Independent Study - Spring 2021

Viraj Joshi

UT Austin

Abstract -

We survey an extension of Prophet Inequalities, Prophet Inequalities with Linear Correlations, one in which traditional threshold algorithms fail to provide optimal guarantees. We provide some interesting experimental results on example datasets. Then, we move onto alternate view of the same problem, robust statistics. Again, we provide verification of performance guarantees with synthetic datasets.

1 Introduction

An investor sees an order book of bids and asks on an arbitrary security is trying to maximize their profit using the common sense principle of "buy low, sell high". Inescapable issues with this strategy are holding too long, leaving the value of the security to fall without exiting their position, or selling early and regretting a rise in price later. This is the optimal stopping problem,that is, given a stream of values where the distribution of each is possibly known, how can we maximize expected profits when we must irrevocably pick a certain item in such stream? In this scenario, the metric we optimize is called a prophet inequality - comparing the performance of an online algorithm (the investor) to that of an omniscient offline one (a Prophet). What if that order book was spoofed with other investors manipulating prices, how can we robustly estimate parameters relevant to our strategy?

1.1 Formal Problem

We are provided a stream $\{X_i\}_{i=1}^n$ of n random variables, where each X_i is independent and non-negative. In a Prophet Inequality problem, the investor or gambler as literature puts it, knows the distribution of each X_i and sees each of the n realizations in order. The gambler must irrevocably pick a realization in the stream or keep examining remaining X_i until all n samples are exhausted. As a prophet inequality problem, we also suppose there is a Prophet who knows all realizations beforehand and simply picks the maximum.

A relevant metric is the **competitive ratio** - the gambler wins an α-fraction of the prophet's pick for a given stream. Then, for all possible sequences of random variables, we seek the infimum of competitive ratios for a given strategy used by the gambler and a set of conditions on the sequence. We call this an **α-competitive strategy**.

Famously, *Kleinberg and Weinberg*[1], came up with the tight upper bound on the threshold strategy when the sequence of n random variables are independent and non-negative. They found simply taking the first sample to exceed a threshold of $\mathbb{E}[max\{X_i\}_{i=1}^n]$ yields 1/2-competitive strategy.

2 Linear Correlations

ISW[2] considers an interesting set of conditions on the sequence provided to the gambler. We are provided a sequence $\{X_i\}_{i=1}^n$ in order, where each X_i linearly depends on m independent non-negative features $\{Y_i\}_{i=1}^m$ that we also know the distributions of. We can characterize this dependence using a matrix A, where **A** is a (n,m) matrix of non-negative weights, such that

$$\mathbf{X} = A \cdot \mathbf{Y}$$

The gambler is provided the matrix, but not the realizations of Y_j.The A matrix is best thought of as capturing market dynamics. By construction,

1. bidder i where i $\in$ [0,n) places an ask of weight A_{ij} on feature j
2. the seller,the one who is accepting/rejecting bids for every X_i, does not have access to Y_j

In this variation of the Prophet Inequality, the paper thoroughly justifies why a single threshold τ by KW[1] or SC[2] would fail to offer a 1/2-competitive strategy. Observing that with linearly correlated X_i, we intuitively know that if we see a X_i over τ, then it might be correlated with some other X_j being large. This is similar to the investor regretting selling their security earlier, when it later increases in price.

2.0.1 A trivial example of this phenomena

Consider that we have an algorithm using the threshold rule that is a **α-competitive strategy**. Suppose,that our independent sequence $\{X_i\}_{i=1}^n$ where X_i Bernoulli(p) for p « $\frac{1}{n}$ that an adversary 'augments', as ISW puts it, the first random variable by some arbitrarily tiny value. Using the median threshold by SC [**?**], the same gambler who achieved a **α-competitive strategy** will have the first value taken because their threshold is is zero,and therefore, results in a very poor competitive ratio.

2.0.2 ISW's findings

Let s_{row} be the upper bound on number of non-zero entries in any row of A.Let s_{col} be the upper bound on number of non-zero entries in any column of A. In other words, s_{row} is the max number of features Y_js that explain X_i, and s_{col} is max number of samples X_is that explain Y_js.

