Homework 4

CSCI 5525: Machine Learning

Due on Nov 21st 11:00am (before the class)

Please type in your info:

- **Name:Anubhav Panda**, **Student ID:5509727**, **Email:panda047@umn.edu**
- **Collaborators, and on which problems**

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to fill in above to specify which problems were a collaborative effort and with whom. (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.

Submission. Submit a PDF using this LaTeX template for written assignment part and submit .py files for all programming part. You should upload all the files on Canvas.

Written Assignment

Instruction. For each problem, you are required to write down a full mathematical proof to establish the claim.

Problem 1. Boosting .

(**16 points**) Boosting explored from a game-theoretic perspective.

Some Background: A two-player zero-sum game is specified by matrix M and the two players are denoted the row player and the column player. The game is played by the row player choosing a row i and the column player choosing a column j, and the *loss* for the row player is $M(i, j)$ which is also the *reward* for the column player. Instead of choosing individual rows/columns, we will allow both players to choose distributions over rows/columns, so if row player choose P and column player choose Q, the loss/reward is

$$\sum_i \sum_j P(i)M(i,j)Q(j) \triangleq M(P,Q)$$

If we are acting as row player, we would like to choose a distribution P that achieves low loss, no matter what column player does. In other words, we would like to choose P that minimizes $\max_Q M(P,Q)$. On the other hand, if we were column player, we would like to choose Q that maximizes $\min_P M(P,Q)$. Von Neumann's celebrated minimax theorem states that in fact both these values are same, or in some sense it does not matter which player goes first in the game:

$$\max_Q \min_P M(P,Q) = \min_P \max_Q M(P,Q) \triangleq V$$

V is referred to as the value of the game. In this problem, we will use boosting to compute the optimal strategy in a particular game.

Let $\mathcal{H}$ be finite, and fix a target concept $c : \mathcal{X} \to \{+1, -1\}$ (not necessarily in $\mathcal{H}$) and sample $S = \{X_i, c(X_i)\}_{i=1}^n$ of size n. (The concept c is just a formulation on how the labels are generated.)

We will form a matrix $M \in \{0,1\}^{n \times |\mathcal{H}|}$ where $M(i,h) = \mathbb{1}\{h(X_i) = c(X_i)\}$. Here, the row player specifies distributions over samples, just as in boosting, and the column player chooses distributions over hypotheses.

a) (**8 points**) Assume that the empirical γ-weak learning assumption holds so that for every distribution P over examples, there exists a hypothesis $h \in \mathcal{H}$ such that $\mathbb{P}_{x\sim P}[h(x) \neq c(x)] \leq 1/2 - \gamma$. What does this mean about the value of the game?
solution
As Given

$$\max_Q \min_P M(P,Q) = \min_P \max_Q M(P,Q) \triangleq V$$

And

$$\sum_i \sum_j P(i)M(i,j)Q(j) \triangleq M(P,Q)$$

AND $M(i,h) = \mathbb{1}\{h(X_i) = c(X_i)\}$ So that it can be combined as

$$\max_Q \min_P \sum_i \sum_j P(i)M(i,j)Q(j) \triangleq M(P,Q)$$

=

$$\max_Q \min_P \sum_i \sum_j P(i)\mathbb{1}\{h(X_i) = c(X_i)\}Q(j) = V$$

From the above equation we get
$\mathbb{P}_{x\sim P}[h(x) = c(x)] = V$

And As given in the question $\mathbb{P}_{x\sim P}[h(x) \neq c(x)] \leq 1/2 - \gamma$
$=1 - \mathbb{P}_{x\sim P}[h(x) = c(x)] \leq 1/2 - \gamma$
$=1 - V \leq 1/2 - \gamma$
$=V \geq 1/2 + \gamma$
So the value of the game will be always greater than or equal to $1/2 + \gamma$

b) (**8 points**) Let Q^* be distribution achieving value of this game, i.e. $\min_P M(P,Q^*) = V$. Since Q^* is distribution over hypotheses, what can we say about empirical error of Q^*?
Solution :
The empirical error can be defined As $\sum_i \sum_j P(i)\mathbb{1}\{h(X_i) \neq c(X_i)\}Q^*$
and we know V=$\sum_i \sum_j P(i)\mathbb{1}\{h(X_i) = c(X_i)\}Q^*$ and from the above bit

