CS 7545: Machine Learning Theory **Fall 2018**

Lecture 11: Online Convex Optimization

Lecturer: Jacob Abernethy *Scribes: Chen Liang and Jiannan Cui*

11.1 Online Gradient Descent

Generalized Experts Setting The process is shown as the following:

Algorithm 1 Generalized Experts

Let $K \subseteq \mathbb{R}^d$ convex and compact.
for $t = 1...T$ **do**
  Algorithm selects $x_t \in K$
  Nature selects loss convex function $f_t : K \to \mathbb{R}$
end for
$\text{Regret}_T := \sum_{t=1}^T f_t(x_t) - \min_{x\in K} \sum_{t=1}^T f_t(x)$

Online Gradient Descent The initialization and update rule are shown as the following

$$x_0 = \text{arbitrary point in } K$$

$$x_{t+1} = \text{proj}_K(x_t - \eta \nabla f_t(x_t))$$

Theorem 11.1 *Let $\nabla_t = \nabla f_t(x_t)$. Assume $\|\nabla_t\|_2 \le G$ where G is some constant, and $\|x_0 - x^*\|_2 \le D$ for any $x^* \in K$ where D is some constant. Then:*

$$R_T(GD) \le GD\sqrt{T}$$

$$\max_{x\in K} ||x_0 - x|| = D \quad \textit{where } D \textit{ is the diameter of the set } K$$

11.2 More Algorithms of Online Convex Optimization

Follow The Leader Consider the update rule

$$x_{t+1} = \arg\min_{x\in K} \sum_{s=1}^{t} f_s(x)$$

Follow The Regularized Leader Consider the update rule

$$x_{t+1} = \arg\min_{x\in K} \eta \sum_{s=1}^{t} f_s(x) + R(x)$$

$R(x)$ is some convex regularizer. Follow The Regularized Leader Algorithm is a generalization of Exponential Weight Algorithm.

Online Mirror Descent Consider the update rule

$$x_{t+1} = \arg\min_{x \in K} \eta < \nabla f_t(x_t), x > + D_R(x, x_t)$$

$D_R(x, x_t)$ is Bregman Divergence. If $D_R(x, x_t) = \frac{1}{2}||x - x_t||^2$, then Online Mirror Descent is a special case of Online Gradient Descent. Alternatively,

$$y_{t+1} = \nabla R^*(\nabla R(x_t) - \eta \nabla f_t(x_t))$$

$$x_{t+1} = \arg\min_{x \in K} D_R(x, y_{t+1})$$

which shows that the gradient of the current x_t regularizer is mapped to dual space and updated, and then mapped back from dual space.

11.3 Example Applications of Online Gradient Descent

Online Linear Regression Consider the process

Algorithm 2 Online Linear Regression

for $t = 1...T$ **do**
  Algorithm selects $\vec{\theta_t} \in \mathcal{R}^d$
  Nature selects $(\vec{x_t}, y_t) \in \mathcal{R}^d \times \mathcal{R}$
  $f_t(\vec{\theta_t}) = \frac{1}{2}(\vec{\theta_t} \cdot \vec{x_t} - y_t)^2$
end for

Online Density Estimation Let $\{P_{\vec{\theta}} : \vec{\theta} \in \Theta \subseteq \mathcal{R}^d\}$ where Θ is convex, and $P_{\vec{\theta}}$ represents a exponential family distribution. Then consider the process:

Algorithm 3 Online Density Estimation

for $t = 1...T$ **do**
  Algorithm selects $\vec{\theta_t} \in \Theta$
  Nature selects $\vec{x_t} \in \mathcal{X}$
  $f_t(\vec{\theta_t}) = -logP_{\vec{\theta_t}}(\vec{x_t})$
end for

Note that a family of distributions is said to belong to a vector exponential family if the probability density function (or probability mass function, for discrete distributions) can be written as

$$P_{\vec{\theta}}(\vec{x}) = exp(\vec{\theta}^\top \phi(\vec{x}) - A(\vec{\theta}))$$

$$\text{where } A(\vec{\theta}) = log \int exp(\vec{\theta}^\top \phi(\vec{x}))dx$$

$$\nabla A(\vec{\theta}) = \mathbb{E}_{\vec{x} \sim P_{\vec{\theta}}}[\phi(\vec{x})]$$

$\vec{\theta}$ are parameters, $\phi(\vec{x})$ can be viewed as feature representations and $A(\vec{\theta})$ can be viewed as a normalizer. For example, if we set $P_{\vec{\theta}}(\vec{x})$ as a standard Gaussian, then $P_{\vec{\theta_t}}(\vec{x_t}) = \frac{exp(-\frac{1}{2}(\vec{\theta_t} - \vec{x_t})^2)}{z_\theta}$ and $f_t(\vec{\theta_t}) = \frac{1}{2}||\vec{\theta_t} - \vec{x_t}||^2$.