ISW[2] proved

1. $\frac{1}{\theta(min(s_{row}, s_{col}))}$ competitive strategy for single item selection exists.
2. When $r >> s_{row}$, where r is the number of items a gambler may pick, there are $\frac{1}{1+o(1)}$ competitive strategies.

In this problem, our updated Prophet Inequality bound looks familiar, that we have a **α-competitive strategy** for some $\frac{1}{f(n, s_{row}, s_{col})}$

$$\mathbb{E}[ALG] \geq \frac{1}{f(n, s_{row}, s_{col})} \cdot \mathbb{E}[max\{X_i\}_{i=1}^{n}]$$

2.1 Using FB Prophet

While most papers focus on algorithms that one could program, with the ease of use of ML libraries, one could apply those. Here, we apply FB Prophet to a time series of daily average New Delhi climate for many years.

2.1.1 Single Choice Maximum

We can view the n samples as a dataset which can be split into the traditional traning, validation, and testing dataset. Then, in the analysis of our Prophet Inequality, we can analyze the problem in two steps.

1. We would use a single model trained on some ϵ-fraction of the sequence and with predictions for the remaining $1 - \epsilon$- fraction sequence, we attempt to beat the Prophet and obtain competitive ratios. This is Figure 1.
2. However, as we recall, we want a ratio over all possible sequences. This would let us know whether we were lucky in 1) and if our strategy is immune to an adversary tricking the model through manipulating the order of the sequence.

To implement (2), we simply retrain our model N times using a shuffled, concatenated dataset of training and testing data. Then, we obtain a N-bootstrap statistic for our competitive ratio for the maximum horizon length(with the start of the horizon being the first sample in the sequence) on a new testing dataset. This is an **important** parameter one could change to get better results because the competitive ratio varies as horizon length change(Figure 1) and is very much dependent on how accurate the model is.

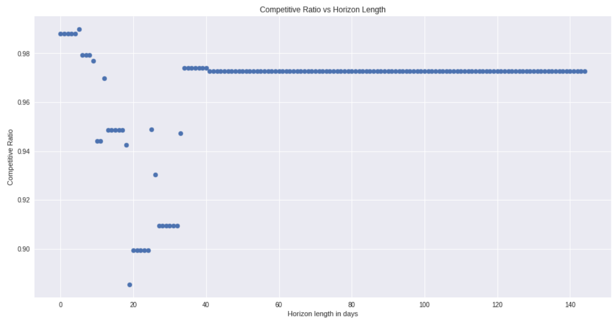

Figure 1. Observe different competitive ratios as horizon changes using FB Prophet

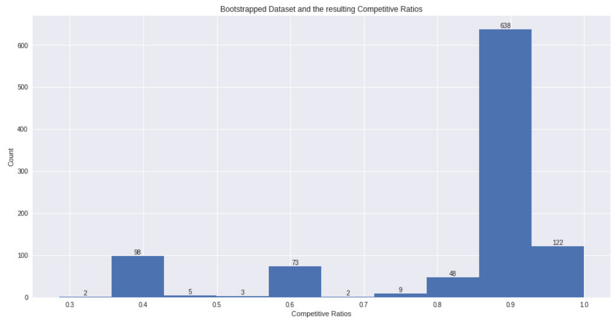

Figure 2. Observe different alpha spreads for the FB-Prophet Strategy for single choice over many trial runs

Table 1. Bootstrapped Mean Competitive Ratio

α-competitive FB Prophet	N = 1000	N = 100
Mean	0.81376	0.75596
Std Deviation of Mean	0.17233	0.18132

Following that procedure, FB Prophet,with very little tuning, yielded a 0.81376-strategy (Table 1), a ratio quite high. This says a gambler cannot hope to win, in expectation, more than 80% of the winnings of a Prophet.Of course, one could compute over a larger number of bootstrap iterations (10000) and other datasets can be tried, but this is promising behavior.

