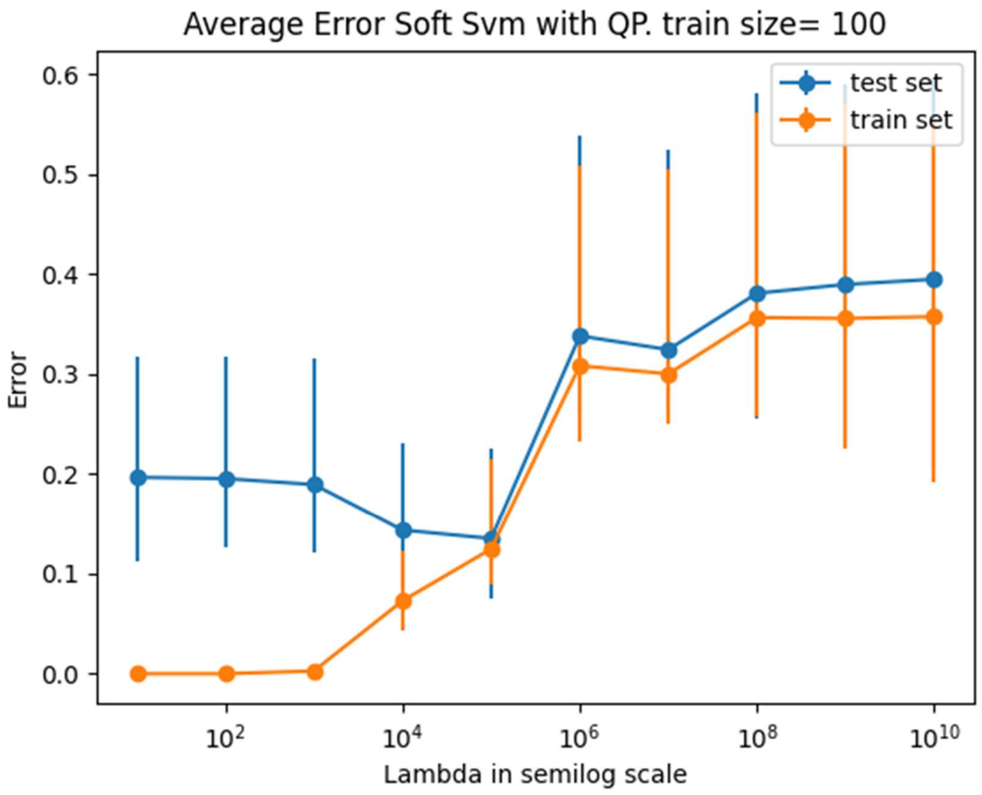
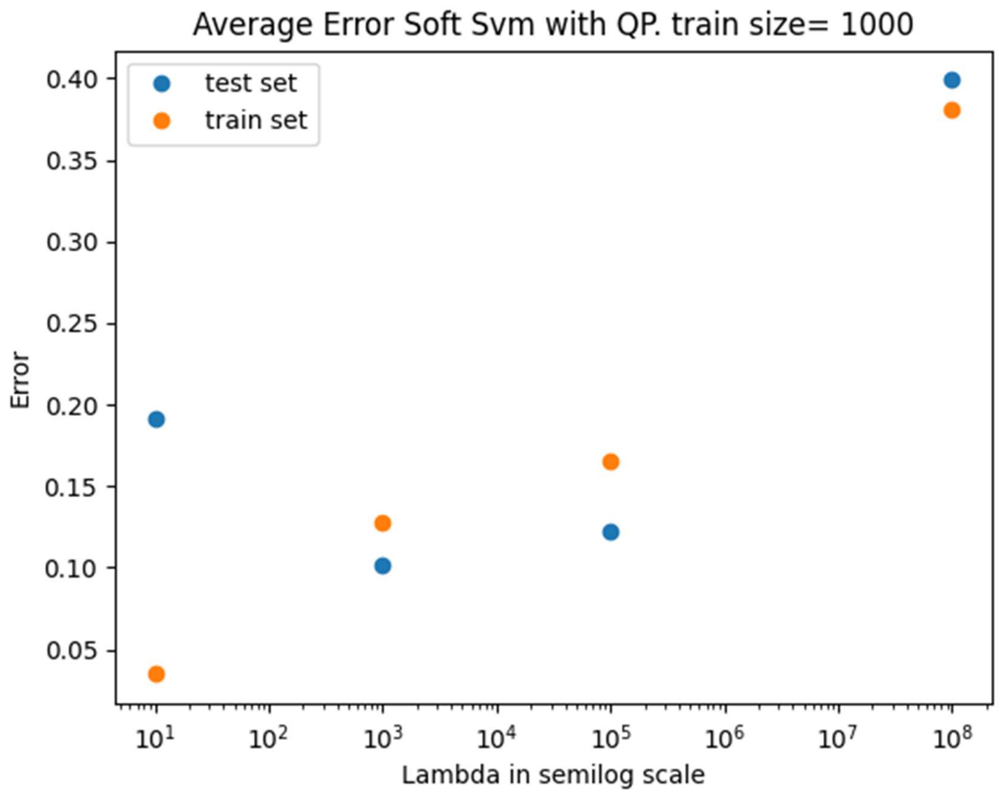
Assignment 2

