1 Section-1

2 Section-2

2.1 Question-1

2.2 Question-2

Problem Recap:

Consider a two-head self-attention layer with inputs $x_1, x_2, x_3, x_4 \in R^2$ and outputs $y_1, y_2, y_3, y_4 \in R^2$. There is no softmax inside the attention.

- **Head 1:** $W_Q^1, W_K^1, W_V^1 = I_{2\times 2}$.
- **Head 2:** $W_Q^2, J_{2\times 2},\ W_K^2 = -J_{2\times 2},\ W_V^2 = I_{2\times 2}$, where J has all entries 1.
- Output projection: $W_O \in R^{2\times 4}$ with all entries 1 .

You were required to answer the following:

1. **Dimensionalities.** What are the sizes of
$$W_Q^1, W_K^1, W_V^1,\ W_Q^2, W_K^2, W_V^2,\ W_O?$$
2. **Numerical outputs.** Compute
$$y_i \ = \ W_O \Big[\underbrace{y_i^{(1)}}_{\text{head1}} \ \Big\| \ \underbrace{y_i^{(2)}}_{\text{head2}} \Big] \quad \text{for } i = 1, 2, 3, 4,$$
where for each head h,
$$y_i^{(h)} = \sum_{j=1}^{4} \big(Q_i^{(h)} \cdot K_j^{(h)}\big)\, V_j^{(h)}, \quad Q_i^{(h)} = W_Q^h\, x_i,\ K_j^{(h)} = W_K^h\, x_j,\ V_j^{(h)} = W_V^h\, x_j.$$
3. **Gradients.** In a simplified single-head, two-input analogue (no W_O, all square), use the chain rule to write
$$\frac{\partial \mathcal{L}}{\partial x_1} \quad \text{in terms of} \quad \frac{\partial \mathcal{L}}{\partial y_1},\ W_Q,\ W_K,\ W_V,\ \{x_j\}.$$

Solution.

1. Since $x_i \in R^2$ and each head does
$$Q, K, V : R^2 \to R^2,$$
each of
$$W_Q^1,\ W_K^1,\ W_V^1,\ W_Q^2,\ W_K^2,\ W_V^2$$
is 2×2. We then concatenate two 2-vectors into a 4-vector and apply
$$W_O : R^4 \to R^2,$$
so
$$W_O \in R^{2\times 4}.$$

2. **Head 1:** since $W_Q^1 = W_K^1 = W_V^1 = I$,

$$y_i^{(1)} = \sum_{j=1}^{4} (x_i \cdot x_j)\, x_j.$$

A quick table of dot-products and sums gives

$$y_1^{(1)} = [12, 12],\ y_2^{(1)} = [-12, -12],\ y_3^{(1)} = [24, 24],\ y_4^{(1)} = [0, 0].$$

Head 2: here

$$Q_i^{(2)} = Jx_i = (\mathbf{1}^\top x_i)\,[1, 1]^\top, \quad K_j^{(2)} = -Jx_j = -(\mathbf{1}^\top x_j)\,[1, 1]^\top,$$

so

$$Q_i^{(2)} \cdot K_j^{(2)} = -2\,(\mathbf{1}^\top x_i)\,(\mathbf{1}^\top x_j).$$

Writing $s_j = \mathbf{1}^\top x_j$, one shows

$$y_i^{(2)} = \sum_{j=1}^{4} \big(-2\, s_i\, s_j\big)\, x_j = -2\, s_i \sum_{j=1}^{4} s_j\, x_j = \begin{cases} [-48, -48], & i = 1, \\ [48, 48], & i = 2, \\ [-96, -96], & i = 3, \\ [0, 0], & i = 4. \end{cases}$$

