CSCI 567 Spring 2024, Exam 1

Instructor: Vatsal Sharan

Time limit: 2 hour and 20 minutes

Problem	1	2	3	4	5	6
Max	21	13	8	8	10	14

1 Multiple choice questions (21 points)

IMPORTANT: *Select ALL options among {A,B,C,D} that you think are correct (no justification needed). You get 0.5 point for selecting each correct option and similarly 0.5 point for not selecting each incorrect option. You get 1 additional point for selecting all four options correctly.*

(1) Which of the following statements are true about gradient descent (GD)?

(A) Increasing the step size/learning rate of GD may help GD converge faster.
(B) Increasing the step size/learning rate of GD may cause it to not converge at all.
(C) GD (with a suitable step size) will converge to an approximate stationary point, even for non-convex functions.
(D) GD can be used to solve logistic regression.

A B C D

(2) Consider a linear model with 100 input features, out of which 10 are highly informative about the label and 90 are non-informative about the label. Assume that all features have values between -1 and 1. Which of the following statements are true?

(A) ℓ_1 regularization will encourage most of the non-informative weights to be exactly 0.0.
(B) ℓ_1 regularization will encourage most of the non-informative weights to be nearly (but not exactly) 0.0.
(C) ℓ_2 regularization will encourage most of the non-informative weights to be exactly 0.0.
(D) ℓ_2 regularization will encourage most of the non-informative weights to be nearly (but not exactly) 0.0.

A D

(3) Which of the following options will decrease the generalization gap (difference between test error and training error) of a machine learning model?
(A) Use more data to learn the model.
(B) Add ℓ_2 regularization on the parameters when learning the model.
(C) Consider a more complex model class, which is a super-set of the original function class.
(D) Simplify the model by reducing its complexity.

A B D

(4) Which of the following statements about kernel methods is correct?
(A) Kernel methods are only applicable to linearly separable datasets.
(B) We need to save the kernel matrix for the training dataset in order to make predictions on test points using kernel methods.
(C) Kernel methods are most useful when the primary objective is to reduce overfitting by simplifying the decision boundary in the original feature space.
(D) Kernel methods allow operating in a high-dimensional feature space without explicitly computing the feature vectors in that space.

D

(5) Which of the following is a kernel function?
(A) $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$
(B) $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^\top \boldsymbol{x}' + 2)^2$
(C) $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x} + \boldsymbol{x}'^\top \boldsymbol{x}'$
(D) $k(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2$

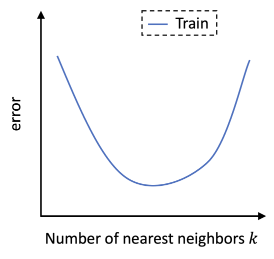
A B

(6) Which of the following are true statements about supervised learning?

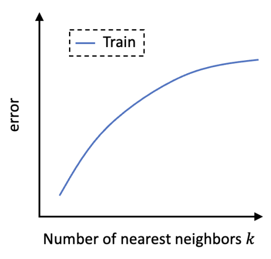
(A) The test set should not be used to train the model, but can be used to tune hyper-parameters.
(B) The generalization gap (difference between test and training errors) generally decreases as the size of the training set increases.
(C) We cannot estimate the risk of a predictor (its average error on the data distribution) solely with the data used to train it.
(D) If training and test data are drawn from different distributions, then low error on the training set may not guarantee low error on the test set even if the size of the training set is sufficiently large.

B C D

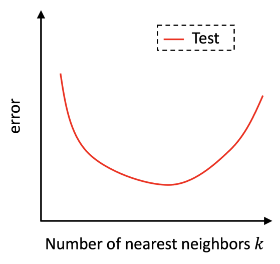
(7) Figure 1 shows various training/test classification error curves as parameters for training k-nearest neighbors are varied. Select all options which represent reasonable relationships between the considered parameters and obtained error(s).



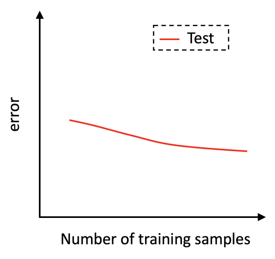
(a)



(b)



(c)



(d)

Figure 1: Possible curves for k-nearest neighbors (select all which are reasonable)

(A) Fig 1a is a reasonable plot of training error as the number of nearest neighbors k used to make a prediction is increased.
(B) Fig 1b is a reasonable plot of training error as the number of nearest neighbors k used to make a prediction is increased.

