR will normally tolerate). Otherwise, just try logistic regression first and see how you do with that simpler model. If logistic regression fails you, try an SVM with a non-linear kernel like a RBF. Besides, The logistic regression comes from generalized linear regression. ~~A good discussion of~~ While the Support Vector Machines algorithm is much more geometrically motivated. Instead of assuming a probabilistic model, we're trying to find a particular optimal separating hyperplane, where we define "optimality" in the context of the support vectors. We don't have anything resembling the statistical model we use in logistic regression here, even though the linear case will give us similar results: really this just means that logistic regression does a pretty good job of producing "wide" margin" ~~cut~~ classifiers, since that's all SVM is trying to do (specifically, SVM is trying to "maximize" the margin between the classes.

ii). cost-functions:

V-SVM: $V\sum_{n=1}^{N} \varepsilon_n + \frac{1}{2}\|w\|^2$

Least Square Regression: $\frac{1}{2}\sum_{n=1}^{N}[t_n - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(x_n)]^2$

constraints:

V-SVM: $0 \le a_n \le \frac{1}{N}$ $\sum_{n=1}^{N} a_n t_n = 0$. $\sum_{n=1}^{N} a_n \ge V$.

least square regression: $w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$ $\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$ $\bar{\phi}_j = \frac{1}{N}\sum_{n=1}^{N} \phi_j(x_n)$

prediction:

V-SVM: $p(t=1|x) = \sigma(Ay(x)+B)$

least square regression: $p(t|x,w,\beta) = N(t|y(x,w),\beta^{-1})$ $E[t|x] = \int t p(t|x)\,dt = y(x,w)$

P5. c) Fast learning algorithm.

By using $p(w|t) = N(w|m_N, S_N)$, we see the posterior distribution as

$p(w|t, X, \alpha) = N(w|0, \Sigma)$. $m = \Sigma \phi^T t$ and $\Sigma = (A + \beta \phi^T \phi)^{-1}$.

we maximum the marginal likelihood function and logarithm the function.

So that we can get $\ln p(t|X, \alpha) = \ln N(t|0, C) = -\frac{1}{2}[N \ln(2\pi) + \ln|C| + t^T C^{-1} t]$

$t = (t_1, \ldots, t_N)^T$, $C = \phi A^{-1} \phi^T$.

There is an isotropic noise having precision β. The marginal likelihood is given by $p(t|\alpha, \beta) = N(t|0, C)$ which $C(C = \frac{1}{\beta} I + \frac{1}{\alpha} \phi \phi^T)$ is the covariance matrix.

Where $t = (t_1, t_2)^T$ and $\phi = (\phi(x_1), \phi(x_2))^T$.

Then we pull out the contribution from α_i in the matrix C.

$$C = \beta^{-1} I + \sum_{j \neq i} \alpha_j^{-1} \phi_i \phi_j^T + \alpha_i^{-1} \phi_i \phi_i^T = C_i + \alpha_i^{-1} \phi_i \phi_i^T$$

$$|C| = |C_{-i}| |1 + \alpha_i^{-1} \phi_i^T C_{-i}^{-1} \phi_i|$$

$$C^{-1} = C_i^{-1} - \frac{C_{-i}^{-1} \phi_i \phi_i^T C_{-i}^{-1}}{\alpha_i + \phi_i^T C_{-i}^{-1} \phi_i}$$

where ϕ_i is the i^{th} column of ϕ and ϕ_n is the n^{th} row of ϕ.

$$L(\alpha) = L(\alpha_{-i}) + \lambda(\alpha_i) \qquad \lambda(\alpha_i) = \frac{1}{2}\left[\ln \alpha_i - \ln(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}\right]$$

$$s_i = \phi_i^T C_{-i}^{-1} \phi_i \qquad q_i = \phi_i^T C_{-i}^{-1} t.$$

s_i is the sparsity and q_i is the quality of ϕ_i. The larger the s_i is, the more likely to be removed from the model.

Analysis of sparsity.

