CSCI567 HW2

Ritu Pravakar and Charlene Yuen

September 2022

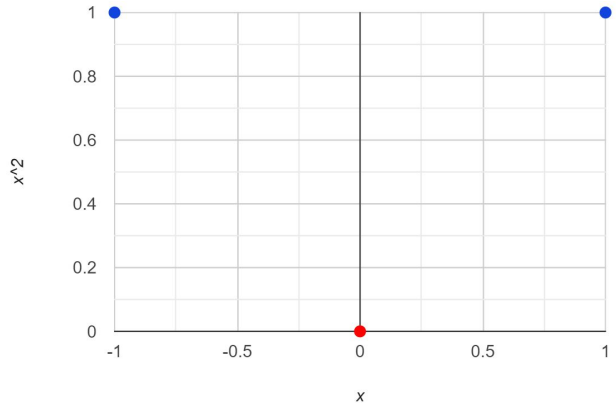
Theory-based Questions

Problem 1: Support Vector Machines (19pts)

1.1 (2pts) Can these three points in their current one-dimensional feature space be perfectly separated with a linear classifier? Why or why not?

No, since the three points can't be separated with a linear function with one degree of complexity. It would need a quadratic (degree 2) or higher complexity function to separate them, or higher dimensions.

1.2 (3pts) Now we define a simple feature mapping $\phi(x) = [x, x^2]^T$ to transform the three points from one-dimensional to two-dimensional feature space. Plot the transformed points in the new two-dimensional feature space. Is there a linear model $\mathbf{w}^T\mathbf{x} + b$ for some $\mathbf{w} \in \mathbb{R}^2$ and $b \in \mathbb{R}$ that can correctly separate the three points in this new feature space? Why or why not?



Yes, you can simply separate the points now that the labels are separated by their y coordinates in two-dimensional space. An example would be $\mathbf{w} = [0, 0]^T$ and $b = 0.5$. b can range from $(-1, 1)$, and the slope of x: w_1 should satisfy $-1 < w_1 < 1$ and w_2 can range from $0 \leq w_2 \leq 2$.

1.3 (2pts) Given the feature mapping $\phi(x) = [x, x^2]^T$, write down the 3×3 kernel/Gram matrix $\mathbf{K}$ for this dataset.

$\phi_\mathbf{1}(x) = [-1, 1]^T$
$\phi_\mathbf{2}(x) = [1, 1]^T$
$\phi_\mathbf{3}(x) = [0, 0]^T$

$$\mathbf{K} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

1.4 (4pts) Now write down the primal and dual formulations of SVM for this dataset in the two-dimensional feature space. Note that when the data is separable, we set the hyperparameter C to be $+\infty$ which makes sure that all slack variables (ξ) in the primal formulation have to be 0 (and thus can be removed from the optimization).

Primal formation (separable): $min_{w,b} \frac{1}{2}\|w\|_2^2$ s.t. $y_i(w^T\phi(x_i) + b) \geq 1, \forall i \in [1,3]$
Dual formation (separable): $max_{\alpha_i} \sum_{i=1}^{3} \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j)$ s.t. $\sum_{i=1}^{3} \alpha_i y_i = 0$ and $\alpha_i \geq 0, \forall i \in [1,3]$

1.5 (5pts) Next, solve the dual formulation exactly (note: while this is not generally feasible, the simple form of this dataset makes it possible). Based on that, calculate the primal solution.

$$\begin{aligned} F(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(&y_1 y_1 \alpha_1 \alpha_1 k(x_1, x_1) + y_1 y_2 \alpha_1 \alpha_2 k(x_1, x_2) + y_1 y_3 \alpha_1 \alpha_3 k(x_1, x_3) \\ &+ y_2 y_1 \alpha_2 \alpha_1 k(x_2, x_1) + y_2 y_2 \alpha_2 \alpha_2 k(x_2, x_2) + y_2 y_3 \alpha_2 \alpha_3 k(x_2, x_3) \\ &+ y_3 y_1 \alpha_3 \alpha_1 k(x_3, x_1) + y_3 y_2 \alpha_3 \alpha_2 k(x_3, x_2) + y_3 y_3 \alpha_3 \alpha_3 k(x_3, x_3)) \\ &= \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}(2\alpha_1^2 + 2\alpha_2^2) \\ &= \alpha_1 + \alpha_2 + \alpha_3 - \alpha_1^2 - \alpha_2^2 \end{aligned}$$