Concat & output: for each i, form $[\, y_i^{(1)} \| y_i^{(2)}] \in R^4$ and multiply by W_O (all ones). Summing the four entries gives

$$y_1 = [-72, -72],\ y_2 = [72, 72],\ y_3 = [-144, -144],\ y_4 = [0, 0].$$

3. **Single-head toy:** let

$$y_i = \sum_{j=1}^{2} (W_Q x_i \cdot W_K x_j)\,(W_V x_j).$$

Then by the chain rule

$$\frac{\partial \mathcal{L}}{\partial x_1} = \sum_{h=Q,K,V} \frac{\partial \mathcal{L}}{\partial y_1}\, \frac{\partial y_1}{\partial h}\, \frac{\partial h}{\partial x_1},$$

and one shows for example

$$\frac{\partial y_1}{\partial x_1} = \sum_j \Big[(W_V x_j)(W_K x_j)^\top W_Q \ +\ (W_Q x_1 \cdot W_K x_j)\, W_V \Big].$$

Thus

$$\frac{\partial \mathcal{L}}{\partial x_1} = \Big[\sum_j (W_V x_j)(W_K x_j)^\top W_Q + \sum_j (W_Q x_1 \cdot W_K x_j)\, W_V \Big]^\top \frac{\partial \mathcal{L}}{\partial y_1}.$$

Marking Rubric for Question 2 (20 marks)

2.3 Question-3

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

Part	Criterion	Marks
(a)	Correct sizes for all six head matrices (2×2)	3
	Correct size for W_O (2×4)	2
(b)	Head 1: correct dot-product sums and $y_i^{(1)}$ values	4
	Head 2: correct form of Q, K and numerical $y_i^{(2)}$	4
(c)	Chain-rule structure, identifying all three Jacobian terms	4
	Correct final expression in terms of W_Q, W_K, W_V, $\frac{\partial \mathcal{L}}{\partial y_1}$	3

Pseudocode for Stochastic Minibatch Gradient Descent

Inputs:

- Dataset $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$
- Learning rate $\eta > 0$
- Batch size m (where $1 \leq m \leq N$)
- Number of epochs E (or other termination criteria)

Output:

- Optimized weight vector $\mathbf{w}$

Initialize weight vector $\mathbf{w}$ (e.g., with small random numbers or zeros).
for epoch from 1 to E **do**
    Shuffle the dataset D.
    for $j = 0, 1, \ldots, \lfloor (N-1)/m \rfloor$ **do**
        Get minibatch $B_j = \{(\mathbf{x}_k, y_k)\}$ for k from $j \cdot m$ to $\min((j+1) \cdot m - 1, N-1)$.
        Let $m_j = |B_j|$ be the actual size of the current minibatch.
        Calculate the gradient of the loss for the current minibatch B_j:
        $\nabla_{\mathbf{w}} L_{B_j} = \frac{1}{m_j} \sum_{(\mathbf{x}_k, y_k) \in B_j} \frac{\partial (y_k - \mathbf{w}^T \mathbf{x}_k)^2}{\partial \mathbf{w}}$
        $\nabla_{\mathbf{w}} L_{B_j} = \frac{1}{m_j} \sum_{(\mathbf{x}_k, y_k) \in B_j} 2(y_k - \mathbf{w}^T \mathbf{x}_k)(-\mathbf{x}_k)$
        $\nabla_{\mathbf{w}} L_{B_j} = -\frac{2}{m_j} \sum_{(\mathbf{x}_k, y_k) \in B_j} (y_k - \mathbf{w}^T \mathbf{x}_k)\mathbf{x}_k$
        Update the weights:
        $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L_{B_j}$
return $\mathbf{w}$

Write one word/phrase answer

1. **What are we (which variable) trying to find out?**
   The weight vector $\mathbf{w}$ (or "optimal weights").
2. **How do we initialize the iterations?**
   Initialize $\mathbf{w}$ randomly or to zeros.
3. **Is this problem convex?**
   Yes (the loss function is convex).
4. **Does the final answer depends on initialization/learning rate?**
   - **Initialization:** No (for convex problems with a unique minimum, it converges to the same global minimum, assuming appropriate learning rate).

- **Learning rate:** Yes (affects convergence speed, stability, and how close to the true optimum the solution gets in a finite number of steps. A very poor learning rate can lead to divergence or a sub-optimal solution).

