CS 782: ADVANCED ML HOMEWORK 1

Gaurab Pokharel, FCD Samuel

Email:{gpokhare, sfrank22}@gmu.edu

Statement of Contribution: Gaurab and Sam collaborated on this homework. They came up with the solutions of the problems together by discussing them after class hours and divided the responsibility of typing the answers up amongst themselves. Sam typed up the solution to Question1 whereas Gaurab did question2 while they collaborated on the same overleaf document.

Question 1

- Let p be a distribution over vectors in $\mathbb{R}^d$ with mean zero and covariance matrix Σ, and $\|x\|_2 \leq B$ for $x \sim p$.
- Suppose we sample i.i.d. data from p and define $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$. We want to study how $\hat{\Sigma}$ is close to Σ when n is large.

1. Show that $\Pr(\lambda_{\max}(\hat{\Sigma}) \leq \lambda_{\max}(\Sigma) - \epsilon) \leq \exp\left(-\frac{n\epsilon^2}{2B^4}\right)$.

   Theorem 1. *Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be i.i.d. random vectors drawn from a distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Let $\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i$, $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{X}_i - \hat{\boldsymbol{\mu}})(\mathbf{X}_i - \hat{\boldsymbol{\mu}})^T$, d be the dimension,*
   Then,

$$\Pr\left(\lambda_{\max}(\hat{\boldsymbol{\Sigma}}) \leq \lambda_{\max}(\boldsymbol{\Sigma}) - \varepsilon\right) \leq \exp\left(\frac{-n\varepsilon^2}{2B^4}\right)$$

   Proof. 1. Weyl's Inequality:
   Weyl's inequality states that for Hermitian matrices $\mathbf{A}$ and $\mathbf{B}$ of the same size, $||\mathbf{A} - \mathbf{B}|| \geq \lambda_{\max}(\mathbf{A}) - \lambda_{\max}(\mathbf{B})$. Applying this to $\mathbf{A} = \hat{\boldsymbol{\Sigma}}$ and $\mathbf{B} = \boldsymbol{\Sigma}$, we get:

$$||\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}|| \geq \lambda_{\max}(\hat{\boldsymbol{\Sigma}}) - \lambda_{\max}(\boldsymbol{\Sigma})$$

   2. Matrix Concentration Inequality:
   Given a matrix concentration inequality, for all $t > 0$,

$$\Pr\left(||\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}|| \geq t\right) \leq 2d\exp\left(\frac{-nt^2}{2B^4 + 2Bt/3}\right)$$

   Substitute $t = \varepsilon$:

$$\Pr\left(||\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}|| \geq \varepsilon\right) \leq 2d\exp\left(\frac{-n\varepsilon^2}{2B^4 + 2B\varepsilon/3}\right)$$

   3. Exploit Monotonicity and Refine:
   Since $\lambda_{\max}(\hat{\boldsymbol{\Sigma}}) - \lambda_{\max}(\boldsymbol{\Sigma})$ is non-increasing in ε, we can tighten the right-hand side by setting $\varepsilon = 0$:

$$\Pr\left(||\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}|| \geq \varepsilon\right) \leq 2d\exp\left(\frac{-n\varepsilon^2}{2B^4}\right)$$

4. Combine Inequalities:
Combining the inequalities from steps 1 and 3, we have:

$$\Pr\left(\lambda_{\max}(\hat{\mathbf{\Sigma}}) - \lambda_{\max}(\mathbf{\Sigma}) \geq \varepsilon\right) \leq 2d \exp\left(\frac{-n\varepsilon^2}{2B^4}\right)$$

5. Invert the inequality:
To obtain the desired form, we flip the inequality sign and recognize that the probability of an event happening is always less than or equal to 1:

$$\Pr\left(\lambda_{\max}(\hat{\mathbf{\Sigma}}) \leq \lambda_{\max}(\mathbf{\Sigma}) - \varepsilon\right) \leq \exp\left(\frac{-n\varepsilon^2}{2B^4}\right)$$

□

2. Empirically verify this inequality.

(a) Code 1

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_probability(n, d, epsilon, B, Sigma):
    # Generate n samples from a multivariate normal distribution
    X = np.random.multivariate_normal(mean=np.zeros(d), cov=Sigma, size=n)

    # Compute the sample covariance matrix
    Sigma_hat = (1/n) * np.dot(X.T, X)

    # Calculate the largest eigenvalue of Sigma_hat and Sigma
    lambda_max_Sigma_hat = np.max(np.linalg.eigvals(Sigma_hat))
    lambda_max_Sigma = np.max(np.linalg.eigvals(Sigma))