Since $\sum_{i=1}^{3} \alpha_i y_i = 0$ and $\alpha_i \geq 0, \forall i \in [1,3]$, $\alpha_3 = \alpha_1 + \alpha_2$.

$$= 2\alpha_1 + 2\alpha_2 - \alpha_1^2 - \alpha_2^2 = (2\alpha_1 - \alpha_1^2) + (2\alpha_2 - \alpha_2^2)$$

To maximize $F(\alpha)$, we take the derivative w.r.t α_1 and α_2.

$$\begin{aligned} \frac{dF}{d\alpha_1} = 2 - 2\alpha_1 = 0 \Rightarrow \alpha_1 = 1 \\ \frac{dF}{d\alpha_2} = 2 - 2\alpha_2 = 0 \Rightarrow \alpha_2 = 1 \\ \Rightarrow \alpha_3 = 2 \end{aligned}$$

The values that maximize $F(\alpha)$ are $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 2$. We then calculate the primal situation by finding values $\mathbf{w}$ and b. To find the those values for the primal solution:

$$\begin{aligned} w = \sum_i \alpha_i y_i \phi_(x_i) = \alpha_1 y_1 \phi_(x_1) + \alpha_2 y_2 \phi_(x_2) + \alpha_3 y_3 \phi_(x_3) \\ = -\begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \\ b = y_i - w \cdot \phi(x_i) = -1 + 2 = 1 \end{aligned}$$

1.6 (3pts) Plot the decision boundary (which is a line) of the linear model $\mathbf{w}^{*T}\mathbf{x} + b^*$ in the two-dimension3al feature space, where $\mathbf{w}^*$ and b^* are the primal solution you got from the previous question. Then circle all support vectors. Finally, plot the corresponding decision boundary in the original one-dimensional space (recall that the decision boundary is just the set of all points x such that $\mathbf{w}^{*T}\phi(x) + b^* = 0$).

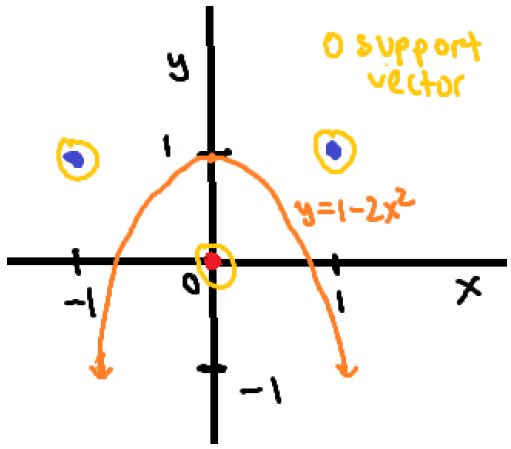


Figure 1: Decision boundary in two-dimensional space

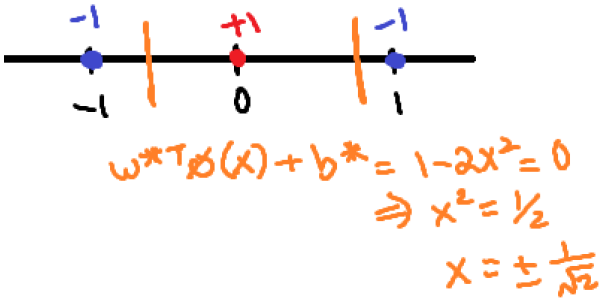


Figure 2: Decision boundary in one-dimensional space

Problem 2: Kernel Composition (6pts)

Prove that if $k_1, k_2 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ are both kernel functions, then $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$ is a kernel function too. Specifically, suppose that $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are the corresponding mappings for k_1 and k_2 respectively. Construct the mapping $\boldsymbol{\phi}$ that certifies k being a kernel function.