$V \geq 1/2 + \gamma$
$\sum_i \sum_j P(i)\mathbb{1}\{h(X_i) = c(X_i)\} \geq 1/2 + \gamma$
$=1 - \sum_i \sum_j P(i)\mathbb{1}\{h(X_i) \neq c(X_i)\} \geq 1/2 + \gamma$
$=1 - empiricalerror(Q^*) \geq 1/2 + \gamma$
$=1/2 - \gamma \geq empiricalerror(Q^*)$

c) (**Hurray!!! Extra credits: 10 points**) Boosting can be viewed as an iterative algorithm to compute Q^* using a weak learner. At every round, we choose P_t, a distribution over samples, and then compute

$$Q_t = \max_Q M(P_t, Q)$$

which is actually a single hypothesis h_t due to linearity. Then we update P_t to be

$$P_{t+1}(x) = \frac{P_t(x)}{Z_t} \times (\exp(-\eta \mathbb{1}\{h_t(x) = c(x)\}))$$

After T rounds, we output $\bar{Q} = \frac{1}{T}\sum_{t=1}^{T} Q_t$ which is a distribution over hypothesis and the final predictor is $H(x) = \text{sign}(\bar{Q}(x))$.

Prove that once $T = \Omega(\log(n)/\gamma^2)$ rounds and for appropriate choice of η, this variant of boosting guarantees (**Hint:** you may find it helpful to look at the Taylor expansion of $\exp(-x)$.)

$$\forall x \in X, \frac{1}{T}\sum_{i=1}^{T} M(x, h_t) > 1/2,$$

which implies that $H(x)$ has zero training error.
solution
measure of distance (or "divergence") between two probability distributions P and ΠP over 1, ... , m called relative entropy, also known as Kullback-Leibler(KL) divergence: it is assumed that $\hat{M}(i,h) = \mathbb{1}\{h(X_i)! = c(X_i)\}$ RE(PII$\acute{P}$)=$\sum_i^m P(i) \log \frac{P(i)}{\acute{P}(i)}$
For any matrix M with m rows and entries in [0, 1], and for any sequence of mixed strategies Q1, ... , QT played by the environment, the sequence of mixed strategies P1, ... , PT produced by algorithm MW(multiplicative weights) with parameter η satisfies $\sum_{t=1}^{T} \hat{M}(P_t, Q_t) \leq min_p a_\eta \sum_{t=1}^{T} \hat{M}(P, Q_t)] + c_\eta RE(P \amalg P_1)$
where $a_\eta = \frac{\eta}{1-\exp(-\eta)}$
and $c_\eta = \frac{1}{1-\exp(-\eta)}$ Under the conditions of above theorem η set to $\log(1 + \sqrt{\frac{2\log m}{T}})$
the average per-trial loss suffered by the learner is
$\frac{1}{T}\sum_{t=1}^{T} \hat{M}(P_t, Q_t) \leq min_p \frac{1}{T}\sum_{t=1}^{T} \hat{M}(P, Q_t)] + \Delta T$Equation 1
Where $\Delta T = \sqrt{\frac{2\log m}{t}} + \frac{\log m}{t}$ Equation 2
If MW is used with P_1 set to the uniform distribution, then its total loss is bounded by
$\frac{1}{T}\sum_{t=1}^{T} \hat{M}(P_t, Q_t) \leq min_p \frac{1}{T}\sum_{t=1}^{T} \hat{M}(P, Q_t)] + C_\eta \log m$
for our case: and it is assumed that m=n As we have seen above
$\mathbb{P}_{x\sim P}[h(x) = c(x)] > 1/2 + \gamma$ and from equation 1
$1/2 + \gamma \leq \frac{1}{T}\sum_{t=1}^{T} M(P_t, Q_t) \leq min_{p\epsilon P} \frac{1}{T}\sum_{t=1}^{T} M(P, Q_t) + \Delta T$
sofor all p we can say $\frac{1}{T}\sum_{t=1}^{T} M(P, Q_t) \geq 1/2 + \gamma - \Delta T > 1/2$ For the above equation to hold we need to prove $\gamma - \Delta T > 0$
$=\gamma > \Delta T$ from equation 2
$=\gamma > \sqrt{\frac{2\log m}{t}} + \frac{\log m}{t}$

