# Problem 1

## 1(a)

Given, Loss function,

$$L(y_i, \hat{y_i}) = log(1 + e^{-y_i \hat{y_i}}) \tag{1}$$

Gradient

$$g_i = \frac{\partial L(y_i, \hat{y_i})}{\partial \hat{y_i}}$$
$$= \frac{\partial}{\partial \hat{y_i}} log(1 + e^{-y_i \hat{y_i}})$$
$$= \frac{-y_i e^{-y_i \hat{y_i}}}{1 + e^{-y_i \hat{y_i}}}$$
$$\Rightarrow g_i = \frac{-y_i e^{-y_i \hat{y_i}}}{1 + e^{-y_i \hat{y_i}}} \tag{2}$$

## 1(b)

Let,

$$\gamma^\star = min_{\gamma \in R} \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2$$

We solve for $y^\star$ below

$$\frac{\partial \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2}{\partial \gamma} = \sum_{i=1}^{n} 2h(x_i)(g_i + \gamma h(x_i)) = 0$$
$$\Rightarrow \sum_{i=1}^{n} g_i h(x_i) + \gamma \sum_{i=1}^{n} h(x_i)^2 = 0$$
$$\Rightarrow \gamma^\star = -\frac{\sum_{i=1}^{n} g_i h(x_i)}{\sum_{i=1}^{n} h(x_i)^2}$$

We find Second derivative to check if minimum is achieved

$$\frac{\partial^2 \sum_{i=1}^{n} (-g_i - \gamma h(x_i))^2}{\partial \gamma^2} = 2h(x_i)^2 > 0$$

Hence, we conclude $\gamma$ can be computed in a closed form solution. Now we solve for $h^\star$

$$h^\star = min_{h \in R}\ y^\star = min_{h \in R} - \frac{\sum_{i=1}^{n} g_i h(x_i)}{\sum_{i=1}^{n} h(x_i)^2}$$

$$\frac{\partial \frac{-\sum_{i=1}^{n} g_i h(x_i)}{\sum_{i=1}^{n} h(x_i)^2}}{\partial h} = -\frac{\sum_{i=1}^{n} g_i h^{'}(x_i)}{\sum_{i=1}^{n} h(x_i)^2} + \frac{\sum_{i=1}^{n} g_i h(x_i) \sum_{i=1}^{n} 2h(x_i) h^{'}(x_i)}{(\sum_{i=1}^{n} h(x_i)^2)^2} = 0$$

Solving the above equation gives us $h^{\star}$. Since $\gamma$ is not at all present in the equation, we can imply that $h^{\star}$ can be derived independent of $\gamma$

## 1(c)

As per Newtons Method, at step $t+1$, we have,

$$\alpha^{*^{t+1}} = \alpha^{*^{t}} - \frac{f'(\alpha)}{f''(\alpha)} \tag{3}$$

$$f(\alpha) = \textstyle\sum_{i=1}^{n} log(1 + e^{-y_i(\hat{y}_i + \alpha h^*(x_i))})$$

$$\begin{aligned} f'(\alpha) &= \frac{\partial \sum_{i=1}^{n} log(1 + e^{-y_i(\hat{y}_i + \alpha h^*(x_i))})}{\partial \alpha} \\ &= \sum_{i=1}^{n} \frac{-y_i h^*(x_i) e^{-y_i(\hat{y}_i + \alpha h^*(x_i))}}{1 + e^{-y_i(\hat{y}_i + \alpha h^*(x_i))}} \end{aligned}$$

$$\Rightarrow f'(\alpha) = \sum_{i=1}^{n} \frac{-y_i h^*(x_i)}{e^{y_i(\hat{y}_i + \alpha h^*(x_i))} + 1} \tag{4}$$

$$f''(\alpha) = \frac{\partial f'(\alpha)}{\partial \alpha} = \sum_{i=1}^{n} \frac{-y_i^2 h^*(x_i)^2 e^{y_i(\hat{y}_i + \alpha h^*(x_i))}}{(e^{y_i(\hat{y}_i + \alpha h^*(x_i))} + 1)^2}$$

$$f''(\alpha) = \sum_{i=1}^{n} \frac{-y_i^2 h^*(x_i)^2 e^{y_i(\hat{y}_i + \alpha h^*(x_i))}}{(e^{y_i(\hat{y}_i + \alpha h^*(x_i))} + 1)^2} \tag{5}$$