2.1.2 Maximum Sum of *r* choices

We apply exactly the same reasoning in (2) towards a gambler whose goal is to pick the top-r options in the sequence.

Table 2. Bootstrapped Mean Competitive Ratio

α**-competitive FB Prophet**	μ	Std. Dev. of μ
r=.6	0.91762	0.01746
r=.4	0.90429	0.02103
r=.2	0.84098	0.02753

In exchange for similar alpha ratios to that of the one-choice scenario, we get a much tighter spread, with far fewer bootstrap iterations.

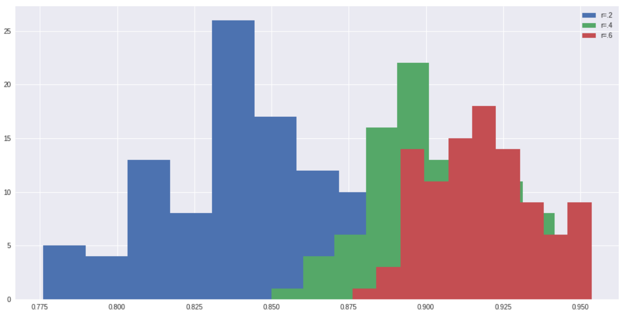

Figure 3. Observe different alpha spreads for the FB-Prophet Strategy over many trials

2.2 Using ISW's Algorithm for Selecting Multiple Samples

Taking a similar approach as 2.1.1 (2), we see that instead of reserving part of the dataset for training a model, potentially missing the maximum to choose, we can now just examine a sequence in order just as *Kleinberg and Weinberg* did.

Table 3. Bootstrapped Mean Competitive Ratio

α**-competitive ISW**	μ	Std. Dev. of μ
r=.4	0.99043	0.00210
r=.2	0.97667	0.00446
r=.1	0.95967	0.01640

Table 3 shows that we obtain a very high α in our α**-competitive ISW**. Furthermore, the spread is order of magnitudes smaller than that out of the single choice sample case of 2.1. What we see here, in fact, is that ISW's algorithm is robust to adversarial interference in the order/distributions of and in the sequence. This is unlike that of FB Prophet, which had a lower alpha strategies at higher proportions of sample taken by the gambler(Table 2).

2.2.1 Caveat in analysis

Unfortunately, relating this to big O notation of ISW is hard because ISW prescribes that m random variables of known distribution and an A matrix is given. However,

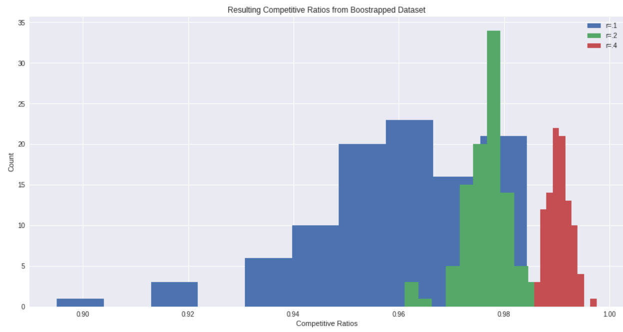

Figure 4. Observe different alpha spreads for the ISW Strategy over many trials

in the real world, one must verify that the dataset used has such properties. One cannot simply guess this. In a time-series, it is not always the case that the ith sample in the sequence can be explained by $\sum_j A_{ij}Y_j$ (a linear combination of unknown features). Instead, we focused on the α in α**-competitive strategy**.

For verifying 2.0.1(2), it would make sense, with our empirical data, that the number of choices is far greater than s_{col} (some features become less related to some samples versus others, especially in physical data like rainfall), so our algorithm is indeed a $\frac{1}{1+O(1)}$**-competitive ISW**.

3 High-Dimensional Robust Mean Estimation

In the real-world, many machine learning models are promised data that have distributional assumptions. Such perfect generation of the data is not always possible.

