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. **Select one:** Neural the Narwhal wants to train a deep neural network to perform machine translation of narwhal *speech* to human *speech*; however, because the narwhal language is very rarely spoken, he only has a very small dataset for the task, so he decides to pre-train his neural network. Which of the following data sources would be the *best* data to pre-train his network with?

    ◯ A large corpus of images showing two narwhals speaking to each other

    ◯ A small corpus of images showing a narwhal speaking to a human

    ◯ A large corpus of text written in the narwhal language (assume the written language is identical to the spoken language)

    ◯ A large corpus of audio recordings containing two narwhals speaking to each other

2. **Short answer:** In 2-3 concise sentences, describe the relationship between unsupervised pretraining and the autoencoder architecture.

3. **Select one:** For a fixed large language model and natural language task, how do few-, one- and zero-shot accuracy tend to relate to one another and how is this affected by model size?

    ◯ Few-shot accuracy, one-shot accuracy and zero-shot accuracy are roughly equivalent and stay roughly constant across all model sizes.

    ◯ Few-shot accuracy, one-shot accuracy and zero-shot accuracy are roughly equivalent and tend to increase with increasing model size.

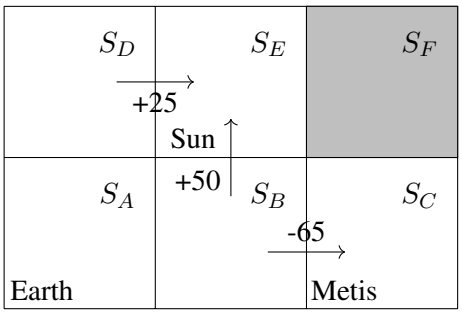
    ◯ Few-shot accuracy is greater than one-shot, which is greater than zero-shot and the differences are roughly constant across different sized models.

    ◯ Few-shot accuracy is greater than one-shot, which is greater than zero-shot and the differences tend to increase with increasing model size.

2 Reinforcement Learning

While attending an ML conference, you meet scientists at NASA who ask you to develop a reinforcement learning agent capable of carrying out a space-flight from Earth to the Sun.

You model this problem as a Markov decision process (MDP). The figure below depicts the state space.



Here are the details:

1. Each grid cell is a state $S_A, S_B, ..., S_F$ corresponding to a position in the solar system. The start state is S_A (Earth). The terminal states include both the S_E (Sun) and S_C (Metis).

2. The action space includes movement `up`/`down`/`left`/`right`. Transitions are **non-deterministic**. With probability 80% the agent transitions to the intended state. With probability 10% the agent slips left *of the intended direction*. With probability 10% the agent slips right *of the intended direction*. For example, if the agent is in state S_B and takes action `left`, it moves to state S_A with 80% probability, it moves to state S_B (left of the intended direction is off the board, so the agent remains where it was) with 10% probability, and it moves to state S_E (right of the intended direction) with 10% probability.

3. It is not possible to move to the blocked state (shaded grey) since it contains another planet. If the agent's action moves them off the board or to the blocked state, it remains in the same state.

4. Non-zero rewards are depicted with arrows. Flying into the Sun from below gives positive reward $R(S_B, \texttt{a}, S_E) = +50 \; \forall \texttt{a} \in \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}\}$, since it is more fuel-efficient than flying into the sun from the left (the agent can use the gravitational field of the planet in S_F and Metis). However, approaching the Sun from below has risks, as flying too close to Metis is inadvisable and gives negative reward $R(S_B, \texttt{a}, S_C) = -65 \; \forall \texttt{a} \in \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}\}$. Note that flying into the Sun from the left still achieves the goal and gives positive reward $R(S_D, \texttt{a}, S_E) = +25$ $\forall \texttt{a} \in \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}\}$. All other rewards are zero.

Below, let $V^*(s)$ denote the value function for state s using the optimal policy $\pi^*(s)$.