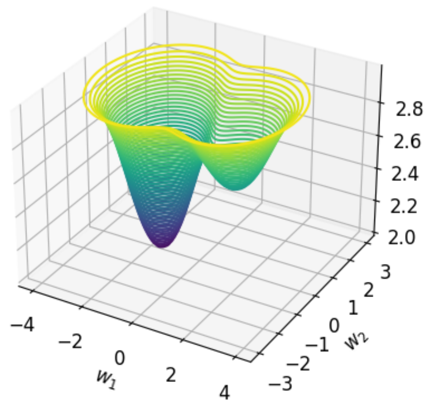
(C) Fig 1c is a reasonable plot of test error as the number of nearest neighbors k used to make a prediction is increased.
(D) Fig 1d is a reasonable plot of test error as the number of training datapoints used to train a k-NN model with $k = 5$ is increased.

B C D

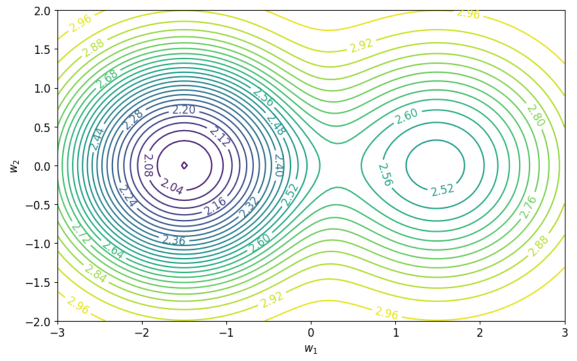
2 Short answer questions (13 points)

2.1 Gradient descent (6 points)

Consider the function depicted in Figs. 2a and 2b. On the left we have a 3-d plot of the function, on the right we have a contour plot of the same function



(a)



(b)

Figure 2: A 3-d plot and contour plot of the same function.

(a) Is the function convex? Explain. How many local minima does the function have? (2 points)

No! One geometric reasoning—we can draw a line between e.g. the two minima, and (some) points on that line will be below the curve. This violates the definition of convexity. The curve has two local minima—and also one saddle point!

Rubric:

- Not convex with correct explanation (+1.5 points)
- 2 local minimas (Note: global minimas are also local minimas but saddle points are not) (+0.5 points)

(b) Suppose we use gradient descent to minimize the function. Will gradient descent always find the global minimizer of the function? Explain. (2 points)

No! Gradient descent only uses local information. The minimum we find will depend on where we initialize our search (i.e. in which basin). The false minimum will be found if we start our GD algorithm closer to that local minimum, because that's where the gradients will lead us. In SGD, we might get lucky with the stochasticity, but GD is deterministic so initialization is the only factor that determines where we end up.

Rubric:

- Gradient descent is not guarenteed to converge to global minimizer (1 point)
- Correct explanation mentioning multiple stationary points and initialization (1 point)

(c) Fig 3 shows the iterates of some run of gradient descent. Comment on the behavior of gradient descent seen in this plot, and suggest how you can improve the convergence. (2 points)

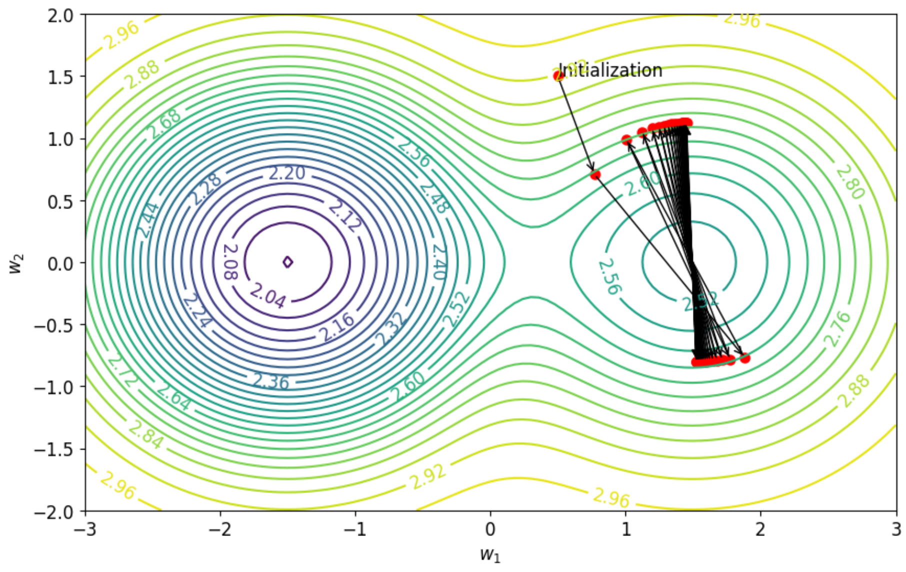


Figure 3: Gradient descent iterates denotes from some initialization denotes with red dots

Oof—some ugly non-convergent behavior. We can observe that the gradients cause a kind of oscillation between two (slowly shifting) points in our loss landscape, across the local minimum. In order to improve the convergence, we'll want to decrease the step size/learning rate parameter. Additionally, if we want to increase our likelihood of discovering the global minimizer, some stochasticity might be in order!

