



1













Profs. Nicolas Flammarion and Martin Jaggi  
Machine Learning – CS-433 - IC  
18.01.2024 from 15h15 to 18h15 in STCC  
Duration : 180 minutes

# Student One

SCIPER: 111111

Do not turn the page before the start of the exam. This document is double-sided, has 20 pages, the last ones are possibly blank. Do not unstaple.

- This is a closed book exam. No electronic devices of any kind.
- Place on your desk: your student ID, writing utensils, one double-sided A4 page cheat sheet if you have one; place all other personal items below your desk.
- You each have a different exam.
- This exam has many questions. We do *not* expect you to solve all of them even for the best grade
- Only answers in this booklet count. No extra loose answer sheets. You can use the last two pages as scrap paper.
- For the **multiple choice** questions, we give :
  - +2 points if your answer is correct,
  - 0 points for incorrect or no answer
- For the **true/false** questions, we give :
  - +1.5 points if your answer is correct,
  - 0 points for incorrect or no answer
- Use a **black or dark blue ballpen** and clearly erase with **correction fluid** if necessary.
- If a question turns out to be wrong or ambiguous, we may decide to nullify it.

Respectez les consignes suivantes   Observe this guidelines   Beachten Sie bitte die unten stehenden Richtlinien		
choisir une réponse   select an answer Antwort auswählen	ne PAS choisir une réponse   NOT select an answer NICHT Antwort auswählen	Corriger une réponse   Correct an answer Antwort korrigieren
  		 
ce qu'il ne faut <b>PAS</b> faire   what should <b>NOT</b> be done   was man <b>NICHT</b> tun sollte		
     		



## First part: multiple choice questions

For each question, mark the box corresponding to the correct answer. Each question has **exactly one** correct answer.

### Linear regression / Loss functions

Figure 1 below shows the three different datasets (A, B, C) and the same linear model used to predict each of these datasets.

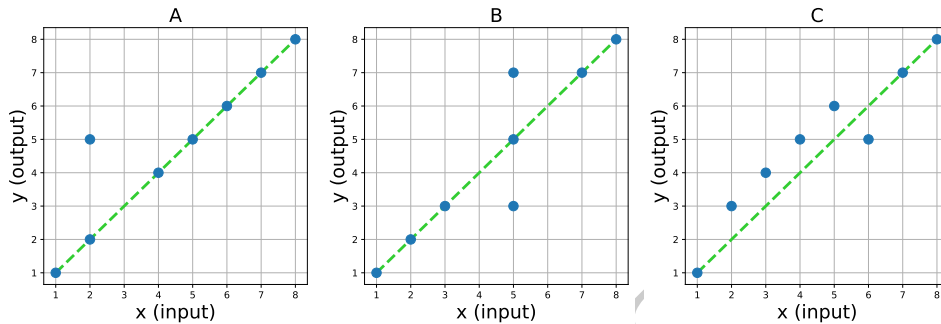


Figure 1: 3 Datasets (A, B, C), and a linear model represented by the dashed line (from  $\mathbb{R}^1$  to  $\mathbb{R}^1$ )

**Question 1** Choose the correct ordering with respect to MSE (Mean Squared Error) loss

- ☐  $\text{MSE}_B > \text{MSE}_A > \text{MSE}_C$
- ☐  $\text{MSE}_B > \text{MSE}_C > \text{MSE}_A$
- ☐ At least two of the MSE losses are equal.
- ☐  $\text{MSE}_A > \text{MSE}_C > \text{MSE}_B$
- ☐  $\text{MSE}_C > \text{MSE}_B > \text{MSE}_A$
- ☒  $\text{MSE}_A > \text{MSE}_B > \text{MSE}_C$
- ☐  $\text{MSE}_C > \text{MSE}_A > \text{MSE}_B$

**Solution:**  $\text{MSE}_A = 9/8$ ,  $\text{MSE}_B = 8/8$ ,  $\text{MSE}_C = 5/8$

**Question 2** Choose the correct ordering with respect to MAE (Mean Absolute Error) loss

- ☐  $\text{MAE}_A > \text{MAE}_C > \text{MAE}_B$
- ☐  $\text{MAE}_B > \text{MAE}_C > \text{MAE}_A$
- ☐  $\text{MAE}_A > \text{MAE}_B > \text{MAE}_C$
- ☐  $\text{MAE}_C > \text{MAE}_A > \text{MAE}_B$
- ☐  $\text{MAE}_B > \text{MAE}_A > \text{MAE}_C$
- ☐ At least two of the MAE losses are equal.
- ☒  $\text{MAE}_C > \text{MAE}_B > \text{MAE}_A$

**Solution:**  $\text{MAE}_A = 3/8$ ,  $\text{MAE}_B = 4/8$ ,  $\text{MAE}_C = 5/8$



## Log-likelihood

**Question 3** Let us assume that our data is generated by the model  $y_n = \mathbf{x}_n^\top \mathbf{w}_{\text{true}} + \varepsilon_n$ , where  $\varepsilon_n$  follows a random distribution such that  $p(y_n | \mathbf{x}_n, \mathbf{w}_{\text{true}}) = \frac{1}{2b} e^{-\frac{1}{b} |y_n - \mathbf{x}_n^\top \mathbf{w}_{\text{true}}|}$ , and  $b$  is a fixed parameter. We would like to maximize log-likelihood to find a good estimate of  $\mathbf{w}_{\text{true}}$ . Which of the following formulations is **NOT** equivalent to MLE?

- ☐  $\operatorname{argmin}_{\mathbf{w}} \sum_{n=1}^N |y_n - \mathbf{x}_n^\top \mathbf{w}|$
- ☐  $\operatorname{argmin}_{\mathbf{w}} \sum_{n=1}^N \frac{1}{b} |y_n - \mathbf{x}_n^\top \mathbf{w}|$
- ☐  $\operatorname{argmax}_{\mathbf{w}} \prod_{n=1}^N p(y_n | \mathbf{x}_n, \mathbf{w})$
- ☒  $\operatorname{argmin}_{\mathbf{w}} \log(p(\mathbf{y} | \mathbf{x}, \mathbf{w}))$

**Solution:** Maximizing the likelihood is the same as maximizing its log and minimizing its negative log. The constant factor  $\frac{1}{b}$  does not change the value.

