Midterm Exam Review

10-701: Introduction to Machine Learning

3/15/2024

1 Decision Trees

1. Suppose you have a training dataset consisting of 3 boolean features X_1, X_2, and X_3 where $X_i \in \{0, 1\}$. Furthermore, assume the label is defined as $Y = X_1 \vee X_2$ i.e., $Y = 1$ if $X_1 = 1$ or $X_2 = 1$ and $Y = 0$ otherwise. Suppose that your training dataset contains all of the 8 possible feature vectors:

X_1	X_2	X_3	Y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

   (a) What is the training error rate of a depth-0 decision tree on this dataset?

   $\frac{1}{4}$

   (b) If your splitting criterion is training error rate, which feature or features would you split on first? Briefly justify your answer.

   All features are equivalently good. No matter which feature you put at the root, the tree will still make 2 mistakes: 4 data points will go left and 4 data points will go right. Both sides will have more 1's than 0's, so the best tree will label both sides as positive (producing the same function as the depth-0 decision tree)

   (c) If your splitting criterion is information gain, which feature or features would you split on first? Briefly justify your answer.

   Either X_1 or X_2: X_3 has zero mutual information while both X_1 and X_2 have mutual information equal to $(-\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}) - \frac{1}{2} > 0$

2. You are given a dataset for binary classification with two features x_1 and x_2. x_1 can have two possible values 0,1 and x_2 can have three possible values 0,1,2. Figure 1 provides a depiction of this dataset.

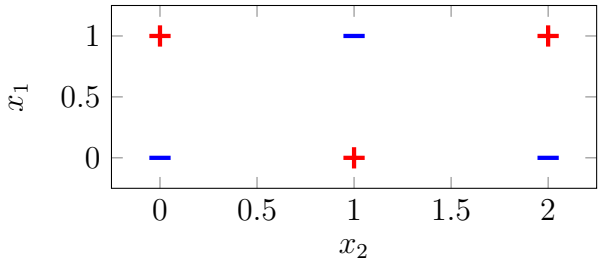


Figure 1: Binary classification dataset.

(a) What is the lowest possible training error rate any decision tree could achieve on this dataset? The tree need not be a binary tree but each node should split on at most one feature.

0

(b) Draw a decision tree that achieves this training error rate.

Many possible solutions exist but the most intuitive one first split on $x_1 < 0.5$ and then makes a ternary split on x_2 down either branch at 0.5 and 1.5.

3. We would like to learn a decision tree. We have n samples for training. You can assume that n is large and we are using continuous features. In the following questions, suppose we only use the first feature, x_1.

   (a) We would like to split according to x_1 at the root with 3 branches: data points are split at the root to three different subtrees by findings values a and b such that the three subtrees are $x_1 \leq a$, $a < x_1 < b$ and $x_1 \geq b$. What is the runtime for finding the values of x_1 that should be used by the root for such a split?

   - ◯ $O(\log n)$
   - ◯ $O(\sqrt{n})$
   - ◯ $O(n)$
   - ◯ $O(n^2)$
   - ◯ $O(n^3)$

   D. To split on continues values we need to find the best thresholds a and b. This requires us to order the values of x_1 and then test all possible splits and so the runtime is $O(n^2)$.

   (b) Following our first split, we would like to *split again* on x_1 in each of the three sub-trees resulting from the split in the previous question. Again, for each subtree we would split three ways as we did in the root. What is the total runtime for determining the optimal splits for ALL three subtrees?

   - ◯ $O(n)$
   - ◯ $O(n^2)$
   - ◯ $O(n^3)$
   - ◯ $O(n^4)$
   - ◯ $O(n^8)$

   B. Let the number of samples assigned to each of the three subtrees be n_1, n_2 and n_3. We know from question 1 that the total run time for each is $O(n_1^2)$, $O(n_2^2)$, $O(n_3^2)$. Since $n_1 + n_2 + n_3 = n$ one of these sets is $O(n)$ and so the total runtime is $O(n^2)$.

(c) For the same dataset we would like to learn the *optimal* tree that:

- Only uses x_1
- Has two levels, a root and a level below it where each splits to three branches (see the figure below).

What is the runtime for computing the *optimal tree*, in terms of minimizing the training error rate?