1. **Fill in the blank:** Report the value of each state (including terminal states) after a single round of **synchronous** value iteration in the table below. Initialize the value table $V^0(s) = 0, \forall s \in \{S_A \dots S_F\}$ and assume $\gamma = 0.9$. Visit each state in *reverse alphabetical order*. Ignore the blocked states. Round your *answers only* to the first decimal place. **Do not round intermediate values when calculating your answers.**

S_D	S_E	S_F
S_A	S_B	S_C

2. **Fill in the blank:** Starting over, report the value of each state for a single round of **asynchronous** value iteration in the table below. Initialize the value table $V(s) = 0, \forall s \in \{S_A \dots S_F\}$ and assume $\gamma = 0.9$. Visit each state in *reverse alphabetical order*. Ignore the blocked states. Round your *answers only* to the first decimal place. **Do not round any intermediate values, including state values, when calculating your answers.**

S_D	S_E	S_F
S_A	S_B	S_C

3. **Fill in the blank:** Below, we give you the value of each state one round before the convergence of **asynchronous** value iteration.[1]

   What is the final value of each state, $V^*(s)$? Be sure to use **asynchronous** value iteration, and visit each state in *reverse alphabetical order*. Ignore the blocked states. Round your *answers only* to the first decimal place. **Do not round any intermediate values, including state values, when calculating your answers.**

S_D 25	S_E 0	S_F
S_A 30	S_B 36	S_C 0

   Your solution:

S_D	S_E	S_F
S_A	S_B	S_C

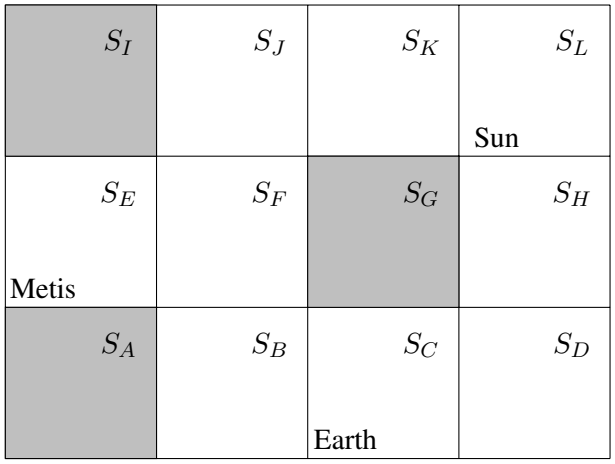
4. **Fill in the blank:** What is the policy, $\pi^*(s)$, that corresponds to $V^*(s)$? Write one of `up`, `down`, `left`, or `right` for each state. If multiple actions are acceptable, choose the one that comes alphabetically first. For terminal states, write `terminal`. Ignore the blocked states.

S_D	S_E	S_F
S_A	S_B	S_C

[1] This is actually one round before the *policy* converges, not *value* convergence. The values we provide are the values after the second iteration, rounded to the nearest whole number for ease of calculation.

3 Q-Learning

Let's consider an environment that is similar to the grid world we saw before, but has more states:



This time, however, suppose we **don't know** the reward function or the transition probability between states. Some rules for this setup are:

1. Each grid cell is a state $S_A, S_B, \ldots, S_L$ corresponding to a position in the solar system.
2. The action space of the agent is: $\{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}\}$.
3. If the agent hits the edge of the board, it remains in the same state. It is not possible to move into blocked states, which are shaded grey, since they contain other planets.
4. The start state is S_C (Earth). The terminal states include both the S_L (Sun) and S_E (asteroid Metis).
5. Use the discount factor $\gamma = 0.9$ and learning rate $\alpha = 0.1$.

We will go through three iterations of Q-learning in this section. Initialize $Q(s, a)$ as below:

$a \setminus s$	S_A	S_B	S_C	S_D	S_E	S_F	S_G	S_H	S_I	S_J	S_K	S_L
Up	0.4	0.1	0.1	0.7	0.0	0.9	0.7	0.8	0.0	0.1	0.8	0.8
Down	1.0	0.8	0.2	0.5	0.1	0.2	0.7	0.2	1.0	0.9	0.1	0.3
Left	0.9	0.4	0.3	0.4	0.9	0.6	0.5	0.1	0.2	0.3	0.9	0.1
Right	0.3	0.8	0.3	0.2	0.0	0.2	0.2	0.3	0.9	0.4	0.2	0.3

1. **Select all that apply:** If the agent were to start in state S_C and act greedily, which action or actions would it take?

   ☐ `up`

   ☐ `down`

   ☐ `left`

   ☐ `right`

2. **Numerical answer:** Beginning at state S_C, you take the action `right` and receive a reward of 0. You are now in state S_D. What is the new value for $Q(S_C, \texttt{right})$, assuming the update for deterministic transitions? If needed, round your answer to the fourth decimal place.

$Q(S_C, \texttt{right})$

3. **Numerical answer:** What is the new value for $Q(S_C, \texttt{right})$, using the temporal difference error update? If needed, round your answer to the fourth decimal place.