## Optimization

**Question 4** We are using Gradient Descent to find the 1-dimensional global minimum  $w^*$  by optimizing the loss function  $\mathcal{L}(w)$  at iteration  $t$ .  $\mathcal{L}(w)$  is strictly convex, so it has a unique minimum. If  $w^t > w^*$ , what is true about the gradient of the loss function,  $\nabla \mathcal{L}(w^t)$ , and the next iteration of the parameter  $w^{t+1}$ ?

- ☒  $\nabla \mathcal{L}(w^t) > 0$  and  $w^{t+1} < w^t$
- ☐  $\nabla \mathcal{L}(w^t) < 0$  and  $w^{t+1} < w^t$
- ☐  $\nabla \mathcal{L}(w^t) > 0$  and  $w^{t+1} > w^t$
- ☐  $\nabla \mathcal{L}(w^t) < 0$  and  $w^{t+1} > w^t$

**Solution:** Take, for example, the MSE loss function. If  $w^t > w^*$ , it means that the current parameter is greater than the optimum and at this point the gradient is positive. In a Gradient Descent optimization step  $w^{t+1} = w^t - \gamma \nabla \mathcal{L}(w^t)$ , the parameter moves closer to the optimum and therefore decreases.

## Logistic Regression

**Question 5** Which of the following statements is **true** about the logistic regression model?

- ☐ Logistic regression gives a max-margin classifier
- ☐ By minimizing negative log-likelihood, we can obtain a closed-form solution for logistic regression
- ☐ In logistic regression, we calculate the weights  $\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ , and then fit responses as  $\hat{\mathbf{y}} = \sigma(\mathbf{X} \hat{\boldsymbol{\theta}})$
- ☒ If we run Gradient Descent to solve a logistic regression task on linearly separable data, the weights will not converge

**Solution:** There is no closed-form solution when minimizing negative log-likelihood for logistic regression. We cannot solve for  $\hat{\boldsymbol{\theta}}$  analytically in logistic regression like in linear regression. Optimization techniques like GD or Newton methods are required. Logistic regression finds any solution that separates two classes. To solve logistic regression, we maximize log likelihood, i.e.  $\max_{\boldsymbol{\theta}} \log \prod_{n=1}^N \sigma(\mathbf{x}_n^\top \boldsymbol{\theta})^{y_n} [1 - \sigma(\mathbf{x}_n^\top \boldsymbol{\theta})]^{1-y_n}$ . For linearly separable case, by increasing  $\|\boldsymbol{\theta}\|$ , one could always increase the likelihood. The weights can thus go to  $\infty$ .

**k-NN**

**Question 6** Recall the bound on the 1-NN generalization error for  $(X, Y) \sim \mathcal{D}$  over  $\mathcal{X} \times \mathcal{Y} = [0, 1]^d \times \{0, 1\}$ :

$$\mathbb{E}_{S_{\text{train}}} [L(f_{S_{\text{train}}})] \leq 2L(f_*) + 4c\sqrt{d}N^{-\frac{1}{d+1}},$$

where  $f_{S_{\text{train}}}$  is the 1-NN classifier with the train dataset  $S_{\text{train}}$ ,  $f_*(\mathbf{x}) = \mathbf{1}_{\eta(\mathbf{x}) \geq \frac{1}{2}}$  is the Bayes optimal classifier and  $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 | X = \mathbf{x})$  is a  $c$ -Lipschitz function.

Now, assume that the same setting is verified but  $X$  is now distributed over  $[0, 1]^d \cap L$  where  $L$  is a  $p$ -dimensional plane inside  $\mathbb{R}^d$  with  $p < d$ . How does the number of samples  $N$  need to scale in terms of  $p$  or  $d$  to guarantee that the worst-case error does not exceed a fixed  $\varepsilon$  with 1-NN classifier?

- ☐  $N \propto p$
- ☐  $N \propto d$
- ☒  $N \propto 2^p$
- ☐  $N \propto 2^d$

**Solution:** This is a 1-NN on a lower-dimensional space, so the number of samples  $N$  need to scale as  $2^p$  for a fixed worst-case error.

**K-means clustering**

**Question 7** Consider  $K$ -means with a modification to the loss seen in class such that each data point  $\mathbf{x}_n$  is now weighted according to function  $w$  with weight  $w(\mathbf{x}_n) \geq 0$ . The optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{z}, \boldsymbol{\mu}} \mathcal{L}(\mathbf{z}, \boldsymbol{\mu}) &= \sum_{n=1}^N w(\mathbf{x}_n) \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 \\ \text{s.t. } \boldsymbol{\mu}_k &\in \mathbb{R}^D, z_{nk} \in \{0, 1\}, \sum_{k=1}^K z_{nk} = 1, \\ \text{where } \mathbf{z}_n &= [z_{n1}, z_{n2}, \dots, z_{nK}]^\top, \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N]^\top, \boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K]^\top \end{aligned}$$

Following the same strategy as for K-means, the new iterative updates become

- ☐  $z_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j w(\mathbf{x}_n) \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } \boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk} \mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$
- ☐  $z_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j w(\mathbf{x}_n) \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } \boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk} w(\mathbf{x}_n) \mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$
- ☐  $z_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } \boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk} \mathbf{x}_n}{\sum_{n=1}^N z_{nk}}$
- ☒  $z_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } \boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk} w(\mathbf{x}_n) \mathbf{x}_n}{\sum_{n=1}^N z_{nk} w(\mathbf{x}_n)}$

**Solution:** As for coordinate descent with  $K$ -means, (1) optimizing for  $z_{nk}$  yields the center closest to  $\mathbf{x}_n$ , its weight doesn't have any effect, and (2) optimizing for the centers yields a weighted average over the points assigned to the cluster. An easy way to eliminate some wrong solutions is to test with say  $w(\mathbf{x}_n) = 2$  which is a constant and shouldn't change the solution



## GMM & K-means

### Question 8

Four simulated datasets are illustrated in Figure 2. As you can see, we have 3 distinct clusters in each dataset. We are interested in applying either K-means clustering or a Gaussian mixture model (GMM) for recovering the correct clustering (meaning that the clustering algorithm correctly assigns each data point to its true underlying cluster). Assuming sufficient initializations, which of the following statements is correct?

- ☐ Only GMM can perfectly cluster Dataset B.
- ☒ Only GMM can perfectly cluster Dataset D.
- ☐ K-means cannot perfectly cluster any of the datasets due to the presence of non-spherical clusters.
- ☐ Only GMM can perfectly cluster Dataset C.
- ☐ Only GMM can perfectly cluster Dataset A.