- ◯ $O(n)$
- ◯ $O(n^2)$
- ◯ $O(n^4)$
- ◯ $O(n^8)$
- ◯ $O(n^9)$

C. A naive solution would be to simply find the best 9 way split, which would entail a total run time is $O(n^8)$. However, we can formulate the problem as a dynamic program where we enumerate all $O(n^2)$ possible splits at the top level, then brute force over the $O(n^2)$ possible splits down each of the subtrees.

2 kNN

1. **True or False:** Regardless of the training dataset size, the true error rate of a 1-NN model will never exceed 2 $*$ the true error rate of the optimal classifier on a binary classification problem with stochastic labels.

   ◯ True

   ◯ False

   False, this condition only holds *with a high probability* as $N \to \text{inf}$

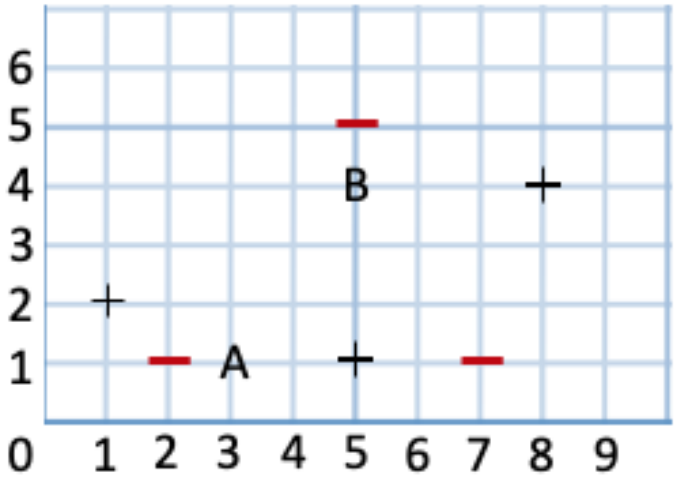
2. Suppose you are deciding between the following kNN models for a binary classification problem where $y \in \{-1, +1\}$ (for both models, assume we use the Euclidean distance):

   - Model 1: Probabilistic 3NN. This model predicts the label of a query data point with probabilities corresponding to the relative proportion of labels in the 3 nearest neighbors, e.g., if the 3 nearest neighbors have labels $\{-1, -1, +1\}$ then this model predicts -1 with probability $2/3$ and $+1$ with probability $1/3$.
   - Model 2: Distance weighted 2NN. This model predicts

   $$\hat{y} = \text{sign}\left(\sum_{(\vec{x}^{(i)}, y^{(i)}) \in N_2(\vec{x})} \frac{y^{(i)}}{\|\vec{x}^{(i)} - \vec{x}\|_2^2 + 1}\right)$$

   where $N_2(\vec{x})$ are the 2 nearest neighbors in the training data to the query data point $\vec{x}$ (the plus one in the denominator is to avoid division by 0). For the purposes of this model, let $\text{sign}(0) = +1$.

   You gather a training dataset of 6 points: 3 red $-$'s, which correspond to label $y = -1$, and 3 black $+$'s, which correspond to label $y = +1$. You also gather a validation dataset consisting of points A and B, both of which have label $+1$.



(a) What is the *expected* training error rate for Model 1? You may express your answer as a fraction if necessary.

Starting with the -1 at (5,5) and going clockwise, the expected pointwise training errors are

$$\frac{2}{3}+\frac{2}{3}+\frac{2}{3}+\frac{2}{3}+\frac{2}{3}+\frac{1}{3}=\frac{11}{3}$$

Thus, the expected training error rate is $\frac{11}{18}$.

(b) What is the training error rate for Model 2? You may express your answer as a fraction if necessary.

0; the trick here is to realize that this is effectively a 1NN model when applied to the training data as the nearest neighbor to a training data point will always be itself and thus, will always have a higher weight than the second nearest neighbor.

(c) What is the *expected* validation error rate for Model 1? You may express your answer as a fraction if necessary.

$\frac{1}{3}$; the expected validation error on both A and B is $\frac{1}{3}$ as the 3 nearest neighbors to A and B both have label sets $\{-1,+1,+1\}$.

(d) What is the validation error rate for Model 2? You may express your answer as a fraction if necessary.

1; again, because no two points are equidistant to either A or B, this model boils down to a 1NN.