The stationary points of marginal likelihood with respect to α_i occur when the derivative is equal to zero.

$$\frac{d\lambda(\alpha_i)}{d\alpha_i} = \frac{\alpha_i^{-1} s_i^2 - (q_i^2 - s_i)}{2(\alpha_i + s_i)^2} = 0 \rightarrow \alpha_i = \frac{s_i^2}{q_i^2 - s_i}$$

If $q_i^2 \leq s_i$ and $\alpha_i = \infty$, then the basis function ϕ_i is already excluded from the model and no action is required. Evaluate the quantities.

$$Q_i = \phi_i^T C^{-1} t. \qquad S_i = \phi_i^T C^{-1} \phi_i$$

$$q_i = \frac{\alpha_i Q_i}{\alpha_i - S_i} \qquad s_i = \frac{\alpha_i S_i}{\alpha_i - S_i}$$

when $\alpha_i = \infty$, we have $q_i = Q_i$ and $s_i = S_i$, we can write

$$Q_i = \beta \phi_i^T t - \beta^2 \phi_i^T \phi \Sigma \phi^T t \qquad S_i = \beta \phi_i^T - \beta^2 \phi_i^T \phi \Sigma \phi^T \phi_i.$$

Problem 6

a)

The basic function of activation function in neural network is to introduce nonlinearity. There are many choices for the specific nonlinear form. The advantage of sigmoid is that the output range is limited, so the data is not easy to diverge in the process of transmission. Of course, there is also a corresponding disadvantage, that is, the gradient is too small at saturation. Another advantage of sigmoid is that the output range is (0, 1), so it can be used as an output layer, and the output represents probability. Besides, another advantage of sigmoid is easy derivation.

b)

i)

Sigmoid and tanh are "saturated activation functions", while relu and its variants are "unsaturated activation functions". The advantages of using "unsaturated activation function" lie in two points:

(1) "unsaturated activation function" can solve the so-called "gradient disappearance" problem.

(2) It can accelerate the convergence speed.

Sigmoid compresses the real value output in the range of [0,1], and the tanh function compresses the real value output in the range of [- 1,1].

Sigmoid function was used very often in history, and the output value range is real number between [0,1]. But now it's not very popular and it's rarely used in practice. The reasons are as follows:

(1) The saturation of sigmoid function makes the gradients disappear.

(2) The sigmoid function output is not "zero centered.".

(3) The calculation of exponential function consumes more computing resources.

tanh function

tanh function is very similar to sigmoid in shape. In fact, tanh is a deformation of sigmoid,

Unlike sigmoid, tanh is "zero centered.". Therefore, in practical application, tanh is better than sigmoid. However, in the case of saturated neurons, tanh did not solve the problem of gradient disappearance.

Advantages: (1) tanh solves the problem that the output of sigmoid is not "Zero Center"

Disadvantages: (1) there is still the problem of supersaturation of sigmoid function. (2) It's still exponential.

ReLU

In recent years, relu functions have become more and more popular. Its full name is rectified linear unit, and its Chinese name is modified linear unit.

Advantages: (1) relu solves the problem of gradient disappearance, at least x is in the positive range, neurons will not be saturated;

(2) Due to the linear and unsaturated form of relu, it can converge quickly in SGD;

(3) The calculation speed is much faster. Relu function has only linear relationship, and does not need exponential calculation. It is faster than sigmoid and tanh in both forward and backward propagation.

Disadvantages: (1) the output of relu is not "Zero Center";

(2) With the training going on, the neuron may die and the weight cannot be updated. This neuronal death is irreversible.

ii)

Sigmoid functions and their combination usually work better in classifiers. Due to the problem of gradient collapse, sigmoid and tanh activation functions need to be avoided in some cases. Relu function is a common activation function, which is most used at present. If we encounter some dead neurons, we can use the leaky relu function, which is always used only in hidden layers. According to experience, we can generally start with the relu

activation function, but if relu can't solve the problem well, we can try other activation functions

c)

The idea of hinge loss is to make the distance between those who fail to classify correctly and correctly classify sufficiently. If the difference reaches a threshold value of Δ Delta Δ, the error of incorrect classification can be considered as 0, otherwise, the calculation error will be accumulated.Hinge loss function is mainly used in support vector machine (SVM), Advantages: stable classification surface, convex function. The classification interval can be maximized.

