Algoritmos e Incerteza **(PUC-Rio INF2979, 2017.1)**

Lecture 8: Online Gradient Descent, Online SVM and Regression

2 October 2018

Lecturer: Marco Molinaro *Scribe: Mauricio Carvalho*

In this lecture, we are going to recap the Online Convex Optimization and present applications of the algorithm. In the first four sections we did a recap of last lecture and on section 5 we showed some applications of the OCO. First we present Online Regression and later an algorithm for Online SVM.

1 Loss Functions

First we have points inside a set of feasible moves, then we have loss functions that will be evaluated using these points $f_1, ..., f_{y-1} \rightsquigarrow p_t \in \mathcal{P}$. The loss in an instant of time t when a point p_t is played is $f_t(p_t)$. Our objective is to minimize the sum of losses:

$$\min \sum_{t=1}^{T} f_t(p_t)$$

The question is how can we produce these points over the time so that the sum of the losses is minimized.

2 Follow the Regularized Leader

The algorithm Follow the Regularized Leader produces these points in a greedy regularized way. This algorithm produces a point that is a minimizer of the losses calculated until time t, but it regularizes the options according to a function R.

$$p_t \in \underset{p \in \mathcal{P}}{Argmin}\{f(p) + ... + f_{t-1}(p) + R(p)\}$$

Basically to decide the next play the algorithm looks for all the loss functions applied previously and chooses the move that goes better on all the accumulated loss functions plus a regularization function.

3 Guarantee for FTRL

The algorithm FTRL has a good guarantee. The maximum loss for this algorithm is at most the minimum total loss of a fixed move plus $O(\sqrt{T}...)$:

$$\sum_{t=1}^{T} f_t(p_t) \leq \min_{p \in \mathcal{P}} \sum_{t} f_t(p) + O(\sqrt{T...})$$

Our loss above the optimal strategy grows in a square rooted way and not linearly, with respect to T. So our average loss above the optimal solution is $\frac{1}{\sqrt{T}}$ and when time grows it tends to zero.

Further reading. In [1] the authors demonstrate that the best guarantee for this type of problem is equivalent to the best probabilistic inequality of Martingales.

4 Main idea of the guarantee: Stability (avoid overfit)

Stability is crucial in learning algorithms. For example, three learning algorithms where stability is crucial:

1- OCO

2- Optimal algorithm for load balancing is a greedy "stabilized" algorithm (log m).

3- PAC learning model: learning is equivalent to stability, demonstrated in [2].

4.1 Gradient Descent Algorithm

To implement the FTRL algorithm we need a solver for convex functions.

Q: How to minimize convex functions?

A: You can follow the gradient of the function, moving to the opposite direction. That strategy is called the gradient descent algorithm. We need to be careful with the stability, so we need to use small steps of size δ to be sure that we are converging to the minimum of the function. You can see an example of that in Figure 1.

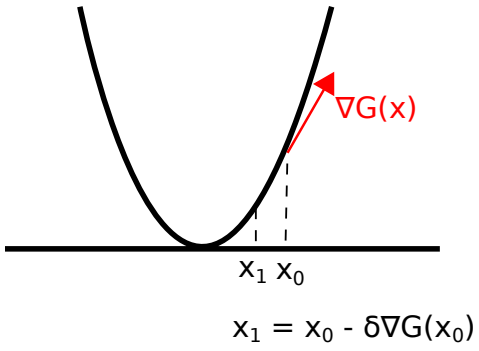

Figure 1: Example of gradient descent.

What if we have restrictions on the points available to be used in the optimization problem? After a step if the point acquired is out of the feasible options we need to find the nearest point in the feasible set. For that we make a projection of the resulting point on the feasible set. Projections will be made minimizing the Euclidian distance. You can see an example of a projection in Figure 2. In this figure, $x_1 = \underset{\mathcal{P}}{Proj}\tilde{x_1}$ (Projection in the feasible set) $\in \underset{x \in \mathcal{P}}{Argmin}\{||x - \tilde{x_1}||\}$. With this we have another optimization problem to solve.

