Online Single Index Model Learning
Final Report

Prakhar Kulshreshtha	**Pranshu Gupta**	**Sandipan Mandal**
13485	13493	13807616

1 Introduction

One of the most fundamental tasks of Machine Learning is learning a Regression model. The most elementary of these models is perhaps that of Linear Regression. It models the output variable $\mathbf{y} \in \mathbb{R}^{\mathbf{m}}$ as a linear function of a feature vector $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$. Formally, we try to estimate parameter $\mathbf{w} \in \mathbb{R}^{\mathbf{mxn}}$ such that $\mathbf{y} = \mathbf{w^T x} \quad , \forall (\mathbf{x}, \mathbf{y}) \in$ Training Set. However, there are many other interesting variants, such as:

- Isotonic Regression (For 1-D data) : a non-decreasing function $u : \mathbb{R} \to \mathbb{R}$ such that $\hat{y_i} = u(x_i)$ where $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$
- Generalized Linear Models : $y = u(\mathbf{w^T x})$, where link function u is known.
- Single Index Models : $y = u(\mathbf{w^T x})$, both u and $\mathbf{w}$ have to be learnt

1.1 Single Index Model

Single Index Model (SIM) is an interesting model where we have to estimate both $u : \mathbb{R} \to \mathbb{R}$ (popularly called the link function) and $\mathbf{w} \in \mathbb{R}^n$ simultaneously, given labels $y \in \mathbb{R}$ corresponding to feature vectors $\mathbf{x} \in \mathbb{R}^n$, so that we get $y = u(\mathbf{w^T x})$. One of the most popular algorithms to learn SIM is SLISOTRON ([Kakade et al., 2011]), which basically iterates multiple times on learning u, and learning $\mathbf{w}$. However, this is an offline algorithm. As per best of our knowledge, at present there is not any online algorithm for SIM learning. An online SIM model will find its applications in stock market settings, inflation modeling. Basically, it will be relevant in all the SIM problems where data is coming in an online fashion.

2 Formal Problem Description

The primary objective of our project is to design an online algorithm for SIM learning having sub-linear regret bound and polynomial time complexity. Initially we'll assume noiseless data and realisable setting. Formally, problem setting is as follows:

- The data is a collection of $(\mathbf{x}_t, y_t)$ pair, where $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$. The data arrives sequentially.
- Adversary is stochastic.
- No noise in data
- The setting is realizable i.e. $\exists u^*, \mathbf{w}^*$ s.t $u^*(\mathbf{w}^*.x_i) = y_i \; \forall i$
- $\|\mathbf{x}_t\| \leq \mathbf{X}$, $\|\mathbf{w}_t \leq \mathbf{W}$, $\|\mathbf{w}^* \leq \mathbf{W}$, and both u_t and u^* are bounded, smooth, and 1-Lipschitz.

3 Background Literature

In this section, we will briefly review some literature from [Kakade et al., 2011][Kalai and Sastry, 2009][Kotłowski et al., 2016] to explain isotonic regression, isotron, SIM, online isotonic regression model,

- [Kotłowski et al., 2016] gives the first online version of isotonic regression problem. The algorithm uses Exponential Weights played over a covering net of isotonic functions and gives a regret bounded of $O(T^{1/3}log^{2/3}(T))$. But the algorithm has certain drawbacks. It assumes the training set be given before hand. The paper also proves the problem to be unlearnable if the training set is not given in advance.

- The first provably efficient method for learning SIMs and GLMs, under the assumption that the data is in fact generated under a GLM and under certain monotonicity and Lipschitz (bounded slope) constraints was **The Isotron Algorithm** [Kalai and Sastry, 2009] given by Kalai and Sastry (2009). The isotron algorithm is inspired from the batch perceptron algorithm. The paper also proves Perceptron problem with a γ-margin to be a special case of the idealized SIM problem. The work introduces two versions of isotron algorihm.

  - The first version requires m training examples and the algorithm runs through the data once before updating the parameter and this is repeated multiple times. At each iteration, first the transfer function is computed using PAV algorithm on the training data and current parameter. This transfer function is used to compute the squared error which in turn is used to update the parameter. The analysis of the algorithm is very similar to perceptron except that at each iteration, it solves PAV which computes the transfer function in time $O(mlog(m))$.
  - The second version requires number of iterations (T) and Tm training examples. The algorithm runs through a mini-batch of m examples before updating the parameter. At each iteration a new mini-batch is used to calculate transfer function using PAV. This transfer function is used to compute the squared error on the mini-batch which in turn is used to update the parameter.