$=\gamma > \sqrt{\frac{2\log m}{t}}$
And from the Above equation we can say that $T = \Omega(\log(n)/\gamma^2)$ for zero training error (proved)

d) (**Let's have more!!! Extra credits: 5 points**) Informally, what does this mean about $\bar{Q}$, what is it converging to as $T \to \infty$?
solution As we have seen above the training Error goes down to zero as T increases And from the bit c we can say that $\frac{1}{T}\sum_{t=1}^{T} M(P_t, Q_t) \geq 1/2 + \gamma$ and
$a_\eta min_{p \epsilon P} \frac{1}{T}\sum_{t=1}^{T} M(P, Q_t) + \frac{c_\eta log(m)}{T} \geq \frac{1}{T}\sum_{t=1}^{T} M(P_t, Q_t)$
where $\eta = 2a$
from the above mentioned equations we can say that
$a_\eta min_{p \epsilon P} \frac{1}{T}\sum_{t=1}^{T} M(P, Q_t) \geq \frac{1}{a_\eta}[1/2 + \gamma - \frac{c_\eta log(m)}{T}]$
$a_\eta min_{p \epsilon P} \frac{1}{T}\sum_{t=1}^{T} M(P, Q_t) \geq 1/2 + \gamma - (1/2 + \gamma)(1 - \frac{1-\exp -2\alpha}{2\alpha}) - \frac{log(m)}{2\alpha T}$
after applying taylor Approximation
$a_\eta min_{p \epsilon P} \frac{1}{T}\sum_{t=1}^{T} M(P, Q_t) \geq 1/2 + \gamma - (1/2 + \gamma)\alpha - \frac{log(m)}{2\alpha T}$
When T is very large such as ∞ the tern $\frac{log(m)}{2\alpha T}$ will go to zero and $\bar{Q}$ will be close to Q^*

Your answer.

Problem 2. PCA.

(**Total 7 points**)

- (**3 point**) Consider the full SVD of $X = USV^\intercal$. Define

$$S_1 := \{D \in \mathbb{R}^{d \times k} : D^\intercal D = I\}, S_2 := \{V^\intercal D : D \in S_1\}.$$

  Show that $S_1 = S_2$.

- (**4 points**) Now use the fact above to show that

$$\max_{D \in S_1} \|XD\|_F^2 = \max_{D \in S_1} \|SD\|_F^2$$

Your answer. a)
As we know $V^T V = I and D^T D = I$ lets start from $D^T D$
$=D^T V V^T D$ (since $VV^T = I$)
$=(VD^T)^T(VD^T)$ (since $VV^T = I$)
from the above statement we can say that $S_2 \subseteq S_1$
As we have already seen before $D = V(V^T D)$ So that $S_1 \subseteq S_2$ We can say that $S_1 == S_2$ from $S_1 \subseteq S_2$ and $S_2 \subseteq S_1$
b)As we have seen in the lecture
$max_{D \in S_1} \|XD\|_F^2 = max_{D \in S_1} \|XVD\|_F^2$
$=max_{D \in S_1} \|USV^T VD\|_F^2$ since$(V^T V = I)$
$=max_{D \in S_1} \|USD\|_F^2$
$=max_{D \in S_1} tr(USD)^T(USD)$
$=max_{D \in S_1} tr((SD)^T U^T U(SD))$ since$(U^T U = I)$
$=\max_{D \in S_1} \|SD\|_F^2$(proved)

Problem 3. Bias-Variance Trade-off.