Rubric:

- Mention gradient descent may not converge because of oscillation across local minimum or gradient descent finding local min instead of global min. (1 point)
- Mention decreasing step size/learning rate to improve convergence or adding stochasticity to find global minimum. (1 point)

2.2 Kernels on strings (4 points)

We consider a kernel function defined over *strings* in this problem. A string is just a sequence of characters, and in this question we will assume that it only consists of lower-case letters. Given any two strings s_1 and s_2, define the kernel

$$k(s_1, s_2) = \mid \{\text{all lower-case letters which appear at least once in both } s_1 \text{ and } s_2\} \mid .$$

Informally, $k(s_1, s_2)$ just counts the number of letters which occur in both the strings ($| \cdot |$ here denotes the cardinality of a set). As an example $k(\text{'machine'},\text{'learning'}) = 4$ since the letters a, e, i, n appear in both the words. Find an explicit feature transformation $\phi(s)$ such that $k(s_1, s_2) = \phi(s_1)^T \phi(s_2)$.

We can define $\phi(s)$ as a 26-d vector where each element i corresponds to a character c_i, and indicates whether or not c_i exists in s. When we take the dot product $\phi(s_1)^T \phi(s_2)$, the only elements that will be counted in the sum are characters c_i where $\phi(s_1)_i = \phi(s_2)_i = 1$; in other words, characters that appear in both strings.

Rubric:

- Give the correct transformation (4 points)
- If the intuition is correct but the transformation is wrong at the end (2 points)
- Wrong transformation and intuition (0 points)

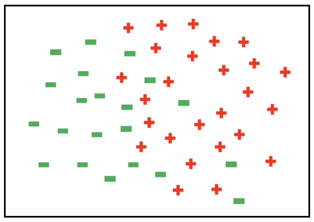
2.3 Linear classification (3 points)

Consider the following program which learns a linear classifier $\boldsymbol{w} \in \mathbb{R}^d$ based on training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$ by taking repeated passes over the data.

Algorithm 1: Linear classifier

```
Input  : A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$
Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$;
Initialize not-converged ← True;
while not-converged do
    Set not-converged ← False
    for $i$ in $\{1, \ldots, n\}$ do
        Make a prediction $\hat{y} = \text{SIGN}(\boldsymbol{w}^T \boldsymbol{x}_i)$ using $\boldsymbol{w}$
        if $\hat{y} \neq y_i$ then
            $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i \boldsymbol{x}_i$
            not-converged ← True
```

Explain why the above algorithm will never terminate on the training dataset in Fig. 4a but will terminate on the dataset in Fig. 4b.


(a)

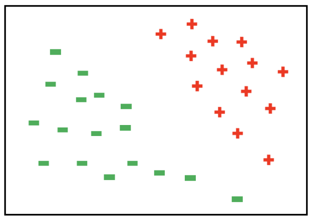

(b)

Figure 4: Two binary classification datasets, where red, plus signs denote label +1, and the green, minus signs denote the label -1.

The algorithm will not terminate on the training data in Fig. 5a because the data is not linearly separable. The algorithm only converges when it arrives at a state where all points are being classified correctly. Since this is impossible when the data is not linearly separable, in each epoch, there will be at least one point (and in Fig. 5a's case, multiple), where updates are required.

Fig. 5b, on the other hand, *is* linearly separable. We know from class that the Perceptron update rule leads to convergence on linearly separable data. No need to provide justification beyond this citation—students are not required to know Perceptron convergence proof.

Rubric:

- Explain that figure (a) is not linearly separable (1.5 points)
- Explain that figure (b) is linearly separable (1.5 points)

3 A machine learning competition (8 points)

Your friend Bob is working on on a machine learning competition on Kaggle, and you are advising him based on your new found machine learning expertise from CSCI567. The competition is a binary classification task, and has a training dataset with n datapoints $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \{\pm 1\}$. You suspect the data may be linearly separable, and advise Bob to train a linear model.

(a) Bob has decided on the following objective function to find the linear predictor $\boldsymbol{w}$:

$$F(\boldsymbol{w}) = \sum_{i=1}^{n} (\textsc{sign}(\boldsymbol{w}^T \boldsymbol{x}_i) - y_i)^2,$$

where $\textsc{sign}(\cdot)$ denotes the sign function. Bob now wants to minimize $F(\boldsymbol{w})$ using stochastic gradient descent (SGD). Your 567 training tells you this is not a good idea. Explain to Bob why it will be difficult to minimize $F(\boldsymbol{w})$ using SGD. (2 points)

The sign function is not differentiable at 0 and has a derivative of 0 everywhere else. Therefore, the model parameters cannot be updated by SGD.

