Homework 3

Zhaoheng Zheng
EECS 545 - Machine Learning

November 9, 2017

1) (a)

$$\begin{aligned}
\ell(\boldsymbol{w}) &= \sum_{i=1}^{n} -y_i log(\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_i}}) - (1-y_i)log(\frac{e^{-\boldsymbol{w}^T\boldsymbol{x}_i}}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_i}}) \\
&= \sum_{i=1}^{n} \big[y_i log(1+e^{\boldsymbol{w}^T\boldsymbol{x}_i}) - (1-y_i)(-\boldsymbol{w}^T\boldsymbol{x}_i) + (1-y_i)log(1+e^{-\boldsymbol{w}^T\boldsymbol{x}_i})\big] \\
&= \sum_{i=1}^{n} \big[-y_i\boldsymbol{w}^T\boldsymbol{x}_i + log(1+e^{\boldsymbol{w}^T\boldsymbol{x}_i})\big]
\end{aligned}$$

So,

$$\begin{aligned}
\nabla\ell(\boldsymbol{w}) &= \sum_{i=1}^{n} \big[\boldsymbol{x}_i(-y_i + \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_i}})\big] \\
&= \sum_{i=1}^{n} [\boldsymbol{x}_i(-y_i + h(\boldsymbol{x_i}))]
\end{aligned}$$

(b)

$$\begin{aligned}
\nabla^2\ell(\boldsymbol{w}) &= \frac{\partial^2\ell}{\partial\boldsymbol{w}\partial\boldsymbol{w}^T} \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i \frac{\partial h}{\partial\boldsymbol{w}^T} \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i \frac{\partial h}{\partial(e^{-\boldsymbol{w}^T\boldsymbol{x}_i})} \frac{\partial(e^{-\boldsymbol{w}^T\boldsymbol{x}_i})}{\partial(\boldsymbol{w}^T\boldsymbol{x}_i)} \frac{\partial(\boldsymbol{w}^T\boldsymbol{x}_i)}{\partial\boldsymbol{w}^T} \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i \cdot \frac{-1}{(1+e^{\boldsymbol{w}^T\boldsymbol{x}_i})^2} \cdot (-e^{\boldsymbol{w}^T\boldsymbol{x}_i}) \cdot \boldsymbol{x}_i^T \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^T \frac{e^{-\boldsymbol{w}^T\boldsymbol{x}}}{(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})^2} \\
&= \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^T h(\boldsymbol{x}_i)(1-h(\boldsymbol{x}_i))
\end{aligned}$$

So for any vector $\boldsymbol{u} \in \mathbb{R}^d$

$$\begin{aligned}\boldsymbol{u}^T \nabla^2 \ell(\boldsymbol{w}) \boldsymbol{u} &= \boldsymbol{u}^T \left[\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T h(\boldsymbol{x}_i)(1 - h(\boldsymbol{x}_i)) \right] \boldsymbol{u} \\ &= \sum_{i=1}^{n} \boldsymbol{u}^T \boldsymbol{x}_i \boldsymbol{x}_i^T \boldsymbol{u} h(\boldsymbol{x}_i)(1 - h(\boldsymbol{x}_i)) \\ &= \sum_{i=1}^{n} (\boldsymbol{x}_i^T \boldsymbol{u})^2 h(\boldsymbol{x}_i)(1 - h(\boldsymbol{x}_i))\end{aligned}$$

Since $0 < h(\boldsymbol{x}_i) < 1$, $\boldsymbol{u}^T \nabla^2 \ell(\boldsymbol{w}) \boldsymbol{u} \geq 0$ for any vector $\boldsymbol{u} \in \mathbb{R}^d$.
So $\nabla^2 \ell(\boldsymbol{w})$ is positive semi-definite, and thus $\ell(\boldsymbol{w})$ is convex and has no local minimum other than the global one.

