Assignment 3

Due Dec 4, 2024 at 11:59pm

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use Google to find solutions for assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment.

Submitting Your Assignment

The assignment must be submitted online on Coursys. You must submit a report in **PDF format**. You may typeset your assignment in LaTeX or Word, or submit neatly handwritten and scanned solutions. We will not be able to give credit to solutions that are not legible.

Useful Information

This page will also appear on the final exam. Please familiarize yourself with it and feel free to refer to it, in case you need any of these results in your work.

Gradient of a function $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ and Jacobian of a function $\vec{f}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}^m$:

$$\frac{\partial f}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}, \quad \frac{\partial \vec{f}(\vec{x})}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{1}$$

Derivative rules:

$$\frac{\partial(\vec{f}(\vec{x})^\top \vec{g}(\vec{x}))}{\partial \vec{x}} = \frac{\partial \vec{f}(\vec{x})}{\partial \vec{x}} \vec{g}(\vec{x}) + \frac{\partial \vec{g}(\vec{x})}{\partial \vec{x}} \vec{f}(\vec{x}) \tag{2a}$$

$$\frac{\partial \vec{f}(\vec{y}_1(\vec{x}), \vec{y}_2(\vec{x}))}{\partial \vec{x}} = \frac{\partial \vec{y}_1}{\partial \vec{x}} \frac{\partial \vec{f}}{\partial \vec{y}_1} + \frac{\partial \vec{y}_2}{\partial \vec{x}} \frac{\partial \vec{f}}{\partial \vec{y}_2} \tag{2b}$$

Common derivatives:

$$\frac{\partial(\vec{a}^\top \vec{x})}{\partial \vec{x}} = \vec{a}, \qquad \frac{\partial(A\vec{x})}{\partial \vec{x}} = A^\top, \qquad \frac{\partial(\vec{x}^\top A \vec{x})}{\partial \vec{x}} = (A + A^\top)\vec{x} \tag{3}$$

Common distributions:

$$x \sim \text{Categorical}(\{p_i\}_{i=1}^n) \Rightarrow p(x_i) = p_i, \quad \forall i \in \{1, \dots, n\} \tag{4a}$$

$$x \sim \text{Binomial}(n, p) \Rightarrow p(x = k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \tag{4b}$$

Ridge regression and maximum a *posteriori* estimation:

$$\vec{w}^* = \arg\min_{\vec{w}} \sum_{i=1}^N (y_i - \vec{w}^\top \vec{x}_i)^2 + \lambda \sum_{i=1}^n w_i^2 \qquad = \arg\min_{\vec{w}} \|\vec{y} - X\vec{w}\|_2^2 + \lambda \|\vec{w}\|_2^2 \tag{5a}$$

$$= \arg\max_{\vec{w}} p\left(\{x_i, y_i\}_{i=1}^N \mid \vec{w}\right) p(\vec{w}) \qquad = (X^\top X + \lambda I)^{-1} X^\top \vec{y} \tag{5b}$$

Support vector machine (SVM):

$$\underset{\vec{w}, b}{\text{minimize}} \quad \frac{1}{2}\|\vec{w}\|_2^2 \tag{6a}$$

$$\text{subject to} \quad y_i(\vec{w}^\top \vec{x}_i - b) \geq 1, \quad \forall i \tag{6b}$$

SVM dual problem:

$$\underset{\vec{\lambda}}{\text{maximize}} \quad \sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \vec{x}_j^\top \vec{x}_i \tag{7a}$$

$$\text{subject to} \quad \sum_{i=1}^N \lambda_i y_i = 0 \tag{7b}$$

$$\lambda_i \geq 0 \quad \forall i \tag{7c}$$

1. **Neural Networks**

Consider a basic multi-layer perceptron (MLP) with three layers, characterized by the following weight matrices:

- $W_0 \in \mathbb{R}^{2\times 2}$
- $W_1 \in \mathbb{R}^{3\times 2}$
- $W_2 \in \mathbb{R}^{1\times 3}$

Answer the following questions regarding the structure and behavior of this neural network.

