# CS689: Machine Learning

## Fall 2022 - Midterm Exam: Solutions

Name: ____________________________________

**Instructions:** Write your name on the exam sheet. The duration of this exam is **two hours**. No electronic devices may be used during the exam. You may consult your paper notes and/or a print copy of texts during the exam. Sharing of notes/texts during the exam is strictly prohibited. Show your work for all derivation questions. Provide answers that are as detailed as possible for explanation questions. Attempt all problems. Partial credit may be given for incorrect or incomplete answers. If you need extra space for answers, write on the back of the **preceding** page. If you have questions at any time, please raise your hand.

| Problem | Topic | Page | Points | Score |
|---|---|---|---|---|
| 1 | Loss Functions | 1 | 10 | |
| 2 | Discriminant Functions | 3 | 10 | |
| 3 | Bias | 5 | 10 | |
| 4 | Differentiability | 7 | 10 | |
| 5 | Convexity | 9 | 10 | |
| 6 | Sub-Differentials | 11 | 10 | |
| 7 | Gradient Descent | 13 | 10 | |
| 8 | Gradient Computations | 15 | 10 | |
| 9 | Regularization | 17 | 10 | |
| 10 | SVC | 19 | 10 | |
| Total: | | | 100 | |

**1.** (*10 points*) **Loss Functions**. Consider the regression loss functions $L_{sqr}(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$ and $L_{abs}(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$ when answering this question.

**a.** (*5 pts*) Give one reason why we might prefer using $L_{sqr}(y, f(\mathbf{x}))$ to learn a regression model instead of $L_{abs}(y, f(\mathbf{x}))$.

**Example Solution:** The squared loss $L_{sqr}(y, f(\mathbf{x}))$ is differentiable, and the absolute loss $L_{abs}(y, f(\mathbf{x}))$ is not. Hence, we can optimize the squared loss more easily using tools for standard differentiable optimization problems.

**b.** (*5 pts*) Give one reason why we might prefer using $L_{abs}(y, f(\mathbf{x}))$ to learn a regression model instead of $L_{sqr}(y, f(\mathbf{x}))$.

**Example Solution:** The linear increase of the absolute loss $L_{abs}(y, f(\mathbf{x}))$ gives this loss function more robustness to outliers than the squared loss $L_{sqr}(y, f(\mathbf{x}))$. The contribution of outliers to the squared loss is quadratic, making their individual contributions to the total loss significantly higher than with the absolute loss. This would make the model more biased towards the outliers in the case of squared loss.

**2.** (*10 points*) **Discriminant Functions**. The following questions pertain to classification discriminant and prediction functions.

**a.** (*5 pts*) In the context of classification models, explain what the difference is between the discriminant function and the prediction function.

**Example Solution:** In supervised learning of classification models, a prediction function $f_\theta(\mathbf{x})$ is used predict the output $y \in \mathcal{Y}$ that corresponds to an input $\mathbf{x} \in \mathcal{X}$.

A discriminant function $g_\theta(\mathbf{x})$ is a function that maps from the input space $\mathcal{X}$ to $\mathbb{R}$. In supervised learning of classification models, a discriminant function is typically used to distinguish between different classes by looking at the sign of the function $g_\theta(\mathbf{x})$. Thus, a discriminant function $g_\theta(\mathbf{x})$ is typically used to define a prediction function $f_\theta(\mathbf{x})$ by composing it with the sign function.

**b.** (*5 pts*) Explain why classification models typically use a loss applied to the discriminant function instead of the prediction function.

**Example Solution:** Applying a loss to the prediction function typically makes the resulting ERM objective function non-differentiable due to both the non-dffierentiability of the prediction function itself. This makes it hard to use optimization methods to learn the model.

On the other hand, applying a differentiable loss to the discriminant function results in an ERM objective function that us differentiable and hence we can easily use numerical optimization methods to learn the model.

**3.** (*10 points*) **Bias**. Suppose we have a linear regression model where $\mathbb{E}_{p_*(y|\mathbf{x})}[y] = \mathbf{x}\mathbf{w}_* + b_*$ and all data cases are independent and identically distributed under $p_*$ such that $p_*(\mathcal{D}) = \prod_{n=1}^{N} p_*(\mathbf{x}_n)p_*(y_n|\mathbf{x}_n)$. Further, suppose that we know the value of $\mathbf{w}_*$, but not $b_*$. Consider the estimator for the bias shown below when answering the following questions:

$$\hat{b} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n\mathbf{w}_*)$$

**a.** (*5 pts*) Briefly explain what it means for $\hat{b}$ to be an unbiased estimator of $b_*$ and give a supporting equation.