(c) The algorithm takes 10 iterations to converge, the final coefficient $\boldsymbol{w}$ is

$$\boldsymbol{w} = [-4.738783 \quad 4.402149 \quad -1.515217]$$

The error curve is shown below:

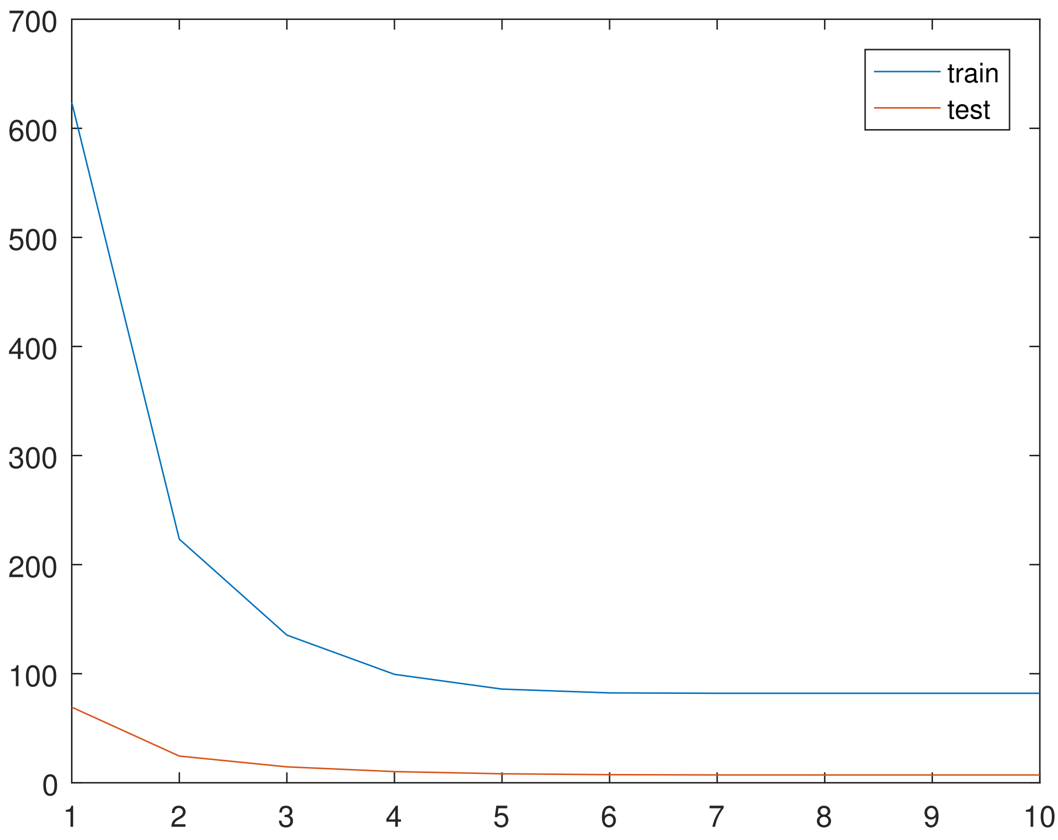

Figure 1: Error on training and test set

Following is the implementation:

```
w = [0 0 0];
train_data = [ones(size(train_x0)) train_x0 train_x1];
train_label = train_y;

test_data = [ones(size(test_x0)) test_x0 test_x1];
test_label = test_y;

loss = zeros(2, 1);
loss(1) = getLoss(train_data, train_label, w);
loss(2) = loss(1) + 1;

eps = 1e-8;
training_losses = [loss(1)];
test_losses = [getLoss(test_data, test_label, w)];

while (abs(loss(1) - loss(2)) > eps)
    loss(1) = loss(2);

    h_w = h(train_data, w)';
    delta_w = sum(-train_data .* repmat(train_label - h_w, 1, 3));
    hessian_w = zeros(3, 3);
    for i = 1 : size(train_data, 1)
        hessian_w = hessian_w + train_data(i, :)' * train_data(i, :) * h_w(i) * (1 - h_w(i));
    end
    w = w - (inv(hessian_w) * delta_w')';
    loss(2) = getLoss(train_data, train_label, w);
    test_losses = [test_losses getLoss(test_data, test_label, w)];
    training_losses = [training_losses, loss(2)];
end

fprintf('Takes %d iterations to converge.\n', size(training_losses, 2));
fprintf('%f\n', w);
x = 1 : size(training_losses, 2);
plot(x, training_losses, x, test_losses);
legend('train', 'test');
```

2) a)

$$L(\theta) = \prod_{i=1}^{n} f(x_i;\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\boldsymbol{x}_i - \alpha)^2}{2\sigma^2} \right)$$

