CSCI 5525: Advanced Machine Learning (Fall 2023)

Homework 5

(Due Tue, Dec. 5, 11:59 PM CST)

1. **(20 points)** In this problem, we consider VC dimension.

   (a) **(5 points)** Let $\mathcal{H} = \{\mathbb{1}[x > a] \mid a \in \mathbb{R}\}$ denote the class of threshold functions where x is the feature and $\mathbb{1}[x > a] = 1$ if $x > a$ and 0 otherwise. What is the VC dimension of $\mathcal{H}$? (You must provide a proof.)

   (b) **(15 points)** Let the feature space be $\mathbb{R}^2$ and $\mathcal{H}$ be a class of binary classifiers h. Each classifier h has a triangular decision boundary where points inside the triangle are classified as positive labels and points outside are classified as negative labels as Figure 1 illustrates. What is the VC dimension of $\mathcal{H}$? (You must provide a proof.)

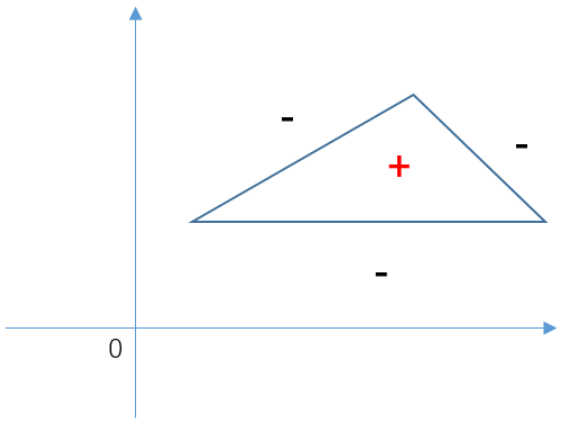

Figure 1: Binary classifier h.

2. **(20 points)** In this problem, we consider the bias-variance tradeoff. The expected test error using a machine learning algorithm, $\mathcal{A}$, and squared loss is:

$$E_{x,y,D}\left[(f_D(x) - y)^2\right]$$
$$= E_{x,D}\left[(f_D(x) - \bar{f}(x))^2\right] + 2\, E_{x,y,D}\left[\left(f_D(x) - \bar{f}(x)\right)\left(\bar{f}(x) - y\right)\right] + E_{x,y}\left[\left(\bar{f}(x) - y\right)^2\right]$$

   where D represents set of training points, (x, y) pairs are test points, and $\bar{f}$ is the expected classifier for algorithm $\mathcal{A}$.

   (a) **(10 points)** Prove $E_{x,y,D}\left[\left(f_D(x) - \bar{f}(x)\right)\left(\bar{f}(x) - y\right)\right] = 0$.

(b) **(10 points)** Proving (a) gives

$$E_{x,y,D}\left[(f_D(x)-y)^2\right] = \underbrace{E_{x,D}\left[(f_D(x)-\bar{f}(x))^2\right]}_{\text{Variance}} + E_{x,y}\left[\left(\bar{f}(x)-y\right)^2\right]$$

where the first term is variance. The second term can further be decomposed as follows:

$$\begin{aligned}&E_{x,y}\left[\left(\bar{f}(x)-y\right)^2\right]\\ &= \underbrace{E_{x,y}\left[(\bar{y}(x)-y)^2\right]}_{\text{Noise}} + \underbrace{E_x\left[\left(\bar{f}(x)-\bar{y}(x)\right)^2\right]}_{\text{Bias}^2} + 2\,E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)(\bar{y}(x)-y)\right].\end{aligned}$$

Prove $E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)(\bar{y}(x)-y)\right]=0$ to complete the bias-variance decomposition which gives

$$\underbrace{E_{x,y,D}\left[(f_D(x)-y)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{x,D}\left[\left(f_D(x)-\bar{f}(x)\right)^2\right]}_{\text{Variance}} + \underbrace{E_{x,y}\left[(\bar{y}(x)-y)^2\right]}_{\text{Noise}} + \underbrace{E_x\left[\left(\bar{f}(x)-\bar{y}(x)\right)^2\right]}_{\text{Bias}^2}.$$

3. **(30 points)** In this problem, we consider uniform convergence. We will prove a stronger generalization error bound for binary classification that uses more information about the distribution. Let us say that P is a distribution over (X,Y) pairs where $X\in\mathcal{X}$ and $Y\in\{+1,-1\}$. Let $\mathcal{H}$ be a finite hypothesis class which contains functions $h:\mathcal{X}\to\{+1,-1\}$ and let ℓ denote the zero-one loss $\ell(\hat{y},y)=\mathbb{1}[\hat{y}\neq y]$. Let $R(h)=E[\ell(h(X),Y)]$ denote the risk and let $h^*=\text{argmin}_{h\in\mathcal{H}}R(h)$. Given n samples, let $\hat{h}_n$ denote the empirical risk minimizer. Here, we want to prove a sample complexity bound of the form:

$$R(h^*)-R(\hat{h}_n)\le c_1\sqrt{\frac{R(h^*)\log(|\mathcal{H}|/\delta)}{n}}+c_2\frac{\log(|\mathcal{H}|/\delta)}{n} \tag{1}$$

for constants c_1,c_2. If $R(h^*)$ is small, this can be a much better bound than the usual excess risk bound. To prove the result, we will use Bernstein's inequality, which is a sharper concentration result.