The 0-1 loss function is mainly used for perceptron

The quadratic loss function is mainly used in least squares (OLS)

Logarithmic loss function (logrithmic loss function, cross entropy loss function, softmax loss function) is mainly used in logistic regression and softmax classification

Exponential loss function is mainly used in AdaBoost ensemble learning algorithm

d)

Jacobian

Sometimes we need to find all of the partial derivatives of a function whose input and output are both vectors. The matrix containing all such partial derivatives is the Jacobian.

Given:

$$\vec{f} : \mathbb{R}^m \to \mathbb{R}^n$$

The Jacobian matrix **J** is given by:

$$J \in \mathbb{R}^{n \times m}$$

$$J_{i,j} = \frac{\partial}{\partial x_j} f(\vec{x})_i$$

Example of Jacobian Matrix

Let's say:

$$\vec{f} : \mathbb{R}^2 \to \mathbb{R}^2$$

The function f, takes in a vector of size 2 and outputs a vector of size 2. The operation can be explained by 2 functions in a vector:

$$\begin{bmatrix} x^2y \\ 5x + sin(y) \end{bmatrix}$$

Where:

$$f_1(x, y) = x^2y$$

$$f_2(x, y) = 5x + sin(y)$$

And the Jacobian Matrix of **f** is:

$$J_f(x, y) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}$$

Hessian

Hessian is a square matrix of second order partial derivatives of a scalar-valued function or scalar field.

It describes the local curvature of a function of many variables.

The loss functions of neural nets are very high dimensional and computing and storing the full Hessian matrix takes n squared memory, which makes it intractable.

The directional second derivative tell us how well we can expect a gradient descent step to perform. We can make a second order Taylor Series approximation to the function f(**x**) around the current point x^0:

… where g is the gradient and H is the Hessian at x^0.

If we use a learning rate of epsilon, then the new point **x**, will be given by:

$$\vec{x^0} - \epsilon\vec{g}$$

… substituting this into our equation, we get:

You can think of each part of the equation like this: there's the original value of the function, the expected improvement due to the slope of the function and then the curvature correction from the second derivative.

Let's call

When alpha is 0 or negative, the Taylor series predicts that increasing epsilon forever will result in the function decreasing forever. This is not true, in practice Taylor expansion is not accurate for large epsilon.

When alpha is positive, solving for the optimal step size yields:

$$\epsilon^* = \frac{\vec{g}^T\vec{g}}{\vec{g}^T\vec{H}\vec{g}}$$

Taylor Series

Taylor Series is a series expansion of a function about a point. The one-dimensional Taylor Series is an expansion of a real function f(x) about a point x= a:

In order for this to be true, the function must meet certain requirements, but given that it meets them, Taylor Series will always work.

Taylor Series also works for complex variables and multivariable expansion.

If you remember your Calculus, there's the whole concept of optimization, where you can start to graph a polynomial from just the equation. You know that if the first derivative at a point is positive that it must be sloping upwards and if the second derivative is positive then it must be accelerating or curving up.

You also know that when a point has a first derivative of 0, that it's a critical point and it must be either a minima or maxima. Which brings you to the point that there can be many minima and maxima. Which are referred to as local/global minima/maxima.

And finally you also know that at a critical point you can determine if its a minima or maxima by finding the second derivate which tells you that the if its negative then it must be a maximum.

So similarly in multivariable state, you can use the Jacobian and the Gradient as the first derivative and the Hessian as the second derivative. And roughly apply the same principles to graph in 3D space.