5. **Suggest some steps to make sure that the implementation converge and reliable.**
   - Monitor training/validation loss.
   - Tune the learning rate (e.g., use a learning rate schedule, try different values).
   - Normalize/standardize input features.
   - Gradient checking (to ensure correct gradient implementation).
   - Ensure proper shuffling of data each epoch.
6. **Suggest an alternate termination criteria (than the one used in the pseudo code).**
   - Stop when the change in loss between epochs falls below a small threshold.
   - Stop when the validation loss starts to increase or plateaus (early stopping).
   - Stop when the magnitude of the gradient (or change in weights) falls below a small threshold.

2.4 Question-4

3 Section-3

3.1 Question-1

3.2 Question-2

Solution to Question 2

We are given 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 the (i, j)-th element of this matrix is $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

(a) Show that K is symmetric

A matrix K is symmetric if $K = K^T$, which means $K_{ij} = K_{ji}$ for all i, j. The (i, j)-th element of K is given by:

$$K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

The (j, i)-th element of K is given by:

$$K_{ji} = \kappa(\mathbf{x}_j, \mathbf{x}_i) = \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i)$$

Since $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ is a scalar (it's a dot product), its transpose is itself:

$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j))^T$$

Using the property $(AB)^T = B^T A^T$, and noting that $(\phi(\mathbf{x}_i)^T)^T = \phi(\mathbf{x}_i)$:

$$\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \phi(\mathbf{x}_j)^T (\phi(\mathbf{x}_i)^T)^T = \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i)$$

Thus, we have:

$$K_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i) = K_{ji}$$

Therefore, K is symmetric.

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

A symmetric matrix K is Positive Semi-Definite (PSD) if for any non-zero vector $\mathbf{z} \in \mathbb{R}^N$, $\mathbf{z}^T K\mathbf{z} \geq 0$. Let $\mathbf{z} = [z_1, z_2, \ldots, z_N]^T$ be an arbitrary non-zero vector.

$$\begin{aligned}
\mathbf{z}^T K\mathbf{z} &= \sum_{i=1}^{N}\sum_{j=1}^{N} z_i K_{ij} z_j \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} z_i \left(\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)\right) z_j \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} (z_i \phi(\mathbf{x}_i)^T)(\phi(\mathbf{x}_j) z_j) \\
&= \left(\sum_{i=1}^{N} z_i \phi(\mathbf{x}_i)^T\right)\left(\sum_{j=1}^{N} z_j \phi(\mathbf{x}_j)\right)
\end{aligned}$$

Let $\mathbf{v} = \sum_{k=1}^{N} z_k \phi(\mathbf{x}_k)$. Then $\mathbf{v}$ is a vector (in the feature space). The expression above can be written as:

$$\mathbf{z}^T K\mathbf{z} = \mathbf{v}^T\mathbf{v} = ||\mathbf{v}||^2$$

Since the squared norm of any vector is non-negative, i.e., $||\mathbf{v}||^2 \geq 0$, we have:

$$\mathbf{z}^T K\mathbf{z} \geq 0$$

Therefore, the Kernel matrix K is Positive Semi-Definite.

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

We are given $\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$ and $\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$. First, calculate $\mathbf{p}^T\mathbf{q}$:

$$\mathbf{p}^T\mathbf{q} = p_1q_1 + p_2q_2$$

Now, we need to compute $(\mathbf{p}^T\mathbf{q})^3$:

$$\begin{aligned}
\kappa(\mathbf{p}, \mathbf{q}) &= (p_1q_1 + p_2q_2)^3 \\
&= (p_1q_1)^3 + 3(p_1q_1)^2(p_2q_2) + 3(p_1q_1)(p_2q_2)^2 + (p_2q_2)^3 \\
&= p_1^3q_1^3 + 3p_1^2p_2q_1^2q_2 + 3p_1p_2^2q_1q_2^2 + p_2^3q_2^3 \\
&= (p_1^3)(q_1^3) + (\sqrt{3}p_1^2p_2)(\sqrt{3}q_1^2q_2) + (\sqrt{3}p_1p_2^2)(\sqrt{3}q_1q_2^2) + (p_2^3)(q_2^3)
\end{aligned}$$

