CSCI 567, Spring 2025
Haipeng Luo

Written Assignment #2

Due: Feb 27, 2025, 11:59 pm, PT

Instructions

Total points: 50

Submission: Solutions must be typewritten or neatly handwritten and submitted through gradescope. You can submit multiple times, but only the last submission counts. It is your responsibility to make sure that you submit the right things, and we will *not* consider any regrading requests regarding mistakes in making submissions.

Recall that you have a total of three "late days" for the entire semester, and you can use at most one late day for each written assignment.

Notes on notation:

- Unless stated otherwise, scalars are denoted by small letter in normal font, vectors are denoted by small letters in bold font, and matrices are denoted by capital letters in bold font.
- $\|.\|$ means L2-norm unless specified otherwise, i.e., $\|.\| = \|.\|_2$.

Academic integrity: Our goal is to maintain an optimal learning environment. You can discuss the written assignments at a high level with others, but you should not look at any other's solutions. Trying to find solutions online or from any other sources (including ChatGPT and other similar tools) is prohibited, will result in zero grade and will be reported. To prevent any future plagiarism, uploading any materials from this course to the Internet is also prohibited, and any violations will be reported. Please be considerate and help us help everyone get the best out of this course.

Problem 1 Multiclass Perceptron (17 points)

Recall that a linear model for a multiclass classification problem with C classes is parameterized by C weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C \in \mathbb{R}^D$. In the lecture, we derived the multiclass logistic regression by minimizing the multiclass logistic loss. In this problem, you need to derive the multiclass perceptron algorithm in a similar way. Specifically, the multiclass perceptron loss on a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N) \in \mathbb{R}^D \times [C]$ is defined as

$$F(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C) = \frac{1}{N}\sum_{n=1}^{N} F_n(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C), \quad \text{where } F_n(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C) = \max\left\{0, \max_{y \neq y_n} \boldsymbol{w}_y^{\mathrm{T}}\boldsymbol{x}_n - \boldsymbol{w}_{y_n}^{\mathrm{T}}\boldsymbol{x}_n\right\}.$$

1.1 To optimize this loss function, we need to first derive its gradient. Specifically, for each $n \in [N]$ and $c \in [C]$, write down the partial derivative $\frac{\partial F_n}{\partial \boldsymbol{w}_c} \in \mathbb{R}^D$ (and your derivation). For simplicity, you can assume that for any n, $\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{x}_n, \ldots, \boldsymbol{w}_C^{\mathrm{T}}\boldsymbol{x}_n$ are always C distinct values (so that there is no tie when taking max over them, and consequently no non-differentiable points needed to be considered). (8 points)

For each n, let $\hat{y}_n = \arg\max_{y \in [C]} \boldsymbol{w}_y^{\mathrm{T}}\boldsymbol{x}_n$. Then by definition, F_n can be written as

$$\begin{cases} 0, & \text{if } \hat{y}_n = y_n, \\ \boldsymbol{w}_{\hat{y}_n}^{\mathrm{T}}\boldsymbol{x}_n - \boldsymbol{w}_{y_n}^{\mathrm{T}}\boldsymbol{x}_n, & \text{else.} \end{cases}$$

Its partial derivative with respect to $\boldsymbol{w}_c$ is then

$$\begin{cases} \boldsymbol{0}, & \text{if } \hat{y}_n = y_n, \\ \boldsymbol{x}_n, & \text{else if } c = \hat{y}_n, \\ -\boldsymbol{x}_n, & \text{else if } c = y_n, \\ \boldsymbol{0}, & \text{else.} \end{cases}$$

Rubrics: 2 points for each of the 4 cases. There are many other ways to write this, such as using indicator functions. It is of course also possible to combine some of these cases (such as the first and the fourth ones).

1.2 Similarly to the binary case, multiclass perceptron is simply applying SGD with learning rate 1 to minimize the multiclass perceptron loss. Based on this information, fill in the missing details in the repeat-loop of the algorithm below (your solution cannot contain implicit quantities such as $\nabla F_n(\boldsymbol{w})$; instead, write down the exact formula based on your solution from the last question). (4 points)

```
Algorithm 1: Multiclass Perceptron
1 Input: A training set $(x_1, y_1), \ldots, (x_N, y_N)$
2 Initialization: $w_1 = \cdots = w_C = 0$
3 Repeat:
4     randomly pick an example $(x_n, y_n)$ and compute $\hat{y}_n = \arg\max_{y \in [C]} w_y^T x_n$
5     if $\hat{y}_n \neq y_n$ then
6         $w_{\hat{y}_n} \leftarrow w_{\hat{y}_n} - x_n$
7         $w_{y_n} \leftarrow w_{y_n} + x_n$
```

Rubrics: 1 point for randomly picking an example, 1 point for computing the prediction $\hat{y}_n$ (which one has to compute either explicitly or implicitly), and 2 points for correctly implementing the rest of SGD. Again, there are many equivalent ways to implement this. Do not deduct points if the gradient is wrong solely due to mistakes from the last question.