Question 2:

a) The plot below shows the graph of average error in 10 iterations of soft SVM with sample size 100, as a function on λ, with error bars showing the maximal and minimal error of the 10 iterations:



b) The plot below shows the value of error of soft SVM with sample size 1000, as a function of λ.



c) We expected that small sample size will have a smaller training error. This is because that as the sample size gets smaller the data may be easier to separate with larger margin which in turn produce a smaller hinge loss and low training error. On the other hand, as the sample size increases it represents the distribution better, thus the algorithm's error gets lower. The results indeed match our expectations, as we can see in the plots at each point we checked training error in both samples, the train error in where the sample size is 100 is smaller. Also the test error is higher in the small sample size.

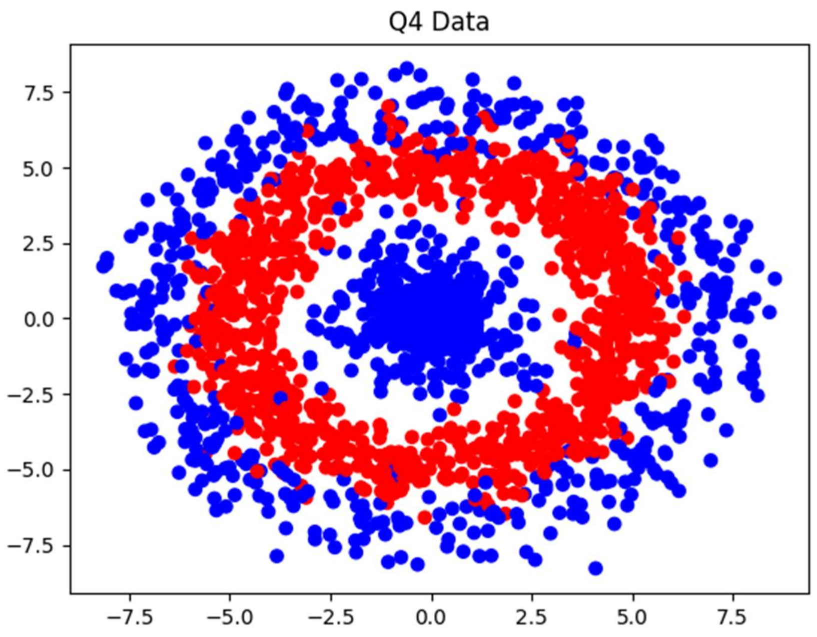
Also we know that large λ implies larger penalty on $||w||$ at the objective $\lambda\left||w|\right|^2 + \ell^h(w, S)$. Meaning it enforces the optimal solution to have a smaller $||w||$, which in turn implies larger margin in which event a correct answer is penalized.

Penalizing correct answers can lead to minimizing an objective that finds a separator that is wrong in many examples. Hence, we expect that larger lambda will imply higher train error for both sample sizes which is exactly what we can see at the plots.

Thinking on the test error, there might be a λ that is the perfect balance when minimizing the objective $\lambda\left||w|\right|^2 + \ell^h(w, S)$, meaning the separator both represents the sample and has large margin as well. We expect that for such λ the error on the test will be optimal, meaning it will be larger for both larger and smaller λs. Also the optimal lambda can change when sample size changes. As we can see at the plots, there is an optimal λ around 10^5 when sample size is 100 and 10^3 when sample size is 1000. We can see that for values of different λ the test error is larger.

<u>Question 4:</u>

a)



It may be a better idea to use a kernel soft SVM instead of linear soft SVM. The reason is there is no 2-dim linear separator for the data, every line that passes through the origin separates approximately half of the red points above it, same for the blue points. It means that this linear classifier will be wrong about half of the examples witch is like guessing the label. On the other hand, there might be a feature map that takes the data to a higher dimension in which a linear separator soft SVM produces implies better results. For instance, a linear separator that its interpretation at 2-dim is two circles where most of red labels are between them.

b) RBF soft SVM results:

$\lambda \quad \backslash \quad \sigma$	0.01	0.5	1
1	0.0805	0.0685	0.088
10	0.0805	0.0685	0.088
100	0.0805	0.0685	0.088

Best parameters are $\lambda = 1$ and $\sigma = 0.5$, test average error for these parameters is: 0.045.