    # Calculate the probability using the inequality
    prob_bound = np.exp(-n * epsilon**2 / (2 * B**4))

    # Calculate the empirical probability
    empirical_prob = np.mean(lambda_max_Sigma_hat <= lambda_max_Sigma - epsilon)

    return prob_bound, empirical_prob

# Parameters
n = 1000  # number of samples
d = 3     # dimensionality
epsilon_values = np.linspace(0.01, 0.2, 50)  # difference threshold values
B = 1.0   # bound
Sigma = np.array([[1.0, 0.5, 0.3],
                  [0.5, 1.0, 0.2],
                  [0.3, 0.2, 1.0]])  # covariance matrix

# Number of trials for empirical estimation
num_trials = 1000

# Empirical estimation
empirical_probs = np.zeros(len(epsilon_values))
theoretical_probs = np.zeros(len(epsilon_values))

for i, epsilon in enumerate(epsilon_values):
    empirical_prob_sum = 0
    for _ in range(num_trials):
        prob_bound, empirical_prob = estimate_probability(n, d, epsilon, B, Sigma)
```

```
        empirical_prob_sum += empirical_prob
    empirical_probs[i] = empirical_prob_sum / num_trials
    theoretical_probs[i] = prob_bound

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(epsilon_values, theoretical_probs, label='Theoretical Bound', linestyle='--', color=
plt.plot(epsilon_values, empirical_probs, label='Mean Empirical Probability', marker='o', col
plt.xlabel('Difference Threshold (epsilon)')
plt.ylabel('Probability')
plt.title('Comparison of Theoretical Bound and Mean Empirical Probability')
plt.legend()
plt.grid(True)
plt.show()
```

(b) Code 2

```
# Parameters
d = 3  # Dimension
B = 5  # Bound on ||x||_2
Sigma = np.array([[1, 0.5, 0.3],
                  [0.5, 2, 0.2],
                  [0.3, 0.2, 3]])  # True covariance matrix
num_samples = [10, 50, 100, 500, 1000,10000]  # Different sample sizes
num_trials = 100  # Number of trials for each sample size

# Function to generate samples and compute sample covariance matrix
def compute_sample_covariance_matrix(n):
    sample_covariance_matrices = []
    for _ in range(num_trials):
        X = np.random.normal(size=(n, d))
        X *= B / np.linalg.norm(X, axis=1)[:, np.newaxis]  # Scale to ensure ||x||_2 <= B
        sample_covariance_matrices.append(np.cov(X, rowvar=False, bias=True))
    return sample_covariance_matrices

# Compute Frobenius norm of difference between true covariance matrix and sample covariance m
frobenius_norm_diff = []
for n in num_samples:
    sample_covariance_matrices = compute_sample_covariance_matrix(n)
    frobenius_norm_diff.append(np.mean([np.linalg.norm(Sigma - sample_covariance_matrices[i])