1.3 At this point, you should find that the parameters $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C$ computed by Multiclass Perceptron are always linear combinations of the training points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, that is, $\boldsymbol{w}_c = \sum_{n=1}^N \alpha_{c,n}\boldsymbol{x}_n$ for some coefficient $\alpha_{c,n}$. Just like kernelized linear regression, this means that one can kernelize multiclass Perceptron as well for any given kernel function $k(\cdot,\cdot)$ (with a corresponding feature mapping $\boldsymbol{\phi}$).

Specifically, fill in the missing details in the repeat-loop of the algorithm below which maintains and updates weights $\alpha_{c,n}, \ \forall c \in [C], n \in [N]$ such that $\boldsymbol{w}_c = \sum_{n=1}^N \alpha_{c,n}\boldsymbol{\phi}(\boldsymbol{x}_n)$ is always the same as what one would get by running Algorithm 1 with $\boldsymbol{x}_n$ replaced by $\boldsymbol{\phi}(\boldsymbol{x}_n)$ for all n. No reasoning is required. Keep in mind that in your solution $\boldsymbol{\phi}(\boldsymbol{x}_n)$ should never appear. (5 points)

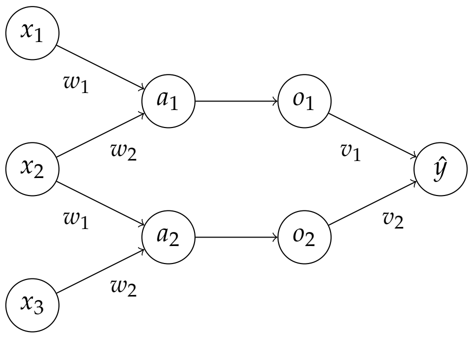
Algorithm 2: Multiclass Perceptron with kernel function $k(\cdot,\cdot)$

```
1 Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
2 Initialize: $\alpha_{c,n} = 0$ for all $c \in [C]$ and $n \in [N]$
3 Repeat:
4     randomly pick an example $(\boldsymbol{x}_n, y_n)$ and compute $\hat{y}_n = \arg\max_{y \in [C]} \left(\sum_{m=1}^N \alpha_{y,m} k(\boldsymbol{x}_m, \boldsymbol{x}_n)\right)$
5     if $\hat{y}_n \neq y_n$ then
6         $\alpha_{\hat{y}_n,n} \leftarrow \alpha_{\hat{y}_n,n} - 1$
7         $\alpha_{y_n,n} \leftarrow \alpha_{y_n,n} + 1$
```

Rubrics: 1 point for randomly picking an example, 2 points for computing the prediction $\hat{y}_n$ using the kernel function, and 2 points for correctly implementing the rest. Again, there are many equivalent ways to implement this. Storing the kernel matrix to avoid repeated calculations is of course acceptable. Do not deduct points if the mistake is solely inherited from the first question.

Problem 2 Backpropagation for CNN (18 points)

Consider the following mini convolutional neural net, where (x_1, x_2, x_3) is the input, followed by a convolution layer with a filter (w_1, w_2), a ReLU layer, and a fully connected layer with weight (v_1, v_2).



More concretely, the computation is specified by

$$\begin{aligned}
a_1 &= x_1 w_1 + x_2 w_2 \\
a_2 &= x_2 w_1 + x_3 w_2 \\
o_1 &= \max\{0, a_1\} \\
o_2 &= \max\{0, a_2\} \\
\hat{y} &= o_1 v_1 + o_2 v_2
\end{aligned}$$

For an example $(\boldsymbol{x}, y) \in \mathbb{R}^3 \times \{-1, +1\}$, the logistic loss of the CNN is

$$\ell = \ln(1 + \exp(-y\hat{y})),$$

which is a function of the parameters of the network: w_1, w_2, v_1, v_2.

2.1 Write down $\frac{\partial \ell}{\partial v_1}$ and $\frac{\partial \ell}{\partial v_2}$ (show the intermediate steps that use chain rule). You can use the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to simplify your notation. (4 points)

$$\begin{aligned}
\frac{\partial \ell}{\partial v_1} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial v_1} && \text{(1 point)} \\
&= \frac{-ye^{-y\hat{y}}}{1 + e^{-y\hat{y}}} o_1 = -\sigma(-y\hat{y}) y o_1 = (\sigma(y\hat{y}) - 1) y o_1 && \text{(1 point)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \ell}{\partial v_2} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial v_2} && \text{(1 point)} \\
&= \frac{-ye^{-y\hat{y}}}{1 + e^{-y\hat{y}}} o_2 = -\sigma(-y\hat{y}) y o_2 = (\sigma(y\hat{y}) - 1) y o_2 && \text{(1 point)}
\end{aligned}$$

Rubrics: Any one of the last three expressions is acceptable.