Substituting (4) and (5) in (3), we get, for step $t+1$,

$$\alpha^{*^{t+1}} = \alpha^* - \frac{f'(\alpha^*)}{f''(\alpha^*)}$$

$$\Rightarrow \alpha^{*^{t+1}} = \alpha^* - \sum_{i=1}^{n} \frac{e^{-y_i(\hat{y}_i + \alpha^* h^*(x_i))} + 1}{y_i h^*(x_i) 1} \tag{6}$$

Substituting (6) in the update step, we get,

$$\hat{y}_i = \hat{y}_i + \left[ \alpha^* - \sum_{i=1}^{n} \frac{e^{-y_i(\hat{y}_i + \alpha^* h^*(x_i))} + 1}{y_i h^*(x_i) 1} \right] h^*(x_i)$$

# Problem 2

## 2(a)

Maximize the flatness means minimize the Norm-2. Hence the Primal optimization formulation is:

$$
\begin{aligned}
&minimize\ \frac{1}{2}||w||_2^2 &(7)\\
&y_i - w^T x - b \leqslant \epsilon &(8)\\
&w^T x + b - y_i \leqslant \epsilon &(9)
\end{aligned}
$$

## 2(b)

To cope with unfeasible constraints, we can introduce Slack variables $\xi_i, \xi_i^*$ that ensures the deviation above $\epsilon$ is taken care of. Hence, the formulation becomes:

$$
\begin{aligned}
minimize\ \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) && (10)\\
y_i - w^T x - b \leqslant \epsilon + \xi_i && (11)\\
w^T x + b - y_i \leqslant \epsilon + \xi_i^* && (12)\\
\xi_i, \xi_i^* \geqslant 0 && (13)
\end{aligned}
$$

The Loss function is called $\epsilon$-insensitive loss function and is given by,

$$
|\xi|_\epsilon = \begin{cases} 0 & \text{If } |\xi| \leqslant \epsilon \\ |\xi| - \epsilon & \text{Otherwise} \end{cases} \tag{14}
$$

## 2(c)

Lagrange function for above Primal form is given by :

$$
L = \frac{1}{2}||w||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}(\eta\xi_i + \eta^*\xi_i^*)
$$
$$
-\sum_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + w^T b) - \sum_{i=1}^{l}\alpha_i^*(\epsilon + \xi_i^* + y_i - w^T b)
$$

Here, $\eta_i, \eta_i^*, \alpha_i$ and $\alpha_i^*$ are the Lagrange multipliers. These have to satisfy positivity constraints.

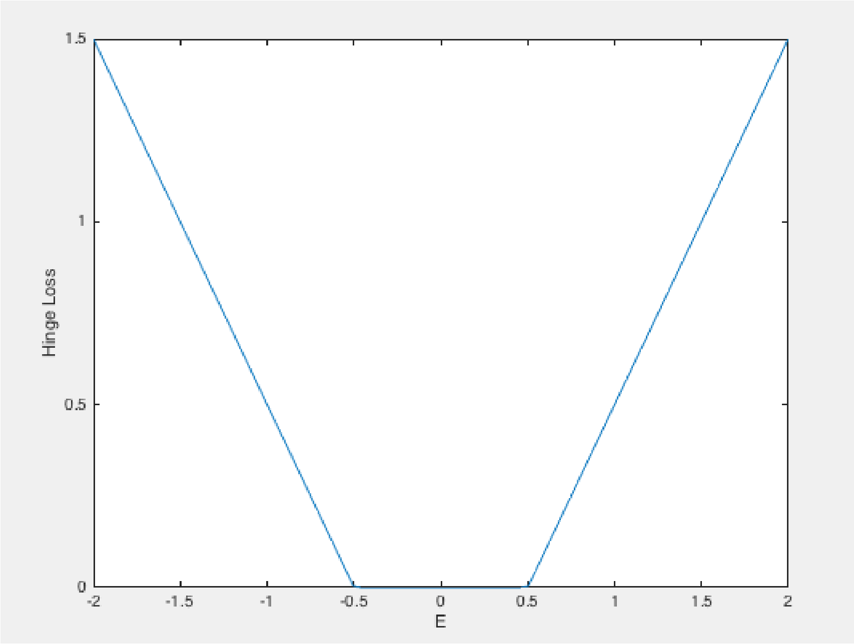


Figure 1: 2.(b) Hinge loss function for $\epsilon = 0.5$

$\eta_i, \eta_i^*, \alpha_i, \alpha_i^* \geqslant 0$
We now take partial derivatives with respect to primal variables.