Rubrics:

- Correct / Mentioned discontinuity (2 points)
- Did not mention that $\textsc{sign}$ function is not differentiable at (only) 0 and has a derivative of 0 everywhere else, or at least recognize that the derivative carries no information (-1 points)
- Did not mention that model parameters can therefore not be updated by SGD due to such properties of its derivative – meaning there is no training (-1 points)

Addendum:

- Mentioned non-convexity (1 point)
- Mentioned only that the gradient cannot be computed / difficult to compute / non-differentiable (0.5 points)

(b) You advise Bob to use the hinge loss $\ell_{hinge}(y\boldsymbol{w}^T x) = \max(1 - y\boldsymbol{w}^T x, 0)$ to learn a linear predictor. Bob follows your advice, and writes the following code to find the gradient with respect to a point.

Algorithm 2: Bob's code to find the gradient for datapoint $(\boldsymbol{x}, y)$

```
Input   : Datapoint $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{\pm 1\}$, current iterate $\boldsymbol{w}$
if $y\boldsymbol{w}^T\boldsymbol{x} > 1$ then
    grad = 0;
end
else
    for $1 \le j \le d$ do
        grad[$j$] = $-y * \boldsymbol{x}[j]$;
        /* Here $\boldsymbol{x}[j]$ denotes the $j$-th coordinate of input $\boldsymbol{x}$    */
    end
end
Output: grad
```

You know that the code will be more simple and computationally efficient if Bob uses vector notation instead of a for loop to iterate over all d coordinates. Write the gradient in vector notation in order to substitute the for loop. (2 points)

If $y\boldsymbol{w}^T\boldsymbol{x} > 1$, then $\texttt{grad} = \mathbf{0}$, where $\mathbf{0}$ is the zero vector of dimension d.
Otherwise, $\texttt{grad} = -y\boldsymbol{x}$.

(c) Bob is not getting high accuracy with a linear predictor, so you advise him to use feature transformation $\phi(\boldsymbol{x})$. Bob suggests using the transformation $\phi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ to every datapoint $\boldsymbol{x}$ for some suitably chosen matrix $\boldsymbol{A} \in \mathbb{R}^{m \times d}$, and then train a linear classifier in the transformed space. Explain to Bob why this transformation will not help at all. (2 points)

The transformation $\phi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ is a linear operation. Linear transformations can rotate, scale, or translate the dataset, but they cannot change the fundamental nature of the data's linear separability. If the data is linearly inseparable in the original space, simply applying a linear transformation will not make it linearly separable.

Rubrics:

- indicating that $\phi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ is a linear operation (1 point)
- explaining linear transform doesn't change linear separability of the data (1 point)
- stating the reason is not increasing the dimension of the data (-1 point)

(d) Based on your advice, Bob is now trying various kernels to fit his data. A polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T\boldsymbol{x}' + c)^m$ seems to be working well, and Bob is picking the degree of the polynomial kernel m based on the value which gets the highest accuracy on the training set. Explain to Bob why this is not good machine learning practice, and why this might cause him to do poorly in his competition. (2 points)

By choosing m solely based on training set accuracy, Bob risks creating a model that is overly complex. A higher-degree polynomial kernel can capture more complex patterns and interactions between features, but it also increases the model's capacity to fit noise in the training data. This can lead to overfitting, where the model performs exceptionally well on the training data but poorly on new data. Thus, having low scores on the Kaggle competition. Since we want ML models perform well on test sets, determining model parameters on training set is not a good ML practice.

Extra (not expected as answer): Bob can split the available data into a training set and a validation set (or use cross-validation techniques) to evaluate the performance of models with different values of m. This helps in selecting a model that performs well on both the training and validation sets, indicating better generalization ability.

Rubrics:

- explaining overfitting might happen (1 point)
- explaining why we avoid overfitting in ML (1 point)

4 Modified logistic regression (8 points)

In class, we defined the logistic loss for a linear predictor $\boldsymbol{w}$ on a labeled datapoint $(\boldsymbol{x}, y)$ as follows,

$$\ell_{log}(\boldsymbol{w}, \boldsymbol{x}, y) = \log(1 + \exp(-y\boldsymbol{w}^T\boldsymbol{x})).$$

Consider the following variation of the logistic loss,

$$\ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}, y) = \begin{cases} \log(1 + \exp(-\boldsymbol{w}^T\boldsymbol{x})) & \text{if } y = 1, \\ 0.01 \log(1 + \exp(\boldsymbol{w}^T\boldsymbol{x})) & \text{if } y = -1. \end{cases}$$

(a) Explain how this modified logistic loss $\ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}, y)$ would differ from the original logistic loss $\ell_{log}(\boldsymbol{w}, \boldsymbol{x}, y)$. (2 points)

We can see that the new logistic loss adds a scalar to reduce the penalty to the negative label data points.

