CS/CNS/EE 253: Advanced Topics in Machine Learning
Topic: Online Gradient Descent **Lecturer:** Daniel Golovin
Scribe: Esther Wang **Date:** Jan. 11, 2010

3.1 Online Convex Programming

Definition 3.1.1 (Convex Set) *A set of vectors $X \subseteq \mathbb{R}^n$ is* **convex** *if for all $x, y \in X$, and all $\lambda \in [0,1]$, $\lambda x + (1-\lambda)y \in X$.*

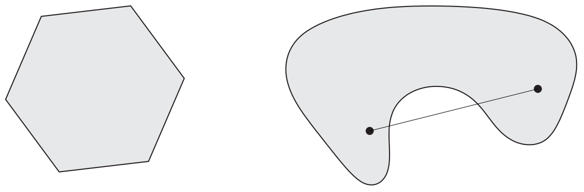

Figure 3.1.1: The hexagon, which includes its boundary is convex. The kidney shaped is not convex because the line segment between two points in the set is not contained in the set [1].

Definition 3.1.2 (Convex function) *For a convex set $\mathcal{X}$, a function $f : \mathcal{X} \to \mathbb{R}$ is* **convex** *if for all $x, y \in \mathcal{X}$, for all $\lambda \in [0,1]$,*

$$\lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y)$$

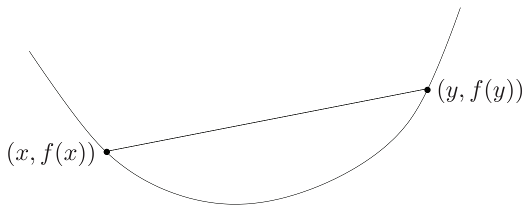

Figure 3.1.2: Graph of a convex function. The segment between any two points on the graph lies above the graph [1].

Definition 3.1.3 (Convex programming problem) *A* **convex programming problem** *consists of a convex feasible set $\mathcal{X}$ and a convex cost function $c : \mathcal{X} \to \mathbb{R}$. The* **optimal solution** *is the solution that minimizes the cost.*

Definition 3.1.4 *An* **online convex programming problem** *consists of a feasible set $\mathcal{X} \subseteq \mathbb{R}^n$ and an infinite sequence $\{c^1, c^2, \dots\}$ where each $c^t : \mathcal{X} \to \mathbb{R}$ is a convex function.*

At each time step t, an **online convex programming algorithm** *selects a vector $x^t \in \mathcal{X}$. After the vector is selected, it receives the cost function c^t.*

Note: $\|x\| = \sqrt{x \cdot x}$

All above definitions were taken from [2].

We have a convex set of experts $\mathcal{X} \subseteq \mathbb{R}^n$ and convex cost functions $c^1, c^2, \ldots, c^T : \mathcal{X} \to [0,1]$.

In online convex programming, the algorithm faces a sequence of convex programming problems with the same feasible set but different cost functions. At each time step the algorithm chooses a point before it observes the cost function.

Since the cost functions can be anything, instead of attempting to choose a point x^i that minimizes the cost function, we try to minimize regret. Regret is calculated by comparing our algorithm to OPT, the optimal feasible fixed point, or equivalently by comparing ourselves against an algorithm that knows all the cost functions in advance but must play the same vector on all rounds [2].

Definition 3.1.5 (Regret) *Given an algorithm and a convex programming problem $(\mathcal{X}, \{c^1, c^2, c^3, \ldots\})$, if $\{x^1, x^2, x^3, \ldots\}$ are the vectors selected by the algorithm, then the regret of the algorithm until time T (i.e. T number of rounds) is*

$$R(T) := \sum_{t=1}^{T} c^t(x^t) - \mathsf{OPT}(T) \tag{3.1.1}$$

where $\mathsf{OPT}(T)$ is the cost of the "static optimum" for the first T rounds, namely

$$\mathsf{OPT}(T) \equiv \min_{x \in \mathcal{X}} \sum_{t=1}^{T} c^t(x)$$

and the average regret is

$$\bar{R}(T) = \frac{R(T)}{T}$$

The first suggested algorithm in class was simply to apply gradient descent:

```
1  for $t = 1$ to $T$
2      $x^{t+1} = x^t - \eta_t \nabla c^t(x^t)$
```

However, there are a few problems with this proposed algorithm. The class suggested the following:

1. An adversary could choose convex functions such that we get stuck at local minima.
2. The gradient of the cost function may not exist.
3. Most importantly, at each time step, x^{t+1} is not necessarily in $\mathcal{X}$.