(**Total 2 points**) In the lecture, **expected test error using a machine learning algorithm, $\boldsymbol{A}$**, can be given as (with respect to squared loss):

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

where D represents set of training points and (x, y) pairs are test points

We can write:

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

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

Proving above 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 1^{st} term is **variance**. The 2^{nd} term can further be decomposed as follows:

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

where 1^{st} term is **Noise** and 2^{nd} term is **Bias²**

b) Prove $E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]=0$

Proving above gives the magic of **Bias-Variance Decomposition**,

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

Your answer. a) As we know $\bar{f}=E[f_D]$(Equation 1) so lets start from
$E_{x,y,D}\left[\left(f_D(x)-\bar{f}(x)\right)\left(\bar{f}(x)-y\right)\right]$
$=E_{x,y,D}[f_D(x)\bar{f}(x) - f_D(x)y - \bar{f}(x)^2 + \bar{f}(x)y]$
$=E_{x,y,D}[f_D(x)\bar{f}(x)] - E_{x,y,D}[f_D(x)y] - E_{x,y,D}[\bar{f}(x)^2] + E_{x,y,D}[\bar{f}(x)y]$
$=E_{x,y,D}[f_D(x)]E_{x,y,D}[\bar{f}(x)] - E_{x,y,D}[f_D(x)]E_{x,y,D}[y] - E_{x,y,D}[\bar{f}(x)]^2 + E_{x,y,D}[\bar{f}(x)]E_{x,y,D}[y]$
and As we know $E_{x,y,D}[E_{x,y,D}[f]] = E_{x,y,D}[f]$
$=E_{x,y,D}[f_D(x)]^2 - E_{x,y,D}[f_D(x)]E_{x,y,D}[y] - E_{x,y,D}[f_D(x)]^2 + E_{x,y,D}[f_D(x)]E_{x,y,D}[y]$

=0(proved)
b) $E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]$
$=E_{x,y}[\bar{f}(x)\bar{y}(x)-\bar{f}(x)y-\bar{y}(x)^2+\bar{y}(x)y]$
As we know $y(x)=E[y/x]=E[y]$
$=E_{x,y}[\bar{f}(x)\bar{y}(x)]-E_{x,y}[\bar{f}(x)y]-E_{x,y}[\bar{y}(x)]^2+E_{x,y}[\bar{y}(x)y]$
$=E_{x,y}[\bar{f}(x)]E_{x,y}y-E_{x,y}[\bar{f}(x)]E_{x,y}y-E_{x,y}[y]^2+E_{x,y}[y]^2$
=0(proved)

Programming Assignment

Instruction. For each problem, you are required to report descriptions and results in the PDF and submit code as python file (.py) (as per the question).

- **Python** version: Python 3.
- Please follow PEP 8 style of writing your Python code for better readability in case you are wondering how to name functions & variables, put comments and indent your code
- **Packages allowed**: numpy, pandas, matplotlib
- **Submission**: Submit report, description and explanation of results in the main PDF and code in .py files.
- Please **PROPERLY COMMENT** your code in order to have utmost readability otherwise **1 MARK** would be deducted

Programming Common

This programming assignment focuses on boosting and bagging approaches.

Problem 4. Boosting.

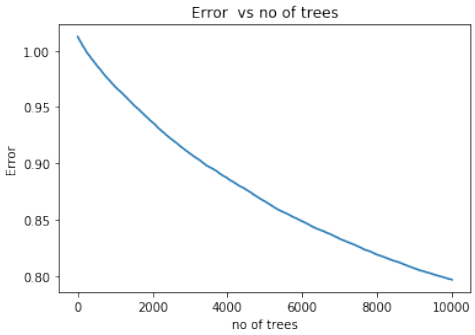
(**10 Points**) Imagine we just have the training dataset visible (no test labels are visible) but a scoring function is made available to you. This score function gives us an idea about how worse our model is doing on the test dataset. Can we achieve better than random performance using **just** the knowledge of how well your model does on test dataset ? If **yes**, then explain how? You have to implement this model. Show how the score changes as we increase the number of weak learners (show a plot).