- The paper[Kakade et al., 2011] introduces a modified version of isotron algorithm named as **SLISOTRON** for SIM learning. The only difference between *isotron-1* and *slisotron* is that in the former the transfer function is computed using PAV while in the later, with *Lipschitz Isotonic Regression* (LIR). It enables the model to fit a smoother function than with simple isotron. The time complexity of LIR is same as PAV ($O(mlog(m))$). It is given as Algorithm 1.

Algorithm 1: SLISOTRON

Input: Training Set $< (x_i, y_i) >_{i=1}^{m}$; Validation Set $< (x_{m+j}, y_{m+j}) >_{j=1}^{s}$
$w^1 \leftarrow 0$
for *t = 1,2,...* **do**
  $u^t = LIR((w^t.x_1, y_1), ..., ((w^t.x_m, y_m))$
  $w^{t+1} = w^t + \frac{1}{m}\sum_{i=1}^{m}(y_i - u^t(w^t.x_i))x_i$
end
Return $argmin_{h^t} \sum_{j=1}^{s}(h^t(x_{m+j}) - y_{m+j})^2$

4 Proposed Algorithm: FTL-SLISOTRON

The algorithm we propose is a simple FTL style extension of of offline SLISOTRON algorithm. Basically, we run SLISOTRON for all $t-1$ points to obtain a $\mathbf{w}_t$ and u_t, which are used to predict $\hat{y^t} = u^t(\mathbf{w}^t\mathbf{x}^t)$. It is given as Algorithm 2. Note that time complexity of SLISOTRON is polynomial in m, the no. of training points. So, FTL-SLISOTRON's time complexity will be polynomial in T.

Algorithm 2: FTL-SLISOTRON

$\mathbf{w}^0 \leftarrow 0$
for *t = 1, 2, ... , T* **do**
  Split the $t-1$ points as held out set $H = \{x_\tau, y_\tau\}_{\tau \in \{1,2,..,m\}}$
  and training set $S_t = \{x_\tau, y_\tau\}_{\tau \in \{m,m+1,..,t-1\}}$, where $m = 0.2\lfloor t-1 \rfloor$
  Receive $\mathbf{x}_t$
  $u^t, \mathbf{w}_t \leftarrow SLISOTRON(S_t, H)$
  Predict $\hat{y}_t = u^t(\mathbf{w}_t \cdot \mathbf{x}_t)$
  Receive y_t
  Incurr loss $l(y_t, \hat{y}_t) = (y_t - \hat{y}_t)^2$
end

5 Regret Bound

After defining the algorithm, we'll now define the notion of regret. Since our adversary is stochastic so we'll obtain a definitional expression of Expected Regret bound with high probability.

5.1 Notion of Regret

We have $\hat{y}_t = h^t(\mathbf{x_t}) = u^t(\mathbf{w}^t\mathbf{x}_t)$ and $\tilde{y}_t = h^*(\mathbf{x_t}) = u^*(\mathbf{w}^*.\mathbf{x}_t)$. Let $l(\hat{y}_t, y_t) = (y_t - \hat{y}_t)^2$. Now, since setting is realizable, $\tilde{y}_t = y_t$. Hence the Regret R_T is defined as

$$\begin{aligned} R_T &= \sum_{t=1}^{T} [l(\hat{y}_t, y_t) - l(\tilde{y}_t, y_t)] \\ &= \sum_{t=1}^{T} [l(\hat{y}_t, y_t) - 0] \\ &= \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 \end{aligned}$$

5.2 Expected Regret

Now, since our adversary is stochastic, we'll come up with an expression for a bound on Expected Regret. We use $\epsilon(h)$ as defined in [Kakade et al., 2011],

$$\epsilon(h) = \mathbb{E}_{(x,y)}[(h(x) - u^*(\mathbf{w}^*.x))^2] = \mathbb{E}_{(x,y)}[(h(x) - y)^2]$$

We define $Z^t = \epsilon(h^t) - l(h^t(\mathbf{x}_t), y_t)$. We'll now show that Z^t is an MDS sequence.