**Solution:** By drawing perpendicular bisectors between the centroids, we can see that in Dataset D, the leftmost cluster considers the perpendicular bisector (this is a drawing idea to understand what k-means does). As a result, K-means cannot perfectly cluster Dataset D and GMM is the only option. In all other datasets, K-means can perfectly cluster the (finite) dataset when we draw the perpendicular bisectors between each pair of centroids.

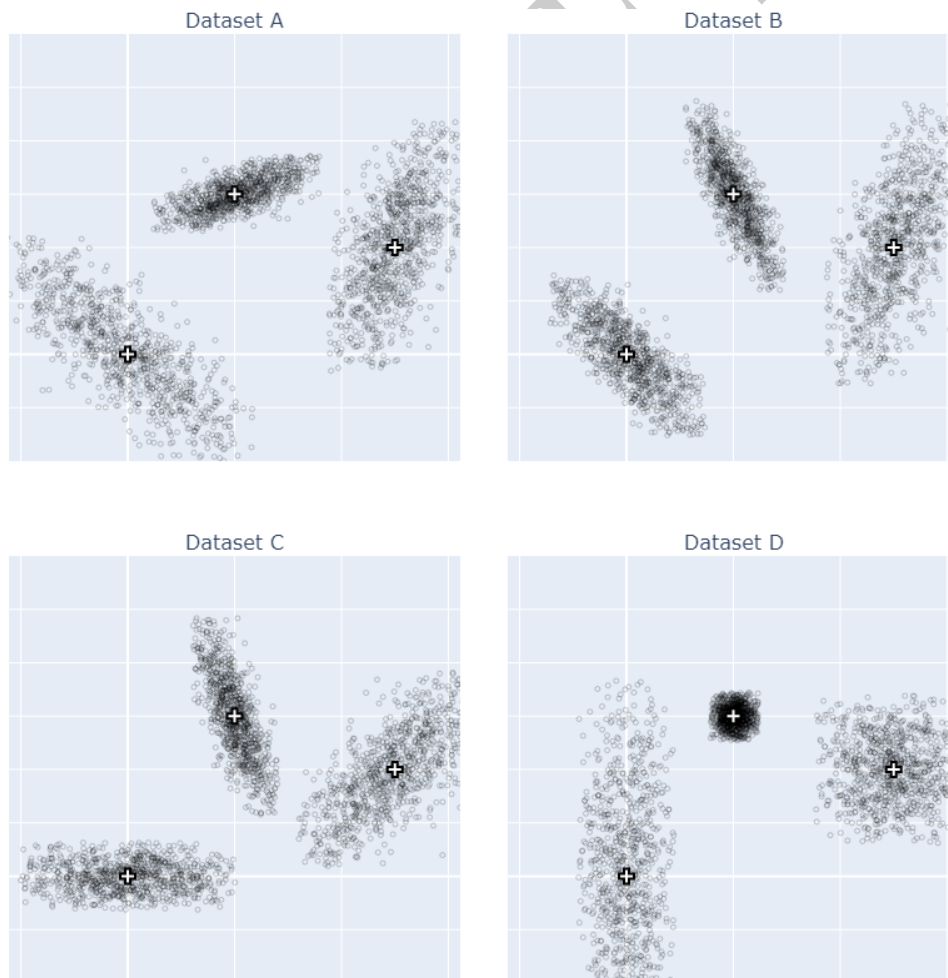


Figure 2: 4 simulated datasets. The center of each cluster is marked with a +.



## Neural Networks & Deep Learning

**Question 9** Consider an alternative attention mechanism where the attention weights  $p_{i,j}$  forming  $\mathbf{P}$  and the outputs  $\mathbf{Z}$  are given by:

$$p_{i,j} = \frac{\text{ReLU}(\mathbf{q}_i \mathbf{k}_j^\top)}{\sum_{t=1}^{T_{in}} \text{ReLU}(\mathbf{q}_i \mathbf{k}_t^\top)}$$
$$\mathbf{Z} = \mathbf{P}\mathbf{V}$$

Here all vectors are row vectors like in the transformer lecture. Ignore cases where the denominator for  $p_{i,j}$  is zero (assume that never happens). Which of the following statements best captures the difference compared to standard attention with  $\mathbf{P} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}})$ ?

- ☐ The alternative mechanism has a different big- $\mathcal{O}$  computational complexity
- ☐ The alternative mechanism does not compute proper, i.e. based on a valid probability distribution, weighted averages of the value tokens
- ☒ The alternative mechanism has  $\mathbf{P}$  that is invariant to a rescaling  $\mathbf{Q} \mapsto \alpha \mathbf{Q}$  for constant  $\alpha \in \mathbb{R}_+$
- ☐ The alternative mechanism is non-linear unlike standard attention
- ☐ The alternative mechanism explicitly accounts for the order of the inputs and therefore shouldn't require positional embeddings unlike standard attention
- ☐ None of the other statements are correct

**Solution:** This attention variant is known as scale-invariant attention and has the property that  $\mathbf{P}$  is invariant to a positive rescaling of either  $\mathbf{Q}$  or  $\mathbf{K}$ .

## Matrix Factorization

**Question 10** Given matrix  $A \in \mathbb{R}^{d \times d}$  with eigenvectors  $(1, 2, 1)^\top$  and  $(1, 1, 0)^\top$ , both with eigenvalue 4, and  $\text{trace}(A) = 2$ . What is the determinant of  $A$ ?

- ☐  $\det(A) = -16$
- ☐  $\det(A) = 128$
- ☐  $\det(A) = 16$
- ☐ The determinant of a matrix cannot be determined, since the dimension of  $A$  is unknown.
- ☐  $\det(A) = -128$
- ☒  $\det(A) = -96$

**Solution:** Since the eigenvectors of  $A$  is in  $\mathbb{R}^3$ , the total number of eigenvalues is 3. Using  $\text{trace}(A) = \lambda_1 + \lambda_2 + \lambda_3$ , we can determine the eigenvalues are  $(4, 4, -6)$ . Thus,  $\det(A) = 4 \times 4 \times (-6) = -96$ .