$Q(S_C, \texttt{right})$

4. **Select all that apply:** Continue to update your Q-function (as calculated by the temporal difference error update) from above. This time, though, assume your run has brought you to state S_H with no updates to the Q-function in the process. If the agent were to act greedily, what action or actions would it take at this time?

   ☐ `up`

   ☐ `down`

   ☐ `left`

   ☐ `right`

5. **Numerical answer:** Beginning at state S_H, you take the action `up`, receive a reward of +25, and the run terminates. What is the new value for $Q(S_H, \texttt{up})$, assuming the update for deterministic transitions? If needed, round your answer to the fourth decimal place.

$Q(S_H, \texttt{up})$

6. **Numerical answer:** What is the new value for $Q(S_H, \texttt{up})$, using the temporal difference error update? If needed, round your answer to the fourth decimal place.

$Q(S_H, \texttt{up})$

7. **Select all that apply:** Continue to update your Q-function (as calculated by the temporal difference error update) from above. You start from state S_C since the previous run terminated, but manage to make it to state S_F with no updates to the Q-function. If the agent were to act greedily, what action or actions would it take at this time?

   ☐ `up`

   ☐ `down`

   ☐ `left`

   ☐ `right`

8. **Numerical answer:** Beginning at state S_F, you take the action `left`, receive a reward of -50, and the run terminates. What is the new value for $Q(S_F, \texttt{left})$, assuming the update for deterministic transitions? If needed, round your answer to the fourth decimal place.

   $Q(S_F, \texttt{left})$

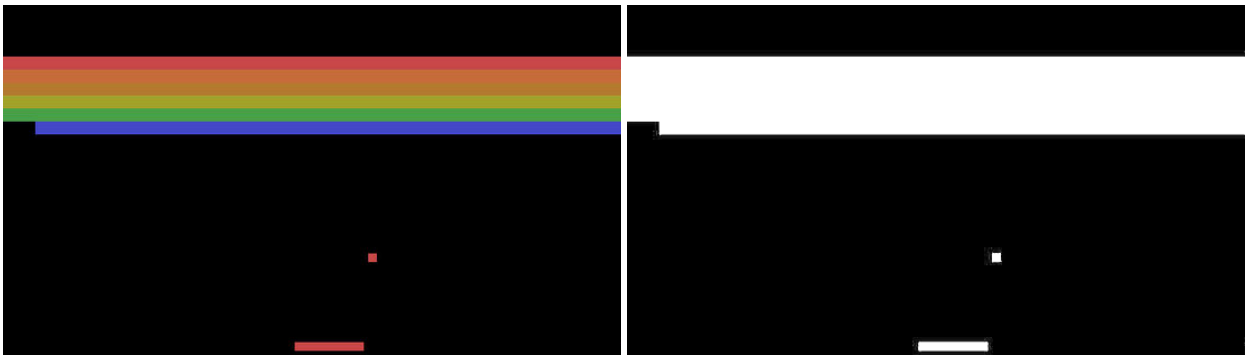
9. **Numerical answer:** What is the new value for $Q(S_F, \texttt{left})$, using the temporal difference error update? If needed, round your answer to the fourth decimal place.

   $Q(S_F, \texttt{left})$

4 Deep Q-Learning

In this question we will motivate learning a parametric form for solving Markov Decision Processes by looking at Breakout, a game on the Atari 2600. The Atari 2600 is a gaming system released in the 1980s, but nevertheless is a popular target for reinforcement learning papers and benchmarks. The Atari 2600 has a resolution of 160×192 pixels. In the case of Breakout, we try to move the paddle to hit the ball in order to break as many tiles above as possible. We have the following actions:

- Move the paddle left
- Move the paddle right
- Do nothing



(a) Atari Breakout (b) Black and white Breakout

Figure 1: Atari Breakout. 1a is what Breakout looks like. We have the paddle in the bottom of the screen aiming to hit the ball in order to break the tiles at the top of the screen. 1b is our transformation of Atari Breakout into black and white pixels for the purpose of some of the following problems.

1. **Numerical answer:** Suppose we are dealing with the black and white version of Breakout[2] as in Figure 1b. Furthermore, suppose we are representing the state of the game as just a vector of pixel values without considering if a certain pixel is always black or white. Since we are dealing with the black and white version of the game, these pixel values can either be 0 or 1.

   What is the size of the state space?

   Answer

2. **Numerical answer:** In the same setting as the previous part, suppose we wish to apply Q-learning to this problem. What is the size of the Q-value table we will need?