(b) Consider the binary classification dataset of points in two dimensions in Fig. 5. Here the red, plus signs denote the label +1, and the green, minus signs denote the label -1.

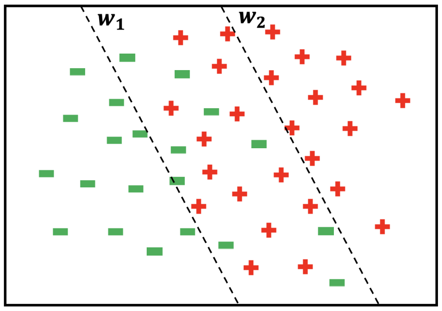


Figure 5: Binary classification dataset

If we train a linear predictor on this data using $\ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}, y)$, then which of $\boldsymbol{w}_1$ or $\boldsymbol{w}_2$ is more likely to be the decision boundary of the linear classifier? Explain. (2 points)

$\boldsymbol{w}_1$. First note that the dataset here is not linearly separable so any linear classifier will make mistakes. Next, the new loss function does not penalize classifying negative points as positive as heavily as it penalizes classifying positive points as negative. Therefore $\boldsymbol{w}_1$ is preferred to $\boldsymbol{w}_2$.

(c) Given a dataset $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, we will use $\ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}, y)$ to learn a linear model $\boldsymbol{w} \in \mathbb{R}^d$. To do this, we can minimize the empirical risk given by $F(\boldsymbol{w})$,

$$F(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}_i, y_i). \tag{1}$$

For a fixed i, write down the gradient $\nabla \ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}_i, y_i)$ of $\ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}_i, y_i)$ with respect to $\boldsymbol{w}$ (show your derivation), then fill in the missing details in the while-loop of the algorithm below which applies SGD to minimize F. (4 points)

The SGD is similar to the update that was given in class. The loss function is the updated version with the scalar when it is negative data.

$$\nabla \ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}_i, y_i) = \begin{cases} -\boldsymbol{x}_i * \frac{\exp(-\boldsymbol{w}^T\boldsymbol{x})}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}_i)} & \text{if } y_i = 1, \\ 0.01 * \boldsymbol{x}_i * \frac{\exp(\boldsymbol{w}^T\boldsymbol{x})}{1+\exp(\boldsymbol{w}^T\boldsymbol{x}_i)} & \text{if } y_i = -1. \end{cases}$$

Algorithm 3: SGD for minimizing Eq. (1)

```
Input : A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, learning rate $\eta > 0$, number of
        iterations $T$
Initialization: $\boldsymbol{w} = \boldsymbol{0}$;
while $T$ iterations are not complete do
    Randomly pick an example $(\boldsymbol{x}_i, y_i)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla \ell_{new-log}(\boldsymbol{w}, \boldsymbol{x}_i, y_i)$
end
```

5 Linear regression with non-uniform regularization (10 points)

Consider a modification of the standard linear regression setup where we have a different regularization penalty on different coordinates of the predictor. Formally, we consider the problem of finding a linear predictor $\boldsymbol{w} \in \mathbb{R}^d$ using a dataset of n datapoints $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, with the following regularized objective,

$$\begin{aligned} G(\boldsymbol{w}) &= \sum_{i=1}^{n} \left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - y_i\right)^2 + \lambda_1 \sum_{j=1}^{d/2} w_j^2 + \lambda_2 \sum_{j=d/2+1}^{d} w_j^2 \\ &= \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda_1 \sum_{j=1}^{d/2} w_j^2 + \lambda_2 \sum_{j=d/2+1}^{d} w_j^2. \end{aligned}$$

Here w_j is the j-th coordinate of the predictor $\boldsymbol{w}$, $\boldsymbol{X}$ is the $n \times d$ matrix whose i-th row is $\boldsymbol{x}_i^{\mathrm{T}}$, $\boldsymbol{y}$ be the n-dimensional column vector whose i-th coordinate is y_i, and $\lambda_1, \lambda_2 > 0$.

(a) Suppose we want to encourage the model to have smaller values in the first $d/2$ coordinates of $\boldsymbol{w}$ compared to the the last $d/2$ coordinates. Which of these would be a suitable relationship between λ_1 and λ_2 to achieve this? (a) $\lambda_1 > \lambda_2$, (b) $\lambda_2 > \lambda_1$. Explain in 1-2 sentences. (2 points)

(a) $\lambda_1 > \lambda_2$ To get the first $d/2$ coordinates to be smaller than the last half, we want to regularize the first $d/2$ coordinates of $\boldsymbol{w}$ more heavily by using a larger regularization coefficient. So λ_1 should be greater than λ_2.