## Bias-Variance Decomposition

**Question 11** Consider data consisting of input-output pairs  $(x, y)$  coming from an unknown distribution  $\mathcal{D}$ , where the input  $x \in \mathcal{X}$  follows an unknown distribution  $\mathcal{D}_x$  and the output  $y \in \mathbb{R}$  is generated as  $y = f(x) + \varepsilon g(x)$ . The functions  $f, g : \mathcal{X} \rightarrow \mathbb{R}$  are unknown, and the random error term  $\varepsilon \sim \mathcal{D}_\varepsilon$  is independent of the input and has mean 0 and variance  $\sigma^2$ . Given a training set  $S$  sampled from  $\mathcal{D}$ , let  $h_S : \mathbb{R}^d \rightarrow \mathbb{R}$  be the predictor learned by our ML algorithm. Let  $L(h_S) = \mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon} \left[ (f(x_0) + \varepsilon g(x_0) - h_S(x_0))^2 \right]$  be the error at some fixed point  $x_0$ . Which of the following answers gives the **correct** expression for the expected error  $\mathbb{E}_{S \sim \mathcal{D}}[L(h_S)] = \mathbb{E}_{S \sim \mathcal{D}, \varepsilon \sim \mathcal{D}_\varepsilon} \left[ (f(x_0) + \varepsilon g(x_0) - h_S(x_0))^2 \right]$ ?

- ☒  $\sigma^2 g(x_0)^2 + \mathbb{E}_{S \sim \mathcal{D}}[(f(x_0) - h_S(x_0))^2];$
- ☐  $\sigma^2 g(x_0)^2 + 2g(x_0)\sigma \mathbb{E}_{S \sim \mathcal{D}}[f(x_0) - h_S(x_0)] + \mathbb{E}_{S \sim \mathcal{D}}[(f(x_0) - h_S(x_0))^2];$
- ☐  $\sigma^2 + (f(x_0) - \mathbb{E}_{S \sim \mathcal{D}}[h_S(x_0)])^2 + \mathbb{E}_{S \sim \mathcal{D}}[(h_S(x_0) - \mathbb{E}_{S' \sim \mathcal{D}}[h_{S'}(x_0)])^2];$
- ☐ None of the above expressions equals  $\mathbb{E}_{S \sim \mathcal{D}}[L(h_S)]$ .

**Solution:** Using the linearity of expectation,

$$\begin{aligned} & \mathbb{E}_{S \sim \mathcal{D}, \varepsilon \sim \mathcal{D}_\varepsilon} [(f(x_0) + \varepsilon g(x_0) - h_S(x_0))^2] \\ &= \mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon} [\varepsilon^2 g(x_0)^2] + \mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon, S \sim \mathcal{D}} [2\varepsilon g(x_0)(f(x_0) - h_S(x_0))] + \mathbb{E}_{S \sim \mathcal{D}} [(f(x_0) - h_S(x_0))^2] \\ &= \sigma^2 g(x_0)^2 + \mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon, S \sim \mathcal{D}} [2\varepsilon g(x_0)(f(x_0) - h_S(x_0))] + \mathbb{E}_{S \sim \mathcal{D}} [(f(x_0) - h_S(x_0))^2], \end{aligned}$$

where we used that  $\mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon} [\varepsilon^2] = \sigma^2$ . Now, using the independence of  $\varepsilon$  and  $S$ , the middle term becomes

$$\mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon, S \sim \mathcal{D}} [2\varepsilon g(x_0)(f(x_0) - h_S(x_0))] = 2g(x_0) \mathbb{E}_{\varepsilon \sim \mathcal{D}_\varepsilon} [\varepsilon] \mathbb{E}_{S \sim \mathcal{D}} [f(x_0) - h_S(x_0)] = 0.$$

## Transformers

**Question 12** Considering a sequence of  $n$  tokens, the computational complexity of the masked attention mechanism in BERT language models is: (select the smallest correct complexity)

- ☐  $\mathcal{O}(n^3)$
- ☐  $\mathcal{O}(n \log n)$
- ☐  $\mathcal{O}(n)$
- ☐  $\mathcal{O}(n^{1/2})$
- ☒  $\mathcal{O}(n^2)$

**Solution:** Masked attention is quadratic in the sequence length  $n$  because it computes the attention between all pairs of tokens.



## Second part: true/false questions

For each question, mark the box (without erasing) TRUE if the statement is **always true** and the box FALSE if it is **not always true** (i.e., it is sometimes false).

**Question 13** (Linear regression) For linear regression with no regularization, **scaling the features** (e.g. as in data normalization) does not change the model's performance, assuming that we can efficiently compute the optimum model in both cases, i.e. numerical stability/efficiency of finding the optimum model is not a concern.

☒ TRUE ☐ FALSE

**Solution:** True. Linear regression models are scale invariant when there is no regularization. The model will learn the same relationship between inputs and outputs regardless of the scale of the data.

**Question 14** (Linear regression) For linear regression with a bias and no regularization, **centering the features** (as in data normalization) does not change the model's performance, assuming that we can efficiently compute the optimum model in both cases, i.e. numerical stability/efficiency of finding the optimum model is not a concern.

☒ TRUE ☐ FALSE

**Solution:** True. Linear regression models are scale invariant when there is no regularization. The model will learn the same relationship between inputs and outputs regardless of the scale of the data.

**Question 15** (Feature expansion) Unnecessary polynomial expansion of input features can lead to underfitting.

☐ TRUE ☒ FALSE

**Solution:** False. It can lead to overfitting.

**Question 16** (Ridge regression) Ridge regression can help mitigate the impact of ill-conditioned data matrices during training, but it does not make the solution sparse. It consists of minimizing the following loss function:  $\sum_{n=1}^N (y_n - \mathbf{x}_n^\top \mathbf{w})^2 + \lambda \|\mathbf{w}\|_2^2$

☒ TRUE ☐ FALSE

**Solution:** True. Lasso regularization makes the solution sparse.

**Question 17** (Gradient Descent) When training a Deep Neural Network, it is better to use classical gradient descent rather than stochastic gradient descent with mini-batches to optimize the parameters of the network.

☐ TRUE ☒ FALSE

**Solution:** False. Deep Learning requires large datasets, and going through all of the data to compute a full gradient is unnecessarily expensive, compared to many cheaper SGD steps. In addition, noisy updates for several reasons seem to perform better in the non-convex landscapes stemming from deep learning training.





**Question 18** (Optimization) The Mean Absolute Error (MAE) loss function is more robust to outliers in the training dataset than the Mean Squared Error (MSE) loss function.

☒ TRUE ☐ FALSE

**Solution:** True, because the MSE computes the squared error, so if the dataset contains outliers the error gets very large. The MAE, on the other hand, does not amplify the error contributions of outliers so much. This was shown in Exercise 2.