$$\partial_b L = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) = 0 \tag{15}$$

$$\partial_w L = w + \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) x_i = 0 \tag{16}$$

$$\partial_{\xi_i} L = C - \alpha_i - \eta_i = 0 \tag{17}$$

$$\partial_{\xi_i^*} L = C - \alpha_i^* - \eta_i^* = 0 \tag{18}$$

Substituting (15), (16), (17) and (18) into Lagrange function, we get the dual form:

$$Maximize \;\; -\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) < x_i, x_j > -\epsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i (\alpha_i - \alpha_i^*)$$

Subject to

$$\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

Also from (16), $w = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i$ and $f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) < x_i, x > +b$

### 2(d)

Given Kernel $k(x, x') := \phi^T(x)\phi(x')$. As the above dual form depends only on the Dot product of $x$, we can transform dot products to the given Kernel function as below:

$$Maximize\ -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k(x_i,x_j)-\epsilon\sum_{i=1}^{l}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{l}y_i(\alpha_i-\alpha_i^*)$$

Subject to

$$\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)=0 \text{ and } \alpha_i,\alpha_i^*\in[0,C]$$

Also from (16), $w=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\phi(x_i)$ and $f(x)=\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)k(x_i,x)+b$

# Problem 3

### 3.1

After transforming the data, we get Train and Test sets with dimensions 2000 x 45 each.

### 3.2

Both trainsvm.m and testsvm.m files are implemented. For $C = 0.1$, we observed a test accuracy of 0.9365

### 3.3

#### (a)

| **C** | **Average Accuracy(%)** | **Average Time** |
|---|---|---|
| $4^{-6}$ | 58.65 | 0.8900 |
| $4^{-5}$ | 90.40 | 0.6067 |
| $4^{-4}$ | 92.30 | 0.7867 |
| $4^{-3}$ | 93.70 | 0.7133 |
| $4^{-2}$ | 94.25 | 0.7567 |
| $4^{-1}$ | 94.15 | 0.8300 |
| 1 | 93.85 | 0.8600 |
| 4 | 93.65 | 0.9033 |
| $4^2$ | 93.55 | 1.0300 |

Increase in C value results in increase in Average accuracy too. But, we observe that the accuracy seems to reach a plateau at some C and starts to decrease. It may be due to

overfitting caused after a particular C, as more weight is put on slack variables.
Increase in C value results in increase of Average Time too. The can be because, as C increases, the model dimension increases, which causes longer compute time.

### (b)

We choose C which gives best Average Accuracy. Hence, we choose $4^{-2}$

### (c)

Test Accuracy = 0.9335

## 3.4

| C | Average Accuracy(%) | Average Time |
|---|---|---|
| $4^{-6}$ | 56.50 | 0.2233 |
| $4^{-5}$ | 91.35 | 0.1867 |
| $4^{-4}$ | 92.70 | 0.1267 |
| $4^{-3}$ | 93.60 | 0.0833 |
| $4^{-2}$ | 94.40 | 0.0633 |
| $4^{-1}$ | 94.50 | 0.0600 |
| 1 | 94.40 | 0.0600 |
| 4 | 94.40 | 0.1033 |
| $4^{2}$ | 94.55 | 1.1567 |

### (a)

As we see from above tables, compared to my SVM, LIBSVM is marginaaly better overall. But LIBSVM does give better performance after the model hits a plateau at some C.

### (b)

LIBSVM is much faster ( 5-10 times faster on average) compared to my implementation of SVM.

## 3.5

### (a)

**Table for Average Accuracy**

| **C/Degree** | 1 | 2 | 3 |
|---|---|---|---|
| $4^{-3}$ | 55.7500 | 55.7500 | 55.7500 |
| $4^{-2}$ | 78.1500 | 78.1500 | 78.1500 |
| $4^{-1}$ | 92.2500 | 92.2500 | 92.2500 |
| 1 | 94.7500 | 94.7500 | 94.7500 |
| 4 | 96.2000 | 96.2000 | 96.2000 |
| $4^{2}$ | 97.0500 | 97.0500 | 97.0500 |
| $4^{3}$ | 96.9500 | 96.9500 | 96.9500 |
| $4^{4}$ | 96.7500 | 96.7500 | 96.7500 |
| $4^{5}$ | 96.4500 | 96.4500 | 96.4500 |
| $4^{6}$ | 96.4500 | 96.4500 | 96.4500 |
| $4^{7}$ | 96.4500 | 96.4500 | 96.4500 |