3. For the points shown in Figure 2 suppose x_1, x_4, x_5, x_7, and x_9 have label $+1$, and the other points have label -1. For a 3NN classifier using the Euclidean distance, what is the LOOCV error?

The predictions are below. The error is 1

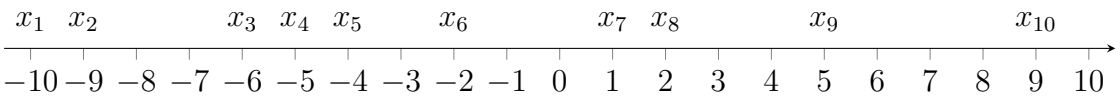


Figure 2

point	ground truth	LOOCV prediction
x_1	+1	-1
x_2	-1	+1
x_3	-1	+1
x_4	+1	-1
x_5	+1	-1
x_6	-1	+1
x_7	+1	-1
x_8	-1	+1
x_9	+1	-1
x_{10}	-1	+1

3 Linear Regression & Regularization

1. Using monthly stock averages, $x_1, x_2, \ldots x_n$, you run linear regression without regularization to estimate some future stock value y. However, your test error rate turns out to be very high. Your friend suggests that maybe the price *differences* between consecutive months may offer better features, so you add additional features $(x_1 - x_2), (x_2 - x_3), (x_3 - x_4), ..., (x_{n-1} - x_n)$. Do you expect an improvement? Briefly justify your answer.

No because the new features are just a linear combination of the original features. Thus, any regressor that could be learned using the new feature set would already have been considered by the original linear regression model.

2. Consider the following linear regression model:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$$

where we assume the residuals are normal and i.i.d.: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$. Recall from lecture that the MLE of $\boldsymbol{w}$ is $\hat{\boldsymbol{w}} = (X^T X)^{-1} X^T y$. Using the fact that $\hat{\boldsymbol{w}}$ is an unbiased estimator of $\boldsymbol{w}$, derive an expression for the variance of $var(\hat{\boldsymbol{w}})$:

By definition,

$$var(\hat{\boldsymbol{w}}) = \mathbb{E}[\hat{\boldsymbol{w}^2}] - (\mathbb{E}[\hat{\boldsymbol{w}}])^2$$

We can calculate $\mathbb{E}[\hat{\boldsymbol{w}}^2]$ as,

$$\begin{aligned}
\mathbb{E}[\hat{\boldsymbol{w}}^2] &= \mathbb{E}[((X^T X)^{-1} X^T y)^2] \\
&= \mathbb{E}[((X^T X)^{-1} X^T (X\boldsymbol{w} + \epsilon))^2] \\
&= \mathbb{E}[(\boldsymbol{w} + (X^T X)^{-1} X^T \epsilon)^2] \\
&= \boldsymbol{w}^2 + 2\mathbb{E}[(X^T X)^{-1}\epsilon] + (X^T X)^{-1}\mathbb{E}[\epsilon^2] \\
&= \boldsymbol{w}^2 + (X^T X)^{-1}\mathbb{E}[\epsilon^2]
\end{aligned}$$

Because $\hat{\boldsymbol{w}}$ is unbiased, $\mathbb{E}[\hat{\boldsymbol{w}}] = \boldsymbol{w}$. Thus, $var(\hat{\boldsymbol{w}}) = \boldsymbol{w}^2 + (X^T X)^{-1}\mathbb{E}[\epsilon^2] - \boldsymbol{w}^2 = (X^T X)^{-1}\sigma^2$.

4 MLE/MAP

1. **True or False**: Recall that the MLE of a parameter $\boldsymbol{\theta}$ given some dataset $\mathcal{D}$ is

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$$

while the MAP estimate of $\boldsymbol{\theta}$ is

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

For every dataset, there will always exist some prior distribution $p(\boldsymbol{\theta})$ for which the MAP estimate is equivalent to the MLE.

- ◯ True
- ◯ False

True; for bounded $\boldsymbol{\theta}$, the uniform distribution works, and for unbounded $\boldsymbol{\theta}$, any unimodal distribution with mode $\arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})$ works.