So,

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^{n} \left[\log \frac{1}{\sqrt{2\pi\sigma^2}} + \left(-\frac{(x_i - \alpha)^2}{2\sigma^2} \right) \right] \\ &= n \log \frac{1}{\sqrt{2\pi\sigma^2}} + \sum_{i=1}^{n} \left(-\frac{(x_i - \alpha)^2}{2\sigma^2} \right) \end{aligned}$$

Thus,

$$\frac{\partial \ell}{\partial \alpha} = \sum_{i=1}^{n} \left(\frac{2(x_i - \alpha)}{2\sigma^2} \right)$$
$$\frac{\partial \ell}{\partial \sigma^2} = -\frac{n}{2} \frac{1}{2\pi\sigma^2} + \sum_{i=1}^{n} \left(\frac{(x_i - \alpha)^2}{2\sigma^4} \right)$$

We let

$$\frac{\partial \ell}{\partial \alpha} = \frac{\partial \ell}{\partial \sigma^2} = 0$$

Then we can get the maximum likelihood estimates of α and σ^2:

$$\hat{\alpha} = \frac{1}{n} \sum_{i=1}^{n} x_i$$
$$\hat{\sigma^2} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\alpha})^2$$

b)

$$\ell(\theta) = \sum_{i=1}^{n} \left[-\frac{1}{2} \log \left((2\pi)^d |\mathbf{\Sigma}| \right) + \left(-\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu})^T \mathbf{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}) \right) \right]$$
$$= \sum_{i=1}^{n} \left[-\frac{1}{2} \left(d \log(2\pi) + \log |\mathbf{\Sigma}| + \boldsymbol{x}_i^T \mathbf{\Sigma}^{-1} \boldsymbol{x}_i - 2\boldsymbol{\mu}^T \mathbf{\Sigma}^{-1} \boldsymbol{x}_i + \boldsymbol{\mu}^T \mathbf{\Sigma}^{-1} \boldsymbol{\mu} \right) \right]$$

So,

$$\frac{\partial \ell}{\partial \boldsymbol{\mu}} = \sum^{n} i = 1 \left[-\frac{1}{2} \left(2\mathbf{\Sigma}^{-1} \boldsymbol{x}_i + (\mathbf{\Sigma}^{-1} + (\mathbf{\Sigma}^{-1})^T) \boldsymbol{\mu} \right) \right]$$

We let

$$\frac{\partial \ell}{\partial \boldsymbol{\mu}} = 0$$

Then we obtain

$$\hat{\boldsymbol{\mu}} = \frac{2\mathbf{\Sigma}^{-1} \sum_{i=1}^{n} \boldsymbol{x}_i}{n(\mathbf{\Sigma}^{-1} + (\mathbf{\Sigma}^{-1})^T)}$$

Since $\mathbf{\Sigma}$ is a symmetric matrix, we have

$$\mathbf{\Sigma}^{-1} = (\mathbf{\Sigma}^{-1})^T$$

Thus, the maximum likelihood estimate of $\boldsymbol{\mu}$ will be

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$$

3) a) *Proof.* In definition,

$$I(X,Y) = \iint p(X,Y)\log\left(\frac{p(X,Y)}{p(X)p(Y)}\right)dXdY$$

So we can obtain,

$$\begin{aligned}
H(X) - H(X|Y) &= -\int p(X)\log p(X)dX + \iint p(X,Y)\log p(X|Y)dXdY \\
&= -\int \log p(X)\int p(X,Y)dYdX + \iint p(X,Y)\log\frac{p(X,Y)}{p(Y)}dXdY \\
&= \iint p(X,Y)\log\frac{p(X,Y)}{p(X)p(Y)}dXdY \\
&= I(X,Y)
\end{aligned}$$

and

$$\begin{aligned}
H(Y) - H(Y|X) &= -\int p(Y)\log p(Y)dY + \iint p(X,Y)\log p(Y|X)dYdX \\
&= -\int \log p(Y)\int p(X,Y)dXdY + \iint p(X,Y)\log\frac{p(X,Y)}{p(X)}dYdX \\
&= \iint p(X,Y)\log\frac{p(X,Y)}{p(X)p(Y)}dXdY \\
&= I(X,Y)
\end{aligned}$$

Therefore,

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

□

b) *Proof.* If $X = f(Y)$ and $Y = f^{-1}(X)$, then we can obtain

$$\begin{aligned}
p(X = f^{-1}(Y)|Y) = p(Y = f(X)|X) = 1 \\
p(X \neq f^{-1}(Y)|Y) = p(Y \neq f(X)|X) = 0
\end{aligned}$$