**Example Solution:** Assuming the data in $\mathcal{D}$ are sampled IID from $p_*$, an estimator of $b_*$ is said to be unbiased if its bias is equal to zero for all values of parameter $b_*$ and any data set $\mathcal{D}$. In the given model, the estimator is unbiased if $\mathbb{E}_{p_*(\mathcal{D})}[\hat{b}] - b_* = 0$.

**b.** (*5 pts*) Prove that $\hat{b}$ is an unbiased estimator of $b_*$.

**Example Solution:** To prove the result, we can show that $\mathbb{E}_{p_*(\mathcal{D})}[\hat{b}] = b_*$. We proceed as follows:

$$\begin{aligned}
\mathbb{E}_{p_*(\mathcal{D})}[\hat{b}] &= \mathbb{E}_{p_*(\mathcal{D})}\left[\frac{1}{N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n\mathbf{w}_*)\right] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p_*(\mathcal{D})}[(y_n - \mathbf{x}_n\mathbf{w}_*)]\\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p_*(y_n,\mathbf{x}_n)}[(y_n - \mathbf{x}_n\mathbf{w}_*)] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p_*(\mathbf{x}_n)}[\mathbb{E}_{p_*(y_n|\mathbf{x}_n)}[(y_n - \mathbf{x}_n\mathbf{w}_*)]\\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p_*(\mathbf{x}_n)}[\mathbb{E}_{p_*(y_n|\mathbf{x}_n)}[y_n] - \mathbf{x}_n\mathbf{w}_*)] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p_*(\mathbf{x}_n)}[(\mathbf{x}_n\mathbf{w}_* + b_*) - \mathbf{x}_n\mathbf{w}_*)]\\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{p_*(\mathbf{x}_n)}[b_*] = b_*
\end{aligned}$$

Note that to be complete, we need to justify the switch from taking the expectation with respect to $p_*(\mathcal{D})$ to taking the expectation this respect to $p_*(y_n, \mathbf{x}_n)$. As shown previously, we can do this using the IID assumption. Let $\mathcal{D}^{-n}$ denote all the data cases except $(\mathbf{x}_n, y_n)$. We have:

$$\begin{aligned}
\mathbb{E}_{p_*(\mathcal{D})}[y_n - \mathbf{x}_n\mathbf{w}_*] &= \mathbb{E}_{p_*(y_n,\mathbf{x}_n)}[\mathbb{E}_{p_*(\mathcal{D}^{-n})}[y_n - \mathbf{x}_n\mathbf{w}_*]]\\
&= \mathbb{E}_{p_*(y_n,\mathbf{x}_n)}[y_n - \mathbf{x}_n\mathbf{w}_*]
\end{aligned}$$

**4.** (*10 points*) **Differentiability**. Consider the squared hinge loss function given by $L_{SH}(z) = (\max(0, 1 - z))^2$. Is this function differentiable? Support your answer with a proof. (Note: you can use a sketch to get intuition for the problem, but "proof by picture" is not acceptable for a final answer.)

**Example Solution:** We observe that the max function has two pieces with boundary at $z = 1$.

$$\max(0, 1 - z) = \begin{cases} 1 - z & \text{for } z < 1 \\ 0 & \text{for } z \geq 1 \end{cases}$$

Hence,

$$(\max(0, 1 - z))^2 = \begin{cases} (1 - z)^2 & \text{for } z < 1 \\ 0 & \text{for } z \geq 1 \end{cases}$$

The two pieces are quadratic and constant respectively, and hence continuous and differentiable in the interiors. We thus need to check for continuity and differentiability at the boundary point between pieces, $z = 1$. To check continuity, we verify that the left and right hand limits of $L_{SH}(z)$ match at $z = 1$.

$$\lim_{z \to 1^-} L_{SH}(z) = \lim_{z \to 1^-} (1 - z)^2 = (1 - 1)^2 = 0 \text{ and } \lim_{z \to 1^+} L_{SH}(z) = \lim_{z \to 1^+} (0) = 0$$

To verify differentiability, we need to verify that the derivatives of the two pieces also match at $z = 1$. We have that the derivative of the constant piece is 0 at $z = 1$, so we need to verify that the derivative of $(1 - z)^2$ is 0 at $z = 1$. We have:

$$\frac{d}{dz}(1 - z)^2|_{z=1} = -2(1 - z)|_{z=1} = -2(1 - 1) = 0$$

**5.** (*10 points*) **Convexity**. Consider the function $f(x) = 1 - x^2$ when answering this question.

**a.** (*5 pts*) According to the definition of convexity, what condition needs to hold for $f(x)$ to be convex?

**Example Solution:** According to the definition of convexity, a function $f : \mathcal{X} \rightarrow \mathcal{R}$ is convex if $\mathcal{X}$ is a convex set and for any two points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, and for any scalar $\alpha \in [0, 1]$ the following condition must be true:

$$f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{x}') \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{x}')$$

**b.** (*5 pts*) Prove that $f(x)$ is *not* convex using the definition of convexity. (Note: you can use a sketch to get intuition for the problem, but "proof by picture" is not acceptable for a final answer. Your final answer must argue algebraically.)