$$\mathbb{E}_{(x,y)}[Z^t | \mathcal{H}^t] = \mathbb{E}_{(x,y)}[\epsilon(h^t) - l(h^t(\mathbf{x}_t), y_t) | \mathcal{H}^t]$$

$$\implies \mathbb{E}_{(x,y)}[Z^t | \mathcal{H}^t] = \epsilon(h^t) - \mathbb{E}_{(x,y)}[l(h^t(\mathbf{x}_t), y_t) | \mathcal{H}^t] = 0$$

Hence, Z^t is an MDS sequence. Next we'll come up with an expression of the regret bound. As we stated in Section 2, $h^t(\mathbf{x}^t)$ is bounded, and hence Z^t is also bounded i.e. $|Z^t| \leq B \ \forall t$. So, using Azuma Hoeffding's inequality, with probability $1 - \delta$

$$|\frac{1}{T}[\sum_{t=1}^{T}(\epsilon(h^t) - l(h^t(\mathbf{x}_t), y_t))]| \leq O(B\sqrt{\frac{lg(1/\delta)}{T}})$$

$$\sum_{t=1}^{T} \epsilon(h^t) \leq \sum_{t=1}^{T} l(h^t(\mathbf{x}_t), y_t) + O(B\sqrt{lg(1/\delta)}\sqrt{T}), \quad \text{(w.p. } 1-\delta)$$

$$\Rightarrow R_s \leq \sum_{t=1}^{T} l(h^t(\mathbf{x}_t), y_t) + O(B\sqrt{lg(1/\delta)}\sqrt{T}), \quad \text{(w.p. } 1-\delta)$$

R_s is the Stochastic Regret. We need to compute worst case bound for first term of RHS to get bound on stochastic regret. We can also find a high probability bound on RHS and then take a union bound on the probability. Rest of the sections are dedicated to solve this bound.

6 Bounding the Loss of FTL-SLISOTRON

For bounding the loss of FTL-SLISOTRON, we need to prove two things:

- Given $t-1$ training points, FTL-SLISOTRON indeed learns a perfect 1-Lipschitz function u_t and a corresponding $\mathbf{w}_t$ on these points.
- After fitting on $t-1$ points it predicts a value on $\mathbf{x}_t$, and incurs a loss. Cumulative sum of these losses should be sublinear.

SLISOTRON is able to learn the best predictor on training points with a very high probability. Hence it should be able to to get the exact fit on training points.

Now we'll give a proof for the regret bound, but it'll stricly be valid only for noiseless case i.e. realisable setting and exact fit for $t-1$ points. Our SLISOTRON-FTL is able to learn u^t, and $\mathbf{w}^t$ such that at step t

$$u^t(w^t \cdot \mathbf{x}_i) = y_i, \forall i = 1, 2, ..., t-1$$

Assume $\|\mathbf{x}\| \leq \mathbf{X}$, $\|\mathbf{w}\| \leq \mathbf{W}$

At iteration t, let $\mathbf{x}_i = argmin_{\mathbf{x}_j} \|\mathbf{x}_t - \mathbf{x}_j\|$ where $j = 1, 2, ..., t-1$, then from 1-Lipschitz constraint for u^t and u^*, (Note that LIR step in SLISOTRON ensures that u^t is Lipschitz [Kakade et al., 2011]) we have,

$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^t(\mathbf{w}^t \cdot \mathbf{x}_i)| \leq |\mathbf{w}^t \cdot \mathbf{x}_t - \mathbf{w}^t \cdot \mathbf{x}_i|$$
$$|u^*(\mathbf{w}^* \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_i)| \leq |\mathbf{w}^* \cdot \mathbf{x}_t - \mathbf{w}^* \cdot \mathbf{x}_i|$$

And since, $|a| \leq |c_1|, |b| \leq |c_2| \implies |a-b| \leq |c_1| + |c_2|$, hence

$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_i)| \leq |\mathbf{w}^t \cdot \mathbf{d}_t| + |\mathbf{w}^* \cdot \mathbf{d}_t|$$
$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_i)| \leq \|\mathbf{w}^t\|\|\mathbf{d}_t\| + \|\mathbf{w}^*\|\|\mathbf{d}_t\| \quad \text{(Cauchy-Schwatz)}$$

