WEEK 3 STUDY GUIDE

10-301/10-601 Introduction to Machine Learning (Summer 2025)
http://www.cs.cmu.edu/~hchai2/courses/10601
Released: Monday, June 2nd, 2025
Quiz Date: Friday, June 6th, 2025
TAs: Andy, Canary, Michael, Sadrishya, and Neural the Narwhal

Summary These questions are meant to prepare you for the upcoming quiz on Learning Theory, Deep Learning, Language Modeling, and Unsupervised Learning. You'll start with some examples of deriving complexity bounds and finding VC-dimensions of hypothesis sets. Then, you'll answer a few questions on convolutional and recurrent neural networks followed by attention and transformers. Finally, you'll work through some questions on k-means clustering and principal component analysis, two commonly-used unsupervised learning algorithms.

Note These questions are entirely optional; you do not need to submit your answers to these questions. However, at least 50% of the questions that will appear in your quiz will be *identical or nearly identical* to questions in this document. Thus, we recommend you to at least attempt every question. Furthermore, we *highly encourage* you to work in groups to solve these questions: because you are not being directly assessed on your solutions, feel free to share solutions and discuss ideas with your peers.

We encourage you to work on this study guide regularly throughout the week; in particular, this study guide is organized in sections where each section corresponds to a particular day's lectures. Here is our recommended schedule for working on this study guide:

1. Learning Theory - after Monday's (6/2) lectures
2. CNNs & RNNs - after Monday's (6/2) lectures
3. Transformers & Autodiff - after Tuesday's (6/3) lectures
4. k-means - after Wednesday's (6/4) lectures
5. Principal Component Analysis - after Thursday's (6/5) lectures

1 Learning Theory

1. (a) Infinite and agnostic

   (b) Infinite $|H|$ and agnostic case of PAC learning: $R \leq \hat{R} + \mathcal{O}\left(\sqrt{\frac{1}{N}\left[(16+1) + \ln\frac{1}{\delta}\right]}\right)$

   (c) D

2. (a) From the true definition of big-O, we can rewrite the following as for some $c \in \mathbb{R}$:

$$\begin{aligned} N &\leq c\left(\frac{1}{\epsilon^2}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right]\right) \\ \implies \epsilon^2 &\leq c\left(\frac{1}{N}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right]\right) \\ \implies \epsilon &\leq \sqrt{c}\left(\sqrt{\frac{1}{N}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right]}\right) \\ \implies \epsilon &= O\left(\sqrt{\frac{1}{N}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right]}\right) \end{aligned}$$

   Since $\sqrt{c}$ is also in $\mathbb{R}$.

   (b) If our sample complexity bound is true, we know that:

$$\begin{aligned} & P(|R(h) - \hat{R}(h)| \leq \epsilon) \geq 1 - \delta \\ \implies & P((R(h) - \hat{R}(h) \leq \epsilon) \wedge (R(h) - \hat{R}(h) \geq -\epsilon)) \geq 1 - \delta \\ \implies & P(R(h) - \hat{R}(h) \leq \epsilon) \geq 1 - \delta \quad \text{Since } P(A \cap B) \leq P(A) \text{ for any events } A, B \\ \implies & P\left(R(h) - \hat{R}(h) \leq O\left(\sqrt{\frac{1}{N}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right]}\right)\right) \geq 1 - \delta \\ \implies & P\left(R(h) \leq \hat{R}(h) + O\left(\sqrt{\frac{1}{N}\left[\text{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right]}\right)\right) \geq 1 - \delta \end{aligned}$$

   as desired.

3. Neural is right. For any placement $S = \{0,1\}^M$, any labeling can be generated using some function f, and therefore 2^M examples can be shattered.

4. (a) C. Consider 4 points along $\mathcal{X}$. For all possible labelings of these points, they are shatterable by H. This is not true for 5 points. Consider the following set of point-label pairs in $\mathcal{X}$: $S = \{(-2,+1), (-1,-1), (0,+1), (1,-1), (2,+1)\}$. It is impossible for any hypothesis in H to perfectly classify S; since every hypothesis in H classifies only the open sets (a,b), (c,d) as positive, any hypothesis would either incorrectly label at least one negative point as positive in order to correctly classify all the positive-class points (e.g. $a = -3, b = -1.5, c = -0.5, d = 2.5$) or incorrectly label at least positive point as negative in order to correctly classify all the negative-class points (e.g. $a = -3, b = -1.5, c = -0.5, d = 0.5$).

   (b) Multiple correct answers for this one, but all should have the form of alternating +1, -1, +1, -1, +1. Partial credit awarded if choice for part (a) is 5 or 6 AND labeling is not shatterable.

2 CNNs & RNNs

1. (a) E

   (b) A, B, D

   (c) 4*4 = 16

   (d) 10*10*16 = 1600 where 10 = ((22-4+2*0)/2)+1 because (image size – filter size + 2 * padding) / stride + 1

   (e) 19*19*16*3=17328

   (f) Because filters are used in feature detection, so sharing the filter allows us to look for the same features across the entire image.

2. (a) `r + i, c + j`

   (b)
   $$\begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix}$$

   (c) $\mathbf{A}^T$

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

4. 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 modeling.

3 Transformers & Autodiff

1. B, C

2. A. In a module-based automatic differentiation system, the `tape.push(self)` command is used to store the reference of the current module being executed during the forward pass.

3. False. Sequence matters in autodiff because we are reversing the sequence during backward

4. C. 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 $T \times d_k$, where T is the number of tokens and d_k is the dimensionality of the key vectors for that head. Option (C) is correct because it is the only one that correctly represents the dimensions of the key matrix without incorporating the number of heads or the original embedding size.

5. B. For each head, the attention mechanism outputs a matrix of size $T \times d_v$ (since each of the T tokens now has a new representation of size d_v from that head). When the outputs from all H heads are concatenated, the resulting matrix has a size of $T \times (H \times d_v)$, hence option (B) is correct. After concatenation, the output matrix often goes through a final linear transformation to reshape it back to $T \times d_{\text{model}}$, but this step is not included in the current question.

6. 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.

4 k-Means

1. (a) A, C

   (b) A

2. (a) C

   (b) D

   (c) 3: 1st, 3rd, and 5th data points

   (d) 2: 2nd, and 4th data points

3. (a) 0

   (b) Any $k \geq 100$

4. (a) k^N

   (b) $10^{1000} * 0.01 = 10^{998}$

5. (a) C. Furthest point initialization is sensitive to outliers because the outlier points will have significantly greater distances to previously chosen cluster centers than other data points (that are probably close to those chosen centers) and thus will be chosen as cluster centers in those iterations.

   (b) A. New centers are chosen iteratively with probability proportional to squared distance from existing centers, which makes k-means++ more likely to sample points from other Gaussian clusters, as they are further away.

5 Principal Component Analysis

1. A. We will have larger-valued features for the higher principal components because there is more spread along those vectors and all components are unit vectors.

2. C and then A (order matters!)

3. D, E. We want to find linearly independent, orthogonal vectors (so we have dot product 0 for any pair of distinct vectors). Orthogonality means that variations in our data are uniquely attributable to our principal components (and the components are uncorrelated), and we can compute our principal component scores independently for each component. Orthogonality alone does not imply that we maximize variance/minimize error, and F is nonsensical.

4. C. PCA preserves as much variance as possible in each of the constructed principal components. It's very possible that the two raw features we pick at random are not in the directions of greatest variation in the data. Then, these features do not preserve the maximum possible amount of variation in the data, which makes them less informative to the model once we perform dimensionality reduction.