We want to find $\phi(\cdot)$ such that $\kappa(\mathbf{p}, \mathbf{q}) = \phi(\mathbf{p})^T\phi(\mathbf{q})$. Let $\phi(\mathbf{p}) = [\phi_1(\mathbf{p}), \phi_2(\mathbf{p}), \phi_3(\mathbf{p}), \phi_4(\mathbf{p})]^T$. Then $\phi(\mathbf{p})^T\phi(\mathbf{q}) = \phi_1(\mathbf{p})\phi_1(\mathbf{q}) + \phi_2(\mathbf{p})\phi_2(\mathbf{q}) + \phi_3(\mathbf{p})\phi_3(\mathbf{q}) + \phi_4(\mathbf{p})\phi_4(\mathbf{q})$. Comparing the terms, we can identify the components of $\phi(\mathbf{p})$:

- $\phi_1(\mathbf{p})\phi_1(\mathbf{q}) = p_1^3q_1^3 \implies \phi_1(\mathbf{p}) = p_1^3$
- $\phi_2(\mathbf{p})\phi_2(\mathbf{q}) = 3p_1^2p_2q_1^2q_2 \implies \phi_2(\mathbf{p}) = \sqrt{3}p_1^2p_2$
- $\phi_3(\mathbf{p})\phi_3(\mathbf{q}) = 3p_1p_2^2q_1q_2^2 \implies \phi_3(\mathbf{p}) = \sqrt{3}p_1p_2^2$
- $\phi_4(\mathbf{p})\phi_4(\mathbf{q}) = p_2^3q_2^3 \implies \phi_4(\mathbf{p}) = p_2^3$

So, the feature mapping $\phi(\mathbf{p})$ can be defined as:

$$\phi(\mathbf{p}) = \begin{bmatrix} p_1^3 \\ \sqrt{3}p_1^2 p_2 \\ \sqrt{3}p_1 p_2^2 \\ p_2^3 \end{bmatrix}$$

Note that other forms are possible by multiplying components by -1 (e.g., $\phi_2(\mathbf{p}) = -\sqrt{3}p_1^2 p_2$ and $\phi_2(\mathbf{q}) = -\sqrt{3}q_1^2 q_2$), but this is a standard representation.

3.3 Question-3

1. Define $z_i = Lx_i \in \mathbb{R}^m$. Then

   $$d_L(x_i, x_j) = \|z_i - z_j\|^2 = \|L(x_i - x_j)\|^2 = (x_i - x_j)^\top L^\top L\,(x_i - x_j).$$

   Considering only class A with index set $\mathcal{S}_A$, we minimize

   $$\min_L \sum_{i,j \in \mathcal{S}_A} \|L(x_i - x_j)\|^2,$$

   where $x_i \in \mathbb{R}^n$, $L \in \mathbb{R}^{m \times n}$, so the objective is scalar.

2. If $L = \mathbf{0}$, then $d_L(x_i, x_j) = 0$ for all pairs and the objective is trivially minimized, but there is no discrimination. To avoid this, impose e.g. the constraint

   $$\|L\|_F^2 = 1 \quad \text{or} \quad \operatorname{trace}(L^\top L) = 1 \quad \text{or} \quad L^\top L = I_n,$$

   or add a penalty $\lambda\|L\|_F^2$.

3. A Mahalanobis distance

   $$d_A(x_i, x_j) = (x_i - x_j)^\top A\,(x_i - x_j)$$

   matches

   $$d_L(x_i, x_j) = \|L(x_i - x_j)\|^2 = (x_i - x_j)^\top (L^\top L)(x_i - x_j),$$

   hence

   $$\boxed{A = L^\top L.}$$

4. To enforce a margin $m > 0$ between class A and B:

   $$\max_{L,m} m \quad \text{s.t.} \quad d_L(x_i, x_k) - \max_{p,q \in \mathcal{S}_A} d_L(x_p, x_q) \geq m,\ \forall\, i \in \mathcal{S}_A,\ k \in \mathcal{S}_B, \quad \|L\|_F^2 = 1.$$

   Alternatively, trade off tightness of A vs. separation from B by

   $$\min_L \sum_{i,j \in \mathcal{S}_A} d_L(x_i, x_j) - \lambda \sum_{i \in \mathcal{S}_A, k \in \mathcal{S}_B} d_L(x_i, x_k) \quad \text{s.t. } \|L\|_F^2 = 1.$$

