WEEK 4 STUDY GUIDE

10-301/10-601 Introduction to Machine Learning (Summer 2025)
http://www.cs.cmu.edu/~hchai2/courses/10601
Released: Monday, June 9th, 2025
Quiz Date: Friday, June 13th, 2025
TAs: Andy, Canary, Michael, Sadrishya, and Neural the Narwhal

Summary These questions are meant to prepare you for the upcoming quiz on Pre-training & Fine-tuning, Reinforcement Learning, and Ensemble Methods. You'll start with a few questions about pre-training and fine-tuning depe learning models. Then, you'll step through a simple grid world and apply value iteration to derive an optimal policy. Next, you'll explore a larger version of the same grid world, which you'll solve using Q-learning. You'll perform some analysis to justify deep Q-learning in the Atari game-playing setting. Finally, you'll conclude with a few questions on random forests and AdaBoost, including a theoretical analysis of both methods.

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. Pre-training, Fine-tuning, & In-context Learning - after Monday's (6/9) lectures
2. Reinforcement Learning - after Tuesday's (6/10) lectures
3. Q-Learning - after Tuesday's (6/10) lectures
3. Deep Q-Learning - after Tuesday's (6/10) lectures
4. Random Forests - after Wednesday's (6/11) lectures
5. AdaBoost - after Wednesday's (6/11) lectures

1 Pre-training, Fine-tuning, & In-context Learning

1. D
2. In unsupervised pretraining, each layer's weights are first learned by treating that layer as the encoder of a 1-layer autoencoder. The weights are learned to minimize the reconstruction error between it's input and output.
3. D

2 Reinforcement Learning

1.

S_D 20	S_E 0	S_F
S_A 0	S_B 33.5	S_C 0

2.

S_D 20	S_E 0	S_F
S_A 25.92	S_B 33.5	S_C 0

3.

S_D 26.35	S_E 0	S_F
S_A 31.1355	S_B 36.2	S_C 0

4.

S_D `down`	S_E `terminal`	S_F
S_A `right`	S_B `up`	S_C `terminal`

3 Q-Learning

1. `left`, `right`
2. $0 + 0.9 * 0.7 = 0.63$
3. $0.3 + 0.1(0 + 0.9 * 0.7 - 0.3) = 0.333$
4. `up`
5. $25 + 0.9 * 0.8 = 25.72$
6. $0.8 + 0.1(25 + 0.9 * 0.8 - 0.8) = 3.292$
7. `up`
8. $-50 + 0.9 * 0.9 = -49.19$
9. $0.6 + 0.1(-50 + 0.9 * 0.9 - 0.6) = -4.379$

4 Deep Q-Learning

1. $2^{160 \times 192} = 2^{30720}$
2. 3×2^{30720}
3. No. The state space is infinite and uncountable.
4. Since the state space is finite (just really huge) we can systematically enumerate all possible states $\mathbf{s}_1, \mathbf{s}_2, \ldots \mathbf{s}_n$ where $n = 2^{30720}$. Then we can construct a feature vector:

$$\begin{bmatrix} \mathbf{1}(\mathbf{s} = \mathbf{s}_1) \\ \mathbf{1}(\mathbf{s} = \mathbf{s}_2) \\ \vdots \\ \mathbf{1}(\mathbf{s} = \mathbf{s}_n) \end{bmatrix}$$

   Clearly each state-action pair will have its own weight, encoding a table lookup. (This is infeasible since the vector will be too big).

5. No, highly correlated. Experience replay.

5 Random Forests

1. False, we want to reduce correlation between trees and combine their predictions to reduce variance. If the trees are too correlated, the predictions will be too similar, and the variance of the ensemble will be higher. If the trees are more uncorrelated, the variance of the ensemble will be lower.
2. B, D
3. A, B, C
4. (a) 0

   (b)

$$\begin{aligned} P(m(x,y) < 0) &= P(m(x,y) - s < -s) \\ &\leq P(m(x,y) - s < -s \vee m(x,y) - s > s) \\ &= P(|m(x,y) - s| \geq s) \\ &\leq \frac{\text{Var}(m(x,y))}{s^2} \end{aligned}$$

   (c) C. The expression becomes $\frac{\rho(1-s^2)}{s^2} = \frac{\rho}{s^2} - \rho$, which decreases with decreasing ρ and increasing s. Note that $0 < s \leq 1$, so the numerator is always non-negative.

6 AdaBoost

1. False. AdaBoost scales weights multiplicatively, so no weight will ever reach 0.
2. False. AdaBoost trains weak learners based on only the current weight distribution and updates weight distributions based on only the current weak learner's error, so they can continue to change even if perfect training accuracy is achieved.
3. (a) $\frac{1}{2} \ln \frac{1-0.5}{0.5} = 0$

   (b) None of the above. The weight updates are multiplications by 1 in both cases.

   (c) A, B. With $\alpha_{t'} = 0$, all points retain the same weights, so we deterministically learn the same weak learner with the same (weighted) error.
4. (a) $M_{t+1}(i) = -\alpha y^{(i)} H_t(x^{(i)}) = -\alpha \text{margin}_t(x^{(i)}, y^{(i)})$

   (b) D, lower signed

   (c) A, increase

   (d) A. AdaBoost will improve margins on points with the lowest signed margins. After training convergence, we are no longer moving points from negative to positive. However, for points that have positive margin, AdaBoost can increase the confidence on these predictions. Assuming that training data and test data come from the same distribution, test data points are likely similar to those in the training data. With a high variance ensemble with many low-confidence regions near training data points, we may misclassify test points that appear as small perturbations of training points. Improving the margin makes a more generalizable classifier and potentially simpler decision boundary. This can improve generalization error.