**Example Solution:** We will show that $f(x)$ is not a convex function by exhibiting values of $x$, $x'$ and $\alpha \in [0, 1]$ where the convexity condition stated above does not hold. Since it needs to hold for all $x$, $x'$ and $\alpha \in [0, 1]$ for the function to be convex, this will prove that the function is not convex by counter example. Let $x = 1$, $x' = -1$ and $\alpha = 0.5$. We thus have $f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{x}') = f(0.5(1) + (1 - 0.5)(-1)) = f(0) = 1$. We also have $\alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{x}') = 0.5 f(1) + 0.5 f(-1) = 0$. Since $1 > 0$, we have found values $x$, $x'$ and $\alpha \in [0, 1]$ such that $f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{x}') > \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{x}')$ and the function is not convex.

**6.** (*10 points*) **Sub-Differentials**. Consider the epsilon insensitive loss function $L_\epsilon(z) = \max(0, |z| - \epsilon)$. What is the sub-differential of this function? Explain your answer.

**Example Solution:** For the epsilon insensitive loss function $L_\epsilon(z)$, and for $\epsilon > 0$, the function $L_\epsilon(z) = \max(0, |z| - \epsilon)$ consists of three pieces, with boundaries at $z = -\epsilon$ and $z = +\epsilon$. We can write the function $L_\epsilon(z)$ as:

$$L_\epsilon(z) = \begin{cases} -z - \epsilon & \text{for } z < -\epsilon \\ 0 & \text{for } -\epsilon \leq z \leq \epsilon \\ z - \epsilon & \text{for } z > \epsilon \end{cases}$$

Within each piece, the function is linear and hence differentiable. The potential points of non-differentiability are at $z = -\epsilon$ and $z = \epsilon$.

The subdifferential corresponding to the piece where $z < -\epsilon$ is the set containing the derivative of $-z - \epsilon$ which is $\frac{d}{dz}(-z - \epsilon) = -1$. Thus the subdifferential $\partial L_\epsilon(z) = \{-1\}$ for $z < -\epsilon$.

The subdifferential corresponding to the piece where $-\epsilon < z < \epsilon$ is the set containing the derivative of 0 which is $\frac{d}{dz}(0) = 0$. Thus the subdifferential $\partial L_\epsilon(z) = \{0\}$ for $-\epsilon < z < \epsilon$.

The subdifferential corresponding to the piece where $z > \epsilon$ is the set containing the derivative of $z - \epsilon$ which is $\frac{d}{dz}(z - \epsilon) = 1$. Thus the subdifferential $\partial L_\epsilon(z) = \{1\}$ for $z > \epsilon$.

The subdifferential at $z = -\epsilon$ is the closed interval $[\frac{d}{dz}(-z - \epsilon)|_{z=-\epsilon}, \frac{d}{dz}(0)|_{z=-\epsilon}] = [-1, 0]$.

The subdifferential at $z = \epsilon$ is the closed interval $[\frac{d}{dz}(0)|_{z=\epsilon}, \frac{d}{dz}(z - \epsilon)|_{z=\epsilon}] = [0, 1]$.

This gives us the following subdifferential function $\partial L_\epsilon(z)$:

$$\partial L_\epsilon(z) = \begin{cases} \{-1\} & \text{for } z < -\epsilon \\ [-1, 0] & \text{for } z = -\epsilon \\ \{0\} & \text{for } -\epsilon < z < \epsilon \\ [0, 1] & \text{for } z = \epsilon \\ \{1\} & \text{for } z > \epsilon \end{cases}$$

**7.** (*10 points*) **Gradient Descent**. Consider the basic gradient descent algorithm shown below for learning the linear regression model under squared loss when answering the following questions.

```
0: Inputs: $\mathcal{D}$, $\alpha$, $T$
1: Initialize $\theta_0 = 0$
2: for $i$ from 1 to $T$:
    2.1: $\mathbf{d}_i \leftarrow -\frac{1}{N}\sum_{n=1}^{N}(y_n - \mathbf{x}_n\theta_{i-1})\mathbf{x}_n^T$
    2.2: $\theta_i \leftarrow \theta_{i-1} + \alpha\mathbf{d}_i$
```

**a.** (*5 pts*) Suppose we run this method and we find that the value of the risk is *increasing* as $i$ increases. Explain how this could happen. Assume that the optimization method and all required computations are implemented correctly.

**Example Solution:** A value of the step size $\alpha$ that is too large can cause the gradient descent algorithm to diverge with the sequence of objective function values increasing instead of decreasing despite everything being implemented correctly.