2. The probability density function of the Pareto distribution is

$$p(x|k,\alpha) = \begin{cases} \frac{\alpha k^\alpha}{x^{\alpha+1}} & x \in [k,\infty) \\ 0 & \text{otherwise} \end{cases}$$
$$k, \alpha \in (0,\infty)$$

Given n independent samples $x_1, x_2, ..., x_n$ drawn from a Pareto distribution, what is the MLE for the parameters k and α? **Hint: first determine the MLE of k, then use that result to in your expression for the MLE of α.**

$$\hat{k}_{MLE} = \_\_\_\_\_$$

$$\hat{\alpha}_{MLE} = \_\_\_\_\_$$

$$\hat{k}_{MLE} = \min_i x_i$$

$$\hat{\alpha}_{MLE} = \frac{n}{\sum_{i=1}^n \ln\left(\frac{x_i}{\hat{k}_{MLE}}\right)}$$

5 Naïve Bayes

1. Suppose you wish to extend the Naïve Bayes model to time series data. Formally, a training data point consists of a binary label Y and D sequentially ordered observations of some binary phenomenon where X_1 occurs before X_2, which occurs before X_3 and so on all the way down to the final observation X_D; each feature X_d is binary.

   You decide to modify the Naïve Bayes assumption such that now, a feature X_d is conditionally independent of all other features given the label Y *and* the previous feature X_{d-1}; the first feature X_1 is conditionally independent of all other features given just the label Y.

   (a) Write down the expression for the joint distribution $P(X, Y)$ under your new Naïve Bayes model. For full credit, your answer must use the modified Naïve Bayes assumption described above.

   $P(X,Y) = P(Y)P(X_1|Y)\prod_{d=2}^{D} P(X_d|X_{d-1}, Y)$

   (b) How many parameters would you need to learn in order to make predictions using your new Naïve Bayes model?

   1 for $P(Y)$, 2 for $P(X_1|Y)$ and 4 for each $P(X_d|X_{d-1}, Y)$ so $3 + 4(D-1)$ or $4D-1$.

2. Suppose we are training a Gaussian Naïve Bayes classifier to distinguish between students taking 10701 and 10601 based on 12 real-valued features. We train two models: M_1 using data from students across both courses in the F23 semester, and M_2 using values from the S23 semester. For F23, we had G_1 students in 10701, and U_1 students from 10601. For S23, we had G_2 and U_2 students respectively.

   Amazingly, M_1 and M_2 learned the same values for all the means and variances!

   Suppose a new student which we did not use in training shows up and we classify them using M_1 and M_2. M_1 predicts they are in 10701 whereas M_2 predicts they are in 10601. Given this, which of the following relationships must be true?

   - ◯ $G_1 > G_2$
   - ◯ $U_2 > U_1$
   - ◯ $G_1 + U_1 > G_2 + U_2$
   - ◯ $\frac{G_1}{G_2} > \frac{U_1}{U_2}$

   Answer: D. For this result to happen we need the prior for 10701 to be higher in M1 than it is in M2. This means that $\frac{G1}{G1+U1} > \frac{G2}{G2+U2}$. All values in this inequality are

positive so this is the same as:

$$\frac{G2+U2}{G2} > \frac{G1+U1}{G1} \implies \frac{U2}{G2}+1 > \frac{U1}{G1}+1 \implies \frac{G1}{G2} > \frac{U1}{U2}$$

6 Logistic Regression

1. Consider a dataset with only binary features, $\mathbf{x} \in \{0,1\}^d$, where feature x_1 is rare and only takes on value 1 in the training dataset if the corresponding the label is 1. If you fit a logistic regression model to this dataset using gradient descent, what value will the coefficient on this feature, $\hat{w}_1$, approach? Briefly justify your answer.

   If a binary feature fired for only label 1 in the training set then, by maximizing the conditional log likelihood, we will make the weight associated to that feature be infinite. This is because, when this feature is observed in the training set, we will want to predict predict 1 irrespective of everything else. This is an undesired behaviour from the point of view of generalization performance, as most likely we do not believe this rare feature to have that much information about class 1. Most likely, it is spurious co-occurrence. Controlling the norm of the weight vector will prevent these pathological cases.