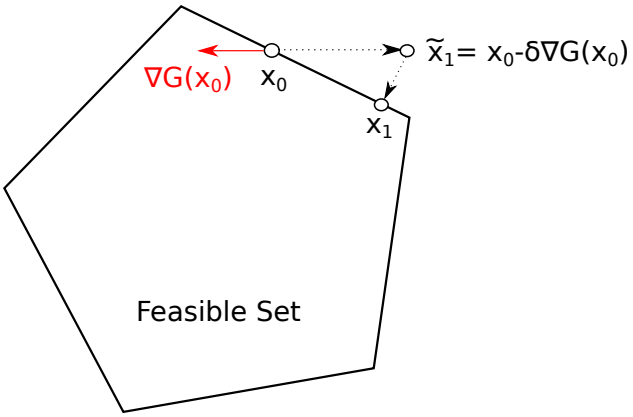

Figure 2: Projection on the feasible set after a step of the OGD.

To solve this optimization problem, of finding the minimal projection of a point, we have some options:

1- If the feasible set $\mathcal{P}$ is simple, we can solve the projections using brute force.

2- ...(another technique not mentioned in the course)

In conclusion, we can implement the FTRL algorithm using a solver (gradient descent) to find a move p_t, but that is an expensive solution. We would like to find an algorithm where in each step you wouldn't have to solve an optimization problem on each iteration to find the best move.

Idea: Run one step of the gradient descent algorithm to each instant of time T.

4.2 Online Gradient Descent

The first step of the algorithm Online Gradient Descent is to find the best move p_t inside the feasible set. For that we need to find $\tilde{p_t}$ using the formula:

$$\tilde{p_t} = p_{t-1} - \delta \nabla f_{t-1}(p_{t-1})$$

The point p_{t-1} already has the weight of all previous gradients recursively. We still need to make small steps to guarantee stability. With the point $\tilde{p_t}$ at hands we can find the the point p_t making a projection.

$$p_t = \underset{\mathcal{P}}{Proj}\tilde{p_t} \text{ (Projection in the feasible set)}$$

What if the functions f_t are too different?

This algorithm looks bad for functions that are too different, look at the Figure 3. The algorithm seems to be unstable in those cases and it can take steps that actually make the total loss worse instead of better. How can we achieve a "good" regret, even in this type of instance?

- OPT is fixed, if the functions are too different, OPT also goes badly, so comparing to OPT it is going well.

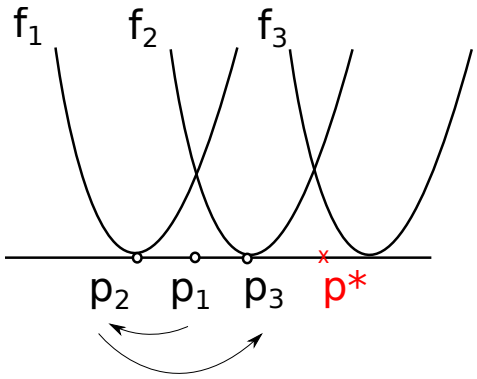

Figure 3: Example of OGD in functions too different.

For that reason it is important to analyse the algorithm using the reference of a fixed OPT. Otherwise it would not be possible to compete with a point that can move accordingly with the functions.

Theorem 4.1. *Online Gradient Descent has regret*$\leq O(\sqrt{T}DG)$*, where D is the diameter of* $\mathcal{P}$*(feasible set):* $\max_{x,y\in\mathcal{P}} ||x-y||^2$*, and G is the biggest gradient of* f_t*.*

We will not prove this theorem on this class.