Note: For getting the score, use the ***score*** function in **test_score.py** script by importing it in your **hw4_boosting.py** file you would submit as solution for this question. The function ***score*** accepts only one parameter which is of type 'numpy.ndarray' and has shape of (N,1) where N is the number of samples predicted passed as argument to this function. Here, we have 21283 samples in testing dataset therefore $N = 21283$.

Keep the **test_score.py** and **true_values.csv** which contains the true values of test dataset in the same location where **hw4_boosting.py** file would be. This ***score*** basically calculates how much error is made in predictions by comparing with true value in **true_values.csv** testing dataset.

Submission: Submit all plots requested and explanation in latex PDF. Submit your program in a file named (hw4_boosting.py).
solution:
As shown in the figure Error Vs no of trees.We can perform better than a random using just the knowledge of score function of all test data ,If we have the access of the whole testdata.If we have the knowledge of partial testdata score then we may/maynot develop a model that performs better than the random model. The reason is we are fitting the weak learners to the model so that the overall Error will decrease.And Based on the score i am generating a random

Figure 1: Error vs no of trees



vector out of 100 random vectors such that their combined score of error is less than the score of previous output vector and the selected vector's combined score is the best score available inside the 100 vectors .

Problem 5. Random forest.

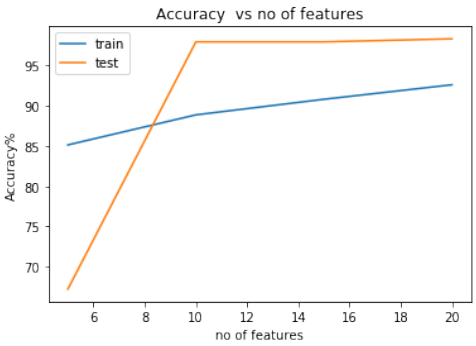
(**Total 25 Points**) For this problem, you will use the Mushroom dataset as in the file Mushroom.csv. You can find this file in the canvas file folder named hw4. It has 2 classes - edible or poisonous and each sample has 22 features. First column is class labels. More details can be found here Link.

Keep first 6000 samples for fitting the model i.e. train and the remaining for testing.

a) (**9 Points**) Implement a random forest of 100 decision trees with max depth = 2 (i.e. 2 layer decision tree). You are allowed to use decision tree classifier from scikit standard library (sklearn link). **No other function from scikit learn is allowed**. Use gini as quality measure for the decision tree splits.

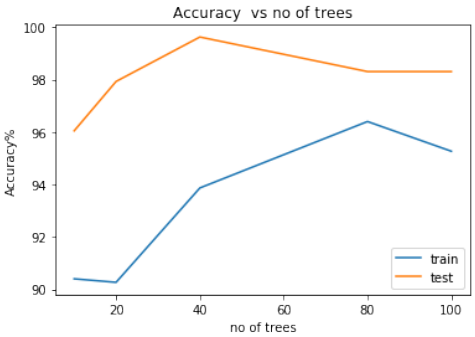
b) (**3 Points**) Vary the size of random feature sets as [5,10,15,20] and fit the model. After the model is fit, plot the accuracy on train and test set vs size of the random feature set.

Figure 2: Accuracy vs no of features



c) (**3 Points**) Vary the number of estimators i.e. number of decision trees as [10, 20, 40, 80, 100] for feature size 20 and plot the accuracy on train and test set for number of estimators.

Figure 3: Accuracy vs No of Trees



d) (**10 Points**) Implement a boosting model similar to above where you just happen to have the knowledge of testing miss-classification error as the **score** for this Mushroom dataset. Plot the bias, variance and expected generalisation error for this implementation as well as for the random forest model as we increase the number of learners.

Submission: Submit all plots requested and explanation in latex PDF. Submit your program in a file named (hw4_random_forest.py).