ROUGH WORK

Part I: Simple objective questions; No details/steps needed [60 min; $12 \times 5 = 60$ points]

1. If a neural network designed for classification outputs $[0.2, 0.1, 0.7, 0.0]^T$ and the target is $[0, 0, 1, 0]^T$. Compute the cross entropy loss.

   Expression: ————

   - $y_i \cdot \log(\hat{y}_i)$

Numerical Value: ————-

log(0.7)

2. Consider a 1D convolution. A convolution layer has 3 input channel of size 1000 each. Output is computed over a 1-D window of length 7. There are 5 output channels. Stride is 3. No bias.

   How many learnable parameters exists in this layer?
   Answer: 7*3*5 or 110 for bias

3. Make the necessary minimal changes (if any required) and rewrite as true sentence in the space provided. Avoid changing the words in bold.

   Adding a momentum term *removes the problem of local minima* **in Neural network training.**

   Adding a momentum term reduces the problem of local minima in Neural network training.

   ————

4. Consider the following question with multiple answers. Given answers could be incorrect. Correctly answer in the space given with an answer approximately of equal length/detail, preferably by rewording or copying one of the given answers. Pick the closest answer and consider rewriting.

   Q: Given the objective/loss of a neural network, why don't we differentiate with respect to the individual weights, equate to zero and find the optimal (may be local) solution in one step?

   Ans1: Yes; it is feasible. We do not do this because back-propagation is simpler to implement in python.

   Ans2: No; this is not possible. Because we do not know how to differentiate the loss with respect to individual weights.

   Possible but not feasible in reality because solving a non linear equation with a large number of variables (number of weights) is almost impossible

   Your Answer: ————

   Ans 2.(1 mark) correct explanation(4 mark)

5. Consider the following question with multiple answers. Given answers could be incorrect. Correctly answer in the space given with an answer approximately of equal length/detail, preferably by rewording or copying one of the given answers. Pick the closest answer and consider rewriting.

   Q: We can initialize all the weights of an MLP as equal (say 1.0). This is considered to be bad. Why?

Ans1: No; This is a perfectly fine initialization

Ans2: Yes; this is not a good initialization. Since weights are equal, we can not find the derivatives

Ans3: Yes; this is not a good initialization. Since weights are equal, all weights learn at constant speed and we will not get a sparse solution as network.

Not a good initialization, all nodes of a layer for all layers except first layer will have same derivative and thus weights will remain always same across a layer. "Symmetry" needs to be broken.

6. Consider the following question with multiple answers. Given answers could be incorrect. Correctly answer in the space given with an answer approximately of equal length/detail, preferably by rewording or copying one of the given answers. Pick the closest answer and consider rewriting.

   Q: Consider an autoencoder where input and output are x. We train this network by minimizing the MSE loss of predicted and actual, using BP. Why does not loss become zero at the end of the training?

   Ans1: Neural networks can not learn identity function.

   Ans2: Training by backpropagation will never allow to make loss zero or near zero.

   Ans3: The reason is the architecture. With more layers (deeper network), the loss will become zero or near zero.

   Loss does not become zero because we only want to minimize the loss on a non-convex surface locally in a neural network. [OR]
   bottleneck layer inability to capture data perfectly causes non-zero loss"

   Your Answer: ——————

7. Consider the following code snippet for convolution

```
conv = nn.Conv1d(in_channels=1,out_channels=5,
                 kernel_size=7,padding=3,bias=True)
input = torch.randn(12,1,100)
output = conv(input)
```

   What is the number of trainable parameters in conv? What is the "output" dimension?