5 Applications

5.1 Online Regression

In this problem we have the following steps:

- On day t, predict $v_t \in [-m, m]$, having access to extra informations $y^t \in \mathbb{R}^d$ about the time t.
- Hypothesis: there is an $\alpha \in \mathbb{R}^d$ with norm $||\alpha|| = 1$, such that $v_t \cong < y^t, \alpha > = \sum_i y_i^t \alpha_i$. Basicallly there exists a vector that the internal product with the extra information makes the best prediction. But we don't know α.
- If you predict $\tilde{v_t}$, the loss is defined by $(\tilde{v_t} - v_t)^2$

Question: How can we minimize the total loss? We have a hypothesis that a good regression exists but we do not have any information about the distribution of the data.

Approach 1:

- $\mathcal{P} = [-m, m]$ (set of feasible v_t)

- $f_t(v) = (v - v_t)^2$ (loss function defined by the quadratic loss)
- run FTRL.

That approach does not use the extra informations.

Approach2: We need to focus on learning the correct α, instead of the $\tilde{v_t}$.

- $\mathcal{P}$ = Euclidian ball of size 1 in $\mathbb{R}^d$
- Run FTRL to produce $\alpha^t \in \mathcal{P}$
- Prediction: $\tilde{v_t} =< y^t, \alpha^t >^{\text{internal product}}$
- $f_t(\alpha) = (< y^t, \alpha > -v_t)^2$

This approach uses the extra information to make the prediction. Using Online Gradient Descent instead of FTRL, what is our guarantee?

Guarantee. Using the OGD guarantee we have:

$$\sum_t f_t(\alpha_t) \leq \min_{\alpha \in \mathcal{P}} \sum_t f_t(\alpha) + O(\sqrt{T}DG)$$

then, adapting to our approach 2

$$\sum_t (\tilde{v_t} - v_t)^2 \leq \min_{\alpha \in \mathcal{P}} \sum_t (< y^t, \alpha > +v_t)^2 + O(\sqrt{T}DG).$$

Finally, we have by assumption,

$$\sum_t (< y^t, \alpha > +v_t)^2 \approx 0.$$

So

$$\sum_t (\tilde{v_t} - v_t)^2 \lesssim O(\sqrt{T}DG).$$

Obs: Even if the hypothesis is not true, the algorithm works and the guarantee is similar: loss $\leq$ loss of the best regression + $O(\sqrt{T}DG)$

5.2 Online SVM

Each data has a set of features and a label, the objective is to make a linear classification of new data like in Figure 4.

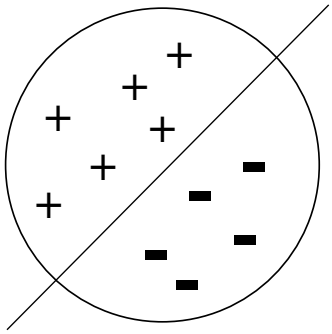

Figure 4: Linear classification.

- Data $(y^t, z_t), y^t \in \mathbb{R}^d$ and $z_t \in \{-1, 1\}$
- Linear classification:

$$< y^t, p >= \begin{cases} > 0, \text{ if } z_t = +1 \\ < 0, \text{ if } z_t = -1 \end{cases} \tag{1}$$

If there is a linear classification with a margin of 1 we have:

$$< y^t, p >= \begin{cases} \geq 1, \text{ if } z_t = +1 \\ \leq -1, \text{ if } z_t = -1 \end{cases} \quad \text{(for p with } ||p|| = 1)$$

This is SVM with hard-margin, as you can see in Figure 5.

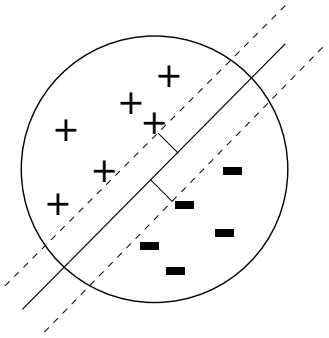

Figure 5: SVM with hard margin.

Problem: What if the data is not linearly separable?

SVM Soft-Margin This is a solution when the problem is not exactly linear separable but is approximately linear separable.