Thus,

$$\begin{aligned}
H(X|Y) &= -\iint p(X,Y)\log p(X|Y)dXdY \\
&= -\iint_{X=f^{-1}(Y)} p(X,Y)\log p(X|Y)dXdY - \iint_{X\neq f^{-1}(Y)} p(X,Y)\log p(X|Y)dYdX \\
&= 0 \\
H(Y|X) &= -\iint p(X,Y)\log p(Y|X)dYdX \\
&= -\iint_{Y=f(X)} p(X,Y)\log p(Y|X)dYdX - \iint_{Y\neq f(X)} p(X,Y)\log p(Y|X)dYdX \\
&= 0
\end{aligned}$$

Therefore,

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) = H(Y)$$

$\square$

c) *Proof.* For the maximum likelihood estimation of $q(x|\theta)$, the loss function will be

$$\ell(\theta) = \sum_{i=1}^{N} \log q(x = x_i|\theta)$$
$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} q(x = x_i|\theta)$$

And

$$\begin{aligned} -\int \hat{p}(x) \log \frac{q(x|\theta)}{\hat{p}(x)} dx &= -\int \hat{p}(x) \log[q(x|\theta)] - \hat{p}(x) \log[\hat{p}(x)] dx \\ &= -\int \hat{p}(x) \log[q(x|\theta)] dx + \int \hat{p}(x) \log[\hat{p}(x)] dx \end{aligned}$$

Since $\hat{p}(x) = \frac{1}{N} \sum\limits_{i=1}^{N} \mathbb{I}[x = x_i]$, we can obtain

$$\begin{aligned} \int \hat{p}(x) \log[q(x|\theta)] dx &= \int \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[x = x_i] \log[q(x|\theta)] dx \\ &= \sum_{i=1}^{N} \log[q(x = x_i|\theta)] \\ &= \ell(\theta) \end{aligned}$$

Because $\int \hat{p}(x) \log[\hat{p}(x)]$ is irrelevant to θ, minimizing the Kullback-Leibler divergence is equivalent to maximizing the likelihood function $\ell(\theta)$, which produces θ_{ML}.

Therefore, the minimum Kullback-Leibler divergence can be obtained by maximum likelihood estimate θ_{ML} given the data. $\square$

d) *Proof.* We can start from getting the maximum of $H(p)$ for any PDF that satisfies following constraints:

$$\int p(x) dx = 1 \tag{1}$$
$$\int x p(x) dx = \mu \tag{2}$$
$$\int (x - \mu)^2 dx = \sigma^2 \tag{3}$$

So the Lagrangian function can be written as

$$
\begin{aligned}
L(p(x), \alpha, \beta, \gamma) = &- \int p(x) \log[p(x)] dx + \lambda_1 \Big(\int p(x) dx - 1 \Big) \\
&+ \lambda_2 \Big(\int x p(x) dx - \mu \Big) + \lambda_3 \Big(\int (x-\mu)^2 p(x) dx - \sigma^2 \Big)
\end{aligned}
$$

So,

$$
\frac{\partial L}{\partial p(x)} = -\log[p(x)] + 1 + \lambda_1 + \lambda_2 x + \lambda_3 (x-\mu)^2
$$

Let $\frac{\partial L}{\partial p(x)} = 0$, we can obtain

$$
p(x) = \exp(1 + \lambda_1 + \lambda_2 x + \lambda_3 (x-\mu)^2)
$$

Put the $p(x)$ back to the constraints Eqn. (1) - (3), we can get the value of Lagrangian multiplier.

$$
\begin{aligned}
\lambda_1 &= -\frac{1}{2}\log(2\pi\sigma^2) - 1 \\
\lambda_2 &= 0 \\
\lambda_3 &= -\frac{1}{2\sigma^2}
\end{aligned}
$$

So the $p(x)$ that maximize $H(p)$ will be $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, which is the PDF of Gaussian distribution.
Therefore, Gaussian distribution has the maximum entropy $H(p)$. $\square$