Online Portfolio Selection Assume there are N stocks. The prices fluctuate from day to day. Let's define

$$\vec{r_t}(i) = \frac{Price_{(t)}(Stock_i)}{Price_{(t-1)}(Stock_i)}$$

where $Price_{(t)}(Stock_i)$ denotes the price of stock i at day t

Algorithm 4 Online Portfolio Selection

for $t = 1...T$ **do**
  Algorithm distributes wealth according to $\vec{w_t} \in \Delta_N$
  Nature updates price arbitrarily
  $f_t(\vec{w_t}) = -log \sum_{i=1}^{N} \vec{r_t}(i)\vec{w_t}(i)$
end for
Regret$_T$ $:= \sum_{t=1}^{T} -log(\vec{r_t} \cdot \vec{w_t}) - \min_{\vec{w} \in \Delta_N} \sum_{t=1}^{T} -log(\vec{r_t} \cdot \vec{w}) = \max_{\vec{w} \in \Delta_N} log \frac{\prod_{t=1}^{T} \vec{w} \cdot \vec{r_t}}{\prod_{t=1}^{T} \vec{w_t} \cdot \vec{r_t}}$

we have a constant log optimal strategy $\vec{w}^*$ at each time. This is called a constant rebalanced portfolio (CRP). This means we must rebalance our investment after the stocks have grown at non-uniform rates to yield a different balance than $\vec{w}^*$.

11.4 Convex Optimization to Online Convex Optimization

We want to solve the following convex optimization problem,

$$min_{x \in K} \ f(x)$$

where f and K is convex. And there is some given no-regret OGD Algorithm: for $t = 1, ..., T$, the algorithm plays x_t; nature plays $f_t(x) = f(x)$; output is $\bar{x}_T = \frac{1}{T} \sum_{t=1}^{T} x_t$.
Claim:

$$f(\bar{x}_T) - min_{x \in K} f(x) \leq \frac{Regret_T}{T}$$

Proof:

$$f(\bar{x}_T) \leq \frac{1}{T} \sum_{t=1}^{T} f(x_t) = \frac{1}{T} \sum_{t=1}^{T} f_t(x_t) = \frac{1}{T} min_{x \in K} \sum_{t=1}^{T} f_t(x) + \frac{Regret_T}{T}$$

$$= min_{x \in K} \frac{1}{T} \sum_{t=1}^{T} f(x) + \frac{Regret_T}{T} = min_{x \in K} f(x) + \frac{Regret_T}{T}$$

Fact: If function f is smooth, GD achieves $O(\frac{1}{T})$ and AGD (Accelerated Gradient Descent) achieves $O\frac{1}{T^2}$. AGD is equivalent to using two regret-min algorithms on the following game:

$$g(x, y) = f^*(y) - x^T y$$

y-player: optimise FTL; x-player: GD.

11.5 Online to Batch Conversion

Online to batch conversion can reduce learning in "stochastic setting" to OCO.
Given data $\vec{x}$ and label y, i.i.d., $(\vec{x}_1, y_1), ..., (\vec{x}_T, y_T) \sim D \in \Delta(\vec{x}, y)$; $\mathcal{H} := \{h_{\vec{\theta}} : \vec{\theta} \in \Theta\}$, where Θ is convex and bounded; loss $\ell(h_{\vec{\theta}}, (\vec{x}, y))$ is convex in $\vec{\theta}$ (e.g. $(\vec{\theta} \cdot \vec{x} - y)^2$). Risk of $\vec{\theta}$ is $\mathscr{L}(\vec{\theta}) =$

$\mathbb{E}_{(\vec{x},y)\sim D}[\ell(h_{\vec{\theta}}, (\vec{x}, y))]$.
We want to find $\hat{\vec{\theta}}$ from T data points, such that $\mathscr{L}(\hat{\vec{\theta}}) - min_{\vec{\theta}^* \in \Theta}\mathscr{L}(\vec{\theta}^*) \leq \epsilon$.
Proposal:

Algorithm 5 Online to Batch Conversion

Input: $(\vec{x}_t, y_t) \sim D\ for\ all\ t = \{1, ..., T\}$**.**
Output: $\hat{\vec{\theta}} = \frac{1}{T}\sum_{t=1}^{T}\vec{\theta}_t$
  for $t = 1...T$ **do**
    Algorithm chooses $\vec{\theta}_t \in \Theta$**;**
    Algorithm observes $(\vec{x}_t, y_t)$**;**
    Algorithm experiences loss given by convex loss function $f_t(\vec{\theta}_t) := \ell(h_{\vec{\theta}_t}, (\vec{x}_t, y_t))$**;**
    Update by using OCO $\vec{\theta}_{t+1} = Projection_{\Theta}(\vec{\theta}_t - \eta\nabla f_t(\vec{\theta}_t))$**;**
  end for

Claim: The online to batch conversion guarantees

$$\mathscr{L}(\hat{\vec{\theta}}) - \mathscr{L}(\vec{\theta}^*) \leq \frac{\mathbb{E}[Regret_T]}{T}$$