1.1 Basic MLP

a) How many weights are there in this neural network? Show the calculation.

b) Compute the output of the network $\hat{y}$ for the input $x = (1, 1)^\top$. Use the weights:

$$W_0 = \begin{bmatrix} 0.1 & 0.4 \\ -0.5 & 0.6 \end{bmatrix}, \quad W_1 = \begin{bmatrix} 0.2 & -0.1 \\ 0.5 & 0.4 \\ 0.3 & -0.6 \end{bmatrix}, \quad W_2 = [0.7, -0.4, 0.9]$$

Assume ReLU activation and show the calculation steps.

c) Compare the effect of using ReLU versus sigmoid activation in layer 1 only. Compute the output for the same network and input x, and summarize the differences.

d) Compute $\frac{\partial L}{\partial W_2}$, assuming $L = \frac{1}{2}(\hat{y} - y)^2$. Show its shape and intermediate steps for the forward and backward passes.

1.2 Revised MLP-1: MLP with Missing Weights

a) For the following network structure, calculate the output $\hat{y}$ for $x = (1, 1)^\top$:

$$W_0 = \begin{bmatrix} 0.2 & 0.3 \\ \star & -0.1 \end{bmatrix}, \quad W_1 = \begin{bmatrix} 0.4 & \star \\ \star & 0.6 \\ -0.2 & 0.1 \end{bmatrix}, \quad W_2 = [0.3, 0.5, \star]$$

Missing entries (denoted $\star$) are zero. Use ReLU activation.

b) Assume layer-wise normalization (mean subtraction, variance normalization) applied to z_1 before activation. How does it affect the output? Compute for the network above.

c) Debugging and Improving a Neural Network with PyTorch

The code file `mlp_weights.py` provides a PyTorch code template for you to experiment with the multi-layer perceptron (MLP) from part (b), which contains missing weights. The network is implemented with explicit forward passes for more transparency during training.

Task: Debug and modify the given neural network so that it converges effectively during training. Initially, the missing weights are set to zero, which prevents the network from learning effectively. Your goal is to make modifications that enable successful convergence and analyze the outcomes.

Constraints:

- Use stochastic gradient descent (SGD) as the optimizer.
- Maintain the three linear transformations defined by weight matrices W_0, W_1, and W_2. Do not add or remove layers.
- You may modify elements within each layer (e.g., activation functions, adding bias terms), but keep the given structure of the weight matrices intact.

Steps to Follow:

i) **Initial Run:** Run the provided network as-is. Document the loss curve and explain why the network fails to converge.

ii) **Modify and Observe:** Make one modification at a time. Document each noteworthy change, including plots showing the effects on loss and parameter evolution.

iii) **Analyze:** Compare the different modifications. Which ones improved convergence? Discuss how the initial zero values for weights led to degenerate learning, and explain how your modifications addressed these issues.

Note: Since we have only a single input and output, the "stochastic" element of SGD is not present here—it effectively functions as standard gradient descent.

In Your Report:

- Describe the initial issues observed.
- Summarize the modifications you made and why.
- Include plots for the loss curve and parameter evolution.
- Analyze the impact of different changes, focusing particularly on the role of initialization in achieving effective learning.

Time Frame: Spend approximately 30-60 minutes experimenting until you achieve a satisfactory solution.

d) Exploring the Role of Stochasticity in Overcoming Initialization Issues

Consider the problem of zero initialization in neural networks. Could true stochastic gradient descent (SGD) overcome the issue of zero-initialized weights? To better understand this, please answer the multiple-choice questions by editing the text file `sgd_mc_answers.txt` located in the A3 folder. Don't answer them in the report you submit. Below are the multiple-choice questions included in the file:

a) Why is stochastic gradient descent (SGD) called "stochastic"?

(a) Because it uses random mini-batches of data to approximate the gradient.

(b) Because it computes the gradient over the entire dataset.

(c) Because the learning rate changes at each step.

(d) Because it always converges faster.

b) What is the effect of initializing all weights to zero in a neural network?

(a) All neurons in a layer receive identical gradients and learn the same features.

(b) The network will converge faster due to symmetry.

