A Semester of Robust Statistics

Joshua Dudley

Spring 2021

Abstract

This is an independent-study course supervised by Prof. Yu Cheng. The main topic of the course is high-dimensional robust statistics. When a small fraction of the input samples is adversarially corrupted, even for the most basic statistical problems like estimating the mean and covariance of a distribution, classic statistical estimators either have dimension-dependent error guarantees or are hard to compute in the worst case. From an introductory level, we explain the basic pathway into understanding the field of Robust Statistics.

Chapter 1

Introduction

1.1 Basic Concepts

From an introductory statistics class, one is introduced to the concept of a Normal/Gaussian distribution. The Normal distribution, $\mathcal{N}(\mu, \sigma)$, is a continuous distribution with a mean (μ) and standard deviation (σ). However, there exists a generalization to our Normal Distribution, called the Multivariate Gaussian Distribution, that scales into higher dimensions. The Multivariate Gaussian distribution, $\mathcal{N}(\mu, \Sigma) \in R^d$, is a distribution that has a center μ, a 1xd vector, and a covariance matrix Σ, a dxd matrix. Using the idea of a Multivariate Gaussian Distribution, we can begin to talk about the foundation of robust statistics.

The first task one might ask would be: Given a set of N independent samples from the normal distribution D = $\mathcal{N}(\mu, \sigma)$, with a sample mean of $\bar{\mu}$ and sample standard deviation of $\bar{\sigma}$, estimate the true mean. Knowing the central limit theorem, if given a sufficiently large sample size, we can approximate the $\mu = \bar{\mu}$. Otherwise, we know that $\bar{\mu}$ is off at most the standard error of the mean: $\bar{\sigma}/\sqrt{N}$. In the high dimensional case where we are given a set of N' independent samples from the Spherical Gaussian D' = $\mathcal{N}(\mu$,I), we can see that for any x $\in$N that $\|x\|$ for any sample is at approximately $\sqrt{d}$. Hence we can safely say that with high probability that the sample mean will have a L2-error of O($\sqrt{d/N}$). Although we can estimate the mean within a certain bound, there are 2 critical flaws:

1. As we scale into high dimensions, d can become very large causing the bound to become extended quite far.

2. We are under the assumption that our sampled data perfectly represents the sampled distribution with no corruption.

1.2 ϵ corruption

In modern day, we see data coming from all fields. When data collection occurs, bad data can come anywhere from human error to natural outliers. In light of all the possible contaminations models robust statistics aims to minimize the margin of error for key characteristic from corrupted sets of data. This leads us into our definition of an ϵ corrupted set.

Definition 1.1. Given $0 < \epsilon < 1/2$ and a family of distributions $\mathcal{D}$ on R^d, the adversary operates as follows: The algorithm specifies some number of samples N, and N samples $X_1, X_2, ..., X_N$ are drawn from some (unknown) D $\in \mathcal{D}$. The adversary is allowed to inspect the samples, removes ϵN of them, and replaces them with arbitrary points. This set of N points is then given to the algorithm. We say that a set of samples is ϵ-corrupted if it is generated by the above process.

In Robust Mean Estimation, we aim to create an algorithm that takes in an ϵ-corrupted set and outputs a vector $\hat{\mu}$ such that it can minimize the l2-error of $\|\hat{\mu} - \mu\|$ (generated vector - true mean).

1.3 Mean vs Median in the presence of ϵ corruption

When looking for key characteristics to describe a distribution, it is natural to select the mean and median. However, we will show that with the presence of corruption, the mean and median have drastically different behavior. Given a small random dataset sampled from $\mathcal{N}$(0,1) of N = 10 with the following code

```
import numpy as np

mu, sigma, n = 0, 1, 10

s = np.random.normal(mu, sigma, n)
print(s, np.mean(s), np.median(s))
```

A sample output
s $\Rightarrow$[-0.43500325, -2.56214603, -0.0716686 , 2.53557785, 0.09287595, -0.23981618, 0.94681163, -0.64767749, -0.99527079, -0.18717869]
Sample mean $\Rightarrow \bar{\mu}$ = -0.15634956054535826
Sample Median $\Rightarrow \bar{\sigma}$ = -0.213497435.
We want to test what would happen with the addition of an adversary. Given the following code to corrupt

```
import numpy as np
import random

mu, sigma, n = 0, 1, 10
```

```
epsilon = 0.1
n_infected = int(epsilon * n)

s = np.random.normal(mu, sigma, n)

locations = np.random.choice(n, n_infected , replace = False)
for location in locations:
    s[location] = 100

print(s, np.mean(s), np.median(s))
```