(b) For some diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{d \times d}$, show that the objective can be written as follows in matrix form,

$$G(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \boldsymbol{w}^T\boldsymbol{D}\boldsymbol{w}. \tag{2}$$

Your answer should specify what each entry of $\boldsymbol{D}$ should be. (2 points)

$\boldsymbol{D}$ should be a diagonal matrix with λ_1 on the first $d/2$ diagonal entries and λ_2 on the last $d/2$ diagonal entries. To show that this is correct, we need to show that $\boldsymbol{w}^T\boldsymbol{D}\boldsymbol{w} = \lambda_1 \sum_{j=1}^{d/2} w_j^2 + \lambda_2 \sum_{j=d/2+1}^{d} w_j^2$.

$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{D}\boldsymbol{w} &= \begin{bmatrix} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_d \end{bmatrix} \underbrace{\begin{bmatrix} \lambda_1 & & & & & \\ & \ddots & & & & \\ & & \lambda_1 & & & \\ & & & \lambda_2 & & \\ & & & & \ddots & \\ & & & & & \lambda_2 \end{bmatrix}}_{\boldsymbol{D}} \begin{bmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_d \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_d \end{bmatrix} \begin{bmatrix} \lambda_1\boldsymbol{w}_1 \\ \vdots \\ \lambda_1\boldsymbol{w}_{d/2} \\ \lambda_2\boldsymbol{w}_{d/2+1} \\ \vdots \\ \lambda_2\boldsymbol{w}_d \end{bmatrix} \end{aligned}$$

$$= \lambda_1 \sum_{j=1}^{d/2} w_j^2 + \lambda_2 \sum_{j=d/2+1}^{d} w_j^2$$

(c) Let $\boldsymbol{w}^* = \operatorname{argmin}_{w \in \mathbb{R}^d} G(\boldsymbol{w})$. Show that the closed-form solution $\boldsymbol{w}^*$ which minimizes Eq 2, is given by the following: (4 points)

$$\boldsymbol{w}_* = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{D}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}.$$

We'll find $\boldsymbol{w}^*$ by setting $\nabla G(\boldsymbol{w}) = 0$.

$$G(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \boldsymbol{w}^T\boldsymbol{D}\boldsymbol{w}$$
$$= \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y} + \boldsymbol{w}^T\boldsymbol{D}\boldsymbol{w}$$
$$= \boldsymbol{w}^T(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{D})\boldsymbol{w} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y}$$

$$\nabla G(\boldsymbol{w}) = \nabla \boldsymbol{w}^T(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{D})\boldsymbol{w} - \nabla 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \nabla \boldsymbol{y}^T\boldsymbol{y}$$
$$= \Big((\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{D}) + (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{D})^T\Big)\boldsymbol{w} - 2X^T\boldsymbol{y}$$
$$= 2(\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{D})\boldsymbol{w} - 2X^T\boldsymbol{y}$$

Setting $\nabla G(\boldsymbol{w}) = 0$ and solving for $\boldsymbol{w}$, we get $\boldsymbol{w}^* = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{D}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$.

(d) For this last part, assume $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$. Let β_j be the j-coordinate of the vector $\boldsymbol{X}^T\boldsymbol{y}$ (i.e., the inner product of the j-th row of $\boldsymbol{X}^T$ with $\boldsymbol{y}$). Derive an expression for the j-coordinate w_j of $\boldsymbol{w}^*$ (your expression can involve $w_j, \lambda_1, \lambda_2$ and β_j). (2 points)

From part (c) we know that $\boldsymbol{w}_* = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{D}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$. Plugging in $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$ and computing the expression we get,

$$\boldsymbol{w}_* = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} + \boldsymbol{D}\right)^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$
$$= (I + \boldsymbol{D})^{-1} \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

$$= \begin{bmatrix} 1+\lambda_1 & & & & & \\ & \ddots & & & & \\ & & 1+\lambda_1 & & & \\ & & & 1+\lambda_2 & & \\ & & & & \ddots & \\ & & & & & 1+\lambda_2 \end{bmatrix}^{-1} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{1+\lambda_1} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{1+\lambda_1} & & & \\ & & & \frac{1}{1+\lambda_2} & & \\ & & & & \ddots & \\ & & & & & \frac{1}{1+\lambda_2} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\beta_1}{1+\lambda_1} \\ \vdots \frac{\beta_{d/2}}{1+\lambda_1} \\ \frac{\beta_{d/2+1}}{1+\lambda_2} \\ \vdots \\ \frac{\beta_d}{1+\lambda_2} \end{bmatrix}$$

So altogether, we get

$$w_j = \begin{cases} \frac{\beta_j}{1+\lambda_1}, & \text{if } j \le d/2 \\ \frac{\beta_j}{1+\lambda_2}, & \text{if} j > d/2 \end{cases}$$

6 Support Vector Machines (14 points)

Consider a binary classification dataset with four points $\{(x_i, y_i), i \in [4]\}$ where $x_i \in \mathbb{R}$ and $y \in \{\pm 1\}$. We will take $(x_1, y_1) = (-2,\ -1)$, $(x_2, y_2) = (-1, 1)$, $(x_3, y_3) = (1/2,\ 1)$, $(x_4, y_4) = (3, -1)$. The points are shown in the figure below.

