# Homework 4

CSCI 5525: Machine Learning

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

Please type in your info:

- **Name**, **Student ID**, **Email**
- **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 $\gamma$-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?

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 $Q^*$?

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 $\eta$, 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.

d) (**Let's have more!!! Extra credits: 5 points**) Informally, what does this mean about $\bar{Q}$, what is it converging to as $T \to \infty$?

**Answer.**

a) $\mathbb{P}_{x\sim P}[h(x) \neq c(x)] \leq 1/2 - \gamma \Rightarrow \mathbb{P}_{x\sim P}[h(x) = c(x)] \geq 1/2 + \gamma$.

$$M(P,h) = \mathbb{P}_{x\sim P}[h(x) = c(x)] \geq 1/2 + \gamma \tag{1}$$
$$\min_P \max_h M(P,h) = \min_P \max_Q M(P,Q) = V \tag{2}$$

Combine Eq. (1) and Eq. (2) we can get

$$1/2 + \gamma \leq \min_P \max_Q M(P,Q) \leq V$$
$$\Rightarrow \boxed{V \geq 1/2 + \gamma}$$

b)

$$M(P,Q^*) = V \geq 1/2 + \gamma$$
$$\Rightarrow \mathbb{P}_{x\sim P, h\sim Q^*}[h(x) \neq c(x)] = 1 - V \leq 1/2 - \gamma$$

So the empirical error would be less than or equal to $(1/2 - \gamma)$

c)

d) $\frac{1}{T}\sum_{i=1}^{T} M(x,h_t)$ is the fraction of weak hypotheses that correctly classify x. From part (c), we can see that $\frac{1}{T}\sum_{i=1}^{T} M(x,h_t) > 1/2$, which means that the majority of weak classifiers $h_t(x) = c(x)$ for $t = 1, ..., T$ is greater than 1/2. Therefore $H(x) = sign(\bar{Q}(x)) = c(x)$ for all x.

## 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

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

**Answer.**

1.

$$(V^\intercal D)^\intercal (V^\intercal D) = D^\intercal V V^\intercal D = I$$

Let $D' = V^\intercal D$ be the matrices in $S_2$. Then,

$$S_2 = \{D' \in \mathbb{R}^{d\times k} : D'^\intercal D' = I\} \Rightarrow S_1 = S_2$$

2.

$$\begin{aligned}\max_{D\in S_1} \|XD\|_F^2 &= \max_{D\in S_1} \|USV^\intercal D\|_F^2 \\ &= \max_{D\in S_1} \|USD\|_F^2\end{aligned}$$

From the definition of Frobenius norm,

$$\|USD\|_F^2 = \sqrt{tr[(USD)^\intercal (USD)]} = \sqrt{tr[D^\intercal S^\intercal U^\intercal USD]} = \sqrt{tr[D^\intercal S^\intercal SD]} = \|SD\|_F^2$$

Therefore,

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

## 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:

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

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:

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

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

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**,

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

**Answer.**

1.

$$\begin{aligned}E_{x,y,D}\left[\left(f_D(x)-\bar{f}(x)\right)\left(\bar{f}(x)-y\right)\right]&=E\left[f_D(x)\bar{f}(x)\right]-E\left[\bar{f}(x)^2\right]+E\left[\bar{f}(x)y\right]-E\left[f_D(x)y\right]\\&=\bar{f}(x)E\left[f_D(x)\right]-\bar{f}(x)^2+E\left[E[f_D(x)]y\right]-E\left[f_D(x)y\right]\\&=\bar{f}(x)^2-\bar{f}(x)^2+E\left[f_D(x)y\right]-E\left[f_D(x)y\right]\\&=0\end{aligned}$$

2.

$$\begin{aligned}E_{x,y}\left[\left(\bar{f}(x)-\bar{y}(x)\right)\left(\bar{y}(x)-y\right)\right]&=E\left[\bar{f}(x)\bar{y}(x)\right]-E\left[\bar{f}(x)y\right]+E\left[\bar{y}(x)y\right]-E\left[\bar{y}(x)^2\right]\\&=\bar{f}(x)\bar{y}(x)-\bar{f}(x)\bar{y}(x)+\bar{y}(x)^2-\bar{y}(x)^2\\&=0\end{aligned}$$

## 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).

**Answer.** I first create a slightly perturbed prediction from test data and calculate its initial score. Then, I ran a loop for 1000 times, and in each iteration, a new slightly perturbed prediction is created. If the new prediction has lower score than the initialized prediction, add the new prediction for boosting. The weight for new prediction is calculated according to its

score. The lower the score, the higher the weight is and vice versa. The training result is as the following figure.