Since k_1 and k_2 are kernel functions, they satisfy the following for some $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$:

$$
\begin{aligned}
k_1(\mathbf{x}, \mathbf{x}') &= \phi_1(\mathbf{x})^T \phi_1(\mathbf{x}') \\
k_2(\mathbf{x}, \mathbf{x}') &= \phi_2(\mathbf{x})^T \phi_2(\mathbf{x}') \\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^T \phi_1(\mathbf{x}')\phi_2(\mathbf{x})^T \phi_2(\mathbf{x}') \\
&= (\sum_{i=1}^{d} \phi_1(x)_i \phi_1(x')_i)(\sum_{j=1}^{d} \phi_2(x)_j \phi_2(x')_j) \\
&= \sum_{i=1}^{d} \phi_1(x)_i \phi_1(x')_i \phi_2(x)_i \phi_2(x')_i \\
&= \sum_{i=1}^{d} (\phi_1(x)_i \phi_2(x)_i)(\phi_1(x')_i \phi_2(x')_i)
\end{aligned}
$$

Let $\phi'(x) = \phi_1(x)\phi_2(x)$ for which $\phi' : \mathbb{R}^d \to \mathbb{R}^M$ since $(\mathbb{R}^d \to \mathbb{R}^M) * (\mathbb{R}^d \to \mathbb{R}^M) \to \mathbb{R}^M$. Thus:

$$
k(\mathbf{x}, \mathbf{x}') = \phi'(\mathbf{x})^T \phi'(\mathbf{x}')
$$

Since we can write $k(x, x')$ as the inner product of a function $\phi' : \mathbb{R}^d \to \mathbb{R}^M$, it satisfies the definition of a kernel function. ϕ' is thus the mapping that certifies k being a kernel function.

Programming-based Questions

Problem 3: Regularization (31pts)

3.1 (2pts) We will first setup a baseline, by finding the test error of the linear regression solution $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$ without any regularization. This is the closed-form solution for the minimizer of the objective function $f(\mathbf{w})$. (Note the formula is simpler than what we saw in the last homework because now $\mathbf{X}$ is square as $d = n$). Report the training error and test error of this approach, *averaged over 10 trials*. For better interpretability, report the normalized error $\hat{f}(\mathbf{w})$ rather than the value of the objective function $f(\mathbf{w})$, where we define $\hat{f}(\mathbf{w})$ as

$$\hat{f}(\mathbf{w}) = \frac{\| \mathbf{X}\mathbf{w} - \mathbf{y} \|_2}{\| \mathbf{y} \|_2}.$$

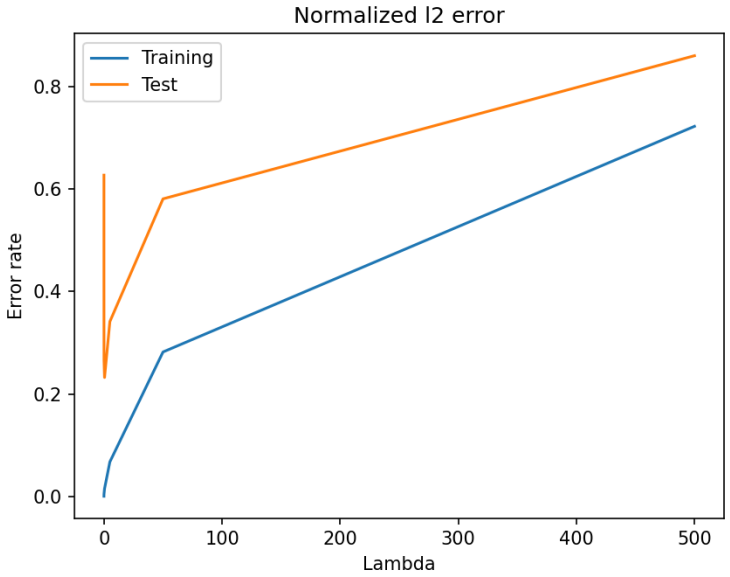
Note on averaging over multiple trials: We're doing this to get a better estimate of the performance of the algorithm. To do this, simply run the entire process (including data generation) 10 times, and find the average value of $\hat{f}(\mathbf{w})$ over these 10 trials.