Table 1: Marking Rubric for Question 3

Part	Criterion	Marks
(a)	Correct expression of transformed distance d_L	2
	Clear minimization objective over class A pairs	2
	Dimensionality check (scalar objective, matrix sizes consistent)	1
(b)	Reasoning of collapse when $L = 0$	2
	Valid constraint or regularization suggestion	3
(c)	Correct relation $A = L^\top L$	3
	Brief explanation why this factorization recovers the metric	2
(d)	Formulation of a margin-based objective or equivalent constraint	3
	Clear inclusion of both the separation objective and constraint	2

3.4 Question-4

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 three out of four the following questions [60 min; 3 × 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]

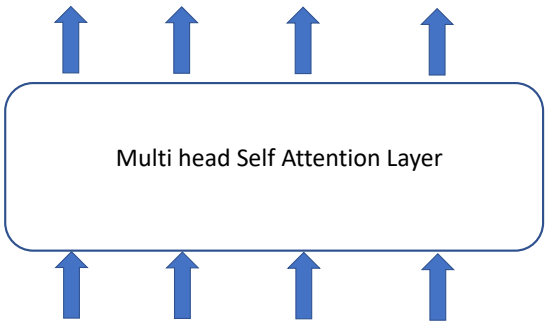
   (a) How many trainable weights are there in this neural network?

   (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. $[5 + 8 + 7]$

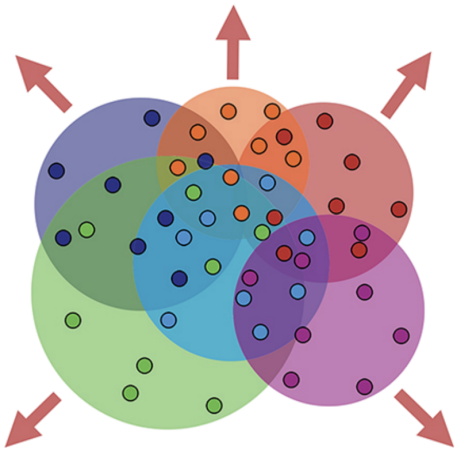
   (a) What are the dimensionality of all the seven learnable matrices. (no bias anywhere)

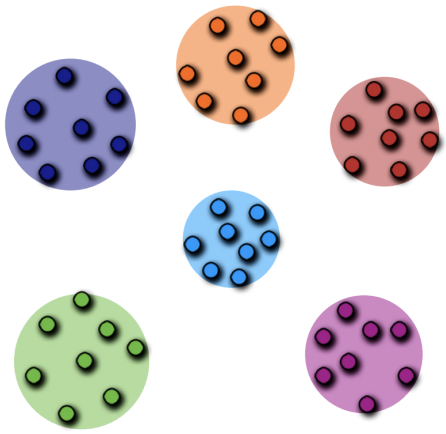
   (b) Compute $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$ if $\mathbf{x}_1 = [1, 1]^T, \mathbf{x}_2[-1, -1]^T, \mathbf{x}_3 = [2, 2]^T, \mathbf{x}_4 = [0, 0]^T$

   (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?

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:

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

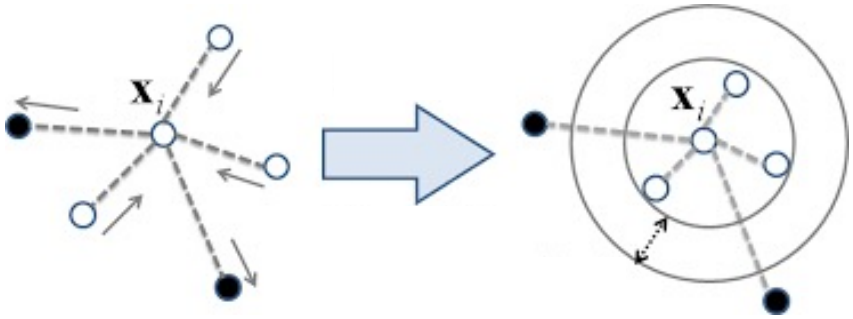
2. We know that a kernel $\kappa(\mathbf{p}, \mathbf{q}) = \phi(\mathbf{p})^T \phi(\mathbf{q})$.

   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