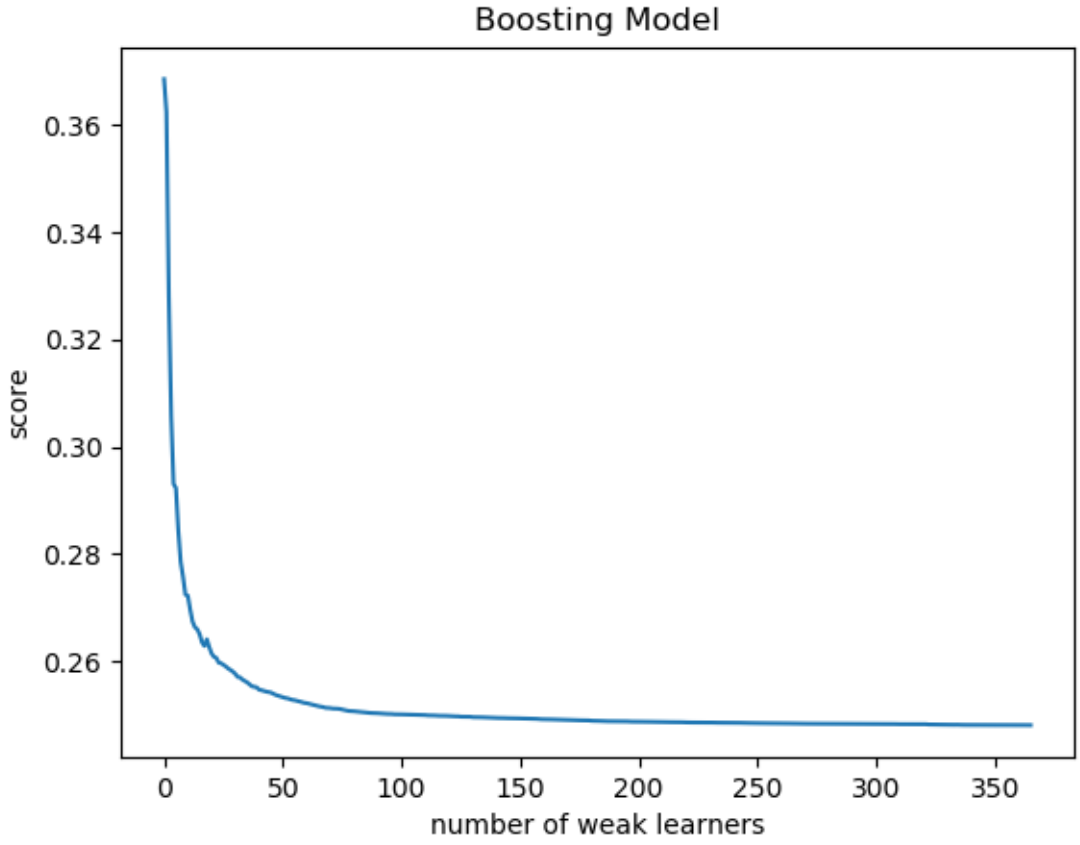


Figure 1: Boosting Result

## 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.

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.