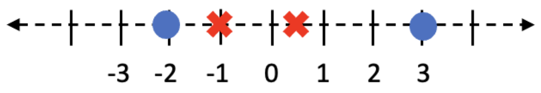


Figure 6: Dataset of points in 1 dimension. Blue circles denote a label of -1, red crosses denote +1.

(a) Can the four points shown in Figure 6, in their current one-dimensional feature space, be perfectly separated with a linear separator? Why or why not? (2 points)

No, they cannot be separated using a linear separator. Linear separator has the form $\text{SIGN}(wx + b) \equiv \text{SIGN}(x - \alpha)$ where $\alpha = -b/w$. α is in one of the intervals $\{-\inf, -2\}, \{-2, -1\}, \{-1, 1/2\}, \{1/2, 3\}$ as the intervals exhaust the real number line. One can see that no matter which interval we have α falls in, we always will have atleast one of the points misclassified.

(b) Now we define a simple feature mapping $\phi(x) = [x, x^2]^T$ to transform the four points from a one to a two-dimensional feature space. Plot the transformed points in the new two-dimensional feature space. You can fill in the 2-D plot below.

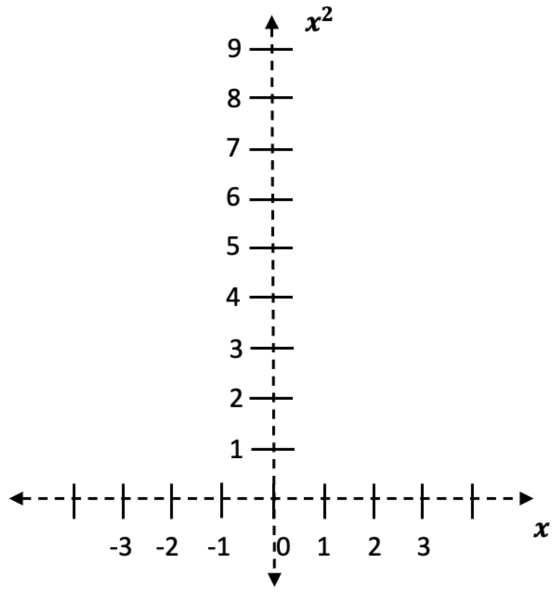


Figure 7: 2-D plot

Verify that the points are now separable in this new feature space by providing any linear decision boundary that separates the points. (2 points)

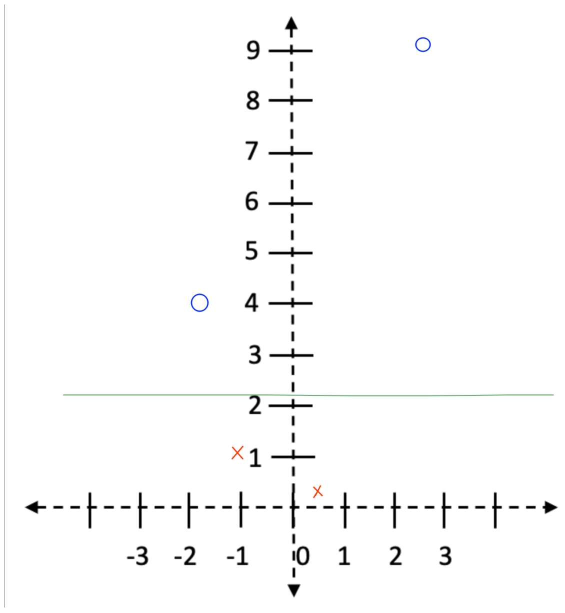


Figure 8: 2-D transformed plot

(c) For the rest of this question, we will use the feature mapping $\phi(x) = [x, x^2]^T$ from the previous part, and let $k(x_i, x_j)$ be the kernel function associated with this feature mapping. Fill in the 3 missing entries c_1, c_2, c_3 in the 4×4 kernel matrix $\boldsymbol{K}$ of the four data points based on the kernel function $k(x_i, x_j)$: (2 points)

$$\boldsymbol{K} = \begin{bmatrix} 20.0 & 6.0 & c_1 & 30.0 \\ 6.0 & c_2 & -0.25 & 6.0 \\ c_3 & -0.25 & 0.3125 & 3.75 \\ 30.0 & 6.0 & 3.75 & 90.0 \end{bmatrix}$$