The first "problem" cannot arise because we are dealing with convex functions over a convex set, and the sum of convex functions is convex. Thus any local minima are automatically global minima.

We defer the second problem for now, by assuming the cost functions are differentiable everywhere. In the next lecture we'll talk about using *subgradients* to remove this assumption.

To deal with the third problem, we modify the proposed gradient descent algorithm so that x^{t+1} is projected back into $\mathcal{X}$ at each time step.

Algorithm 3.1.6 (Greedy Projection [2]) *Choose an arbitrary $x^1 \in \mathcal{X}$ and a sequence of learning rates $\eta_1, \eta_2, \dots \in \mathbb{R}^+$. At time step t, after acquiring a cost function, choose the next vector x^{t+1} according to:*

$$x^{t+1} = \mathrm{Proj}_{\mathcal{X}}(x^t - \eta_t \nabla c^t(x^t))$$

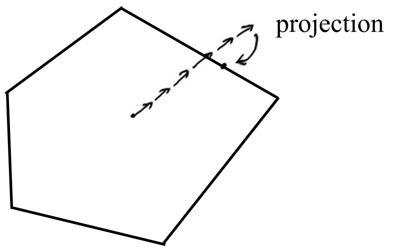

Figure 3.1.3: Projection is defined as $\mathrm{Proj}_{\mathcal{X}}(y) = \mathrm{argmin}_{x\in\mathcal{X}} \|x - y\|$, the closest point to y in $\mathcal{X}$. If there are several such points, any one may be chosen.

The goal is to prove that the average regret of the Greedy Projection approaches zero.

Theorem 3.1.7 *If $\eta_t = \frac{1}{\sqrt{t}}$, then the regret of the Greedy Projection algorithm is*

$$R(T) \leq \frac{D^2\sqrt{T}}{2} + G^2\left(\sqrt{T} - \frac{1}{2}\right)$$

where $D = \max_{x,y\in\mathcal{X}} \|x - y\|$ is the diameter of $\mathcal{X}$ and $G \equiv \max_{x\in\mathcal{X}, 1\leq t\leq T} \|\nabla c^t(x)\|$.

Proof: (Please refer to [2] for details)

1. Without loss of generality, $c^1, \dots, c^T$ are linear. To prove this, note we can replace $c^t(x)$ with linear function $g^t(x) = c^t(x^t) + \nabla c^t(x^t)\cdot(x - x^t)$. This can only increase our regret (or leave it the same) in each round, since $g^t(x)$ is a linear lower bound for $c^t(x)$ and $g^t(x^t) = c^t(x^t)$. That is, $g^t(x) \leq c^t(x)$ for all x, and in particular, $g^t(x^*) \leq c^t(x^*)$, where $x^* = \mathrm{argmin} \sum_{t=1}^T c^t(x)$ is a static optimum. Thus $\sum_{t=1}^T \left(c^t(x^t) - c^t(x^*)\right) \leq \sum_{t=1}^T \left(g^t(x^t) - g^t(x^*)\right)$, and it suffices to bound the RHS.

2. $\Phi(\mathrm{Proj}_{\mathcal{X}}(y)) \leq \Phi(y)$ if $y \in \mathcal{X} \Rightarrow \mathrm{Proj}_{\mathcal{X}}(y) = y$, else $\mathrm{Proj}_{\mathcal{X}}(y)$ gets us closer to the optimal point. See figure 3.1.5. We omit the proof of this geometric fact.

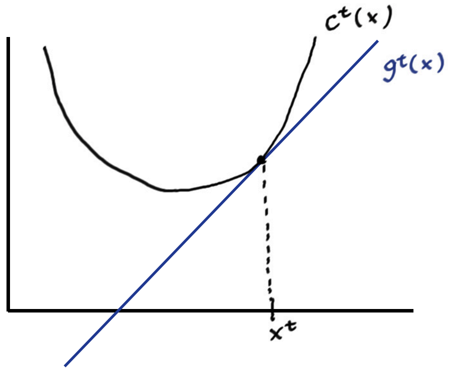

Figure 3.1.4: Replace $c^t(x)$ with linear function $g^t(x) = c^t(x^t) + \nabla c^t(x^t) \cdot (x - x^t)$.