   Answer

[2]Play a "Google"-Doodle version here

3. **Short answer:** Now assume we are dealing with the colored version of Breakout as in Figure 1a. Now each pixel is a tuple of real valued numbers between 0 and 1. For example, black is represented as $(0, 0, 0)$ and white is $(1, 1, 1)$.

   Is it possible to represent all our Q-values with a table holding one value for every (state, action) pair?

   Answer

   Suppose rather than storing many separate Q-values for similar states, we want to share information between states. Instead of individual entries in a table, we can learn parameters $\mathbf{w}$ that parameterize some approximation $q(s, a; \mathbf{w})$ of the true Q-values.

   Let us define $q_\pi(s, a)$ as the true action value function of the current policy π. Assume $q_\pi(s, a)$ is given to us by some oracle. Also define $q(s, a; \mathbf{w})$ as the action value predicted by the function approximator parameterized by $\mathbf{w}$. Clearly we want to have $q(s, a; \mathbf{w})$ be close to $q_\pi(s, a)$ for all (s, a) pairs we see. This is just our standard regression setting. That is, our objective function is just the Mean Squared Error:

$$J(\mathbf{w}) = \frac{1}{2}\frac{1}{N}\sum_{s\in\mathcal{S}, a\in\mathcal{A}} \left(q_\pi(s, a) - q(s, a; \mathbf{w})\right)^2. \tag{1}$$

   Because we want to update for each example stochastically[3], we get the following update rule:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \left(q(s, a; \mathbf{w}) - q_\pi(s, a)\right) \nabla_{\mathbf{w}} q(s, a; \mathbf{w}). \tag{2}$$

   However, more often than not we will not have access to the oracle that gives us our target $q_\pi(s, a)$. So how do we get the target to regress $q(s, a; \mathbf{w})$ on? One way is to bootstrap an estimate of the action value under a greedy policy using the function approximator itself. That is to say

$$q_\pi(s, a) \approx r + \gamma \max_{a'} q(s', a'; \mathbf{w}) \tag{3}$$

   where r is the reward observed from taking action a at state s, γ is the discount factor and s' is the state resulting from taking action a at state s. This target is often called the Temporal Difference (TD) target, and gives rise to the following update for the parameters of our function approximator in lieu of a tabular update:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \underbrace{\left(q(s, a; \mathbf{w}) - \underbrace{\left(r + \gamma \max_{a'} q(s', a'; \mathbf{w})\right)}_{\text{TD Target}} \right)}_{\text{TD Error}} \nabla_{\mathbf{w}} q(s, a; \mathbf{w}). \tag{4}$$

[3] This is not really stochastic, you will be asked in a bit why.

4. **Short answer:** Consider the setting where we can represent our state by some vector $\mathbf{s}$, and for each action, we learn a linear approximation from states to Q-values. That is:

$$q(\mathbf{s}, a; \mathbf{w}) = \mathbf{w}_a^T \mathbf{s} \tag{5}$$

Again, assume we are in the black and white setting of Breakout as in Figure 1b. Show that tabular Q-learning is just a special case of Q-learning with a linear function approximator by describing a construction of $\mathbf{s}$. (**Hint**: Engineer features such that Eq. (5) encodes a table lookup)

Answer

5. **Short answer:** Stochastic Gradient Descent works because we can assume that the samples we receive are independent and identically distributed. Is that the case here? If not, why and what are some ways you think you could combat this issue?

Answer

5 Random Forests

1. **True or False:** In a random forest, it is generally better for the trees to be highly correlated, as this reduces variability.

   ◯ True

   ◯ False

2. **Select all that apply:** Which of the following is true about OOB error?

   □ OOB error is calculated on a held-out dataset separate from the dataset used to generate bootstrap samples

   □ OOB error is the aggregated value of the errors of subsets of the ensemble on samples those subsets were not trained on

   □ Cross-validation error is the same as OOB error

   □ OOB error is a valid method of estimating true error

   □ None of the above

3. **Select all that apply:** Which of the following are hyperparameters that can be tuned in a random forest?