Theorem 1 *(Bernstein's inequality). Let $X_1,...,X_n$ be i.i.d. real-valued random variables with mean zero, and such that $|X_i|\le M$ for all i. Then, for all $t>0$*

$$\mathbb{P}\left[\sum_{i=1}^n X_i\ge t\right]\le\exp\left(-\frac{t^2/2}{\sum_{i=1}^n E\left[X_i^2\right]+Mt/3}\right). \tag{2}$$

(a) **(15 points)** Using this inequality, show that with probability at least $1-\delta$

$$|\bar{X}|\le\sqrt{\frac{2E\left[X^2\right]\log(2/\delta)}{n}}+\frac{2M\log(2/\delta)}{3n} \tag{3}$$

where $\bar{X}=\frac{1}{n}\sum_{i=1}^n X_i$ and X_i's satisfy the conditions of Bernstein's inequality.

(b) **(15 points)** Then, use Eq. (3) and the union bound to show Eq. (1).

4. **(30 points)** In this problem, we consider multi-armed bandits. We will compare 3 multi-armed bandit algorithms (ϵ-greedy, UCB, and Thompson Sampling) and a random agent applied to simulated environments. We will experiment with 2 environments one with Gaussian rewards and one with Bernoulli rewards.

   Write Python code to implement each of the following bandit algorithms ϵ-greedy, UCB, and Thompson Sampling in classes `MyEpsilonGreedy`, `MyUCB`, `MyThompsonSampling`. Apply each algorithm, and also a random agent (one which chooses arms uniformly at random), to the simulated environment defined in class `MyEnvironment.py`. For each algorithm and environment, run 10 repetitions each for $T = 500$ rounds and compute the cumulative regret for each repetition in a list of length T. (Remember, cumulative regret is the sum of instantaneous regrets where instantaneous regret in round t is defined as $\mu_{i^*} - \mu_{i_t}$ where $i^* = \text{argmax}_{i \in [n]} \mu_i$ is the optimal arm, i_t is the arm pulled in round t, and μ_i is the mean reward for arm i.) Plot the average cumulative regret and standard deviation (as error bars) across the repetitions. You should get a single plot with 4 curves (one for each bandit algorithm and the random agent) where the x-axis is the number of rounds T and the y-axis is the average cumulative regret. (Make sure to include a legend to denote which curves were generated from which algorithm.) What can you conclude from the plot and why? Do your results confirm the theoretical regret bounds discussed in class?

   In addition to `MyEpsilonGreedy.py`, `MyUCB.py`, and `MyThompsonSampling.py` (details below), add your code to `hw5_q4.py` and use it to test.

   The bandit algorithms must be defined in classes and have the following functions:

```
class MyEpsilonGreedy:
   def __init__(self, num_arms, epsilon):
       ...
   def pull_arm(self):
       ...
   def update_model(self, reward):
       ...


class MyUCB:
   def __init__(self, num_arms):
       ...
   def pull_arm(self):
       ...
   def update_model(self, reward):
       ...


class MyThompsonSampling:
   def __init__(self, num_arms):
       ...
   def pull_arm(self):
       ...
   def update_model(self, reward):
       ...
```

For each class, `num_arms` is the number of arms as an integer, `reward` is a real-valued number generated from the environment, and the function `pull_arm()` must return an integer denoting the arm to pull. For `MyEpsilonGreedy`, the parameter `epsilon` is a value in $[0, 1]$. Please include the two plots (one for each of the Gaussian and Bernoulli environments) in your PDF report.

Instructions

You must complete this homework assignment individually. You may discuss the homework at a high-level with other students but make sure to include the names of the students in your README file. You may not use any AI tools (like GPT-3, ChatGPT, etc.) to complete the homework. Code can only be written in Python 3.6+; no other programming languages will be accepted. One should be able to execute all programs from the Python command prompt or terminal. Make sure to include a requirements.txt, yaml, or other files necessary to set up your environment. Please specify instructions on how to run your program in the README file.

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures, if any, should be included in the PDF report.

In your code, you can only use machine learning libraries such as those available from scikit-learn as specified in the problem description. You may use libraries for basic matrix computations and plotting such as numpy, pandas, and matplotlib. Put comments in your code so that one can follow the key parts and steps in your code.

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**
  1. `[YOUR_NAME]_hw5_solution.pdf`: A document which contains solutions to all problems.
  2. `hw5_q4.py`, `MyEpsilonGreedy.py`, `MyUCB.py`, and `MyThompsonSampling.py`: Code for Problem 4.
  3. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, any assumptions you are making, and any other necessary details.
  4. Any other files, except the data, which are necessary for your code.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems at a high level only. Each person must write up the final solutions individually. You need to list in the README.txt which problems were a collaborative effort and with whom. Please refer to the syllabus for more details. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online,
- Look up things/post on sites like Quora, StackExchange, etc.