Average training error over 10 trials: 2.4941836515287468e-14
Average testing error over 10 trials: 0.9708595499259829

3.2 (7pts) We will now examine ℓ_2 regularization as a means to prevent overfitting. The ℓ_2 regularized objective function is given by the following expression:

$$\sum_{i=1}^{m} (\mathbf{w}^{\mathbf{T}}\mathbf{x_i} - y_i)^2 + \lambda \|\mathbf{w}\|_2^2.$$

As discussed in class, this has a closed-form solution $\mathbf{w} = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$. Using this closed-form solution, present a plot of the normalized training error and normalized test error $\hat{f}(\mathbf{w})$ for $\lambda = \{0.0005, 0.005, 0.05, 0.5, 5, 50, 500\}$. As before, you should average over 10 trials. Discuss the characteristics of your plot, and also compare it to your answer to (**3.1**).



Both graphs exhibit the general trend of increasing error as the value of λ increases. However, the training error spikes up when $\lambda = 50$ and increases a little more until $\lambda = 500$. Overall, the training error is greater than the test error for all given λ. The l2 regularized training error values are overall greater than (3.1) due to the extra regularization factor, but it reduces overfitting, as shown by the lower test error rate (< 0.97) for all

given values of λ.

3.3 (7pts) Run stochastic gradient descent (SGD) on the original objective function $f(\mathbf{w})$, with the initial guess of $\mathbf{w}$ set to be the all 0's vector. Run SGD for 1,000,000 iterations for each different choice of the step size, $\{0.00005, 0.0005, 0.005\}$. Report the normalized training error and the normalized test error for each of these three settings, averaged over 10 trials. How does the SGD solution compare with the solutions obtained using ℓ_2 regularization? Note that SGD is minimizing the original objective function, which does *not* have any regularization. In Part (**3.1**) of this problem, we found the *optimal* solution to the original objective function with respect to the training data. How does the training and test error of the SGD solutions compare with those of the solution in (**3.1**)? Can you explain your observations? (It may be helpful to also compute the normalized training and test error corresponding to the true coefficient vector $\mathbf{w}^*$, for comparison.)

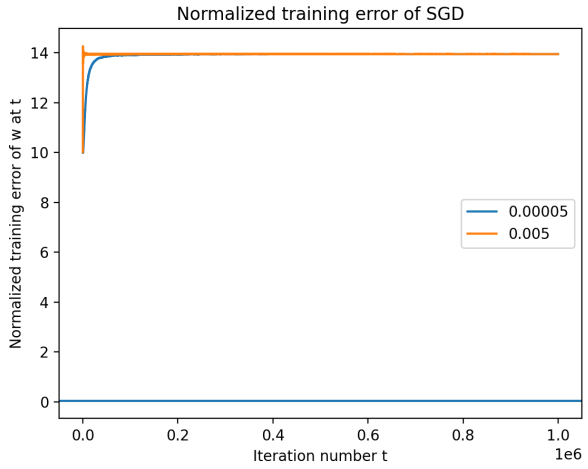
Normalized train error over 10 trials:
step size 0.00005: 14.107285667458859
step size 0.0005: 14.108981186618824
step size 0.005: 14.110961092682606

Normalized test error over 10 trials:
step size 0.00005: 43.80886642508211
step size 0.0005: 44.58075995018656
step size 0.005: 46.20769350072017

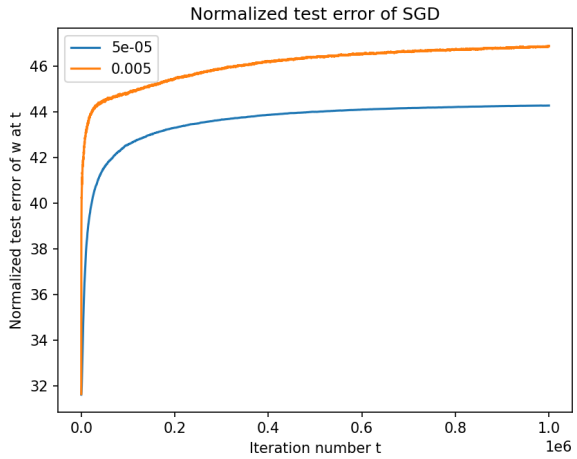
The SGD solution has larger training and test error compared to l2 regularization, since it doesn't regularize to reduce overfitting and also has stochasticity since it randomly chooses a data point. It also has higher training and test error than (3.1), since the stochasticity results in unexpected values which may not match up with simply completing least squares.