   Ans: 40, 12 * 5 * 100

8. Consider this model:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(10,4, bias=False)
        self.fc2 = nn.Linear(4, 2, bias=True)
    def forward(self, x):
        x = self.fc1(x)
        x = nn.ReLU()(x)
        x = self.fc2(x)
        return x
```

   What is the number of trainable parameters in this model

   Ans: ——————

   50

9. Consider a problem of deciding an email as SPAM or NON-SPAM. We have a trained classifier/solution that scores each email as $\mathbf{w}^T\mathbf{x}$. However, our job will be to define a threshold so that we can say $\mathbf{w}^T\mathbf{x} \geq \theta$ for SPAMs

   For the 10 emails that we have the scores (i.e., $\mathbf{w}^T\mathbf{x}$) are

$$0.7, 0.65, 0.92, 0.75, 0.45, 0.8, 0.3, 0.8, 0.1, 0.1$$

and we know the truth/label as (SPAM (+) and NON-SPAM (-))

$$+, -, +, +, -, +, +, +, -, -$$

For θ as 0.5 and 0.2, find the "Precision" and "Recall".

Ans: (0.5): Precision: ————-; Recall: ————-

Ans: (0.2): Precision: ————-; Recall: ————-

10. Identify, none, one or more correct answers:

We define/know the weighted Euclidean distance

$$\tau(\mathbf{x}, \mathbf{y}) = [\mathbf{x} - \mathbf{y}]^T [\mathbf{A}][\mathbf{x} - \mathbf{y}]$$

Where $\mathbf{x}$ and $\mathbf{y}$ are vectors in d dimension $\mathbf{A}$ is a square matrix

(a) If $\mathbf{A}$ is a non-identity diagonal matrix. Then "dist" is a scaled version of Euclidean distance or "$\tau = \alpha$ Euclidean distance"

(b) If $k < d$ and $A = \sum_{i=1}^{k} \mathbf{u}_i \mathbf{u}_i^T$, with $\mathbf{u}_i$s as orthonormal. Then A is rank deficient and Euclidean τ can not be computed.

(c) If $k < d$ and $A = \sum_{i=1}^{k} \mathbf{u}_i \mathbf{u}_i^T$, with $\mathbf{u}_i$s as orthonormal. Then τ is equivalent to dimensioality reduction $\mathbf{x}' = \mathbf{W}\mathbf{x}$ with $\mathbf{W}$ as $k \times d$ matrix with $\mathbf{u}_i^T$ as the i th row.

(d) When $\mathbf{A}$ is non-diagonal symmetric matrix, $\tau(\mathbf{x}, \mathbf{y}) = \tau(\mathbf{y}, \mathbf{x})$.

(e) All the above are TRUE.

Ans: ACD

11. Bag I contain w_1 white and b_1 black balls. Bag II contains w_2 white and b_2 black balls.

A ball is drawn at random from one of the bags, and it is found to be white. What is the probability that it was drawn from Bag I.

(a) $\frac{w_1}{w_1+b_1}$

(b) $\frac{w_1(w_1+b_1)}{w_1(w_1+b_1)+w_2(w_2+b_2)}$

(c) $\frac{w_1(b_1+b_2)}{w_1(b_1+b_2)+w_2(b_1+b_2)}$

(d) $\frac{w_1(w_2+b_2)}{w_1(w_2+b_2)+w_2(w_1+bd_1)}$

(e) None of the above

Ans: D

12. What is "Self-Supervised Learning" ? Give an example. (not more than 3-4 sentences)

Ans:

Part II: Answer <u>three out of four</u> the following questions [60 min; $3 \times 20 = 60$ points]

1. **MLP** Consider an MLP neural network with four inputs; one hidden layer of 5 neurons and one output neuron. Hidden layer has ReLU activation and output neuron has linear activation. All neurons in the hidden and output layers have bias.

   Let W_1 be the weights from input to hidden layer and W_2 be the weights from hidden to the output layer.

   In W_1, all elements in the odd numbered rows is +1. All elements in the even numbered rows is -1.

   In W_2, all elements are +1. [4 + 4 + 2 + 8 + 2]

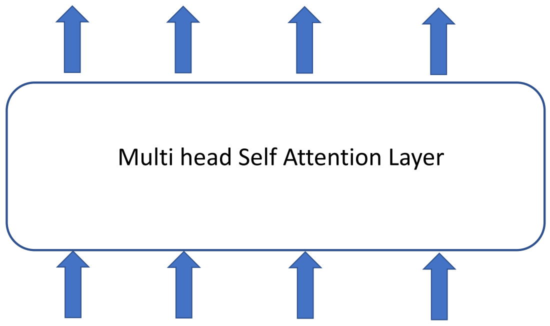
   (a) How many trainable weights are there in this neural network?

   (b) Find output for an input $[1, 2, 1, 4]^T$

   (c) If the target (truth) was 15, find a squared error loss $\mathcal{L}$ (i.e, $(t - o)^2$)