**Answer.**

a) Pleas see parts (b) and (c)

b) From the result we can see that when the feature size is too small, the accuracy is lower. This is because when the random feature size is too small, the model would underfit.

```
feature size: 5
train accuracy: 88.56666666666668 %
test accuracy: 49.90583804143126 %
feature size: 10
train accuracy: 98.66666666666667 %
test accuracy: 97.92843691148776 %
feature size: 15
train accuracy: 94.8 %
test accuracy: 98.30508474576271 %
feature size: 20
train accuracy: 94.8 %
test accuracy: 98.30508474576271 %
```

Figure 2: Boosting Result

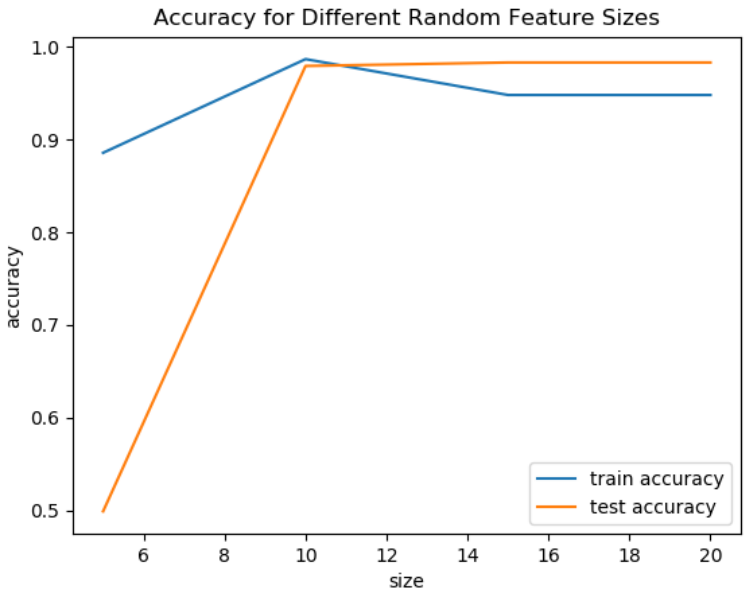


Figure 3: Boosting Result

c) From the result, we can see that the number of trees doesn't affect the accuracy.

```
number of trees: 10
train accuracy: 94.8 %
test accuracy: 96.045197740113 %
number of trees: 20
train accuracy: 94.8 %
test accuracy: 98.30508474576271 %
number of trees: 40
train accuracy: 94.8 %
test accuracy: 98.30508474576271 %
number of trees: 80
train accuracy: 94.8 %
test accuracy: 98.30508474576271 %
number of trees: 100
train accuracy: 94.8 %
test accuracy: 98.30508474576271 %
```

Figure 4: Boosting Result

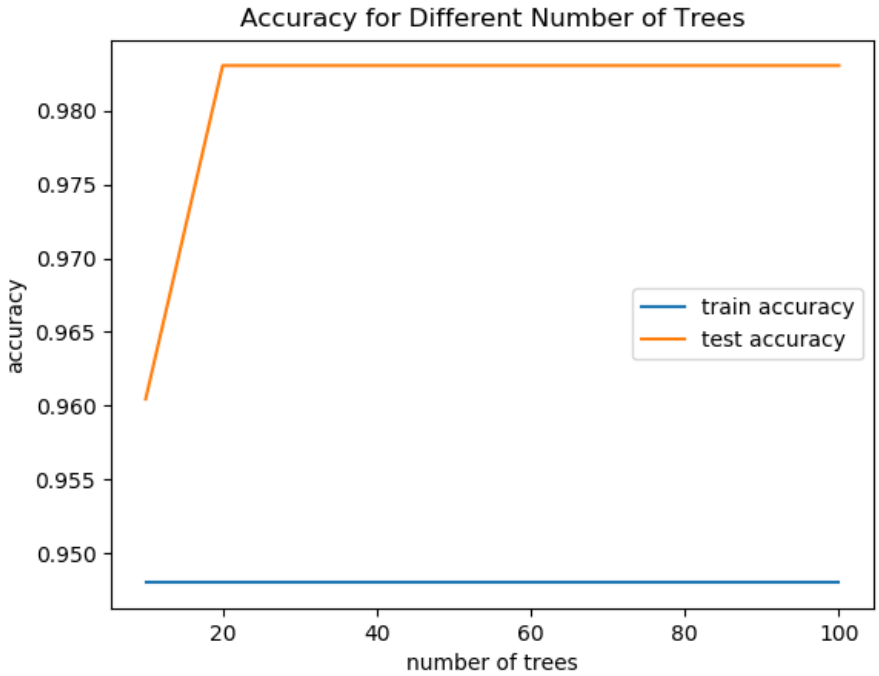


Figure 5: Boosting Result

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