**Question 19** (Logistic regression) One step of SGD for Logistic Regression will result in a change to the parameters only if the current example is incorrectly classified.

☐ TRUE ☒ FALSE

**Solution:** False. Recall the gradient of a logistic regression model:  $\nabla L(\mathbf{w}) = (\sigma(\mathbf{x}^\top \mathbf{w}) - y) \mathbf{x}$ , we update model parameters as long as  $\sigma(\mathbf{x}^\top \mathbf{w})$  is not equal to  $y \in \{0, 1\}$ .

**Question 20** (Classification) There are many classification problems that are not linearly separable. Tricks like kernel methods or adding a regularizer allow us to still separate the classes with a linear classifier.

☐ TRUE ☒ FALSE

**Solution:** Slightly ambiguous question, not counted here. Generally False. Kernel method is indeed a valid trick, but not adding a regularizer. Regularizer is used to prevent overfitting, which is not the case here.

**Question 21** (Kernels) The kernel trick allows computing inner products in infinite-dimensional feature spaces for specific choices of the kernel function.

☒ TRUE ☐ FALSE

**Solution:** True. An example is the RBF kernel discussed in the lecture.

**Question 22** (Support vectors) Let  $\mathcal{D}$  be a  $d$ -dimensional linearly separable binary classification dataset. Recall that Hard-SVM finds the maximum-margin hyperplane that is equidistant to two hyperplanes that form the boundaries of the margin. Let  $S$  be the number of data points on these two hyperplanes obtained with the Hard-SVM classifier on  $\mathcal{D}$ . Then, it is always true that  $S \in [2, 2d]$ .

☐ TRUE ☒ FALSE

**Solution:** False. There could be arbitrarily large support vectors if they are collinear: consider a 2-dimensional dataset where there are two clusters on lines  $x = -1$  and  $x = 1$  with varying  $y$ .

**Question 23** (Duality) Let  $G(w, \alpha) : \mathcal{W} \times \mathcal{A} \rightarrow \mathbb{R}$  be linear in both  $\alpha$  and  $w$ , where  $\mathcal{A}$  and  $\mathcal{W}$  are convex and compact domains. Then, the following holds:

$$\max_{\alpha \in \mathcal{A}} \min_{w \in \mathcal{W}} G(w, \alpha) = \min_{w \in \mathcal{W}} \max_{\alpha \in \mathcal{A}} G(w, \alpha)$$

☒ TRUE ☐ FALSE

**Solution:** True.  $G$  is convex in  $w$  and concave in  $\alpha$ , so the saddle point is the same as the min-max.



**Question 24** (EM Algorithm) The Expectation-Maximization (EM) algorithm is guaranteed to converge to the global maximum likelihood solution for fitting GMMs.

☐ TRUE ☒ FALSE

**Solution:** False. The EM algorithm for GMMs may converge to a local maximum likelihood solution, not necessarily the global maximum likelihood solution.

**Question 25** (GMM) The probability of the latent assignment  $z_n$  of the data point  $x_n$  being the  $k$ -th component of a GMM with  $K$  components is given by:

$$P(z_n = k | X = x_n) = \pi_k \cdot \mathcal{N}(x_n | \mu_k, \Sigma_k)$$

☐ TRUE ☒ FALSE

**Solution:** False. The probability should be normalized by the sum of the probabilities of all components:

$$P(z_n = k | X = x_n) = \frac{\pi_k \cdot \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \cdot \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

**Question 26** (CNN receptive field) Average pooling with a  $2 \times 2$  window and stride 2 increases the size of the receptive field compared to the receptive field of the current layer.

☒ TRUE ☐ FALSE

**Solution:** True or False. In general, such a pooling operation increases the size of the receptive field by the same amount as a  $2 \times 2$  convolution. However there is a corner case which we didn't rule out in the question. Here is an example of a CNN, where we first do convolutions that reduce the big image  $H \times W$  to  $1 \times 1$ . This  $1 \times 1$  image will depend on all the pixels of the original image and will have the receptive field  $H \times W$ . Then we can apply transposed convolutions or pad to scale the image to  $2 \times 2$ . After that we can apply the average pooling from the question statement and get that the receptive field won't increase.

So there is a network, where we apply an average  $2 \times 2$  pooling and the receptive field doesn't increase, so I think the answer to the question should be False.

This question has therefore be removed from the exam this year.

**Question 27** (Transformer) The computational complexity of a forward pass through a standard, single-headed, encoder-only transformer of depth  $L$ , embedding dimension  $D$ , sequence length  $S$  and using a batch size of  $N$  is:  $\mathcal{O}(NLS D^2)$

☐ TRUE ☒ FALSE

**Solution:** False. The computational complexity is  $\mathcal{O}(NLS D^2 + NLS^2 D)$  where the second term comes from the attention mechanism.

**Question 28** (Adversarial Training) Adversarial training typically results in a decreased (standard) accuracy on a test set drawn from the same distribution as the training and validation sets.

☒ TRUE ☐ FALSE

**Solution:** True. Robustness often comes at the cost of standard accuracy as non-robust features can be useful and generalize to the test set.



**Question 29** (Text Representation Learning) FastText is a supervised learning algorithm to learn sentence representation. Given a sentence  $s_n = (w_1, w_2, \dots, w_m)$ , its bag-of-words representation  $\mathbf{x}_n \in R^{|V|}$  in the vocabulary  $V$ , and the classification label  $y_n \in \{\pm 1\}$ . The training objective is:

$$\min_{\mathbf{W}, \mathbf{Z}} \mathcal{L}(\mathbf{W}, \mathbf{Z}) = \sum_{s_n \text{ is a sentence}} f(y_n \mathbf{W} \mathbf{Z}^\top \mathbf{x}_n)$$

This training objective is convex w.r.t. the joint parameters  $W$  and  $Z$  if the function  $f$  is linear,

☐ TRUE ☒ FALSE

**Solution:** False. The objective is non-convex because the matrix factorization is non-convex w.r.t.  $W$  and  $Z$ , even if  $f$  is entry-wise identity or another simple linear function.