The transformed points are (-2,4), (-1,1), (1/2,1/4) and (3,9)

$$\boldsymbol{K} = \begin{bmatrix} 20.0 & 6.0 & k(x_1,x_3) = \phi(x_1)^T\phi(x_3) & 30.0 \\ 6.0 & k(x_2,x_2) = \phi(x_2)^T\phi(x_2) & -0.25 & 6.0 \\ k(x_3,x_1) = \phi(x_3)^T\phi(x_1) & -0.25 & 0.3125 & 3.75 \\ 30.0 & 6.0 & 3.75 & 90.0 \end{bmatrix} \quad (3)$$

$$\boldsymbol{K} = \begin{bmatrix} 20.0 & 6.0 & x_1x_3 + (x_1x_3)^2 & 30.0 \\ 6.0 & x_2^2 + x_2^4 & -0.25 & 6.0 \\ x_3x_1 + (x_3x_1)^2 & -0.25 & 0.3125 & 3.75 \\ 30.0 & 6.0 & 3.75 & 90.0 \end{bmatrix} \quad (4)$$

$$\boldsymbol{K} = \begin{bmatrix} 20.0 & 6.0 & \mathbf{0} & 30.0 \\ 6.0 & \mathbf{2} & -0.25 & 6.0 \\ \mathbf{0} & -0.25 & 0.3125 & 3.75 \\ 30.0 & 6.0 & 3.75 & 90.0 \end{bmatrix} \quad (5)$$

(d) Recall that the dual SVM formulation for separable data $\{(x_i, y_i), i \in [4]\}$ is given by,

$$\max_{\{\alpha_i\}} \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j k(x_i, x_j)$$
$$\text{s.t.} \quad \sum_i \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0, \quad \forall\, i \in \{1, 2, 3, 4\}.$$

Use your 2-d plot from part (b) to conclude which of the α_i should be non-zero in this dual formulation (you do not need to solve the dual problem for this part). *(Hint: Use the fact that SVM finds the max-margin solution for separable data, and the support vectors for separable data are the points that are tight with respect to the max-margin constraints.)* (2 points)

α_3 and α_4 should be 0 since those points are not tight with respect to the max-margin constraints.

(e) Using the kernel function from part (c), write down the dual formulation for the dataset. Please use your answer from the previous part to simplify the dual formulation by only considering the α_i that should be non-zero. (2 points)

$$\max_{\alpha_1,\alpha_2} \quad \alpha_1 + \alpha_2 - \tfrac{1}{2}\left(\alpha_1^2 y_1^2 k(x_1, x_1) + \alpha_2^2 y_2^2 k(x_2, x_2) + \alpha_1\alpha_2 y_1 y_2 k(x_1, x_2) + \alpha_2\alpha_1 y_2 y_1 k(x_2, x_1)\right)$$
$$= \max_{\alpha_1,\alpha_2} \quad \alpha_1 + \alpha_2 - \tfrac{1}{2}\left(\alpha_1^2 y_1^2 k(x_1, x_1) + \alpha_2^2 y_2^2 k(x_2, x_2) + 2\alpha_1\alpha_2 y_1 y_2 k(x_1, x_2)\right)$$
$$= \max_{\alpha_1,\alpha_2} \quad \alpha_1 + \alpha_2 - \tfrac{1}{2}\left(\alpha_1^2 k(x_1, x_1) + \alpha_2^2 k(x_2, x_2) - 2\alpha_1\alpha_2 k(x_1, x_2)\right)$$
$$= \max_{\alpha_1,\alpha_2} \quad \alpha_1 + \alpha_2 - \tfrac{1}{2}\left(20\alpha_1^2 + 2\alpha_2^2 - 12\alpha_1\alpha_2\right)$$
$$= \max_{\alpha_1,\alpha_2} \quad \alpha_1 + \alpha_2 - 10\alpha_1^2 - \alpha_2^2 + 6\alpha_1\alpha_2$$

Constraints -
$\alpha_1 y_1 + \alpha_2 y_2 = 0$ and $\alpha_1, \alpha_2 > 0$
$\implies \alpha_1 = \alpha_2$ and $\alpha_1, \alpha_2 > 0$

(f) Solve your dual formulation to find the optimal value for all $\{\alpha_i, i \in [4]\}$. (4 points)

Let $\alpha_1 = \alpha_2 = \alpha$, then we have
$\max_{\alpha} \quad \alpha + \alpha - 10\alpha^2 - \alpha^2 + 6\alpha^2 \;=\; \max_{\alpha} \quad 2\alpha - 5\alpha^2$
The maximizer of this quadratic is clearly $\alpha = \alpha_1 = \alpha_2 = \frac{1}{5}$.