4) a)

$$
\begin{aligned}
\ell(\boldsymbol{w}) &= \log P(Y|\tilde{\boldsymbol{X}}; \boldsymbol{w}) \\
&= \sum_{i=1}^{n} c_i \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(y_i - \boldsymbol{w}^T \tilde{\boldsymbol{x}}_i)^2}{2\sigma^2} \Big) \\
&= \sum_{i=1}^{n} c_i \log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} c_i (y_i - \boldsymbol{w}^T \tilde{\boldsymbol{x}}_i)^2
\end{aligned}
$$

So it is equivalent to minimizing $J(\boldsymbol{w}) = \sum_{i=1}^{n} c_i (y_i - \boldsymbol{w}^T \tilde{\boldsymbol{x}}_i)^2$

Let

$$
\frac{\partial J}{\partial b} = \sum_{i=1}^{n} -2c_i (y_i - \beta^T \boldsymbol{x}_i - b) = 0
$$

we can obtain

$$
\hat{b} = \frac{\sum_{i=1}^{n} c_i (y_i - \beta^T \boldsymbol{x}_i)}{\sum_{i=1}^{n} c_i} = \bar{y} - \beta^T \bar{\boldsymbol{x}}
$$

where $\bar{\boldsymbol{x}}$ and $\bar{y}$ are weight averages,

$$\bar{\boldsymbol{x}} = \frac{\sum_{i=1}^{n} c_i \boldsymbol{x}_i}{\sum_{i=1}^{n} c_i}, \quad \bar{y} = \frac{\sum_{i=1}^{n} c_i y_i}{\sum_{i=1}^{n} c_i}$$

Denote $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}, \quad \tilde{y}_i = y_i - \bar{y}$,

$$\begin{aligned}
J(\boldsymbol{w}) &= \sum_{i=1}^{n} c_i (y_i - \bar{y} - \beta^T (\boldsymbol{x}_i - \bar{\boldsymbol{x}}))^2 \\
&= \sum_{i=1}^{n} c_i (\tilde{y}_i - \beta^T \tilde{\boldsymbol{x}}_i)^2 \\
&= (\tilde{Y} - \tilde{\boldsymbol{X}}\beta)^T C (\tilde{Y} - \tilde{\boldsymbol{X}}\beta) \\
&= \tilde{Y}^T C \tilde{Y} - 2\tilde{Y}^T C \tilde{\boldsymbol{X}}\beta + (\tilde{\boldsymbol{X}}\beta)^T C (\tilde{\boldsymbol{X}}\beta) \\
&= -\frac{1}{2}\beta^T A \beta + U^T \beta + V
\end{aligned}$$

where

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1 \\ \vdots \\ \tilde{\boldsymbol{x}}_n \end{bmatrix}, \quad \tilde{Y} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_n \end{bmatrix}, \quad A = \tilde{\boldsymbol{X}}^T C \tilde{\boldsymbol{X}}, \quad U = -\tilde{\boldsymbol{X}} C \tilde{Y}, \quad V = \frac{1}{2}\tilde{Y}^T C \tilde{Y}$$

So,

$$\hat{\beta} = -A^{-1}U = \left(\tilde{\boldsymbol{X}}^T C \tilde{\boldsymbol{X}}\right)^{-1} \tilde{\boldsymbol{X}} C \tilde{Y}$$

Thus we get the maximum likelihood estimation of $\boldsymbol{w} = \begin{bmatrix} b, & \beta^T \end{bmatrix}^T$

$$\begin{aligned}
\hat{\beta} &= \left(\tilde{\boldsymbol{X}}^T C \tilde{\boldsymbol{X}}\right)^{-1} \tilde{\boldsymbol{X}} C \tilde{Y} \\
\hat{b} &= \bar{y} - \beta^T \bar{\boldsymbol{x}}
\end{aligned}$$

b) When the noise has different variance σ_i for each i, the maximum likelihood function becomes

$$\begin{aligned}
\ell(\boldsymbol{w}) &= \sum_{i=1}^{n} c_i \log \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - \boldsymbol{w}^T \tilde{\boldsymbol{x}}_i)^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{n} c_i \log \frac{1}{\sqrt{2\pi\sigma_i^2}} - \sum_{i=1}^{n} \frac{c_i}{2\sigma_i^2} (y_i - \boldsymbol{w}^T \tilde{\boldsymbol{x}}_i)^2
\end{aligned}$$

So we can simply let $c_i' = \frac{c_i}{2\sigma^2}$, turning the problem into the same weighted optimization problem as that we solved in a).