3.4 (10pts) We will now examine the behavior of SGD in more detail. For step sizes $\{0.00005, 0.005\}$ and 1,000,000 iterations of SGD,

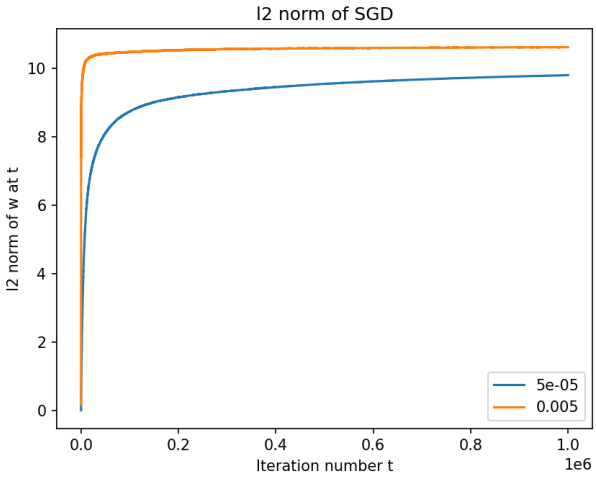
(i) Plot the normalized training error vs. the iteration number. On the plot of training error, draw a line parallel to the x-axis indicating the error $\hat{f}(\mathbf{w}^*)$ of the true model $\mathbf{w}^*$.



(ii) Plot the normalized test error vs. the iteration number. Your code might take a long time to run if you compute the test error after every SGD step—feel free to compute the test error every 100 iterations of SGD to make the plots.



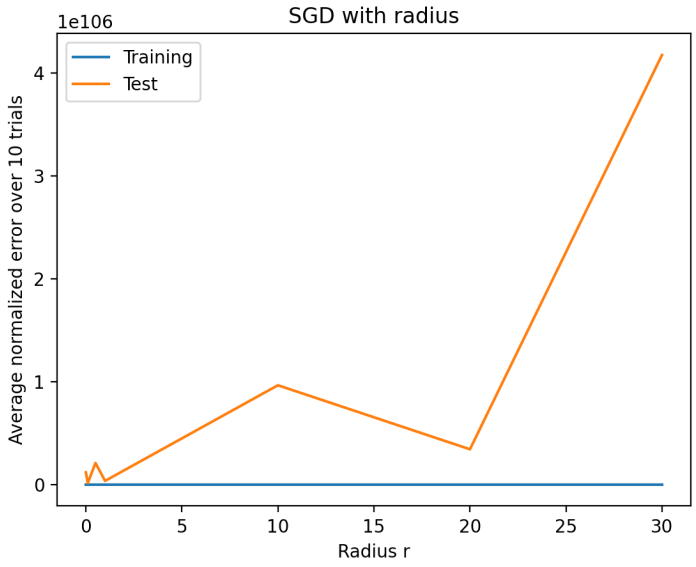
(iii) Plot the ℓ_2 norm of the SGD solution vs. the iteration number.



Comment on the plots. What can you say about the generalization ability of SGD with different step sizes? Does the plot correspond to the intuition that a learning algorithm starts to overfit when the training error becomes too small, i.e. smaller than the noise level of the true model so that the model is fitting the noise in the data? How does the generalization ability of the final solution depend on the ℓ_2 norm of the final solution?

The generalization ability of SGD seems to be better with a lower stepsize (0.00005) with a corresponding smaller test error than a larger step size (0.005), as shown by (ii). Yes, the plots correspond to the intuition that overfitting happens when training error is small, since the smaller training error in (i) corresponds with a larger test error due to overfitting in (ii). A smaller l2 norm of the final solution corresponds with better generalization ability, as shown by the results of step size 0.00005 in (iii).

3.5 (5pts) We will now examine the effect of the starting point on the SGD solution. Fixing the step size at 0.00005 and the maximum number of iterations at 1,000,000, choose the initial point randomly from the d-dimensional sphere with radius $r = \{0, 0.1, 0.5, 1, 10, 20, 30\}$ (to do this random initialization, you can sample from the standard Gaussian $N(0, \mathbf{I})$, and then renormalize the sampled point to have ℓ_2 norm r). Plot the average normalized training error and the average normalized test error over 10 trials vs r. Comment on the results, in relation to the results from part (**3.2**) where you explored different ℓ_2 regularization coefficients. Can you provide an explanation for the behavior seen in this plot?