   □ Number of trees trained

   □ Number of points used to train each decision tree

   □ Size of feature subsets used to train each decision tree

   □ Which features are used for splits in each decision tree

   □ None of the above

4. In this question, we will now derive an error bound for random forests in the case of **binary classification**. Given a random forest of B trees $\{h_i(x)\}_{i=1}^B$ and a sample (x, y) drawn from some data distribution $\mathcal{D}$, define the classification margin as:

$$m(x,y) = \frac{1}{B}\left(\sum_{i=1}^{B} \mathbb{I}[h_i(x) = y] - \sum_{i=1}^{B} \mathbb{I}[h_i(x) \neq y]\right)$$

In words, the margin $m(x, y)$ is the difference between the average vote for the correct label and the average vote for the incorrect label.

(a) **Fill in the blank:** For any example (x, y), the example is classified incorrectly if and only if $m(x, y) \leq$ ____. Assume majority vote ties are classified incorrectly.

Answer

(b) Observe that $P_{(x,y)\sim\mathcal{D}}\big(m(x, y) \leq c\big)$, where c is your answer to part (a), corresponds to the generalization error of the ensemble. Additionally, for an ensemble $\{h_i(x)\}_{i=1}^B$, define the strength of the ensemble as $s = \mathbb{E}_{(x,y)\sim\mathcal{D}}[m(x, y)]$.

Assume $s > 0$. Derive a bound for the generalization error in the form $P(m(x, y) < c) \leq d$, where d is an expression in terms of s and $\text{Var}(m(x, y))$. Show your work.

Hint: You should use Chebyshev's inequality, which states that for any random variable X with finite expectation and variance and any constant $a > 0$, we have

$$P(|X - \mathbb{E}[X]| \geq a) \leq \frac{\text{Var}(X)}{a^2}$$

Generalization error bound

(c) **Select one:** Through some additional manipulation, it is possible to show that $\text{Var}(m(x, y)) \leq \overline{\rho}(1 - s^2)$, where $\overline{\rho}$ is the mean correlation between trees in the ensemble. Substitute this into your bound from part (b). Which of the following describes how the error bound is affected by s and $\overline{\rho}$?

- ◯ The error bound gets smaller as $\overline{\rho}$ increases and s increases.
- ◯ The error bound gets smaller as $\overline{\rho}$ increases and s decreases.
- ◯ The error bound gets smaller as $\overline{\rho}$ decreases and s increases.
- ◯ The error bound gets smaller as $\overline{\rho}$ decreases and s decreases.

6 AdaBoost

1. **True or False:** Consider some training point $(x^{(i)}, y^{(i)})$ to the AdaBoost algorithm. If for all t, the weak learner h_t learned during training at time t correctly classifies $h_t(x^{(i)}) = y^{(i)}$, there will eventually be a finite time t such that the weight assigned to $x^{(i)}$ in the training distribution ω_t reaches exactly 0.

   ◯ True

   ◯ False

2. **True or False:** If the ensemble learned by AdaBoost reaches perfect training accuracy, all weak learners created in subsequent iterations will be identical (i.e., they will produce the same output on any input). Assume we are using deterministically selected weak learners.

   ◯ True

   ◯ False

3. Assume we use a deterministic training procedure for weak learners. Suppose for some iteration t' of AdaBoost we find that the weak classifier learned by the algorithm at time t' has error $\epsilon_{t'} = 0.5$ of the weak learner $h_{t'}$ on the training distribution weighted by $\omega_{t'}$.

   (a) **Numerical answer:** What weight $\alpha_{t'}$ will AdaBoost assign to the classifier $h_{t'}$ from above?

   $\alpha_{t'}$

   (b) **Select all that apply:** In which of the following cases will $\omega_{t'+1}^{(i)} > \omega_{t'}^{(i)}$ (in other words, in which of the following cases will the weight of training sample $(x^{(i)}, y^{(i)})$ *strictly* increase from time step t' to $t' + 1$)?

   □ $h_{t'}(x^{(i)}) = y^{(i)}$ ($h_{t'}$ classifies $x^{(i)}$ correctly)

   □ $h_{t'}(x^{(i)}) \neq y^{(i)}$ ($h_{t'}$ classifies $x^{(i)}$ incorrectly)

   □ None of the above.

   (c) **Select all that apply:** Which of the following are true about the next iteration of the AdaBoost algorithm?

   □ The errors $\epsilon_{t'+1}$ and $\epsilon_{t'}$ are equivalent

   □ The weak learners $h_{t'+1}$ and $h_{t'}$ will be equivalent (i.e., they will have the same output for every input)

   □ None of the above

4. In the following question, we will examine the generalization error of AdaBoost using a concept known as the *classification margin*.