Denote

$$C' = diag(\frac{c_1}{2\sigma_1^2}, \cdots, \frac{c_n}{2\sigma_n^2})$$

The maximum likelihood estimate will be

$$\hat{\beta} = \left(\tilde{\boldsymbol{X}}^T C' \tilde{\boldsymbol{X}}\right)^{-1} \tilde{\boldsymbol{X}} C' \tilde{Y}$$
$$\hat{b} = \bar{y} - \beta^T \bar{\boldsymbol{x}}$$

where

$$\bar{\boldsymbol{x}} = \frac{\sum_{i=1}^{n} \frac{c_i}{2\sigma_i^2} \boldsymbol{x}_i}{\sum_{i=1}^{n} \frac{c_i}{2\sigma_i^2}}, \quad \bar{y} = \frac{\sum_{i=1}^{n} \frac{c_i}{2\sigma_i^2} y_i}{\sum_{i=1}^{n} \frac{c_i}{2\sigma_i^2}}$$

Other notations are the same as those in problem a).

5) a) *Proof.* Given the constraints $t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$, we can write it into

$$\xi_i \geq 1 - t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b)$$
$$\xi_i \geq 0$$

Thus we can obtain

$$\xi \geq \max(0, 1 - t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b))$$

So the optimal solution of ξ_i should be

$$\xi_i = \max(0, 1 - t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b))$$

Therefore minimizing

$$\frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{N} \xi_i$$

is equivalent to minimizing

$$\frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{N} \max(0, 1 - t^{(i)}(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b))$$

□

b) *Proof.* if $\xi_i^* > 0$, we can obtain

$$t^{(i)}((\boldsymbol{w}^*)^T \boldsymbol{x}^{(i)} + b^*) = 1 - \xi_i^*$$

Thus the distance from the training data $\boldsymbol{x}^{(i)}$ to the margin hyperplane

$$t^{(i)}((\boldsymbol{w}^*)^T \boldsymbol{x}^{(i)} + b^*) = 1$$

will be

$$distance = \frac{|t^{(i)}((\boldsymbol{w}^*)^T \boldsymbol{x}^{(i)} + b^*) - 1|}{||\boldsymbol{w}^*||} = \frac{1}{||\boldsymbol{w}^*||} \xi_i^*$$

Therefore, the distance is proportional to ξ_i^*. □

c) When $C \to \infty$, to conduct the minimization, we will have

$$t^{(i)}((\boldsymbol{w})^T \boldsymbol{x}^{(i)} + b) < 0, \forall i$$

Thus the minimization problem becomes

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{w}||^2$$

which is equivalent to the hard-margin SVM optimization problem.

d) (i) For soft-margin SVM, the best C is 0.30, and the test accuracy is 77.61%

(ii) In order to train a hard-margin SVM, we set C as $1e8$, getting a accuracy at 64.55%. From the result we can find that soft-margin SVM performs better than the hard-margin one. I guess is may cause by the reason that data points are so close that we can't find a hard-margin SVM to separate them very well.
Following is the implementation:

```
training_label = label(1:500, :);
training_data = diabetesscale(1:500, :);
test_label = label(501:768, :);
test_data = diabetesscale(501:768, :);

constant = linspace(0.1, 2, 20);
losses = [];

for i = 1 : 20
    rng(42);
    model = fitcsvm(training_data, training_label, 'KernelFunction', 'linear', 'KernelScale', 1, 'BoxConstraint', constant(i), 'CrossVal', 'on', 'KFold', 5);
    loss = kfoldLoss(model);
    losses = [losses loss];
end
[value, idx] = min(losses);
best_c = constant(idx);
best_c_model = fitcsvm(training_data, training_label, 'KernelFunction', 'linear', 'KernelScale', 1, 'BoxConstraint', best_c);
pred = predict(best_c_model, test_data);
fprintf('Best C:%.2f\n', best_c);
fprintf('Soft-Margin SVM Test Accuracy: %.2f%%\n', sum(pred==test_label) / 268 * 100);

hard_margin_model = fitcsvm(training_data, training_label, 'KernelFunction', 'linear', 'KernelScale', 1, 'BoxConstraint', 1e8);
pred = predict(hard_margin_model, test_data);
fprintf('Hard-Margin SVM Test Accuracy: %.2f%%\n', sum(pred==test_label) / 268 * 100);
```