(c) Stochastic updates will make neurons learn distinct features.

(d) Zero initialization always helps prevent overfitting.

c) Could true stochastic gradient descent overcome the problem of zero-initialized weights?

(a) Yes, because random mini-batches would introduce sufficient variability.

(b) No, because all neurons have identical weights and thus receive identical gradients, regardless of mini-batch variability.

(c) Yes, because SGD inherently breaks symmetry.

(d) No, but adding bias terms would always fix it.

2. SVM

Support Vector Machines (SVM) Assignment

This assignment will focus on applying Support Vector Machines (SVM) to a dataset representing simplified customer behavior metrics. The objective is to train an SVM classifier to distinguish between two customer segments based on their engagement level with a service.

The dataset, D, contains the following features for each customer:

- x_1: Average number of visits to the service per week.
- x_2: Average spending per visit (in tens of dollars).

The labels are defined as follows:

- $+1$: High-engagement customer segment.
- -1: Low-engagement customer segment.

The given dataset consists of the following samples:

$$D = \{((2,6),1),((6,2),-1),((4,6),1),((5,3),-1)\}$$

Recall the soft-margin SVM formulation:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \forall i$$

a) Feasibility of $(\mathbf{w_1}, b_1)$

Given $\mathbf{w_1} = (0.5, -0.5)$ and $b_1 = 1$, determine whether $(\mathbf{w_1}, b_1)$ is a feasible solution to the soft-margin SVM problem. Verify if all the data points satisfy the constraint for the given $\mathbf{w_1}$ and b_1.

b) Determine the Optimal Solution $(\mathbf{w}^*, b^*)$

Determine the optimal solution $(\mathbf{w}^*, b^*)$ for the soft-margin SVM problem for the given dataset. Use a linear kernel for simplicity. Create a coordinate system showing the data points (marking positive and negative examples), the optimal decision boundary line $\mathbf{w}^{*T}\mathbf{x} - b^* = 0$, and the margin boundaries parallel to the decision boundary at distance $1/\|\mathbf{w}^*\|$ from the decision boundary.

- Assume $C = 1$. Explain the impact of different values of C on the decision boundary.
- Calculate the objective value of the primal solution.

c) Support Vectors and Their Role

Identify the support vectors for the given dataset and explain their role in defining the decision boundary. Which data points are support vectors in this case, and why?

d) Exploring the Dual SVM Solution

Determine the optimal objective value for the dual SVM problem and find the corresponding dual variables $(\lambda_1, \lambda_2, \dots)$. Use the generalized Lagrangian approach to derive the weight vector and explain how the dual formulation relates to the primal SVM solution.

Use the provided Python script (`solve_lambdas.py`) to determine the optimal objective value for the dual SVM problem. Complete the TODO in the code to define the Gram (Kernel) matrix K. Once you have filled this in, run the code to compute the dual variables $(\lambda_1, \lambda_2, \dots)$.

Tasks:

a) **Fill in the Kernel Computation**: Complete the computation for the Gram matrix K in `solve_lambdas.py`.

b) **Run the Code**: Execute the script to determine the optimal λ_i values and identify the support vectors.

c) **Derive the Weight Vector**: Using the generalized Lagrangian for the SVM, take the partial derivative with respect to the weight vector $\boldsymbol{w}$ and set it to zero to derive the weight vector. Refer to the lecture slides on the SVM dual for guidance on the critical point condition involving $\boldsymbol{w}$ and λ_i. This step will help you link the dual variables (λ_i) to the weight vector.

d) **Verification Task**: Use the computed λ_i values from the script to manually verify the optimal weight vector $\boldsymbol{w}^*$ using the derived formula. Compare your calculation to the value computed by the code.

e) **Relationship Between Primal and Dual**: Explain how the derived $\boldsymbol{w}^*$ relates to the primal SVM problem and how the support vectors influence the solution.

f) Considerations for Scaling to Larger Datasets

Discuss how the analysis methods used in this assignment would change if the dataset were significantly larger, for instance, involving 100,000 customers. Consider aspects such as computational efficiency, model complexity, and practical implementation challenges.