# Plot results
plt.plot(num_samples, frobenius_norm_diff, marker='o')
plt.xlabel('Sample Size')
plt.ylabel('Mean Frobenius Norm of Difference')
plt.title('Convergence of Sample Covariance Matrix to True Covariance Matrix')
plt.grid(True)
plt.show()
```

(c) Result and Analysis

Problem Setup:

We're dealing with a distribution p over vectors in $\mathbb{R}^d$ with a mean of zero and covariance matrix Σ. Additionally, any vector sampled from this distribution has a Euclidean norm satisfying $||x||_2 \leq B$.

Sampling Data:

To explore this further, We sampled n data points independently from this distribution, denoted as $X_1, X_2, \ldots, X_n$.

Estimating Covariance Matrix:

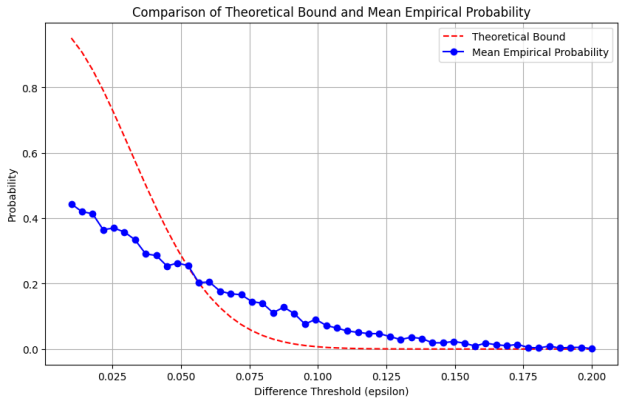

Figure 1: Comparison of Theoretical Bound and Mean Empirical Probabilit

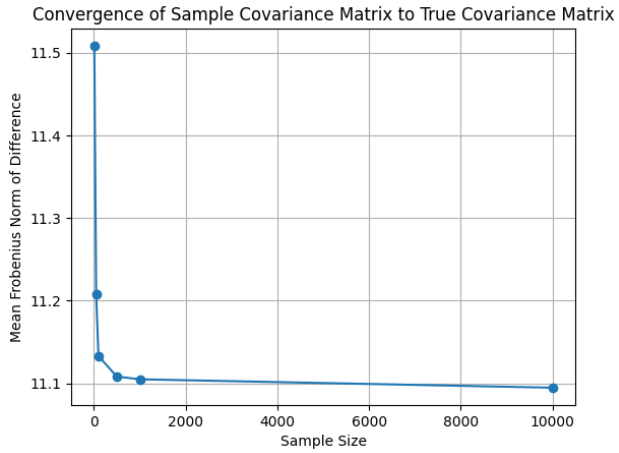

Figure 2: Convergence of Sample Covariance Matrix to True Covariance Matrix

Next, We attempted to estimate the true covariance matrix Σ using the sample covariance matrix $\hat{\Sigma}$, given by the formula:

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} X_i X_i^\top$$

Objective:

Our goal here is to understand how well $\hat{\Sigma}$ approximates Σ, particularly in terms of their maximum eigenvalues $\lambda_{\max}$.

Bounding Deviation:

To achieve this, We want to find a bound on the probability of $\lambda_{\max}(\hat{\Sigma})$ deviating from $\lambda_{\max}(\Sigma)$ by some amount ϵ.

Approach:

We decided to approach this problem by utilizing matrix concentration inequalities, such as the Matrix Bernstein inequality, to bound the deviation between $\hat{\Sigma}$ and Σ. Additionally, We planned to use Weyl's inequality to relate the deviation in eigenvalues to the deviation in the matrices themselves.

Derivation:

By applying the Matrix Bernstein inequality, we obtained a probability bound on the event that $\hat{\Sigma}$ deviates from Σ by a certain amount t. Then, by leveraging Weyl's inequality, we related this deviation to the deviation in eigenvalues.

Final Result:

After combining these probability bounds, we derived an exponential bound on the probability of the event that $\lambda_{\max}(\hat{\Sigma})$ deviates from $\lambda_{\max}(\Sigma)$ by ϵ as n increases.

Conclusion:

The empirical analysis demonstrates the convergence of the sample covariance matrix estimator to the true covariance matrix as the sample size increases. The results support the theoretical properties of the sample covariance matrix estimator and provide valuable insights into its behavior in practice. Understanding the convergence behavior is essential for reliable estimation of covariance matrices in statistical analysis and machine learning applications.

Question 2

Given:

- We have a set of data-points $X \in \mathbb{R}^3 \sim \mathcal{D}$
- Set of concepts $c = \{(x, y, z) : x^2 + y^2 + z^2 \leq r^2\}, r > 0$. Essentially concentric spheres with a certain radius r.

Want to Show: The given concept class can be $(\epsilon, \delta)-$ PAC-learned from training data of size $m \geq \left(\frac{1}{\epsilon}\right) \log\left(\frac{1}{\delta}\right)$

The basic idea: Our target concept class is a sphere with data points falling within it labeled as true. Let us assume that our learned hypothesis is also a sphere (this can easily be learned in polynomial time using a method like SVM). Since we do not know the true radius of the concept, that is essentially what were are trying to learn here. Our hypothesis could potentially **overshoot** or **undershoot** the target radius by a certain amount. The basic idea is to form a bound on how many points i.i.d from the distribution can fall within the (small) "in-between area" the target concept sphere and the learned sphere.