$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_i)| \leq 2\|\mathbf{W}\|\|\mathbf{d}_t\| \quad \text{where} \quad \mathbf{d}_t = \mathbf{x}_t - \mathbf{x}_i$$

We have,

$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_t)| \leq 2\|\mathbf{W}\| \cdot \|\mathbf{d}_t\|$$

Now, taking the sum over T time steps, we have,

$$\sum_{t=1}^{T} \left(u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_t) \right)^2 \leq 4\|\mathbf{W}\|^2 \sum_{t=1}^{T} \|\mathbf{d}_t\|^2$$

If we can bound $\sum_{t=1}^{T} \|\mathbf{d}_t\|^2$ then we would be done. We came up with a proof of sublinear regret but it is not yet verified by instructor so we have **attached it separately**. Now, to get an empirical estimate, we randomly generated vectors inside a unit ball of dimension = 2 and plotted $\sum_{\tau=1}^{t} \|\mathbf{d}_\tau\|^2$ (the upperbound on error) against t (Fig. 6). It comes out to be sublinear in T. However, the proof breaks for the noisy case. For i.i.d. noise with variance σ^2, this proof will result in giving expected $\mathcal{O}(\sigma^2 T)$ regret bound, since for noisy case we'll have:

$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_i) + \eta_t| \leq |\mathbf{w}^t \cdot \mathbf{d}_t| + |\mathbf{w}^* \cdot \mathbf{d}_t| \quad \eta_t \text{ is i.i.d. noise}$$
$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_i)| \leq \|\mathbf{w}^t\|\|\mathbf{d}_t\| + \|\mathbf{w}^*\|\|\mathbf{d}_t\| + |\eta_t|$$
$$|u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_t)| \leq 2\|\mathbf{W}\| \cdot \|\mathbf{d}_t\| + |\eta_t|$$
$$\sum_{t=1}^{T} \left(u^t(\mathbf{w}^t \cdot \mathbf{x}_t) - u^*(\mathbf{w}^* \cdot \mathbf{x}_t) \right)^2 \leq 4\|\mathbf{W}\|^2 \sum_{t=1}^{T} \|\mathbf{d}_t\|^2 + \sum_{t=1}^{T} |\eta_t^2|$$

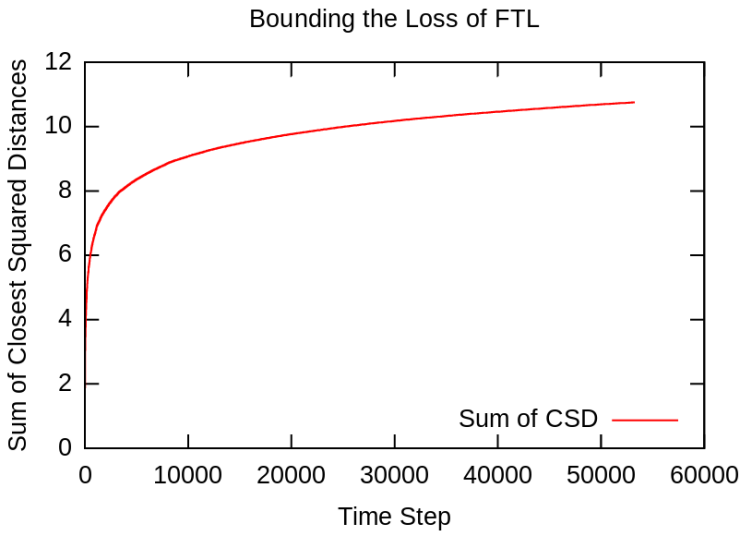

Figure 1: $\sum_{\tau=1}^{t} \|\mathbf{d}_\tau\|^2$ against t

And $\mathbb{E}[\sum_{t=1}^{T} |\eta_t^2|] = \sigma^2 T$ so this will make the expected regret linear in T. Hence this proof doesn't work for noisy case.

7 Parallelograms Based Approach for Bounding the Regret

In this section we try to find the regret bound by visualising the Lipchitz function. We start with the case of simple Online Isotonic Regression via 1-Lipschitz function, and then we attempt to extend it to our Online SIM Learning case.

7.1 Case of Online 'Lipschitz' Istonic Regression

Let's assume a different problem setting, where we have just one dimensional (z_t, y_t) pairs, where $z_t, y_t \in \mathbb{R}$ and $y_t = u(z_t)$ and we want to learn a 1-Lipschitz function u over it. Because u^t is 1-Lipschitz, the scenario is show in the following figure 2.