- Try to minimize the sum of the distances of the wrongly classified data to the hyperplane, like in Figure 6.
- Loss in the data t: $\max\{0, 1- < y^t, p > z_t\}$, this is called hinge loss.

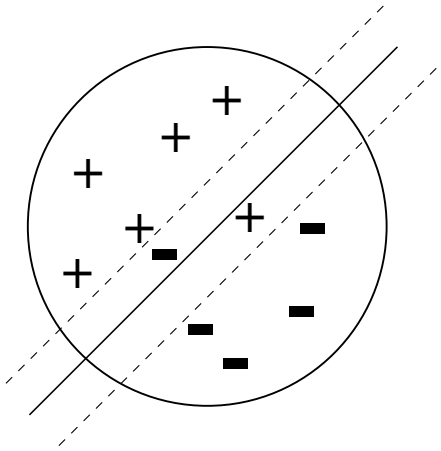

Figure 6: SVM Soft Margin.

Notice that if we have a linear classification with margin 1, then there is no loss:

$$< y^t, p >= \begin{cases} \geq 1 \text{ and } z_t = +1 \Rightarrow \leq 0 (loss = 0) \\ \leq -1 \text{ and } z_t = -1 \Rightarrow \leq 0 (loss = 0) \end{cases}$$

SVM finds p that minimizes the hinge loss in all data.

Online SVM

- Time t has info $(y^1, z_1), ..., (y^{t-1}, z_{t-1}), y^t$ (have all the past data but on time t it receives only the features of the t-th data without the classification).
- Produces p^t and uses it to classify, using (1).

Objective: minimize hinge loss.

Using OCO How can we model this game using OCO?

- $\mathcal{P} =$ euclidian ball in $\mathbb{R}^d$, decides p^t
- $f_t(p)$ = hinge loss (convex)
- Uses OGD to compute $p^1, p^2, ..., p^T$
- Total loss = $\sum_t f_t(p_t) \leq \min_{p \in \mathcal{P}} \sum_t f_t(p) + O(\sqrt{T}DG)$, where $f_t(p)$ is the best offline soft-SVM classification.

Obs: Without the hypothesis of convexity, each expert could have arbitrary loss [0,1]. The regret is $O(\sqrt{T \log m})$.

OCO: hypothesis of convexity. What is the gain of the aditional hypothesis? The algorithm can work with an infinite set $\mathcal{P}$ of hypothesis.

Can we solve the OCO problem using experts? Yes, discretizing the feasible set $\mathcal{P}$, like in Figure 7.

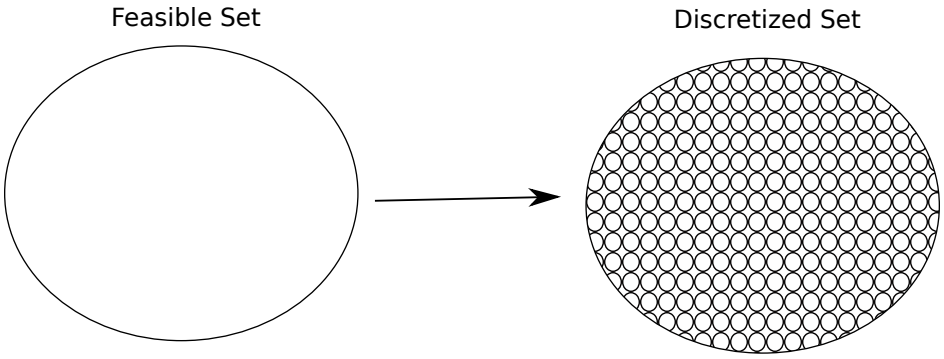

Figure 7: Discretization.

References

[1] A. Rakhlin and K. Sridharan. On equivalence of martingale tail bounds and deterministic regret inequalities. *Proceedings of Machine Learning Research*, 65:1–19, 2017.

[2] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research 11*, pages 2635–2670, 2010.