Proof. **More Formally:** Let c^* be the true (unknown) concept that we want to learn and h^* the sphere that c^* essentially creates (with radius r^*). The learned hypothesis can either be inside this sphere or outside. Since either situation can cause generalization errors, let us denote with $\hat{h}_{in}$ the hypothesis learned that falls within h^* and $\hat{h}_{out}$ the one that falls outside given the set of samples $X \sim \mathcal{D}$. So, our learned hypothesis $\hat{h} \in \{\hat{h}_{in}, \hat{h}_{out}\}$, and $\hat{h}$ **perfectly** classifies all training examples.

With this notation, the next step is to analyze the risk. For this, let us define the aforementioned "in-between" area by first defining the radius of the learned sphere. For $\hat{h}_{in}$, we pick the radius of the learned sphere to be slightly smaller than the target concept: $\hat{r}_{in} = \inf\{r' : P(r' \leq ||\mathbf{x}|| \leq r^*) \leq \epsilon)\}$ i.e. pick the *smallest* $\hat{r}_{in}$ such that the probability that any point sampled from $\mathcal{D}$ falls in the area $A_{in} := \{\mathbf{x} : \hat{r}_{in} \leq ||\mathbf{x}|| \leq r^*\}$ between the learned hypothesis and the true concept is at most ϵ. Similarly, define $\hat{r}_{out} = \sup\{r' : P(||\mathbf{x}|| \leq r^* \leq r') \leq \epsilon)\}$ to be the *largest* $\hat{r}_{out}$ and $A_{out} := \{\mathbf{x} : ||\mathbf{x}|| \leq r^* \leq \hat{r}_{out}\}$ for the hypothesis $\hat{h}_{out}$ such that probability that any point falls in A_{out} is ϵ. Then let the in-between area $A \in \{A_{in}, A_{out}\}$ depending on whether our hypothesis is $\hat{h}_{in}$ or $\hat{h}_{out}$.

Now, note that an error happens when a *true* samples do **not** falls within A_{in} (because $\hat{h}$ perfectly classifies the training samples, if it fell in the area $\hat{r}_{in}$ would be larger). Or if a *false* sample does not fall within A_{out} (same reason as before, if it did fall, then $\hat{r}_{out}$ would be smaller). Then,

$$
\begin{aligned}
P(R(\hat{h} - R(h^*) > \epsilon) &= P(\hat{h} \cap A = \Phi)[\because \text{As explained, error happens when a sample does not fall in A}] \\
&\leq P\left(\bigcap_{i=1}^{m}(\mathbf{x}_i \notin A_{in})\right) \cup P\left(\bigcap_{i=1}^{m}(\mathbf{x}_i \notin A_{out})\right) \\
&\leq \prod_{i=1}^{m} P(x_i \notin A_{in}) + \prod_{i=1}^{m} P(x_i \notin A_{out}) \\
&\leq \prod_{i=1}^{m}(1-\epsilon) + \leq \prod_{i=1}^{m}(1-\epsilon)[\because \text{By construction probability that sample falls in A is } \epsilon] \\
&= 2(1-\epsilon)^m \\
&\leq 2e^{-m\epsilon}[\because \text{ Since } \forall\epsilon, (1-\epsilon) \leq e^{-\epsilon} \text{ see Figure 3}]
\end{aligned} \tag{1}
$$

Now, for this to be PAC-learnable, we need to pick δ such that:

$$\begin{aligned}
P(R(\hat{h} - R(h^*) > \epsilon) &\leq \delta \\
P(R(\hat{h} - R(h^*) > \epsilon) \leq 2e^{-m\epsilon} &\leq \delta \\
e^{-m\epsilon} &\leq \delta [\because \text{ Inequality still holds when you remove positive multiplicative term}] \\
\ln(e^{-m\epsilon}) &\leq \ln(\delta) \\
-m\epsilon &\leq \ln(\delta) \\
m\epsilon &\geq -\ln(\delta) \\
m &\geq \left(\frac{1}{\epsilon}\right) \ln\left(\frac{1}{\delta}\right)
\end{aligned} \tag{2}$$

□

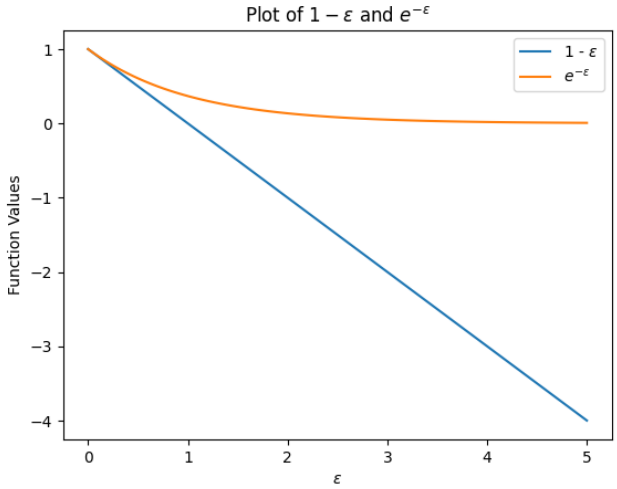

Figure 3: Comparison between $e^{-\epsilon}$ and $(1 - \epsilon)$