With an ϵ of 0.1, the adversary would be able to corrupt 1 data point to make the corrupted set:
s $\Rightarrow$ [-0.43500325, -2.56214603, -0.0716686, 2.53557785, 100, -0.23981618, 0.94681163, -0.64767749, -0.99527079, -0.18717869].
corrupted mean $\Rightarrow \bar{\mu}$ =9.834362845
corrupted median $\Rightarrow \bar{\sigma}$ = -0.213497435
Although only 1 point got corrupted, our example showed that our $\hat{\mu}$ had a net change of 9.99 and our median had a net change of 0. This suggestis the median potentially has a stronger resistance to corruption than the mean and can be used as a more accurate descriptor of the overlying distribution. In the normal distribution case, we can see that on $\mathcal{O}(1/\epsilon^2)$ samples of data, we can see that with high probability $|\hat{\mu} - \mu| < \epsilon$.

This same idea can be also be shown when scaling this principle to high dimensions. Supposed we are experience corruption via the following code:

```
import numpy as np

# N-high dimensional
epsilon = 0.1
dimensions = 50
mean = np.zeros(dimensions, dtype=int)
mean2 = np.ones(dimensions)*100
cov = np.identity(dimensions)

N = int(dimensions/epsilon**2)
N_epsilon= int(dimensions/epsilon)

data = np.random.default_rng().multivariate_normal(mean, cov, N)
corrupt_data = np.random.default_rng().multivariate_normal(mean2, cov, N_epsi

random_locations = np.random.choice(N, N_epsilon, replace = False)

for count , value in enumerate(random_locations):
    data[value] = corrupt_data[count]

print("Mean: ", np.linalg.norm(np.mean(data, axis = 0)))
```

```
print("Median: ", np.linalg.norm(np.median(data, axis = 0)))
```

We see can consistently get a mean ≈ 70 and a median ≈ 1, once again expeimentally showing that the median would be a better descriptor in the high dimensional case.

Using mathmatical techniques, we can create approximations of how far the median will be in the face of corruption.When sampling from a d dimensional guassian distribution, one can first note that for any sample x, $E[\|x\|] = \sqrt{d}$. Becuase we know that for any sample median $\bar{s}$ and true mean s, $|\bar{s} - s| < \mathcal{O}(\epsilon)$, and that the $\|x\|_2 = \sqrt{\Sigma_i (v_i)^2}$. We can see that the $\|\bar{s} - s\| \approx \sqrt{\Sigma_d \epsilon^2} = \epsilon\sqrt{d}$. We note that this approximation gets worse as the dimensions scale into large numbers. This can be problamatic when working in the world of data which often requires high dimensional data.

1.4 (ϵ, δ) stable sets

When trying to create efficient algorithims, first we must be able to get good data to test on. This introduces the idea known as a stable set. One can say a set is a stable set if and only if it satisifes the the stability condition.

Definition 2.1 (Stability Condition) Fix $0 < \epsilon < 1/2$ and $\delta \geq \epsilon$. A finite set $S \subset \mathcal{R}^d$ is (ϵ, δ) - stable (with respect to a distribution X) if for every unit vector $v \in \mathcal{R}^d$ and every $S' \subseteq S$ with $|S'| \leq (1 - \epsilon)|S|$, the following condition holds:

1. $|\frac{1}{|S'|}\Sigma_{x \in S'} v \cdot (x - \mu_X)| \leq \delta$ and
2. $|\frac{1}{|S'|}\Sigma_{x \in S'} v \cdot (x - \mu_X))^2 - 1)| \leq \frac{\delta^2}{\epsilon}$