For the t point received at time step t, the honest adversary must ensure that the it lies within the green region - which is a set of parallelograms. Here, the maximum possible error that can be incurred by the point at t is $\frac{bh}{b+h}$, where b and h are the base and height of the largest parallelogram.

Now, the point may either go inside the largest parallelogram or some other smaller parallelogram. If the adversary puts the point in the largest parallelogram, the worst case error for the next $t + 1^{th}$ point will decrease. Otherwise, the worst case error won't decrease but the probability of the next point being put in the largest parallelogram will increase.

We claim that the worst case loss for each subsequent point must decrease with high probability in such a setting. We show a plot that empirically supports this claim. We have plotted the cumulative regret (in realizable setting it is same as loss) of Online LIR against time steps. It can be observed from the plot that the rate of increase of the cumulative regret drops rapidly. [See Figure 3]

7.2 Extension to FTL-SLISOTRON: An Intuition

Now we'll try to extend this parallelogram-based visualisation into our FTL-SLISOTRON. We make an additional assumption that u^t is not just 1-Lipschitz, but also γ-inverse Lipschitz, where $\gamma \in (0, 1)$. This makes u^t and u strictly increasing function. For time step t, LIR learns a u^t such that $u^t(\mathbf{w}^t \cdot \mathbf{x}^\tau) = y^\tau$ for $\tau = 1, 2,...$ Let $u(z) = y$ then all these points can be represented on z axis.