**b.** (*5 pts*) Explain two different ways that the problem you identify in part (a) could be fixed.

**Example Solution:** One simple way to fix this problem is to reduce the step size $\alpha$. To find an appropriate step size $\alpha$, we can keep reducing $\alpha$ until the objective function decreases monotonically. This will work, but can be slow.

Another option is to use backtracking line search to dynamically adjust the step size at each iteration such that the sequence of objective function values monotonically decreases. To make sure this is efficient, we can use the Armijo rule to determine how far to back track.

**8.** (*10 points*) **Gradient Computations**. Suppose you implement the gradient computation for the standard linear logistic regression model with $\theta = [\mathbf{w}; b]$. You apply it to the data set shown below and compute the gradient at $\theta = [0, ..., 0]$ obtaining the answer $[-1.89859806, -1.71562749, -1.02561427, -1.0292008]^T$. You know without performing any other computations that your implementation has a bug. Explain what the bug is.

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $Y$ |
|---|---|---|---|---|
| 0.45860042 | 0.86083707 | 0.05370958 | 0.18891054 | -1 |
| 0.37934915 | 2.13886237 | 0.87653097 | 1.55244148 | -1 |
| 0.48755482 | 0.97964581 | 0.30012958 | 0.954377 | 1 |
| 0.07122404 | 0.79045198 | 1.84551752 | 0.0552564 | -1 |
| 0.37935095 | 1.6494246 | 1.12112722 | 0.47161681 | -1 |
| 0.15713379 | 1.51819191 | 1.85767571 | 0.72313994 | 1 |

**Example Solution:** The gradient vector is a vector of length equal to the parameter vector. In this case the parameters vector is $\theta = [\mathbf{w}; b]$. Based on the data table, we have four data dimensions indicated by $X_1, X_2, X_3, X_4$. This means $D = 4$. For linear logistic regression, the length of $\mathbf{w}$ is $D$. This means the parameter vector and thus the gradient vector should have length 4+1=5. The stated gradient vector has length 4 instead of length 5, so there is clearly a bug with the implementation.

**9.** (*10 points*) **Regularization**. Consider the linear regression model learned using squared loss with a 2-norm squared regularizer as shown below. Suppose we add the regularization parameter $\lambda$ to the model parameters so that $\theta = [\mathbf{w}; b; \lambda]$ and we use the L-BFGS optimizer to learn all of the model parameters (including the regularization parameter $\lambda$) by minimizing the regularized risk, as shown below. Assume that the optimization method and all required computations are implemented correctly. This learning approach has a severe problem. Explain what it is.

$$\hat{\theta} = \arg\min_{\theta} \left( \left( \frac{1}{N} \sum_{n=1}^{N} (y_n - (\mathbf{x}_n \mathbf{w} + b))^2 \right) + \lambda \|\mathbf{w}\|_2^2 \right)$$

**Example Solution:** If we include $\lambda$ as an optimization variable without constraints, the objective function becomes unbounded below. That can be seem by letting $\mathbf{w}$ be any non-zero value and observing that in that case $\|\mathbf{w}\|_2^2 > 0$. In the limit as $\lambda$ goes to negative infinity, the value of the risk will thus go to negative infinity. An optimizer will fail to converge under this conditions and the parameters learned will be arbitrary, a highly severe problem.

Note that if we learned the model with the constraint that $\lambda \geq 0$, the result would be $\lambda_* = 0$. To see this assume that the optimal value of the regularized risk is obtained when $\lambda_* > 0$ and call the optimal regularized risk value $r_*$. Now consider setting $\lambda'_* = 0$. The risk for this value of $\lambda$ is $r_* - \lambda_* \|\mathbf{w}_*\|_2^2 \leq r_*$, giving a contradiction. Thus, even with the constraint $\lambda \geq 0$, there is still a severe problem in that the value of the regularization parameter can not be usefully set using this approach.

**10.** (*10 points*) **SVC**. Suppose we fit a linear support vector classifier to a data set using hinge loss minimization and the standard 2-norm squared regularizer. Suppose that for any value of $C$ the training error rate is 0.5. Assume that all methods are implemented correctly when answering the following questions.

**a.** (*5 pts*) Explain what could be causing the model to perform poorly on the training set.

**Example Solution:** The training data might not be linearly separable when we try to fit a linear support vector classifier to the data leading to a training error rate of 0.5 no matter what the regularization constant $C$ is.

**b.** (*5 pts*) Suggest an approach that could potentially lead to improved performance on the training set.

**Example Solution:** We could try using a non-linear extension of the SVM such as a basis expansion or the use of a kernel. These approaches can result in non-linear decision boundaries and could potentially lead to improved performance if we can find a basis expansion or kernel that matches the structure of the data. We could also switch the model completely to any other non-linear classifier such as a neural network or logistic regression with a basis expansion or kernel.