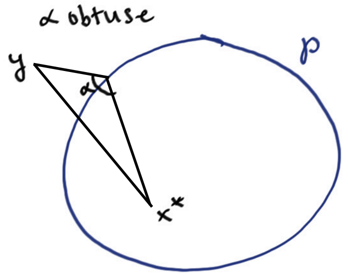

Figure 3.1.5: $\mathrm{Proj}_{\mathcal{X}}(y)$ gets us closer to the optimal point x^*.

Define $y^t \equiv x^t - \eta_t \nabla c^t(x^t)$.

$$\begin{aligned}
\Phi(x^{t+1}) - \Phi(x^t) &\leq \Phi(y^t) - \Phi(x^t) \\
&\leq \|x^t - \eta_t \nabla c^t(x^t) - x^*\|^2 - \|x^t - x^*\|^2 \\
&= \|(x^t - x^*) - \eta_t \nabla c^t(x^t)\|^2 - \|x^t - x^*\|^2 \\
&= -2\eta_t \nabla c^t(x^t) \cdot (x^t - x^*) + \eta_t^2 \cdot \|\nabla c^t(x^t)\|^2 \\
&= -2\eta_t r^t + \eta_t^2 \cdot \|\nabla c^t(x^t)\|^2
\end{aligned}$$

In the last line we substituted the expression for regret in round t:

$r^t = g^t(x^t) - g^t(x^*) = -\nabla c^t(x^t) \cdot (x^* - x^t) = \nabla c^t(x^t) \cdot (x^t - x^*)$

Rearranging the equation, we get the expression for the upper bound of regret at round t:

$$\begin{aligned} r^t &\leq \frac{1}{2\eta_t}[\Phi\left(x^t\right) - \Phi\left(x^{t+1}\right)] + \frac{\eta_t}{2}\|\nabla c^t\left(x^t\right)\|^2 \\ &\leq \frac{1}{2\eta_T}[\Phi\left(x^t\right) - \Phi\left(x^{t+1}\right)] + \frac{\eta_t}{2}\|\nabla c^t\left(x^t\right)\|^2 \end{aligned}$$

The last inequality requires η_t to be non-increasing, so that $1/\eta_t \leq 1/\eta_T$. Additionally, if we set $\eta_t = \frac{1}{\sqrt{t}}$ then

$$\begin{aligned} \sum_{t=1}^{T} r^t &\leq \frac{1}{2\eta_T}[\Phi(x^0) - \Phi(x^{T+1})] + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t \\ &= \frac{D^2\sqrt{T}}{2} + G^2\sqrt{T} \\ &= \frac{D^2}{\eta_T} + G^2\sum_{t=1}^{T}\eta_t \end{aligned}$$

Note $1/\eta_T = \sqrt{T}$ and it is not difficult to show that $\sum_{t=1}^{T} 1/\sqrt{t} \leq \sqrt{T} - 1/2$. This completes the proof. ■

It turns out we can prove a more general result, which intuitively states that this algorithm does well against all slowly changing sequences of solutions (and thus does extremely well for slowly changing environments). Specifically, we can bound the regret against any sequence of solutions $z^1, z^2, \ldots, z^T$, that is $\sum_{t=1}^{T}(c^t(x^t) - c^t(z^t))$, by

$$\sum_{t=1}^{T}(c^t(x^t) - c^t(z^t)) \leq \frac{D^2\sqrt{T}}{2} + G^2\sqrt{T} + 2DL(z^1, \ldots, z^T)\sqrt{T}$$

where $L(z^1, \ldots, z^T) = \sum_{t=1}^{T}\|z_{t+1} - z_t\|$. The proof is similar to the one above. See [2] for details.

An example application of the greedy projection algorithm is online SVM.

References

[1] S.P. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004. freely available for download at `http://www.stanford.edu/~boyd/cvxbook/`.

[2] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 928–936, 2003.