Online Convex Optimization

Gautam Goel, Milan Cvitkovic, and Ellen Feldman

CS 159

4/5/2016

The General Setting

Online Learning

for $t = 1, 2, \ldots$

receive question $\mathbf{x}_t \in \mathcal{X}$

predict $p_t \in D$

receive true answer $y_t \in \mathcal{Y}$

suffer loss $l(p_t, y_t)$

The General Setting

Online Learning

for $t = 1, 2, \ldots$

  receive question $\mathbf{x}_t \in \mathcal{X}$

  predict $p_t \in D$

  receive true answer $y_t \in \mathcal{Y}$

  suffer loss $l(p_t, y_t)$

(Cover) Given only the above, learning isn't always possible

Some Natural Restrictions

- *Realizability*
  - There is some hypothesis that makes no mistakes
  - Implies simple algs like `Consistent` and `Halving`

Some Natural Restrictions

- *Realizability*
  - There is some hypothesis that makes no mistakes
  - Implies simple algs like `Consistent` and `Halving`
- *Randomization*
  - Our predictions are made via a probability distribution, and the environment does not control the randomness
  - Suggests Expected Regret as a performance metric

Recall:

$$\mathrm{Regret}_T(h^\star) = \sum_{t=1}^{T} l(p_t, y_t) - \sum_{t=1}^{T} l(h^\star(x_t), y_t)$$

$$\mathrm{Regret}_T(\mathcal{H}) = \max_{h^\star \in \mathcal{H}} \mathrm{Regret}_T(h^\star)$$

Some Natural Restrictions

- *Realizability*
  - There is some hypothesis that makes no mistakes
  - Implies simple algs like `Consistent` and `Halving`
- *Randomization*
  - Our predictions are made via a probability distribution, and the environment does not control the randomness
  - Suggests Expected Regret as a performance metric
- *Non-adversarial*
  - Loss functions are sampled iid from some distribution
  - This is stochastic gradient descent

Unifying Framework

`Online Convex Optimization (OCO)`

input: A convex set S

for $t = 1, 2, \ldots$

  predict a vector $\mathbf{w}_t \in S$

  receive a convex loss function $f_t : S \to \mathbb{R}$

  suffer loss $f_t(\mathbf{w}_t)$

Unifying Framework

Online Convex Optimization (OCO)
input: A convex set S
for $t = 1, 2, \ldots$
predict a vector $\mathbf{w}_t \in S$
receive a convex loss function $f_t : S \to \mathbb{R}$
suffer loss $f_t(\mathbf{w}_t)$

- *Randomization* is just OCO with the probability simplex

$$S = \{w \in \mathbb{R}^d : w \geq 0, \|w\| = 1\}$$

and a 'surrogate loss function' f_t that upper-bounds the original loss function.

Unifying Framework

Online Convex Optimization (OCO)

input: A convex set S
for $t = 1, 2, \ldots$
  predict a vector $\mathbf{w}_t \in S$
  receive a convex loss function $f_t : S \to \mathbb{R}$
  suffer loss $f_t(\mathbf{w}_t)$

- *Randomization* is just OCO with the probability simplex

$$S = \{w \in \mathbb{R}^d : w \geq 0, \|w\| = 1\}$$

  and a 'surrogate loss function' f_t that upper-bounds the original loss function.

- *Realizability* additionally assumes $\exists \mathbf{w} \text{ s.t. } f_t(\mathbf{w}) = 0$

How to Learn in OCO

The Naïve Approach

Follow-The-Leader (FTL)

$$\forall t, \quad \mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w}\in S} \sum_{i=1}^{t-1} f_i(\mathbf{w}) \quad \text{(break ties arbitrarily)}$$

- For quadratic optimization problems, in which $f_t(\mathbf{w}) = \frac{1}{2}\|\mathbf{w} - \mathbf{z}_t\|_2^2$, FTL has regret $O(\log T)$.

$$\text{Regret}_T(\mathbf{u}) = \max_{\mathbf{u}} \sum_{t=1}^{T} f_t(\mathbf{w}_t) - \sum_{t=1}^{T} f_t(\mathbf{u})$$

The Naïve Approach

Follow-The-Leader (FTL)

$$\forall t, \quad \mathbf{w}_t = \operatorname*{argmin}_{\mathbf{w} \in S} \sum_{i=1}^{t-1} f_i(\mathbf{w}) \quad \text{(break ties arbitrarily)}$$

- For linear optimization problems, in which $f_t(\mathbf{w}) = \mathbf{z}_t^T \mathbf{w}$,

  FTL has unbounded regret.

  Just imagine a 1-dimensional $\mathbf{z}_t^T$ alternating between -1 and 1 each round.

Follow-the-Regularized-Leader (FoReL)

- Regularization stabilizes the solution of follow-the-leader
- Goal: eliminate the jumpiness!