   Throughout the question, use the following definitions:

   - T: The number of iterations used to train AdaBoost.
   - N: The number of training samples.
   - $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(N)}, y^{(N)})\}$: The training samples with binary labels ($y^{(i)} \in \{-1, +1\}$).
   - $\omega_t^{(i)}$: The weight assigned to training example i at time t. Note that $\sum_i \omega_t^{(i)} = 1$.
   - h_t: The weak learner constructed at time t (a function $X \to \{-1, +1\}$).
   - ϵ_t: The weighted (by ω_t) error of h_t.
   - $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$: The normalization factor for the distribution update at time t.
   - $\alpha_t = \frac{1}{2}\ln((1-\epsilon_t)/\epsilon_t)$: The weight assigned to the learner h_t in the composite hypothesis.
   - $H_t(x) = \left(\sum_{t'=1}^{t} \alpha_{t'} h_{t'}(x)\right) / \left(\sum_{t'=1}^{t} \alpha_{t'}\right)$: The majority vote of the weak learners, rescaled based on the total weights.
   - $g_t(x) = \text{sign}(H_t(x))$: The voting classifier decision function.

   For a binary classification task, assume that we use a probabilistic classifier that provides a probability distribution over the possible labels (i.e. $p(y|x)$ for $y \in \{+1, -1\}$). The classifier output is the label with highest probability. We define the *classification margin* for an input as the signed difference between the probability assigned to the correct label and the incorrect label $p_{correct} - p_{incorrect}$, which takes on values in the range $[-1, 1]$. Recall from recitation that $\text{margin}_t(x^{(i)}, y^{(i)}) = y^{(i)} H_t(x^{(i)})$.

   Note: Consistency points will not be awarded for this question.

   (a) **Math:** Recall the update AdaBoost performs on the distribution of weights:

   - $\omega_1^{(i)} = 1/N$
   - $\omega_{t+1}^{(i)} = \omega_t^{(i)} \dfrac{\exp(-y^{(i)}\alpha_t h_t(x^{(i)}))}{Z_t} = \dfrac{1}{N}\left(\prod_{t'=1}^{t} \dfrac{1}{Z_{t'}}\right) \exp(-\sum_{t'=1}^{t} y^{(i)}\alpha_{t'} h_{t'}(x^{(i)}))$

   We define $C_{t+1} = \dfrac{1}{N}\left(\prod_{t'=1}^{t} \dfrac{1}{Z_{t'}}\right)$ and $M_{t+1}(i) = -\sum_{t'=1}^{t} y^{(i)}\alpha_{t'} h_{t'}(x^{(i)})$. We then have

   $$\omega_{t+1}^{(i)} = C_{t+1} \exp(M_{t+1}(i))$$

   Let $\alpha = \sum_{t'=1}^{t} \alpha_{t'}$. Rewrite $M_{t+1}(i)$ in terms of $\text{margin}_t(x^{(i)}, y^{(i)})$ and α. (Hint: first rewrite $M_{t+1}(i)$ in terms of $y^{(i)}, \alpha, H_t, x^{(i)}$, then apply our given formula for the margin).

   Your Answer

(b) **Select one:** Note that C_{t+1}, α are treated as positive constants with respect to the input points. Using the classification margin and the above formulation of the weights assigned by AdaBoost, fill in the blanks to describe which points AdaBoost assigns high weight to at time t.

At time t, AdaBoost assigns higher weight to points $x^{(i)}$ with _____ _____ value of margin on the current ensemble classifier (i.e., $\text{margin}_t(x^{(i)}, y^{(i)})$).

- ◯ higher absolute
- ◯ higher signed
- ◯ lower absolute
- ◯ lower signed

(c) **Select one:** This weighting behavior causes the margins of the points you chose in part (b) to _____.

- ◯ increase
- ◯ decrease
- ◯ stay the same

(d) **Select all that apply:** How does this change in the margins explain the empirical result of test error continuing to decrease after training error has converged?

- □ AdaBoost can continue to increase the confidence of its predictions, particularly on lower confidence training examples, which makes it less likely at test time to misclassify points similar to those it has seen in the training set.
- □ Adaboost continues to increase confidence of its predictions on high confidence training examples only, leading to a highly accurate classifier which, at test time, will outperform the training error.
- □ Adaboost lowers the confidence in high confidence areas by continuing to train on low confidence training examples. This averages out the confidence between high and low confidence training examples, overall creating a more generalizable classifier.
- □ The test error does not continue to decrease after the training error has converged as the model stops updating once we reach convergence.
- □ None of the above.