Soft SVM results:

λ	Average Error
1	0.50028
10	0.50028
100	0.50028

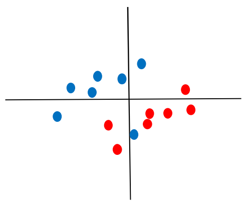
Best parameter is $\lambda = 1$, test average error for this parameters is: 0.5.

c) The error of RBF approach was much lower than linear soft SVM. Just as expected, the error for soft SVM was approximately 0.5 which is similar to guessing, this is exactly what we expected in section a.

One reason why RBF SVM might get better validation error is when the data is 'far' from separable, in a sense that there is no good linear predictor that achieve small error (like in our data sample). If the sample represents well the distribution then we expect also bad results on validation. On the other hand, there may be a good separator in some higher dimensional feature space which its inner product is the Gaussian kernel. In that case, RBF SVM will achieve better results on the validation (assuming the sample represents the distribution well.

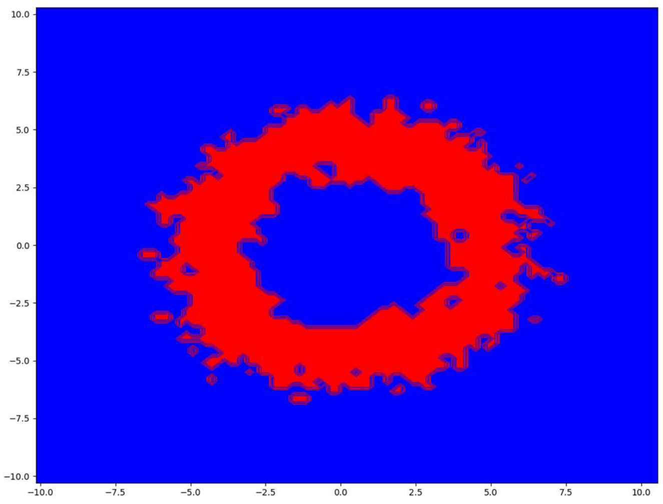
If the original data has a good linear separator in the first place but the problem is not separable, like data is arranged linearly with some noise. If we use RBF SVM it might give too much weight to the noise, meaning noisy points will make more influence (depends on the choice of σ which defines the radius of influence). This in turn **may cause overfitting**, making a smaller error on the sample set but large error on the validation set. An illustration that explains our point can be as follows:

The predictor returned from RBF SVM may lead to overfitting due to influence of the low blue labeled point, while linear predictor will be good for this dataset.

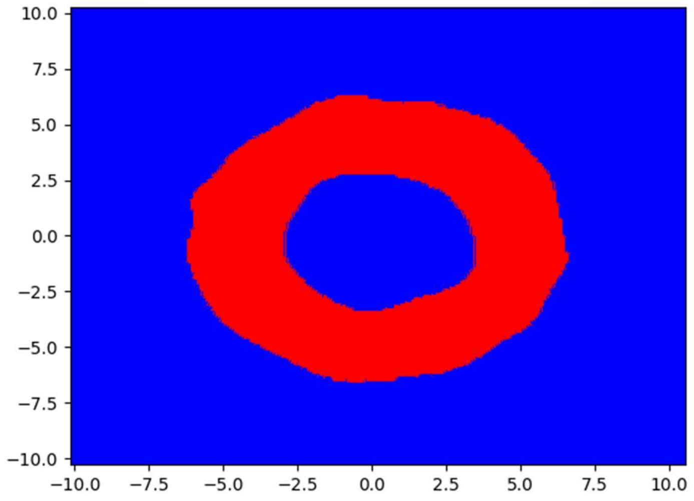


d)