Remember also that you can introduce artificial constraints to your graphs as the problem suggests a real physical phenomena. After all we live in a constrained physical universe. Here we treat these constraint points as critical points. This greatly helps us in finding the global minima and maxima as it reduces our search space.

e)

Generally speaking, mathematicians like to generalise concepts and results up to the maximal point that they can, *to the limits of their usefulness*. That is, when mathematicians develop a concept, and find that one or more useful theorems apply to that concept, they will generally seek to generalise the concept and results more and more, until they get to the point where further generalisation would render the results inapplicable or no longer useful. As can be seen from your list, the exponential family has

a number of useful theorems attached to it, and it encompasses a wide class of distributions. This is sufficient to make it a worthy object of study, and a useful mathematical class in practice.

By exponential family, I mean the distributions which are given as

$f(x|\theta)=h(x)\exp\{\eta(\theta)T(x)-B(\theta)\}f(x|\theta)=h(x)\exp\{\eta(\theta)T(x)-B(\theta)\}$

whose support doesn't depend on the parameter $\theta\theta$. Here are some advantages:

(a) It incorporates a wide variety of distributions.

(b) It offers a natural sufficient statistics $T(x)T(x)$ according to the Neyman-Fisher theorem.

(c) It makes possible to provide a nice formula for the moment generating function of $T(x)T(x)$.

(d) It makes it easy to decouple the relationship between the response and predictor from the conditional distribution of the response (via link functions).

In particular, the exponential family distributions always have conjugate priors, and the resulting posterior predictive distribution has a simple form. This makes is an extremely useful class of distributions in Bayesian statistics. Indeed, it allows you to undertake Bayesian analysis using conjugate priors at an extremely high level of generality, encompassing all the distributional families in the exponential family.

examples:

Bernoulli distribution and uniform distribution.

f)

The common feature of SVM and RVM is that they have sparse solutions, so the prediction of new data only depends on the kernel function calculated on a subset of training data. This subset is support vector for SVM and correlation vector for RVM.

The important property of SVM is that the determination of its model parameters corresponds to a convex optimization problem, so many local solutions are global optimal solutions. However, SVM does not provide posterior probability, and the important property of RVM is that the Bayesian method is introduced into RVM to provide the output of posterior probability, and can often produce more sparse solutions (the prediction speed is faster on the test set). SVM often needs to use cross validation method to determine the model complexity parameter C. for RVM, another advantage of introducing Bayesian method is to omit the step of model selection. But RVM often needs more training time because of the operation of inverse matrix.

g)

The *Kullback-Leibler divergence* (hereafter written as KL divergence) is a measure of how a probability distribution differs from another probability distribution. Classically, in Bayesian theory, there is some *true distribution* P(X); we'd like to estimate with an *approximate distribution* Q(X). In this context, the KL divergence measures the distance from the approximate distribution Q to the true distribution P.

Mathematically, consider two probability distributions P,Q on some space XX. The Kullback-Leibler divergence from Q to P (written as DKL(P∥Q))

DKL(P∥Q)=Ex∼P[logP(X)Q(X)]DKL(P‖Q)=Ex∼P[logP(X)Q(X)]

Properties of KL Divergence

There are some immediate notes that are worth pointing out about this definition.

The KL Divergence is **not symmetric**: that is DKL(P∥Q)≠DKL(Q∥P)DKL(P‖Q)≠DKL(Q‖P). As a result, it is also **not a distance metric**.

The KL Divergence can take on values in [0,∞][0,∞]. Particularly, if PP and QQ are the exact same distribution (Pa.e.=QP=a.e.Q), then DKL(P∥Q)=0, and by symmetry DKL(Q∥P)=0. In fact, with a little bit of math, a stronger statement can be proven: if DKL(P∥Q)=0, then Pa.e.=QP=a.e.Q.

In order for the KL divergence to be finite, the support of PP needs to be contained in the support of QQ. If a point xx exists with Q(x)=0 but P(x)>0, then DKL(P∥Q)=∞

Independently of the interpretation, the KL divergence is always defined as a *specific* function of the cross-entropy (which you should be familiar with before attempting to understand the KL divergence) between two distributions (in this case, probability mass functions)

$$DKL(P \| Q) = -\sum_{x \in X} p(x) \log q(x) + \sum_{x \in X} p(x) \log p(x) = H(P,Q) - H(P) DKL(P \| Q) = -\sum_{x \in X} p(x) \log q(x) + \sum_{x \in X} p(x) \log p(x) = H(P,Q) - H(P)$$

where $H(P,Q)$is the cross-entropy of the distribution P and Q and $H(P) = H(P,P) H(P) = H(P,P)$. The KL is not a metric, given that it does not obey the triangle inequality. In other words, in general, $DKL(P \| Q) \neq DKL(Q \| P)$.