   (d) Find $\frac{\partial L}{\partial v}$, where v is the weight from first input to first hidden neuron.

   (e) Assuming a learning rate of 0.1, write the update equation for v and find the new value numerically.



2. Consider a multi head (two heads) self attention layer. $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ are the four inputs and $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$ are the outputs. $\mathbf{x}_i \in R^4$ (for the simplicity of numerical computation, assume no softmax within the self attention layer)

   - Head 1: All the three matrices W_Q, W_K and W_V are initialized as identity (square) matrices.
   - Head 2: All the three matrices W_Q, W_K and W_V have exactly the same size as of head 1. But W_Q has all its elements 1. Similarly W_K has all its elements -1 and W_V is identity.
   - W_O has all its elements initialized as 1. [5 + 8 + 7]

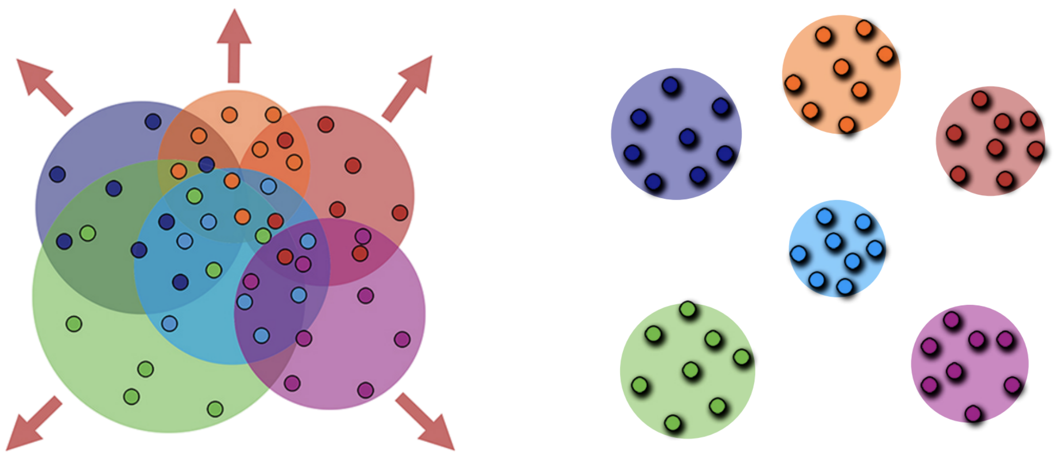
   (a) What are the dimensionality of all the seven learnable matrices. (no bias anywhere)

   (b) Compute $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$ if $\mathbf{x}_1 = [1, 1]^T, \mathbf{x}_2[-1, -1]^T, \mathbf{x}_3 = [2, 2]^T, \mathbf{x}_4 = [0, 0]^T$

   (c) Consider a similar simplified architecture to what is shown in figure. but with only two inputs and two outputs; and only one head. no $\mathbf{W_O}$ and W_Q, W_K and W_V are square matrices. Using chain rule find $\frac{\partial L}{\partial \mathbf{x}_1}$ if we know $\frac{\partial L}{\partial \mathbf{y}_i}$ (Advice: This question may be more involved; partial grades will be given based on how far you go. Avoid spending too much time.)

3. **Algorithm** You are given N examples $(\mathbf{x_i}, y_i)$. Write the **pseudocode** for stochastic minibatch gradient descent based minimization of the loss $L = \sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x_i})^2$.

   Write one word/phrase answer: (i) what are we (which variable) trying to find out? (ii) how do we initialize the iterations? (iii) is this problem convex? (iv) does the final answer depends on initialization/learning rate? (v) Suggest some steps to make sure that the implementation converge and reliable. (vi) suggest an alternate termination criteria (than the one used in the pseudo code). [4 + 2 + 2 + 2 + 2 + 4 + 4]



4. Design of a loss function: Consider a six class classification problem. We can design a neural network and train them using backpropagation. However, along with minmizing the loss on the given samples, it is perceived as if samples/classes can be well seperated as in figure (on right), it is preferred. [5 + 5 + 5 + 5]

   (a) Suggest a loss function for classification of six classes in a neural network setting.

   (b) Suggest how the geometric intuition here (figure) could be expressed analytically as a term to be added to the above loss function.

   (c) Is your loss function differentiable? Can this be computed in a single sample mode?

   (d) We often use a mini-batch SGD for training. Can we have a better mini batch construction (or sampling) for this type of losses?

Design of a Loss Function for Six-Class Classification

(a) Loss function for six-class classification

The most general and widely used loss for multi-class classification is the **Cross-Entropy Loss**. Assuming we use a softmax layer at the output:

Let:

- $\mathbf{z} \in \mathbb{R}^6$ be the logits output from the neural network.
- $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$, where