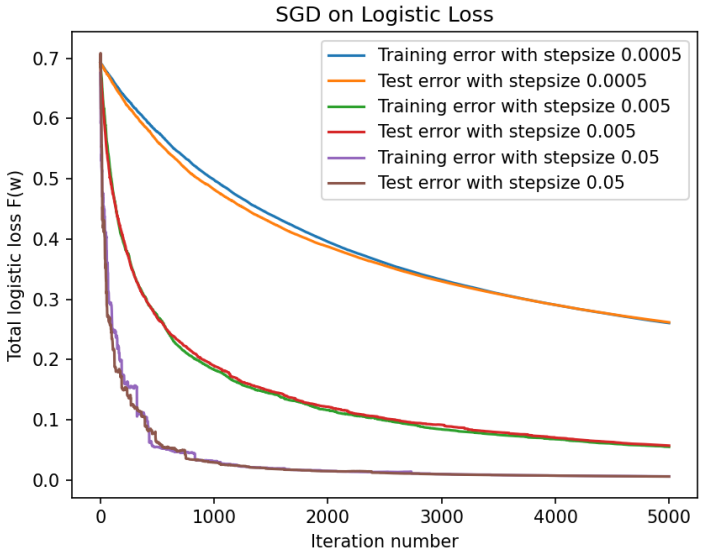


There seems to be zero to no training error, but the test error spikes up exponentially as the radius increases and the stochasticity of the starting point potentially impacting the error as well. This corresponds with (3.2) with the greater test error than training error, and greater error in general when λ controlling regularization is greater. Similar to λ, as r increases, the error also increases on the SGD solution.

Deliverables for Problem 3: Code for the previous parts as a separate Python file `Q3.py`. Training and test error for part 3.1. Plots for part 3.2, 3.4 and 3.5. Training and test error for different step sizes for part 3.3. Explanation for parts 3.2, 3.3, 3.4, 3.5.

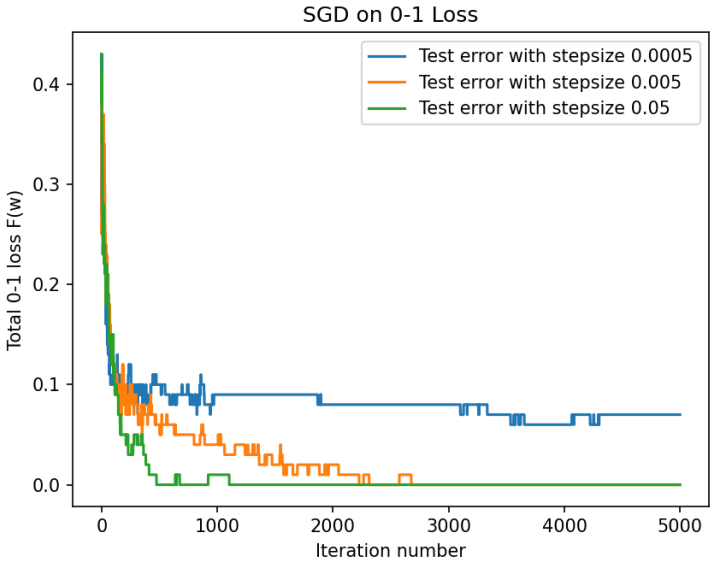
Problem 4: Logistic Regression (11 pts)

4.1 (7pts) As the algorithm proceeds, compute the value of the objective function on the train and test data at each iteration. Plot the objective function value on the training data vs. the iteration number for all 3 step sizes. On the same graph, plot the objective function value on the test data vs. the iteration number for all 3 step sizes. (The deliverable is a single graph with 6 lines and a suitable legend). How do the objective function values on the train and test data relate with each other for different step sizes? Comment in 3-4 sentences.



The objective function values of training and test data are very close as shown by the graph, with training error appearing to be on par with test error. Though it seems that test error seems to increase over training error gradually and converge to similar values to the training error, if not very slightly larger. A smaller stepsize (0.0005) gives greater error compared to a larger stepsize (0.05) in this case, as the larger stepsize manages to converge to a lower minimum faster than the smaller stepsize can in the 5000 iterations given.