**Question 30** (Generalization) When the loss function  $\ell \in [a, b]$  is bounded, Hoeffding's Inequality allows us to bound the difference between the true error  $L_{\mathcal{D}}(f_{S_{\text{train}}})$  and the training error  $L_{S_{\text{train}}}(f_{S_{\text{train}}})$ , which leads to

$$\mathbb{P}_{S_{\text{train}}} \left[ |L_{\mathcal{D}}(f_{S_{\text{train}}}) - L_{S_{\text{train}}}(f_{S_{\text{train}}})| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{\text{train}}|}} \right] \leq \delta.$$

☐ TRUE ☒ FALSE

**Solution:** Hoeffding's Inequality allows us to bound the difference between the true error  $L_{\mathcal{D}}(f_{S_{\text{train}}})$  and the validation error  $L_{S_{\text{test}}}(f_{S_{\text{train}}})$ :

$$\mathbb{P}_{S_{\text{test}}} \left[ |L_{\mathcal{D}}(f_{S_{\text{train}}}) - L_{S_{\text{test}}}(f_{S_{\text{train}}})| \geq \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{\text{test}}|}} \right] \leq \delta.$$

We cannot use Hoeffding's Inequality to bound the difference between  $L_{\mathcal{D}}(f_{S_{\text{train}}})$  and  $L_{S_{\text{train}}}(f_{S_{\text{train}}})$  for two reasons. First, if  $(X_i, Y_i)_{i=1}^n$  are the random independent training samples, then  $L_{S_{\text{train}}}(f_{S_{\text{train}}}) = \sum_{i=1}^n \ell(f_{S_{\text{train}}}(X_i), Y_i)$ . Now, the random variables  $\ell(f_{S_{\text{train}}}(X_i), Y_i)$  are not longer independent since  $f_{S_{\text{train}}}$  depends on the whole training set. Second,  $L_{\mathcal{D}}(f_{S_{\text{train}}})$  is not the expectation of  $L_{S_{\text{train}}}(f_{S_{\text{train}}})$ . Hence, Hoeffding's Inequality is not applicable.

**Question 31** (Model Selection) Suppose we want to optimize the hyperparameter  $\lambda$  over the values  $\lambda_1, \dots, \lambda_K$  and suppose the loss function  $\ell \in [a, b]$  is bounded. Let  $\lambda_{\hat{k}}$  be the hyperparameter value leading to the predictor  $f_{\hat{k}}$  with the smallest training error and let  $\lambda_{k^*}$  be the hyperparameter value leading to the predictor  $f_{k^*}$  with the smallest true error  $L_{\mathcal{D}}(f_{k^*})$ . Then, using Hoeffding's Inequality, we get

$$\mathbb{P}_{S_{\text{train}}} \left[ L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^*}) + 2\sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2|S_{\text{train}}|}} \right] \leq \delta.$$

☐ TRUE ☒ FALSE

**Solution:** Hoeffding's Inequality allows us to bound the difference between the true errors  $L_{\mathcal{D}}(f_{\hat{k}})$  and  $L_{\mathcal{D}}(f_{k^*})$ , where  $f_{\hat{k}}$  is the predictor coming from the hyperparameter  $\lambda_{\hat{k}}$  with the smallest validation error  $L_{S_{\text{test}}}(f_{\hat{k}})$ . Then, we get that

$$\mathbb{P}_{S_{\text{test}}} \left[ L_{\mathcal{D}}(f_{\hat{k}}) \geq L_{\mathcal{D}}(f_{k^*}) + 2\sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2|S_{\text{test}}|}} \right] \leq \delta.$$



**Question 32**

(Self-supervised learning) When GPT models are trained on next token prediction with teacher-forcing, during training, each predicted token is added to the input sequence and fed back as input to the model in an autoregressive manner.

☐ TRUE      ☒ FALSE

**Solution:** In GPT models, predicting the next token in an autoregressive manner is used for inference only, while for training the model is fed with the entire sequence of original tokens and the masked (causal) attention prevents the model from cheating. This is also called teacher-forcing

DRAFT



## Third part, open questions

Answer in the space provided! Your answer must be justified with all steps. Leave the check-boxes empty, they are used for the grading.

### 1 Neural networks

**Question 33:** (3 points.) Let's denote the input to the network as  $\mathbf{x} \in \mathbb{R}^d$ , network parameters as  $\mathbf{W} \in \mathbb{R}^{m \times d}$  and  $\mathbf{v} \in \mathbb{R}^m$ , and let's define some additional variables  $\gamma, \beta \in \mathbb{R}^m$ . Consider a two-layer network  $f(\mathbf{x}) = \mathbf{v}^\top \phi(\mathbf{W}\mathbf{x})$  where  $\phi(z_i) = (z_i - \gamma_i)/\beta_i$  which can be seen as using batch normalization with fixed batch statistics  $\gamma_i$  and  $\beta_i$  instead of a standard activation function. Assume that we are using the squared loss  $\ell(\mathbf{x}, y) = (f(\mathbf{x}) - y)^2$ . Answer the following questions with a proper derivation or a counter-example:

- (a) Is the network  $f$  an affine (i.e., linear with a bias term) function of the input  $\mathbf{x}$ ?
- (b) Is the loss  $\ell$  convex with respect to parameter  $\mathbf{W}$ ?
- (c) Let's define  $\gamma$  as the vector whose every coordinate equals the average of the pre-activation values  $\frac{1}{m} \sum_i \mathbf{w}_i^\top \mathbf{x}$ . Does this change the answer to question (a)? Does this change the answer to question (b)?

☐ 0 ☐ 1 ☐ 2 ☒ 3

**Solution:**

- (a) (1 point) Yes, the network is affine in  $\mathbf{x}$ . Let the division sign denote the elementwise division between two vectors, then we have:  $f(\mathbf{x}) = \mathbf{v}^\top \phi(\mathbf{W}\mathbf{x}) = \mathbf{v}^\top (\mathbf{W}\mathbf{x} - \gamma)/\beta = \underbrace{(\mathbf{v}/\beta)^\top \mathbf{W}}_{\mathbf{a}^\top} \mathbf{x} - \underbrace{(\mathbf{v}/\beta)^\top \gamma}_b = \mathbf{a}^\top \mathbf{x} - b$ .
- (b) (1 point) Yes, the loss is convex with respect to  $\mathbf{W}$  since the objective with respect to  $\mathbf{W}$  is equivalent to linear regression:

$$\ell(\mathbf{x}, y) = (f(\mathbf{x}) - y)^2 = \left( \underbrace{(\mathbf{v}/\beta)^\top \mathbf{W}}_{\mathbf{v}'^\top} \mathbf{x} - \underbrace{b}_{b'} \right)^2 = \left( \underbrace{\text{vec}(\mathbf{v}' \mathbf{x}^\top)}_{\mathbf{x}'^\top} \underbrace{\text{vec}(\mathbf{W})}_{\mathbf{w}'} - b' \right)^2 = (\mathbf{x}'^\top \mathbf{w}' - b')^2$$

Alternatively, one can check if the Hessian of  $\ell$  is positive semi-definite.