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{6} e^{z_j}}$$

- $y \in \{1, 2, \ldots, 6\}$ is the true class label.

Then the standard **Cross-Entropy Loss** is:

$$\mathcal{L}_{\text{CE}} = -\log(\hat{y}_y)$$

This encourages the network to assign high probability to the correct class.

(b) Adding geometric intuition as a regularization term

If the figure suggests geometrically well-separated classes (e.g., clusters or class centers far from each other), we can add a separation-promoting regularization term.

One widely accepted method is to introduce a **Center Loss** or a **Margin-based Loss**. A general form of such a geometric regularizer is:

$$\mathcal{L}_{\text{geom}} = \frac{1}{2} \left\| \mathbf{f} - \mathbf{c}_y \right\|^2$$

Where:

- $\mathbf{f} \in \mathbb{R}^d$ is the feature embedding of the sample before the final layer.
- $\mathbf{c}_y$ is the learned center for class y in the feature space.

This term pulls samples of the same class closer to their class center, improving intra-class compactness.

To encourage inter-class separation, we can use:

$$\mathcal{L}_{\text{inter}} = \sum_{i \neq j} \frac{1}{\left\| \mathbf{c}_i - \mathbf{c}_j \right\|^2}$$

This penalizes class centers that are too close, encouraging them to spread out in the embedding space.

The total loss becomes:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{geom}} + \lambda_2 \mathcal{L}_{\text{inter}}$$

where λ_1, λ_2 are tunable hyperparameters.

(c) Differentiability and single-sample computation

- **Differentiability:** Yes, the total loss is differentiable. All components — cross-entropy, Euclidean distance, and inverse norm — are differentiable with respect to the network parameters.
- **Single-sample mode:**
  - $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{geom}}$ can be computed per sample.
  - $\mathcal{L}_{\text{inter}}$ requires information about multiple class centers and is typically updated across a batch or over several iterations.

(d) Better mini-batch construction

To effectively train with such a loss, especially those involving both intra-class compactness and inter-class separation, we can use better sampling strategies:

Class-uniform mini-batches: For a dataset with $C = 6$ classes, sample K instances from each class per batch. This ensures:

- Each class center gets sufficiently updated.
- Inter-class terms are computed more reliably.
- Intra-class variation is represented in every batch.

Such sampling is commonly used in metric learning, contrastive learning, and face recognition tasks where geometric losses are crucial.

Part III: Answer <u>three out of four</u> the following questions [60 min; $3 \times 20 = 60$ points]

1. Given N training samples $\{(\mathbf{x}_i, y_i)\}$ where $\mathbf{x}_i \in R^d$ and $y_i \in \{-1, +1\}$, geometrically derive SVM (primal) objective. State the constraints.

   Given: Training samples:

$$\{(x_i, y_i)\}_{i=1}^N, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, +1\}$$

Step 1: Define the hyperplane

A separating hyperplane is defined as:

$$w \cdot x + b = 0$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

Step 2: Decision function

The decision function for classification:

$$f(x) = \text{sign}(w \cdot x + b)$$

Step 3: Margin calculation

For a point x_i, the distance to the hyperplane is:

$$d(x_i) = \frac{|w \cdot x_i + b|}{||w||}$$

For correctly classified points:

$$y_i(w \cdot x_i + b) > 0$$

Step 4: Support vectors and canonical hyperplane