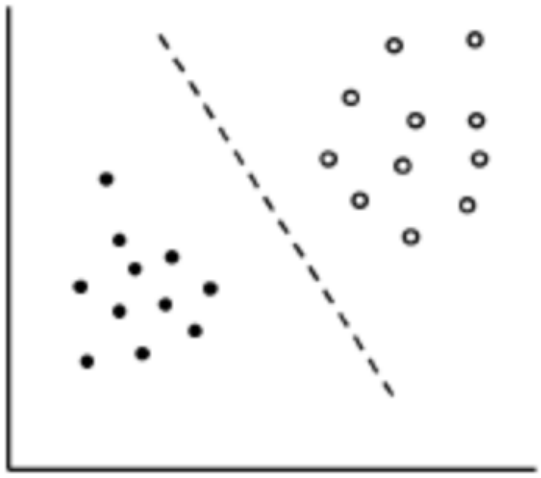
2. A generalization of logistic regression to a multiclass settings involves expressing the per-class probabilities $P(y = c|x)$ as the softmax function $\frac{\exp(w_c^T x)}{\sum_{d \in C} \exp(w_d^T x)}$, where c is some class from the set of all classes C and w_c is a class-specific weight vector.

   Consider a binary classification problem with labels encoded as $y \in \{0,1\}$. Rewrite the expression above for this situation to end up with expressions for $P(Y = 1|x)$ and $P(Y = 0|x)$ that we derived in class for binary logistic regression.

$P(y=1|x) = \frac{\exp(w_1^T x)}{\exp(w_0^T x)+\exp(w_1^T x)} = \frac{\exp((w_1-w_0)^T x)}{1+\exp((w_1-w_0)^T x)} = \frac{\exp(w^T x)}{1+\exp(w^T x)} = p$
Therefore, $1-p = \frac{1}{1+\exp(w^T x)}$

7 Regularization

1. The plot below shows data from two classes (filled and unfilled circles) separated by a linear decision boundary (dashed) defined by a two-dimensional weight vector and a bias term.



Suppose we apply L_2 regularization *to the bias term only* i.e., the two-dimensional weight vector remains the same for the regularized and unregularized settings. Which of the following best describes the new decision boundary that would result from this change. Assume that the amount of regularization is mild.

- ◯ farther from the origin; parallel to the previous line
- ◯ farther from the origin; not parallel to the previous line
- ◯ closer to the origin; parallel to the previous line
- ◯ closer to the origin; not parallel to the previous line

C

2. Complete the following paragraph about L_1 and L_2 regularization by circling the best of the provided options for each of blanks:

When applied to linear regression, L_1 regularization tends to learn denser / sparser weight vectors than L_2 regularization because the constraint surface is non-convex / square-shaped. The optimal weight vector for linear regression with L_1 regularization can / cannot be solved for in closed form.

When applied to linear regression, L_1 regularization tends to learn sparser weight vectors than L_2 regularization because the constraint surface is square-shaped. The optimal weight vector for linear regression with L_1 regularization cannot be solved for in closed form.

8 Neural Networks & Backpropagation

1. Your friend wants to build a fully-connected one-hidden-layer Neural Network. Her inputs have 9 features. She wants 7 hidden neurons in the first layer and 2 neurons in the output layer generated using a softmax function. Each layer is a linear layer fully-connected to the previous one and includes a bias term. Each scalar counts as one parameter.

   (a) What is the number of parameters she will need to create this neural network? Each scalar counts as one parameter.

   [ ]

   (9+1)x7 for the first layer is 70, (7+1)x2 for the output layer is 16, so the total is 86.

   (b) Your friend shares with you that the output she is trying to predict is binary. Using this information, describe an architecture that can learn the same decision boundary while reducing the number of parameters.

   She could make the output layer be only one node which contains the sigmoid function. Using a threshold of 0.5 will then recreate the predictions of the softmax.

2. Your friend trains a Neural Network on some training dataset and achieves an almost perfect training accuracy. However they perform very poorly when it comes to test time. In the boxes below, write either **{Increase, Decrease or N/A}** to indicate what they should do with their model to reduce the amount of overfitting (N/A here means changing the specified quantity won't effect overfitting). *You should assume the training and test data are representative of the true underlying distribution of the data.*

   [ ] the number of training data

   [ ] the number of test data

   [ ] the regularization constant

   [ ] the number of hidden layers

   [ ] the number of neurons in each hidden layer

[ ] the number of nodes in the output layer

Increase
N/A
Increase
Decrease
Decrease
N/A

3. In backpropagation, why is it necessary that we loop through the layers in reverse-order, moving from the output layer backwards towards the input layer?