Given that a neural network is trained to output the mean (which can be a scalar or a vector) and the variance (which can be a scalar, a vector or a matrix), why don't we use a metric like the MSE to compare means and variances? When you use the KL divergence, you don't want to compare just numbers (or matrices), but probability distributions (more precisely, probability densities or mass functions), so you will not compare just the mean and the variance of two different distributions, but you will actually compare the distributions

examples

Cross entropy and relative entropy

h)

Regularization (traditionally in the context of shrinkage) adds prior knowledge to a model; a prior, literally, is specified for the parameters. Augmentation is also a form of adding prior knowledge to a model; e.g. images are rotated, which **you** know does not change the class label. Increasing training data (as with augmentation) decreases a model's variance. Regularization also decreases a model's variance. They do so in different ways, but ultimately both decrease regularization error.

Section 5.2.2 of Goodfellow et al's Deep Learning proposes a much broader definition:

> Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

There is a tendency to asssociate regularization with shrinkage because of the term "l-p norm regularization"...perhaps "augmentation regularization" is equally valid, although it doesn't roll off the tongue.Regularization (traditionally in the context of shrinkage) adds prior knowledge to a model; a prior, literally, is specified for the parameters. Augmentation is also a form of adding prior knowledge to a model; e.g. images are rotated, which **you** know does not change the class label. Increasing training data (as with augmentation) decreases a model's variance. Regularization also decreases a model's variance. They do so in different ways, but ultimately both decrease regularization error.

Section 5.2.2 of Goodfellow et al's Deep Learning proposes a much broader definition:

> Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

There is a tendency to asssociate regularization with shrinkage because of the term "l-p norm regularization"...perhaps "augmentation regularization" is equally valid, although it doesn't roll off the tongue.

Problem 7

Supervised learning method can be divided into generation method and discriminant method. The models learned are called generative model and discriminant model respectively.