- (c) (1 point) Both answers are unchanged since  $\gamma$  only contributes linearly to the bias term  $b$ . The function  $f$  remains linear in  $\mathbf{W}$  and thus the loss remains convex in  $\mathbf{W}$ .

**Question 34:** (2 points.) The GeLU activation function (popular for transformer architectures) is usually implemented as  $\phi(z) = z\sigma(cz)$  where  $z \in \mathbb{R}$ ,  $c \approx 1.702$ , and  $\sigma(z) = \frac{1}{1+e^{-z}}$ .

- (a) Write down the derivative  $\frac{d\phi(z)}{dz}$  and simplify the resulting expression.
- (b) What is the value of  $\frac{d\phi(z)}{dz}$  when  $z \rightarrow -\infty$  and  $z \rightarrow \infty$ ? Show intermediate calculations of the limits.
- (c) Discuss the relationship between the derivatives of GeLU and ReLU. For which inputs  $z$  they are similar and for which  $z$  they differ?

☐ 0 ☐ 1 ☒ 2

**Solution:**



(a) (1 point) Direct computation gives us:

$$\frac{d\phi(z)}{dz} = z \frac{d\sigma(cz)}{dz} + \sigma(cz) = z \frac{d(1 + e^{-cz})^{-1}}{dz} + \sigma(cz) \quad (1)$$

$$= -z \frac{e^{-cz} \cdot (-c)}{(1 + e^{-cz})^2} + \sigma(cz) = cz \frac{e^{-cz}}{(1 + e^{-cz})^2} + \sigma(cz) \quad (2)$$

$$= cz\sigma(cz)(1 - \sigma(cz)) + \sigma(cz) = \sigma(cz)(1 + cz(1 - \sigma(cz))) \quad (3)$$

(b) (0.5 points) We show the derivation step-by-step:

$$\lim_{z \rightarrow -\infty} \frac{d\phi(z)}{dz} = \lim_{z \rightarrow -\infty} \frac{cze^{-cz}}{(1 + e^{-cz})^2} = \lim_{z \rightarrow -\infty} \frac{cz}{(1 + e^{-cz})(1 + e^{cz})} \quad (4)$$

$$= \lim_{z \rightarrow -\infty} \frac{cz}{1 + e^{-cz}} = \lim_{z \rightarrow -\infty} \frac{c}{-ce^{-cz}} = \lim_{z \rightarrow -\infty} -e^{cz} = 0 \quad (5)$$

$$\lim_{z \rightarrow \infty} \frac{d\phi(z)}{dz} = \lim_{z \rightarrow \infty} \left[ \frac{cze^{-cz}}{(1 + e^{-cz})^2} + \sigma(cz) \right] = \lim_{z \rightarrow \infty} \frac{cz}{(1 + e^{-cz})(1 + e^{cz})} + \lim_{z \rightarrow \infty} \sigma(cz) \quad (6)$$

$$= \lim_{z \rightarrow \infty} \frac{cz}{1 + e^{cz}} + 1 = \lim_{z \rightarrow \infty} \frac{c}{ce^{cz}} + 1 = \lim_{z \rightarrow \infty} e^{-cz} + 1 = 1 \quad (7)$$

(c) (0.5 points) Both have the same asymptotic behavior when  $z \rightarrow -\infty$  and  $z \rightarrow \infty$  but a bit different behavior close to 0. GeLU can be seen as a differentiable approximation of ReLU but with a small negative part close to 0.

## 2 GMMs with weights

We consider a modification of the Gaussian Mixture Model seen in class where each data point  $\mathbf{x}_n \in \mathbb{R}^d$  is now weighted according to function  $w$  with weight  $w(\mathbf{x}_n) \geq 0$ . As in the lectures we use Expectation-Maximization and are interested in deriving the updates to optimize this new objective.

$$\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) := \sum_{n=1}^N w(\mathbf{x}_n) \log \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Where

$$\boldsymbol{\theta} = (\pi_1, \dots, \pi_K, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K),$$

$$\sum_{k=1}^K \pi_k = 1, \pi_k \geq 0, \text{ and}$$

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-d/2} \det(\boldsymbol{\Sigma})^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

We have derived the  $E$ -step for you, serving as a lower bound to  $\mathcal{L}(\boldsymbol{\theta})$ . Below, you will find the maximization problem of the  $M$ -step, which you must solve to obtain the new parameter updates.

$$\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^N w(\mathbf{x}_n) \sum_{k=1}^K q_{kn}^{(t)} [\log \pi_k + \log \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

Where  $q_{kn}^{(t)} = \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}{\sum_{k=1}^K \pi_k^{(t)} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})}$ . For simplicity you can assume that  $d = 1$  (in that case  $\boldsymbol{\Sigma} = \sigma^2$ ).

**Question 35:** (1 point.) Derive  $\boldsymbol{\mu}_k^{(t+1)}$ .

☐ 0 ☒ 1



**Solution:** This can be obtained in closed form by setting the derivative of the objective with respect to  $\mu_k$  to zero.

$$\mu_k^{(t+1)} = \frac{\sum_n w(\mathbf{x}_n) q_{kn}^{(t)} \mathbf{x}_n}{\sum_n w(\mathbf{x}_n) q_{kn}^{(t)}}$$

**Question 36:** (1 point.) Derive  $\Sigma_k^{(t+1)}$ .

☐ 0 ☒ 1

**Solution:** This can be obtained in closed form by setting the derivative of the objective with respect to  $\sigma^2$  to zero.

$$\left(\sigma^{(t+1)}\right)^2 = \frac{\sum_n w(\mathbf{x}_n) q_{kn}^{(t)} (\mathbf{x}_n - \mu_k^{(t+1)})^2}{\sum_n w(\mathbf{x}_n) q_{kn}^{(t)}}$$

**Question 37:** (1 point.) Derive  $\pi_k^{(t+1)}$ .

☐ 0 ☒ 1

**Solution:** Using the constraint that  $\sum_k \pi_k = 1$  we can use a Lagrange multiplier to obtain a closed form solution.

$$\pi_k^{(t+1)} = \frac{\sum_{n=1}^N w(\mathbf{x}_n) q_{kn}^{(t)}}{\sum_{n=1}^N w(\mathbf{x}_n)}$$

**Question 38:** (1 point.) If  $w$  was an integer function (i.e.  $w(\mathbf{x}_n) \in \mathbb{N}_{>0}$ ), can the weighted objective be solved using only a routine for solving the unweighted objective and a data preprocessing method? Justify your answer.