4.2 (2pts) So far in the problem, we've minimized the logistic loss on the data. However, remember from class that our actual goal with binary classification is to minimize the classification error given by the 0/1 loss, and logistic loss is just a surrogate loss function we work with. We will examine the average 0-1 loss on the test data in this part (note that the average 0-1 loss on the test data is just the fraction of test datapoints that are classified incorrectly). As the SGD algorithm proceeds, plot the average 0-1 loss on the test data vs. the iteration number for all 3 step sizes on the same graph. Also report the step size that had the lowest final 0-1 loss on the test set and the corresponding value of the 0-1 loss.



Both stepsizes 0.05 and 0.005 had the lowest final 0-1 loss of 0 on the test error.

4.3 (2pts) Comment on how well the logistic loss act as a surrogate for the 0-1 loss.

The logistic loss acts as a pretty good surrogate as it showed the relationship between a lower test error for 5000 iterations with a lower stepsize since it can converge faster than a smaller stepsize. Its results are similar to the results from the 0-1 loss function, but it is convex and thus can be easily optimized, compared to the nonconvex NP-hard properites of using 0-1 loss by itself.

Deliverables for Problem 4: Code for the previous parts as a separate Python file `Q4.py`. Plot (with 6 lines) and associated discussion for 4.1. Plot (with 3 lines) and associated discussion for 4.2. Discussion for 4.3.

Problem 5: Classifier Comparison (8 pts)

5.1 (2pts) Notice that MOON and CIRCLES are not linearly separable. Do linear classifiers do well on these? How does SVM with RBF kernel do on these? Comment on the difference.

Overall, the linear classifiers don't do as well as SVM with RBF kernel. Since the points on the MOON are still somewhat cohesive, linear classifiers end up with accuracies of 0.88 and 0.9 with C=0.025 and 0.25 respectively

on test data. However, SVM with RBF does even better with 0.95 and 0.97 with C=0.25 and 1 respectively on test data for MOON points. Points on CIRCLES are not so easily classified, so both linear classifiers don't do well with accuracies of 0.4, while SVM with RBF does better with 0.68 and 0.88 with C=0.25 and 1 respectively.

5.2 (2pts) Try various values of the penalty term C for the SVM with linear kernel. On the LINEARLY_SEPARABLE dataset, how does the train and test set accuracy relate to C? On the LINEARLY_SEPARABLE dataset, how does the decision boundary change?

For the linear kernel, it seems that larger values of C will result in greater train and test accuracy overall on the LINEARLY_SEPARABLE dataset, since a larger C value will penalize misclassifications more severely. However, since a larger C may overfit to training data and has less regularization, the training accuracy may be higher than the test accuracy. With C=0.025, accuracies are 0.8 (training) and 0.85 (test), and with C=0.25, accuracies are 0.93 (training) and 0.9 (test). The decision boundary changes by fitting more to the training data with a higher C value, which works in favor of test sets similar to the training set on a linearly separable dataset.

5.3 (2pts) Try various values of the penalty term C for the SVM with RBF kernel. How does the train and test set accuracy relate to C? How does the decision boundary change?

For the LINEARLY_SEPARABLE set, it seems that a lower value of C=0.25 with more regularization gives better accuracies of 0.97 (train) and 0.9 (test) compared to C=1 with 0.93 (train) and 0.85 (test) for SVM with RBF kernel. For linearly inseparable sets, having a penalty term C greatly increases the train and test accuracies since it is more difficult to separate the dataset with a linear classifier. For the CIRCLE dataset, having C=1 gives 0.97 (train) and 0.88 (test), which are much better than results from C=0.25 giving 0.87 (train) and 0.68 (test). The decision boundary changes by no longer overfitting to the training data, allowing the classifier to generalize much better to test data.

5.4 (2pts) Try various values of C for Logistic Regression (Note: C is the inverse regularization strength). Do you see any effect of regularization strength on Logistic Regression? Hint: Under what circumstances do you expect regularization to affect the behavior of a Logistic Regression classifier?

It does not seem to have any affect on classification accuracies for logistic regression on the linearly inseparable datasets with small values of C. However, it does have a slight increase in test accuracy for a larger value of C. Regularization should impact logistic regression classification by minimizing the effects of overfitting to the training data and thus increasing generalization to unseen test data, otherwise the training accuracy will generally be larger than the test accuracy.