Generative model and discriminant model

definition

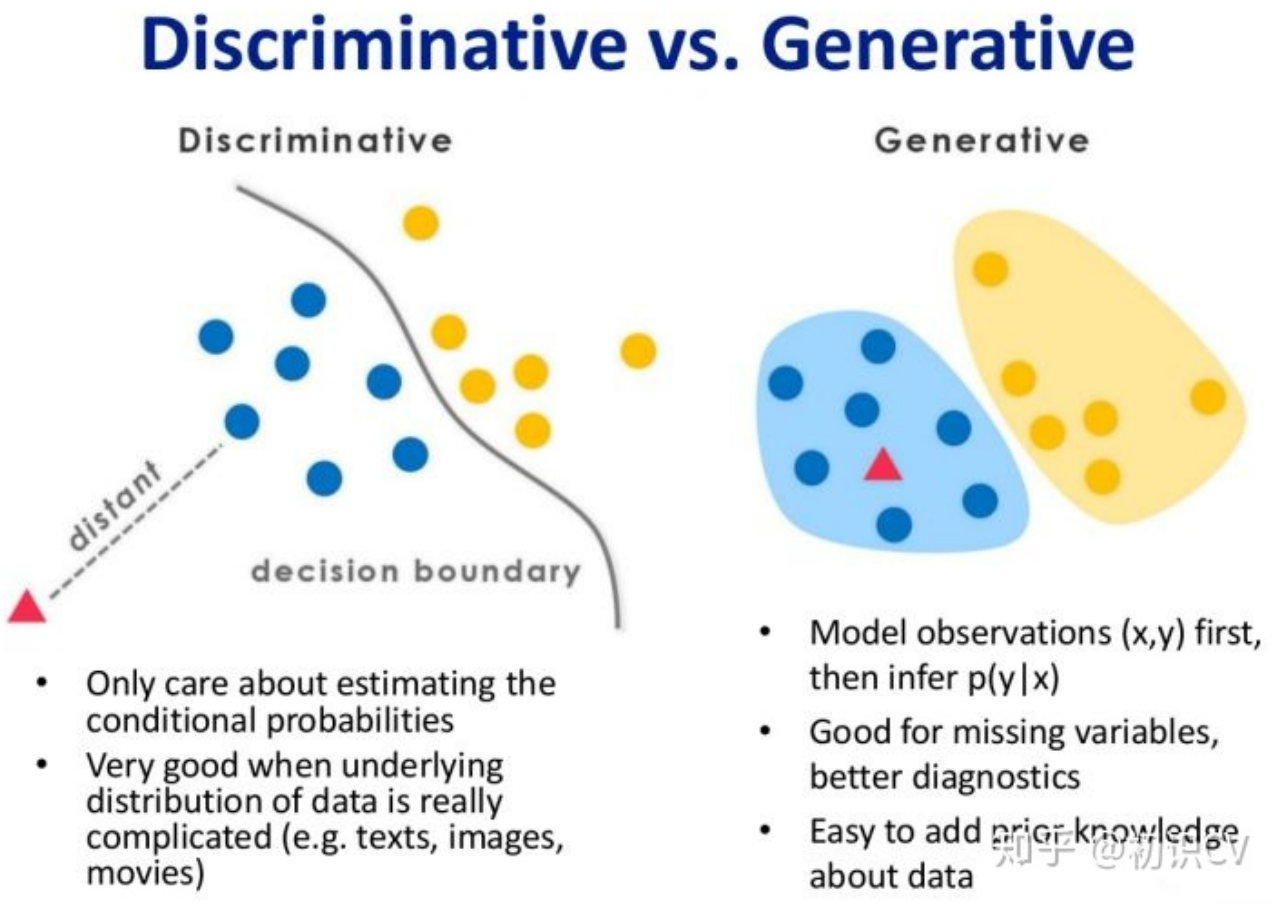

Fig. 1: left discriminant method, right generation method

The generation method (left side of Figure 1) learns the joint probability distribution P(X,Y) from the data, and then obtains the conditional probability distribution P(Y|X) as the prediction model, that is, the generation model: P(Y|X)=P(X,Y)/P(X)

It is called generation method because the model represents the generation relationship of a given input [Formula] and an output [Formula]. The typical generating models are naive Bayes method and Markov model.

For *example*: to determine whether a sheep is a goat or a sheep, the generative model is to first learn a goat model according to the characteristics of the goat, then learn a sheep model according to the characteristics of the sheep, and then extract features from the sheep, put it into the goat model to see what the probability is, and put it into the sheep model to see what the probability is.

The decision function f(X) or conditional probability distribution P(Y|X) is directly learned from the data as the prediction model, i.e. the discriminant model. The relationship between the discriminant methods is the given input X, what kind of output Y should be predicted. Typical discriminant models include k-nearest neighbor, perceptron, logistic regression model, maximum entropy model, support vector machine, lifting method and conditional random field.

For *example*: to determine whether a sheep is a goat or a sheep, the method of discriminant model is to learn the model from the historical data, and then to predict the probability that the sheep is a goat and a sheep by extracting the characteristics of the sheep.

difference

The generating method can restore the joint probability distribution P(X,Y), but the discriminant method cannot.

The learning convergence speed of the generation method is faster, that is, when the sample size increases, the learned model can converge to the real model faster.

When there are hidden variables, the generation method can still be used to learn, but the discriminant method can not be used.

The discriminant method directly learns the conditional probability P(Y|X) or decision function f(X), and when it directly faces the prediction, it often has higher learning accuracy.

Because the discriminant method can directly learn the conditional probability P(Y|X) or the decision function f(X), it can abstract the data in various degrees, define the characteristics and use the features, so the discrimination method can simplify the learning problem.