☐ 0 ☒ 1

**Solution:** Yes, by duplicating each data point  $\mathbf{x}_n$   $w(\mathbf{x}_n)$  times the unweighted objective and the unweighted algorithm updates become equal to the weighted objective and the weighted algorithm updates.



### 3 Diffusion Models

You are given the Markov chain associated with a diffusion model  $\{X_0, X_1, \dots, X_T\}$  with  $X_0 \sim p_{\text{data}}$  and  $p_{\text{data}} \in \mathcal{P}(\mathbb{R}^d)$  the distribution of the data. The forward transition, which is the conditional distribution such that  $X_{t+1} \sim p(\cdot|X_t)$ , is chosen to be Gaussian with density  $p(x_{t+1}|x_t) = \mathcal{N}(x_{t+1}; \alpha x_t, (1 - \alpha^2)\mathbb{I}_d)$  and  $\alpha > 0$ .

**Question 39:** (1 point.) Given this choice of the forward transition, what is the distribution of  $X_T$  when  $T \rightarrow \infty$ ? (answer only, no derivation is required)

☐ 0 ☒ 1

**Solution:**

$$X_T \sim \mathcal{N}(0, \mathbb{I}_d)$$

To create a generative model, we want to sample from the backward process using ancestral sampling and approximating the backward transition with a Taylor expansion such that  $p(x_t|x_{t+1}) \approx \mathcal{N}(x_t; (2 - \alpha)x_{t+1} + (1 - \alpha^2)\nabla \log p(x_{t+1}), (1 - \alpha^2)\mathbb{I}_d)$ .

**Question 40:** (1 point.) Explain in words why the approximate backward transition in the given form still cannot be computed exactly.

☐ 0 ☒ 1

**Solution:** The marginal  $p(x_t)$  at each step  $t$  is analytically intractable and therefore we do not have access to it.

We approximate the score  $\nabla \log p(x_t)$  using a Neural network  $s_\theta(x_t)$  with parameters  $\theta$  and solving the following regression problem:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{2} \mathbb{E}_{X_t} [\|\nabla \log p(X_t) - s_\theta(X_t)\|^2] \quad (\text{L})$$

**Question 41:** (3 points.) Show how the denoising score matching loss in the above equation (L) can be rewritten in a tractable form that only requires the conditional density  $p(X_t|X_0)$ :

☐ 0 ☐ 1 ☐ 2 ☒ 3

**Solution:** The solution is given by the following short derivation as seen in the lecture. Defining:

$$\mathcal{L}(\theta) := \frac{1}{2} \mathbb{E}_{X_t} [\|\nabla \log p(X_t) - s_\theta(X_t)\|^2]$$

we can rewrite it as:

$$\begin{aligned} \mathcal{L}(\theta) &= C_1 + \frac{1}{2} \mathbb{E}_{X_t} [\|s_\theta(X_t)\|^2] - \mathbb{E}_{X_t} [\nabla \log p(X_t)^\top s_\theta(X_t)] \\ &= C_1 + \frac{1}{2} \mathbb{E}_{X_t} [\|s_\theta(X_t)\|^2] - \int \nabla \log p(X_t)^\top s_\theta(X_t) p(X_t) dx_t \end{aligned}$$

Using that  $p(x_t) = \int p_0(x_0) p(x_t|x_0) dx_0$  we get:

$$\nabla \log p(X_t) = \mathbb{E}_{X_0|X_t} [\nabla \log p(X_t|X_0)].$$





Substituting in the DSM-loss:

$$\begin{aligned}
\mathcal{L}(\theta) &= C_1 + \frac{1}{2} \mathbb{E}_{X_t} [\|s_\theta(X_t)\|^2] - \mathbb{E}_{X_0} \mathbb{E}_{X_t|X_0} [\nabla \log p(X_t|X_0)^\top s_\theta(X_t)] \\
&= C_1 + \frac{1}{2} \mathbb{E}_{X_0} \mathbb{E}_{X_t|X_0} [\|s_\theta(X_t) - \nabla \log p(X_t|X_0)\|^2] \\
&\quad - \frac{1}{2} \underbrace{\mathbb{E}_{X_0} \mathbb{E}_{X_t|X_0} [\|\nabla \log p(X_t|X_0)\|^2]}_{\text{const.}} \\
&= C_2 + \frac{1}{2} \mathbb{E}_{X_0} \mathbb{E}_{X_t|X_0} [\|s_\theta(X_t) - \nabla \log p(X_t|X_0)\|^2]
\end{aligned}$$

Given our Gaussian choice for the forward transition, we know that the density at each time step  $t$  conditioned on the initial point is also Gaussian  $p(x_t|x_0) = \mathcal{N}(x_t; \alpha^t x_0, (1 - \alpha^{2t})\mathbb{I}_d)$  and therefore  $X_t = \alpha^t X_0 + \sqrt{1 - \alpha^{2t}} \xi_t$  with  $\xi_t \sim \mathcal{N}(0, \mathbb{I}_d)$ .

**Question 42:** (1 point.) Explain why the score matching loss can be interpreted as learning to predict the added noise from the noised data, remember that  $\nabla \log p(x_t|x_0) = -\frac{(x_t - \alpha^t x_0)}{(1 - \alpha^{2t})}$ .

☐ 0 ☒ 1

**Solution:** By simply substituting  $X_t$  we can see that  $\nabla \log p(X_t|X_0) = -\frac{\xi_t}{\sqrt{1 - \alpha^{2t}}}$ . If we now parameterize the score function as  $s_\theta(X_t) = -\frac{\hat{\xi}_\theta(X_t)}{\sqrt{1 - \alpha^{2t}}}$  then the problem is equivalent to:

$$\theta^* = \operatorname{argmin}_{\theta} \frac{1}{2} \mathbb{E}_{X_0} \mathbb{E}_{X_t|X_0} [\|\hat{\xi}_\theta(X_t) - \xi_t\|^2]$$