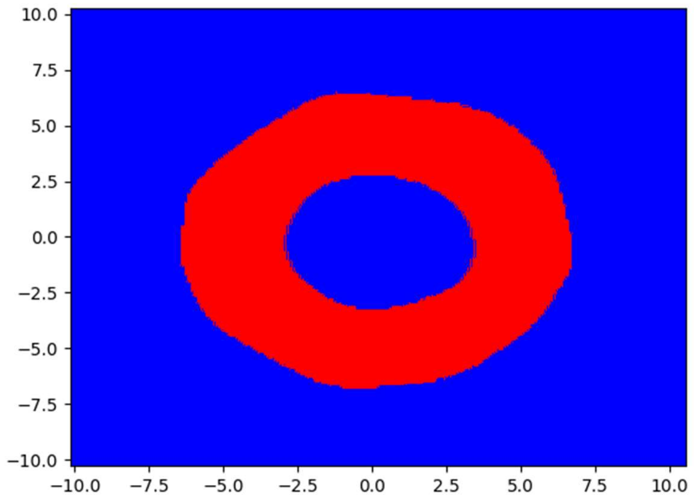
Prediction on grid for $\sigma = 0.01$



Prediction on grid for $\sigma = 0.5$



Prediction on grid for $\sigma = 1$



e) In class we saw that σ can be thought as the "radius of influence" of each x_i. Also there is a trade-off in the size of σ:
Small σ can have a fine-tuned boundary which implies larger estimation error.
Larger σ makes the function smoother which implies larger approximation error.

This explains well the differences between our plots. On the first plot where $\sigma = 0.01$ we can see the fine-tuned boundary, even if there is a single 'blue' surrounded by many 'reds' in the sample, the predictor tags its area with 'blue', which reduces the approximation error while increasing the estimation error. On the second plot where $\sigma = 0.5$ the boundary is much smoother due to the 'radius of influence' of each x_i. We can see that there are no 'holes' of 'blue' in the middle of the red circle leading this time to larger approximation error but probably smaller estimation error. The last plot where $\sigma = 1$ has even smoother boundary the before again due to the radius of influence.

<u>**Question 5:**</u>

a) Define the minimization problem:

$$Minimize_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^m}\ \lambda\|\mathrm{w}\|^2 + \sum_{i=1}^{m}\xi_i^2$$

$$s.t\ \forall i, \qquad y_i < w, x_i > \geq 1-\xi_i, and\ \xi_i \geq 0$$

Denote the penalty for w: $P(w) := \lambda\|w\|^2 + \sum_{i=1}^{m}\left(\ell^h\big(w,(x_i,y_i)\big)\right)^2$

<u>Claim:</u> The minimization problem above returns $\widehat{w}$ that minimizes $P(w)$.

<u>Proof:</u>

The constraints are equivalent to $\xi_i \geq \max\{0, 1 - y_i < w, x_i >\} = \ell^h\big(w,(x_i,y_i)\big)$.
Hence, $\xi_i^2 \geq \ell^h\big(w,(x_i,y_i)\big)^2$ since $\ell^h$is nonnegative, in the optimal solution $\xi_i^2 = \ell^h\big(w,(x_i,y_i)\big)^2$.
Thus, the minimization is equivalent to:

$$Minimize_{w\in\mathbb{R}^d}\ \lambda\|\mathrm{w}\|^2 + \sum_{i=1}^{m}\left(\ell^h\big(w,(x_i,y_i)\big)\right)^2 = P(w)$$

b) The quadratic program will be defined as follows:

For $I_1 \in \mathbb{R}^{d\times d}$ and $I_2 \in \mathbb{R}^{m\times m}$ identity matrices define $H \in \mathbb{R}^{(d+m)\times(d+m)}$ to be: $H = \begin{pmatrix} 2\lambda I_1 & 0 \\ 0 & 2I_2 \end{pmatrix}$
Notice that H is indeed a positive defined matrix.

Define $u = (0,0,\dots,0) \in \mathbb{R}^{d+m}$.

For $C = \begin{pmatrix} y_1x_1(1) & \cdots & y_1x_1(d) \\ \vdots & \ddots & \vdots \\ y_mx_m(1) & \cdots & y_mx_m(d) \end{pmatrix} \in \mathbb{R}^{m\times d}$ define $A = \begin{pmatrix} C & I_1 \\ 0 & I_1 \end{pmatrix} \in \mathbb{R}^{(m+m)\times(d+m)}$.

And finally $v = (1,1,\dots,1,0,0,\dots 0,) \in \mathbb{R}^{2m}$ (m times 1 and m times zero vector).

So the quadratic is:

$$Minimize_{z\in\mathbb{R}^{d+m}}\frac{1}{2}z^T Hz + < u, z >$$

$$s.t \qquad\qquad Aw > v$$

Here we think of the vector z as $z = (w(1), \dots, w(d), \xi_1, \dots, \xi_m)$.