Given n samples and a collection of distributions $\mathcal{D}$, we are provided a sequence $\{X_i\}_{i=1}^n$ drawn from $D \in \mathcal{D}$, and an adversary is able to corrupt some ϵ-fraction of the data arbitrarily. We want $||\mu - \hat{\mu}||_2 = O(\epsilon * \sqrt{log\frac{1}{\epsilon}})$, where μ is the true mean and $\hat{\mu}$, to be as small as possible. One might think MLE is a good approach; it results in a distribution closest in KL-divergence to that of the corrupted dataset. However, this is not necessary close in KL-divergence to the uncorrupted distribution the samples were drawn from.

3.1 Iterative Filtering

Diakonikolas and Kane [DK][3][4] provides a $O(\epsilon \cdot \sqrt{log\frac{1}{\epsilon}})$method to estimate the true mean of an ϵ-corrupted dataset generated from a d-dimensional $\mathcal{N}(\mu, I)$ with $||\mu - \hat{\mu}||_2 = O(\epsilon \cdot \sqrt{log\frac{1}{\epsilon}})$. If the adversary places samples $\theta(\sqrt{d})$ from the mean, then we can use the median of the set to remove points $\Omega(\sqrt{d})$ from the mean.

However, removing any more points could remove inliers. So what should we do?

They reason that it is not necessary to remove all of the corrupted points, only those that make the empirical mean differ significantly from the true mean. This can only happen when the adversary places all outliers in the same direction in our subspace. Otherwise, if outliers were chosen in random directions, they would not affect μ significantly because there is not enough information to distinguish from inliers.

In fact, DK has a commonality in all of their algorithms, that the distribution of samples projected in the direction of the top eigenvector-eigenvalue $(\lambda^*, v*)$ of the empirical covariance matrix can be used for removing outliers in the dataset. In the Filtering case, such a distribution can be used to find a threshold. This threshold is what determines what is an outlier or not.

One can use the code linked in the Conclusion to see how DK's algorithm[3][4] performs under different ϵ and corruptions.

3.1.1 Iterative Filtering 1d Gaussian

Unfortunately, using something like Geometric median or the Tukey median incurs an error of $O(\sqrt{d})$, so we must rely on DK's method.

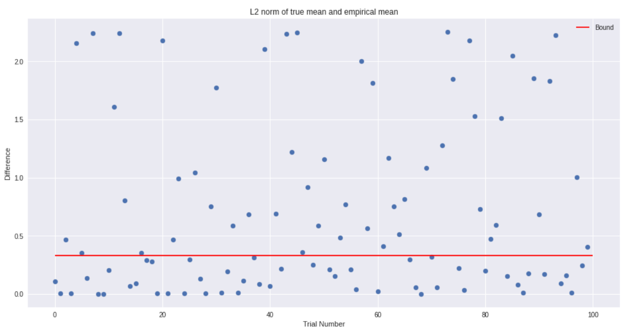

Figure 5. Robust Mean Estimation with $\epsilon = .3; \mu = 10; \hat{\mu} = 11.49972$ from D = 1D Gaussian and corrupted by Gamma Distribution.

In Figure 5, we can see that most points fall below the bound of $C * \epsilon * \sqrt{log\frac{1}{\epsilon}}$, where $C = 1$, below. So a method meant for higher-dimensional mean estimation works in the single dimensional case, albeit that DK prescribes this be a "one-shot" algorithm, the l2-norm is smaller by repeating the algorithm every step (see 4.0.1[2]).

3.1.2 Iterative Filtering d-Gaussian

In Figure 6 and 7, we are robustly estimating the mean with $\epsilon = .1$ and μ = np.random.randn(12). Our data is from a D = 12-Gaussian and corrupted by a N(10,1000) distribution.

Using the same code with minor tweaks, we appear to obtain a less than stellar result.