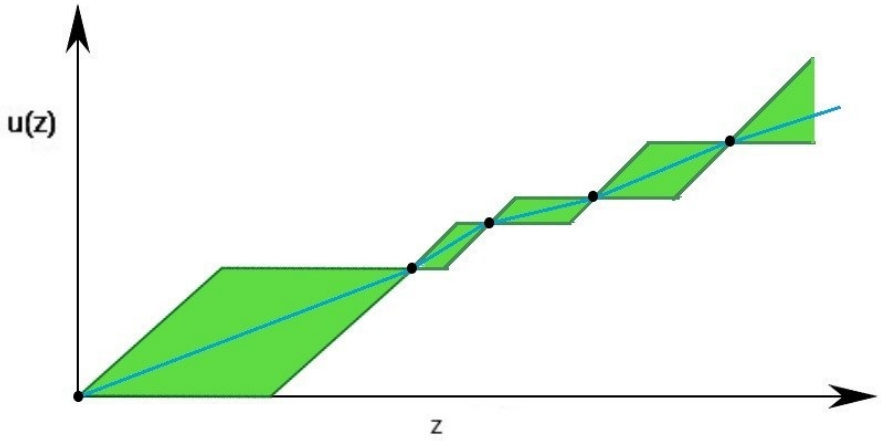

Figure 2: Online Lipschitz Isotonic Regression at frozen time step t in FTL

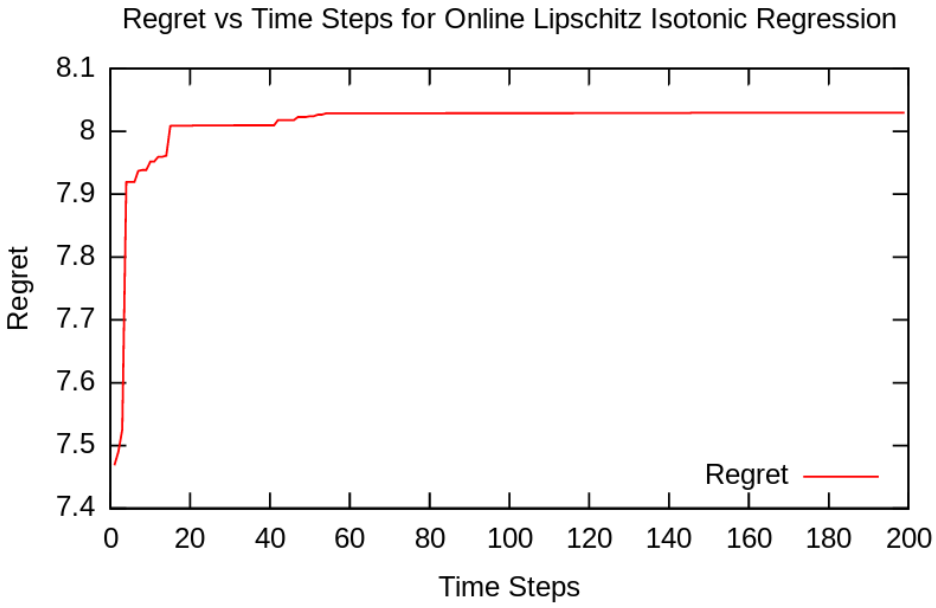

Figure 3: Online Lipschitz Isotonic Regression with w frozen at time step t in FTL

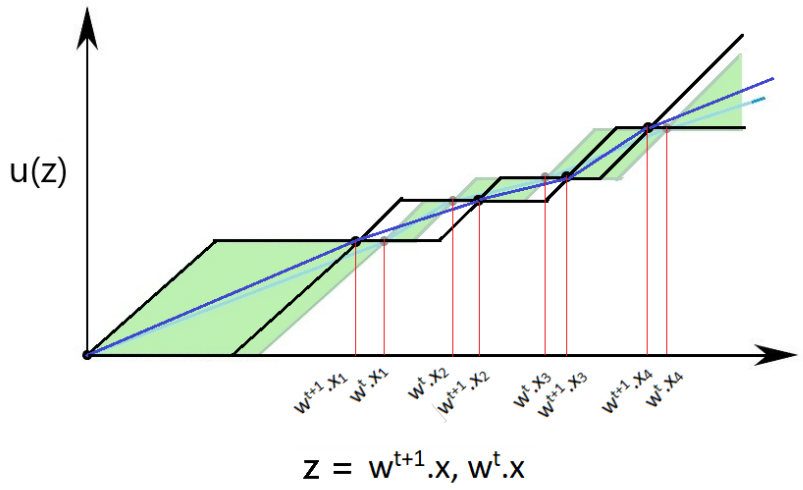

Figure 4: Two parallelograms corresponding to positions of points at time step t (faded) and then the position of same points at $t + 1$.

Now as we get the new $\mathbf{w}^{t+1}$, the individual points ($\mathbf{w}^t \cdot \mathbf{x}^\tau$) can translate independently on z axis, because $\mathbf{w}^{t+1}$ is different from $\mathbf{w}^t$ (Figure 4). However, the relative orders of t points can't change (since u_t is a strictly increasing function). So, as no. of points increase, the z axis will become more populous, and hence update in $\mathbf{w}$ will become more restictive. This implies that updation of $\mathbf{w}$ will be stable. So after a few time steps the $\mathbf{w}_t$ won't change much so we can use the analysis of Online 'Lipschitz' Istonic Regression here. Note however, that FTL-SLISOTRON doesn't guarantee a strictly increasing u_t so it has to be modified slightly.

8 Conclusion

In this project we came up with an interesting problem of Online SIM Learning, and utilised SLISOTRON to make a simple FTL based online algorithm SLISOTRON-FTL. Then we derived the expression for Expected Regret bound (with high probability) in stochastic setting, using Azuma-Hoeffding's Inequality. Then we tried to come up with a proof for the realisable, noiseless setting. We aren't sure about the correctness of the proof but we have verified some claims experimentally. We also looked at the problem of Online Isotonic Regression for 1-Lipschitz function, and how it extends into our problem of Online SIM Learning, provided u and all learnt u^t are inverse-Lipschitz as well.

On the closing note, we would say that Online SIM Learning is a rather unexplored setting despite being so important. Also, if one is trying to extend our work, we would recommend to implement the full SLISOTRON-FTL algorithm and do experimental verifications on it first, which we weren't able to do due to time constraints.

References

[Kakade et al., 2011] Kakade, S. M., Kalai, A. T., Kanade, V., and Shamir, O. (2011). Efficient learning of generalized linear and single index models with isotonic regression. *Advances in Neural Information Processing Systems*.

[Kalai and Sastry, 2009] Kalai, A. T. and Sastry, R. (2009). The isotron algorithm: High-dimensional isotonic regression. *COLT*.

[Kotłowski et al., 2016] Kotłowski, W., Koolen, W. M., and Malek, A. (2016). Online isotonic regression. *29th Annual Conference on Learning Theory*.