In laymans terms, Definition 2.1 is saying that for any set S, for any subset with the cardinality of $(1-\epsilon)$, if all the subset sample means are less than *delta*, and all the subset sample variances are $1 \pm \frac{\delta^2}{\epsilon}$ in every direction, then we can say that the set is (ϵ, δ) - stable. We can see this by reducing the conditions. For the first condition:

$$|\frac{1}{|S'|}\Sigma_{x \in S'} v \cdot (x - \mu_X)| \leq \delta]$$

We can see, that the unit vector is a constant that we can take out.

$$|v\frac{1}{|S'|}\Sigma_{x \in S'} (x - \mu_X)| \leq \delta]$$

We can see that $\frac{1}{|S'|}\Sigma_{x \in s'}$ x is the sample mean over s', so we can call that $\mu_{s'}$.

$$|v(\mu_{s'} - \mu_X)| \leq \delta]$$

Finally we can note that becuse v is a unit vector, we can reduce that away. Allowing for us to be left with

$$\|\mu_{s'} - \mu_X\| \leq \delta]$$

A similar process can be done with the second condition to result in $|\sigma_{s'} - 1| \leq \frac{\delta^2}{\epsilon}$ or $\sigma_{s'}| \leq 1 \pm \frac{\delta^2}{\epsilon}$

Using the defintion of stability, there is been found conditions which the empirical mean is certifiably close to the true mean.

Lemma 2.4 (Certificate for Empirical Mean) Let S be an $(\epsilon, \delta) - \textit{stableset}$ with respect to a distribution X, for some $\delta \geq \epsilon \geq 0$. Let T be an ϵ-corrupted version of S. Let μ_T and Σ_T be the emprical mean and covariance of T. If the largest eigenvalue of Σ_T is at most $1 + \lambda$, for some $\lambda \geq 0$, then $\|\mu_T - \mu_X\| \leq \mathcal{O}(\delta + \sqrt{\epsilon\lambda})$.Which can be tested using the following code:

```
import numpy as np
import math

# sampling properties
epsilon = 0.1
#delta = epsilon * math.sqrt(math.log(1/epsilon)) # proposition 2.2
delta = math.sqrt(epsilon) # proposition 2.3

# distribution properties
dimensions = 50
mean = np.zeros(dimensions, dtype=int)
cov = np.identity(dimensions)

# taking N samples
#N = int(dimensions/epsilon**2) # proposition 2.2
N = int(dimensions/epsilon) # proposition 2.3

S_prime = np.random.randint( (1-epsilon)*N , N) # grabs random number from (
S_prime_locations = np.random.choice(N, S_prime , replace = False) # picks S_

data = np.random.default_rng().multivariate_normal(mean, cov, N)
# the actual sampling


# finds the sum of all means and variances
mean_sum = 0
variance_sum = 0
for location in S_prime_locations:
    mean_sum += np.mean(data[location] - mean)
    variance_sum = np.var(data[location]- mean, ddof = 0)

# finds the average
average_mean_diff = mean_sum / S_prime
average_variance_diff = variance_sum / S_prime

# is the sampled dataset epsilon delta stable?
```

```
stable = False

if abs(average_mean_diff) < delta and abs(average_variance_diff) < (delta**2),
    stable = True

print("Stable? ", stable)
# Lemma 2.4
if stable:
    # corrupting data

    # corrupt distribution
    mean2 = np.ones(dimensions)
    N_2 = int(N * epsilon)
    corrupt_cov = np.identity(dimensions)

    corrupt_data = np.random.default_rng().multivariate_normal(mean2, corrupt
    corrupt_locations = np.random.choice(N,N_2, replace = False)

    for count, value in enumerate (corrupt_locations):
        data[value] = corrupt_data[count]

    # corrupted mean and covariance
    new_mean = np.linalg.norm(np.mean(data, axis = 0))
    new_covariance = np.cov(data)

    # grabs the largest eigenvalue of covariance
    eigen_values, eigenvectors = np.linalg.eig(new_covariance)

    largest_eigen = np.linalg.norm(max(eigen_values))

    if largest_eigen >= 1:
        difference = np.linalg.norm(new_mean)
        if difference < delta + math.sqrt( epsilon * (largest_eigen -1)):
            print("Certificate for empirical mean")
```

What Lemma 2.4 is saying is that, if given an ϵ corrupted set of points T, and has a bounded covariance. The sample mean of T is approximately close to the true mean.