2.2 Write down $\frac{\partial \ell}{\partial w_1}$ and $\frac{\partial \ell}{\partial w_2}$ (show the intermediate steps that use chain rule). The derivative of the ReLU function is $H(a) = \mathbb{I}[a > 0]$, which you can use directly in your answer. (6 points)

$$\begin{aligned}
\frac{\partial \ell}{\partial w_1} &= \frac{\partial \ell}{\partial a_1}\frac{\partial a_1}{\partial w_1} + \frac{\partial \ell}{\partial a_2}\frac{\partial a_2}{\partial w_1} && \text{(1 point)}\\
&= \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_1}\frac{\partial o_1}{\partial a_1}\frac{\partial a_1}{\partial w_1} + \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_2}\frac{\partial o_2}{\partial a_2}\frac{\partial a_2}{\partial w_1} && \text{(1 point)}\\
&= (\sigma(y\hat{y}) - 1)y(v_1 H(a_1)x_1 + v_2 H(a_2)x_2) && \text{(1 point)}
\end{aligned}$$

Similarly

$$\begin{aligned}
\frac{\partial \ell}{\partial w_2} &= \frac{\partial \ell}{\partial a_1}\frac{\partial a_1}{\partial w_2} + \frac{\partial \ell}{\partial a_2}\frac{\partial a_2}{\partial w_2} && \text{(1 point)}\\
&= \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_1}\frac{\partial o_1}{\partial a_1}\frac{\partial a_1}{\partial w_2} + \frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial o_2}\frac{\partial o_2}{\partial a_2}\frac{\partial a_2}{\partial w_2} && \text{(1 point)}\\
&= (\sigma(y\hat{y}) - 1)y(v_1 H(a_1)x_2 + v_2 H(a_2)x_3). && \text{(1 point)}
\end{aligned}$$

Rubrics: Again, other equivalent expressions are acceptable.

2.3 Using the derivations above, fill in the missing details of the repeat-loop of the Backpropagation algorithm below that is used to train this mini CNN. (8 points)

Algorithm 3: Backpropagation for the above mini CNN

```
1 Input: A training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$, learning rate $\eta$
2 Initialize: set $w_1, w_2, v_1, v_2$ randomly
3 Repeat:
4     randomly pick an example $(\boldsymbol{x}_n, y_n)$ (the three features of $\boldsymbol{x}_n$ are denoted by $x_{n1}$, $x_{n2}$, and $x_{n3}$.)
5     Forward propagation: compute                                   (4 points)
          $a_1 = x_{n1}w_1 + x_{n2}w_2, a_2 = x_{n2}w_1 + x_{n3}w_2$
          $o_1 = \max\{0, a_1\}, o_2 = \max\{0, a_2\}, \hat{y} = o_1v_1 + o_2v_2$
6     Backward propagation: update                                   (4 points)
          $w_1 \leftarrow w_1 - \eta(\sigma(y_n\hat{y}) - 1)y_n(v_1H(a_1)x_{n1} + v_2H(a_2)x_{n2})$
          $w_2 \leftarrow w_2 - \eta(\sigma(y_n\hat{y}) - 1)y_n(v_1H(a_1)x_{n2} + v_2H(a_2)x_{n3})$
          $v_1 \leftarrow v_1 - \eta(\sigma(y_n\hat{y}) - 1)y_no_1$
          $v_2 \leftarrow v_2 - \eta(\sigma(y_n\hat{y}) - 1)y_no_2$
```

Rubrics:

- Deduct 1 point for writing x_1, x_2, x_3 instead of x_{n1}, x_{n2}, x_{n3} in the forward propgagation.
- Deduct 1 point for writing x_1, x_2, x_3, y instead of $x_{n1}, x_{n2}, x_{n3}, y_n$ in the backward propgagation.
- Deduct 2 points if updating w_1/w_2 with the updated value of v_1/v_2.
- Do not deduct points for using the wrong gradients solely due to mistakes from previous two questions.

Problem 3 Support Vector Machines (15 points)

Consider a dataset consisting of points in the form of (x, y), where x is a real value, and $y \in \{-1, 1\}$ is the class label. There are only three points $(x_1, y_1) = (-1, -1)$, $(x_2, y_2) = (1, -1)$, and $(x_3, y_3) = (0, 1)$, shown in Figure 1.