The gradient of the loss with respect to the weights in some layer ℓ depends on the gradient with respect to all of the weights in downstream layers or layers closer to the output layer. Hence in order to compute these earlier gradients using the chain rule, backpropagation has to first compute the gradient with respect to weights in later layers.

4. When training a neural network, which of the following techniques is/are *primarily* used to address the nonconvexity of the loss function?

   - ☐ Momentum
   - ☐ Weight decay
   - ☐ Random restarts
   - ☐ Early termination
   - ☐ None of the above

A and C

9 CNNs & RNNs

1. What is the *primary* purpose of a convolutional layer in a CNN?

   ◯ To reduce the dimensions of the input image.

   ◯ To detect features such as edges and textures in the input image.

   ◯ To classify the input image into various categories.

   ◯ To flatten the input image across channels.

   B

2. Suppose you are building a CNN that takes input images with 3 channels, each of size 20×20 pixels. Your first convolutional layer takes the 3 input channels and produces 5 output channels; it uses 4×4 filters. It also uses a padding of 2 (along all sides of the image) and a stride of 2 (in both dimensions).

   (a) How many parameters are there in this convolutional layer, including the bias terms?

   The size of this convolutional layer can be represented as $5 \times 3 \times 4 \times 4$, giving 240 kernel parameters. We also have one bias term for each output channel for a total of 245 parameters.

   (b) What is the dimensionality of the output of this convolutional layer? Include the channel dimension in your answer and make sure you clearly indicate which dimension is the channel dimension.

   Channel $\times$ Height $\times$ Width $= 6 \times 11 \times 11$

3. Which of the following statements about RNNs is/are correct?

   - ☐ Training RNNs is difficult because of vanishing and/or exploding gradients.
   - ☐ Gradient clipping is an effective technique to address the vanishing gradient problem.
   - ☐ RNNs differ from feed-forward neural networks in that they have an additional weight matrix connecting hidden layers across time-steps.
   - ☐ RNNs can process sequences of arbitrary length, while feed-forward neural networks can not.
   - ☐ None of the above.

   A, C and D; gradient clipping is used to address exploding gradients

4. Briefly describe the bidirectional RNN architecture and describe why it is *not* appropriate for language modelling.

   Bi-directional RNNs have hidden-layer connections between both the previous and the next time-steps. This means that bi-directional RNNs are incapable of performing next token prediction, a crucial task in language modelling.

10 Attention & Transformers

1. For a fixed input-size and embedding dimension, which of the following statements is *not* true about multi-head attention relative to single-head attention?

   - ☐ Multi-head attention is more suitable for parallel computation than single-head attention.
   - ☐ Multi-head attention layers have more total parameters in their query, key and value matrices than single-head attention layers.
   - ☐ Multi-head attention is compatible with scaled dot-product attention while single-head attention is not.
   - ☐ Multi-head attention are able to capture more diverse relationships between tokens than single-head attention.
   - ☐ None of the above

   B and C

2. Suppose you have a transformer model that employs multi-head attention. The inputs to your model are sequences of T tokens and each token is represented by a d_M-dimensional embedding. Your model has H heads and for each head, the dimensionality of the key and query vectors is d_K and the dimensionality of the output vectors is d_V.

   (a) What are the dimensions of the key matrix *for one of the attention heads* i.e., K^h where $K^h = XW^h$?

   $T \times d_K$

   For a single attention head, the key matrix transforms the input embeddings into a new space defined by d_K. This transformation is applied separately for each token, resulting in a key matrix of dimension $T \times d_k$.

   (b) What is the dimensionality of the multi-headed attention output *before any sort of concatenation*?

   $T \times d_V \times H$.

   For each head, the attention mechanism outputs a matrix of size $T \times d_V$; given H heads, before concatenating the results, we have a $T \times d_V \times H$-dimensional tensor as the output.

3. Briefly explain why positional encodings are used in transformer models.

In a standard transformer model, because each token attends to every other token, the output is order invariant. This can lead to suboptimal behavior as the relative position of tokens in a sequence can have an impact on their meanings. Positional encodings are used to include information about where in the sequence a given token is to address this shortcoming.