We scale w and b so that for the closest points (support vectors):

$$w \cdot x + b = +1 \quad \text{(positive support vector)}$$

$$w \cdot x + b = -1 \quad \text{(negative support vector)}$$

Step 5: Margin width calculation

The distance from each margin to the hyperplane is:

$$\frac{1}{||w||}$$

So, total margin width is:

$$\frac{2}{||w||}$$

Step 6: Classification constraints

For all samples:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i \in \{1, 2, ..., N\}$$

Step 7: Optimization objective

To maximize margin, minimize $||w||$, or for convenience:

$$\min_{w,b} \frac{1}{2}||w||^2$$

Step 8: Primal SVM (Hard Margin)

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(w \cdot x_i + b) \geq 1, \quad \forall i \end{aligned}$$

Step 9: Soft Margin SVM (Non-separable data)

Introduce slack variables $\xi_i \geq 0$ and penalty parameter C:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

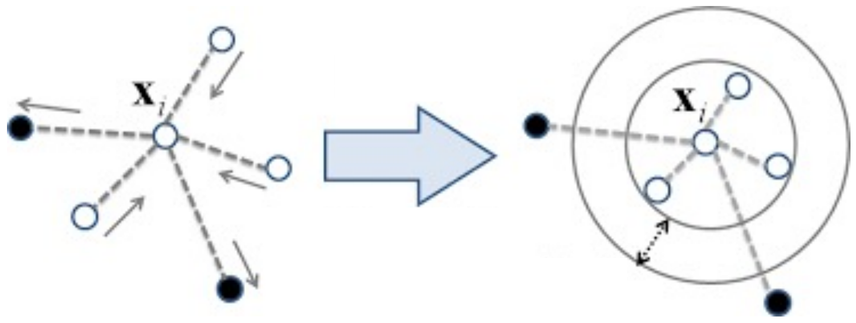
2. We know that a kernel $\kappa(\mathbf{p}, \mathbf{q}) = \phi(\mathbf{p})^T \phi(\mathbf{q})$.

   Consider a data set of N samples $\{\mathbf{x}_i\}$ and a Kernel Matrix K such that (i, j) th element of this matrix is $\kappa(\mathbf{x}_i, \mathbf{x}_j)$. [2 + 8 + 10]

   (a) Show that K symmetric

   (b) Show that Kernel matrix (i.e., K) is PSD (Positive Semi Definite)

   (c) Find $\phi(\cdot)$ if $\kappa(\mathbf{p}, \mathbf{q}) = (\mathbf{p}^T \mathbf{q})^3$ and $\mathbf{p} = [p_1, p_2]^T$; $\mathbf{q} = [q_1, q_2]^T$



3. Consider a data set of N samples $\{\mathbf{x}_i\}$ where samples are either positive or negative (two class). (Classes need not be well separated as in the figure). We are using a nearest neighbour algorithm for classification using a simple distance function $d(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{x}_i - \mathbf{x}_j]^T [\mathbf{x}_i - \mathbf{x}_j]$ [5 + 5 + 5 + 5]

   (a) However, if we can "preprocess" the data as shown in the figure such that all class A (empty circle in figure) come closer, it will help. Let us do this using a feature transformation $\mathbf{z}_i = \mathbf{L}\mathbf{x}_i$. Now, write the distance by including $\mathbf{L}$. Considering just class A: How do we formulate the (maximization/minimization) problem for finding $\mathbf{L}$. (No need to solve. Make sure your function is scalar valued. Check dimensionalities!).

   (b) What are the pitfalls if $\mathbf{L}$ is NULL (all elements zero)? How do we constrain $\mathbf{L}$ to avoid this pitfall?

   (c) It is argued that $d(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{x}_i - \mathbf{x}_j]^T \mathbf{A} [\mathbf{x}_i - \mathbf{x}_j]$ can do this job. If so, relate $\mathbf{A}$ and $\mathbf{L}$

   (d) To minimize any potential confusion, it is good to give a safe "margin" for all samples in class B (filled circles in figure) from that of class A. Write an objective function and optimization problem to achieve that.

4. **Gradient Descent Optimization** [9 + 2 + 9]

(a) Consider a function $f(w) = w^2 + 2w + 1$. We want to minimize $f(w)$ using GD. Initialize w^0 as 5. Choose the learning rate η as 0.1. Show two iterations i.e., compute the value of w after first and second iterations.

(b) Suggest a learning rate better than 0.1 so that it converges faster.

(c) Suggest a learning rate that will take you to the optimal solution in one step i.e., w^1. How do we arrive at this "optimal" learning rate, in general?

ROUGH WORK

ROUGH WORK

ROUGH WORK