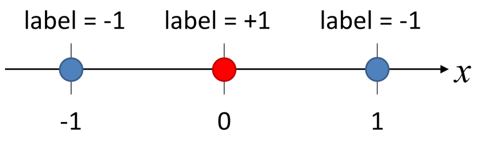


Figure 1: Three data points considered in Problem 1

3.1 Can these three points in their current one-dimensional feature space be perfectly separated with a linear classifier of the form $\text{SGN}(wx + b)$ for some $w, b \in \mathbb{R}$? Why or why not? (2 points)

No. A one-dimensional linear model $\text{SGN}(wx + b)$ is equivalent to a simple threshold function of the form

$$f(x) = \begin{cases} +1 & \text{if } x \geq \theta \\ -1 & \text{else} \end{cases} \qquad \text{or} \qquad f(x) = \begin{cases} -1 & \text{if } x \geq \theta \\ +1 & \text{else} \end{cases}$$

for some threshold $\theta \in \mathbb{R}$. It is thus clear that if we want points x_1 and x_2 to be correctly classified, then x_3 must be incorrectly classified. (Other correct reasoning gets full points as well.)

3.2 Now consider the feature mapping $\boldsymbol{\phi}(x) = [x, x^2]^T$ (convince yourself that the data are now linearly separable in this new feature space). Write down the 3×3 kernel/Gram matrix $\mathbf{K}$ for this dataset. (2 points)

The kernel function is $k(x, x') = \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(x) = xx' + (xx')^2$, so the Gram matrix is $\mathbf{K} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

3.3 Next, write down the 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 $+\infty$. (You have to plug in the actual data instead of just showing the generic dual formulation.) (2 points)

General dual formulation of SVM for separable data is:

$$\begin{aligned} \max_{\alpha} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_n) \\ \text{s.t.} \quad & \alpha_n \geq 0, \ \forall n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

Plugging in the specific dataset gives:

$$\begin{aligned} \max_{\alpha_1,\alpha_2,\alpha_3 \geq 0} \quad & \alpha_1 + \alpha_2 + \alpha_3 - \alpha_1^2 - \alpha_2^2 \\ \text{s.t.} \quad & \alpha_1 + \alpha_2 = \alpha_3 \end{aligned}$$

Rubrics:

- okay to write the solution directly without first writing down the general form.
- 1 point for the objective and 1 point for the constraints.
- do not deduct points if the mistake is solely due to the incorrect Gram matrix from the last question.

3.4 Next, solve the dual formulation exactly (note: while this is not generally feasible as discussed in the lecture, the simple form of this dataset makes it possible). (3 points)

Eliminating the dependence on α_3 using the constraint $\alpha_1 + \alpha_2 = \alpha_3$, we arrive at the objective

$$\max_{\alpha_1,\alpha_2 \geq 0} 2\alpha_1 - \alpha_1^2 + 2\alpha_2 - \alpha_2^2.$$

Clearly we can maximize over α_1 and α_2 separately, which gives $\alpha_1^* = \alpha_2^* = 1$ and thus $\alpha_3^* = 2$.

Rubrics: Any derivation reaching the same solution works. Do not deduct points if the dual formulation is wrong due to the last question as long as it is solved correctly here.

3.5 Given a test point $x = 2$, what is the prediction of this SVM? Follow the steps below to derive your answer: first, compute the primal solution b^*; then, compute $k(x_n, x)$ for all $n = 1, 2, 3$ and the test point $x = 2$; finally, calculate the prediction. Note that to get full credits, you have to avoid directly operating in the feature space that corresponds to the kernel function. (6 points)

Since all of the three examples satisfy $0 < \alpha_n < C = +\infty$, based on Lecture 6, any one of the following gives us the correct answer:

$$b^* = y_1 - \sum_{n=1}^{3} \alpha_n^* y_n k(x_n, x_1) = -1 + 2 - 0 - 0 = 1,$$
$$b^* = y_2 - \sum_{n=1}^{3} \alpha_n^* y_n k(x_n, x_2) = -1 - 0 + 2 - 0 = 1,$$
$$b^* = y_3 - \sum_{n=1}^{3} \alpha_n^* y_n k(x_n, x_3) = 1 - 0 - 0 - 0 = 1.$$

Next, we calculate $k(x_n, x)$ for all $n = 1, 2, 3$ and the test point $x = 2$:

$$k(x_1, 2) = -2 + (-2)^2 = 2, \quad k(x_2, 2) = 2 + 2^2 = 6, \quad k(x_3, 2) = 0.$$

Finally, the prediction can be found by $\text{SGN}\left(\sum_{n=1}^{3} \alpha_n^* y_n k(x_n, 2) + b^*\right) = \text{SGN}\left(-2 - 6 + 0 + 1\right) = -1.$

Rubrics:

- 2 points for finding the correct value of b^*.
- 3 points for computing $k(x_n, x)$ correctly for all $n = 1, 2, 3$.
- 1 point for the correct final prediction.
- Similarly, do not deduct points if the mistake is solely due to the wrong dual solution from the last question.