Question 6:

In order that the Representer Theorem will hold for an objective it must be in the following form:

$$\begin{aligned}&Minimize_w f(< w, \psi(x_1) >, \dots, < w, \psi(x_m) >) + R(||w||)\\ &where\ f: \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}\ is\ a\ function, and\\ &R: \mathbb{R}_+ \to \mathbb{R}\ is\ a\ monotonic\ non-decreasing\ function.\end{aligned}$$

Notice that in our case $\sum_{i=1}^{m} \exp^{|<w,x_i>-y_i|}$ is a function of $\{< w, x_i >\}_i$ so we can define:

$$f(< w, \psi(x_1) >, \dots, < w, \psi(x_m) >) = \sum_{i=1}^{m} \exp^{|<w,x_i>-y_i|}$$

Also $r||w||_2^4$ is a function of $||w||_2$ so if we define $R\left(||w||_2\right) = r||w||_2^4$ than we have a function $R: \mathbb{R}_+ \to \mathbb{R} \cup \{\infty\}$. For any w, w' if $||w||_2 \le ||w'||_2$ then for $r \ge 0$ we have $r||w||^4{}_2 \le r||w'||_2^4$ and for $r < 0$ we get $r||w||^4{}_2 > r||w'||_2^4$. Hence, in order for R to be a monotonic non-decreasing function we need to take r to be non-negative real number $r \ge 0$.

So for every $r \ge 0$ the Representer Theorem holds for the objective.

Question 7:

a) Assume towards contradiction that there exists a feature mapping ψ such that $K(x, x') =< \psi(x), \psi(x') >$. For x such that $x(1) \ne 0$ so $< \psi(x), \psi(x) >= K(x, x) = -x(1)^2 < 0$ which is a violation of an inner product property that $0 \le < w, w >$.

b) Notice that $K(x, x') = K(x', x)$ meaning that:
$\big(x(1) + x(2)\big)\big(x'(3) + x'(4)\big) = \big(x(1) + x(2)\big)\big(x'(3) + x'(4)\big)$ which is not true in general. For instance, take $x = (1, \dots, 1) \in \mathbb{R}^d$ and $x' = (1,2,3, \dots, d) \in \mathbb{R}^d$ so $K(x, x') = 14 \ne 6 = K(x', x)$. So if we assume that $K(x, x') =< \psi(x), \psi(x') >$ for some feature map ψ we get that $< \psi(x), \psi(x') > \ne < \psi(x'), \psi(x) >$ which violates the property of inner product $< w, v >=< v, w >$.

c)

Kernel Perceptron Algorithm:

Input A Gram matrix G where $G_{i,j} = K\big(x_i, x_j\big) \equiv < \psi(x_i), \psi\big(x_j\big) >$
Output $\alpha(1), \dots, \alpha(m)$ such that $w = \sum_{i=1}^{m} \alpha(i)\psi(x_i)$

```
1: $t \leftarrow 1$, $\big(\alpha^1(1), \dots, \alpha^1(m)\big) \leftarrow (0, \dots 0)$
2: while $\exists i\ s.t.\ y_i(\sum_{j=1}^{m} \alpha(j)^t K\big(x_i, x_j\big)) \le 0$ do
3:     $\alpha^{t+1}(i) \leftarrow \alpha^t(i) + y_i$
4:     $t \leftarrow t + 1$
5: end while
6: Return $\alpha^t(1), \dots, \alpha^t(m)$
```

Why does the update make sense? Notice that:

$$y_i \sum_{j=1}^{m} \alpha(j)^{t+1} K(x_i, x_j) = y_i(\sum_{j=1}^{m} \alpha(j)^{t} K(x_i, x_j) + y_i K(x_i, x_i)) = y_i \sum_{j=1}^{m} \alpha(j)^{t} K(x_i, x_j) + y_i^2 K(x_i, x_i)$$

$$= y_i \sum_{j=1}^{m} \alpha(j)^{t} K(x_i, x_j) + K(x_i, x_i)$$

So each update moves $y_i \sum_{j=1}^{m} \alpha(j)^{t+1} K(x_i, x_j)$ closer to being positive.

Also, if the algorithm stops then $y_i(\sum_{j=1}^{m} \alpha(j)^{t} K(x_i, x_j)) > 0$ and:

$$h_w(\psi(x)) = sign(< w, \psi(x) >) = sign\left(\sum_{j=1}^{m} \alpha(j)^{t+1} K(x_i, x_j)\right)$$

So $y_i(\sum_{j=1}^{m} \alpha(j)^{t} K(x_i, x_j)) > 0$ means that $h_w(\psi(x))$ is correct. Thus is the algorithm stops it indeed found a separator $w = \sum_{i=1}^{m} \alpha(i)\psi(x_i)$ in feature space.