**Table for Average Time**

| **C/Degree** | 1 | 2 | 3 |
|---|---|---|---|
| $4^{-3}$ | 0.2333 | 0.5067 | 0.7633 |
| $4^{-2}$ | 0.2633 | 0.4933 | 0.7633 |
| $4^{-1}$ | 0.2067 | 0.3900 | 0.5700 |
| 1 | 0.1133 | 0.2300 | 0.3500 |
| 4 | 0.0800 | 0.1633 | 0.2433 |
| $4^{2}$ | 0.0733 | 0.1433 | 0.2200 |
| $4^{3}$ | 0.0667 | 0.1333 | 0.2000 |
| $4^{4}$ | 0.0667 | 0.1333 | 0.1967 |
| $4^{5}$ | 0.0633 | 0.1333 | 0.1967 |
| $4^{6}$ | 0.0667 | 0.1333 | 0.2033 |
| $4^{7}$ | 0.0667 | 0.1300 | 0.1967 |

**(b)**

**Table for Average Accuracy**

| **C/Gamma** | $4^{-7}$ | $4^{-6}$ | $4^{-5}$ | $4^{-4}$ | $4^{-3}$ | $4^{-2}$ | $4^{-1}$ |
|---|---|---|---|---|---|---|---|
| $4^{-3}$ | 73.2500 | 73.2500 | 73.2500 | 73.2500 | 73.2500 | 73.2500 | 73.2500 |
| $4^{-2}$ | 92.3000 | 92.3000 | 92.3000 | 92.3000 | 92.3000 | 92.3000 | 92.3000 |
| $4^{-1}$ | 93.4000 | 93.4000 | 93.4000 | 93.4000 | 93.4000 | 93.4000 | 93.4000 |
| 1 | 95.1000 | 95.1000 | 95.1000 | 95.1000 | 95.1000 | 95.1000 | 95.1000 |
| 4 | 96.2500 | 96.2500 | 96.2500 | 96.2500 | 96.2500 | 96.2500 | 96.2500 |
| $4^2$ | 96.9000 | 96.9000 | 96.9000 | 96.9000 | 96.9000 | 96.9000 | 96.9000 |
| $4^3$ | 96.9000 | 96.9000 | 96.9000 | 96.9000 | 96.9000 | 96.9000 | 96.9000 |
| $4^4$ | 97.0500 | 97.0500 | 97.0500 | 97.0500 | 97.0500 | 97.0500 | 97.0500 |
| $4^5$ | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 |
| $4^6$ | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 |
| $4^7$ | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 | 96.5000 |

**Table for Average Time**

| **C/Gamma** | $4^{-7}$ | $4^{-6}$ | $4^{-5}$ | $4^{-4}$ | $4^{-3}$ | $4^{-2}$ | $4^{-1}$ |
|---|---|---|---|---|---|---|---|
| $4^{-3}$ | 0.2433 | 0.4733 | 0.7067 | 0.9367 | 1.1700 | 1.4067 | 1.6600 |
| $4^{-2}$ | 0.2000 | 0.4167 | 0.5933 | 0.7733 | 0.9533 | 1.1333 | 1.3167 |
| $4^{-1}$ | 0.1167 | 0.2333 | 0.3567 | 0.4700 | 0.5967 | 0.7133 | 0.8300 |
| 1 | 0.0900 | 0.1733 | 0.2700 | 0.3633 | 0.4600 | 0.5567 | 0.6533 |
| 4 | 0.0667 | 0.1333 | 0.2067 | 0.2733 | 0.3533 | 0.4200 | 0.4867 |
| $4^2$ | 0.0667 | 0.1333 | 0.2000 | 0.2633 | 0.3367 | 0.4000 | 0.4667 |
| $4^3$ | 0.0633 | 0.1267 | 0.1867 | 0.2467 | 0.3067 | 0.3600 | 0.4167 |
| $4^4$ | 0.0567 | 0.1133 | 0.1700 | 0.2267 | 0.2833 | 0.3433 | 0.3967 |
| $4^5$ | 0.0600 | 0.1167 | 0.1733 | 0.2367 | 0.2967 | 0.3533 | 0.4100 |
| $4^6$ | 0.0600 | 0.1267 | 0.2067 | 0.2667 | 0.3300 | 0.3967 | 0.4733 |
| $4^7$ | 0.0700 | 0.1433 | 0.2100 | 0.2733 | 0.3433 | 0.4133 | 0.4800 |

Comparing outputs from Polynomial and RBF Kernels, we see that 97.05% is the highest accuracy achieved. Polynomial Kernel achieves this accuracy for $C = 4^2$. So we choose:
Kernel: Polynomial
C: $4^2$
Degree: 2
Using above parameters in LIBSVM, we get 94.9500% accuracy on Test dataset.