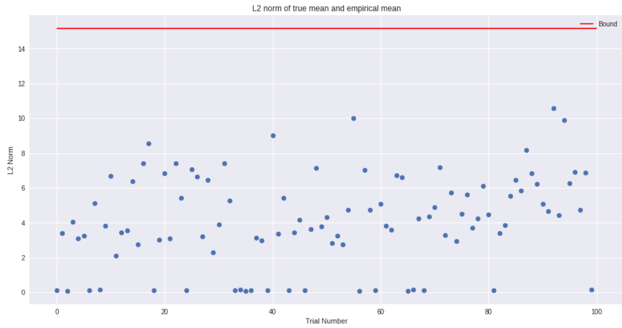

Figure 6. Observe various L2 norm differences between $\hat{\mu}$ and μ

In Figure 6, we can see that all points fall below the bound of $C \cdot \epsilon \cdot \sqrt{log\frac{1}{\epsilon}}$ where $C = 100$ below. Here, we can see much more variability in our estimated μ. Some are very close, tending to zero error, while others are very much differing in every component,resulting in a high L2 norm.

However, correctness of the algorithm is not that the error is small in every run of such an algorithm, but that it outputs a small error with high probability. Over many trials, we can confirm this by observing what iterative filtering outputs on a component-wise mean. Happily, $\hat{\mu}_i \approx \mu_i$, and so while Figure 6 appears to paint a grim outlook for this algorithm, the theory and practice have agreed to produce a stellar result on average. So a researcher or engineer in real-life would simply average results over many runs to obtain a dataset drawn from their expected distribution.

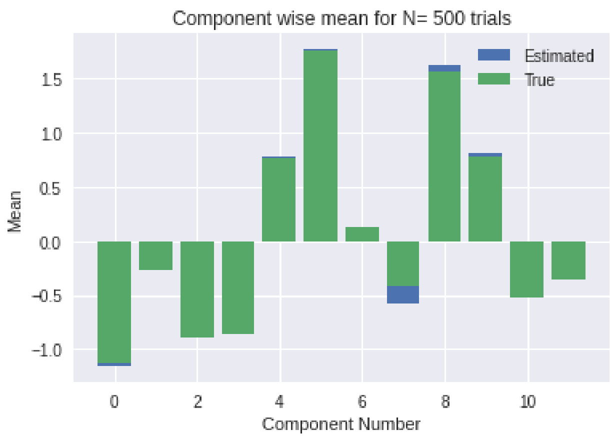

Figure 7.

To note,

1. This method converges very fast in the higher-dimesional case, typically 1 iteration, unlike when the dataset is originally drawn from a 1d Gaussian.
2. When finding a threshold in order to later remove points, we take the dot product of a point drawn from a (n,m) dataset and the top eigenvector of the empirical covariance matrix several times. Obviously, the dot product of (n,1) and (1,m) vector does not compute. Typically n»m, so in the code, one sees a truncation of the length of the eigenvector to that of m. DK did not mention a remedy to this.

4 Conclusion

4.0.1 Computer and Code used

All code was run in a standard Google Colab account

1. Prophet Inequalities
2. Robust Statistics

4.0.2 Future work

I implemented a solution to the higher-dimensional secretary problem we discussed earlier in the semester. The two methods I tried did not have the best results (can be seen in 4.0.1[2]), so I left them out.

DK[3][4] has an algorithm to learn the mean of a set of ϵ-corrupted samples drawn from a mixture of k-Gaussians with identical spherical covariance matrices. As this method is a convex programming algorithm, this would be interesting to implement.

I was reading The Micro-Price: A High Frequency Estimator of Future Prices, a new type of price to predict midprices, in the short-term, of securities. This is of particular importance to HFT firms, and as they are often depicted simultaneously as the boogeyman and savior of markets, I wanted to know more about their practices. Unfortunately, I ran into the issue of forgetting a lot of what I learned in my intro to Stochastic Processes.

References

[1] R. Kleinberg and M. Weinberg. Matroid prophet inequalities. 2012.

[2] Nicole Immorlica, Sahil Singla, and Bo Waggoner. Prophet inequalities with linear correlations and augmentations, 2020.

[3] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability, 2019.

[4] Ilias Diakonikolas and Daniel M. Kane. Recent advances in algorithmic high-dimensional robust statistics, 2019.