- Regularization function: $R: S \rightarrow \mathbb{R}$
- $\boldsymbol{w}_t = \operatorname{argmin}_{\mathbf{w} \in S} \sum_{i=1}^{t-1} f_i(\boldsymbol{w}) + R(\boldsymbol{w})$

Special Case: Linear Loss Functions

- FoReL: $\boldsymbol{w}_t = \operatorname{argmin}_{\mathbf{w} \in S}[\sum_{i=1}^{t-1} f_i(\boldsymbol{w}) + R(\boldsymbol{w})]$

- Let $f_t(\boldsymbol{w}) = < \boldsymbol{w}, \boldsymbol{z}_t >$, $S = \mathbb{R}^d$, $R(\boldsymbol{w}) = \frac{1}{2\eta}||\boldsymbol{w}||_2^2$

$$\boldsymbol{w}_{t+1} = \operatorname{argmin}_{\boldsymbol{w}}\left[\sum_{i=1}^{t} f_i(\boldsymbol{w}) + R(\boldsymbol{w})\right]$$
$$= \operatorname{argmin}_{\boldsymbol{w}}\left[\sum_{i=1}^{t} < \boldsymbol{w}, \boldsymbol{z}_i > + \frac{1}{2\eta}\left||\boldsymbol{w}|\right|_2^2\right]$$

- Set derivative equal to zero:
- $0 = \sum_{i=1}^{t} \boldsymbol{z}_i + \frac{1}{2\eta} 2\boldsymbol{w}$ → $\boldsymbol{w}_{t+1} = -\eta \sum_{i=1}^{t} \boldsymbol{z}_i$

Special Case: Linear Loss Functions

$$\boldsymbol{w}_{t+1} = -\eta \sum_{i=1}^{t} \boldsymbol{z}_i = -\eta \sum_{i=1}^{t-1} \boldsymbol{z}_i - \eta \boldsymbol{z}_t = \boldsymbol{w}_t - \eta < \boldsymbol{z}_t, t > 1$$

- Note that $f_t(\boldsymbol{w}) = < \boldsymbol{w}, \boldsymbol{z}_t > \rightarrow \nabla f_t(\boldsymbol{w}_t) = z_t$

- This gives us the formula for gradient descent for linear functions:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla f_t(\boldsymbol{w}_t)$$

Special Case: Linear Loss Functions

How can we bound the regret?

- $\text{Regret}_T(\boldsymbol{u}) = \sum_{t=1}^{T}(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}))$
- Let $f_t(\boldsymbol{w}) = < \boldsymbol{w}, \boldsymbol{z}_t >,\ S = \mathbb{R}^d,\ R(\boldsymbol{w}) = \frac{1}{2\eta}||\boldsymbol{w}||_2^2$
- For all $\boldsymbol{u}$, we have the following regret bound:

$$\text{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_2^2$$

Special Case: Linear Loss Functions

How can we bound the regret?

- $\text{Regret}_T(\boldsymbol{u}) = \sum_{t=1}^{T}(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}))$
- Let $f_t(\boldsymbol{w}) = < \boldsymbol{w}, \boldsymbol{z}_t >,\ S = \mathbb{R}^d,\ R(\boldsymbol{w}) = \frac{1}{2\eta}||\boldsymbol{w}||_2^2$
- For all $\boldsymbol{u}$, we have the following regret bound:

$$\text{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_2^2$$

Regularization term

Loss functions have greater magnitude

Digression: Proof of Lemma 2.3

Lemma 2.3. Let $\mathbf{w}_1, \mathbf{w}_2, \ldots$ be the sequence of vectors produced by FoReL. Then, for all $\mathbf{u} \in S$ we have

$$\sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})).$$

- Define a function f_0 to be equal to the regularization function R
- Then, running FoReL on $\mathrm{f}_1, \ldots, \mathrm{f}_T$ is equivalent to running FTL on $f_0, f_1, \ldots, f_T$
- Apply Lemma 2.1, which is:

$$\sum_{t=0}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \sum_{t=0}^{T} (f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}))$$

Digression: Proof of Lemma 2.3, continued

Lemma 2.3. Let $\mathbf{w}_1, \mathbf{w}_2, \ldots$ be the sequence of vectors produced by FoReL. Then, for all $\mathbf{u} \in S$ we have

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1})).$$

$$\sum_{t=0}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \sum_{t=0}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}))$$

- Since $f_0 = R$, this is equivalent to:

$$\sum_{t=1}^{T}\big(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u})\big) + R(\boldsymbol{w}_0) - R(\boldsymbol{u}) \leq \sum_{t=1}^{T}\big(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{w}_{t+1})\big) + R(\boldsymbol{w}_0) - R(\boldsymbol{w}_1)$$

Special Case: Linear Loss Functions (Thm. 2.4)

- From Lemma 2.3: $$\text{Regret}_T(\mathbf{u}) \leq R(\mathbf{u}) - R(\mathbf{w}_1) + \sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{w}_{t+1}))$$

- By definitions of R and f_t:

$$\text{Regret}_T(u) \leq \frac{1}{2\eta}||\boldsymbol{u}||_2^2 - \frac{1}{2\eta}||\boldsymbol{w}_1||_2^2 + \sum_{t=1}^{T} \langle \boldsymbol{w}_t - \boldsymbol{w}_{t+1}, \boldsymbol{z}_t \rangle$$

$$\leq \frac{1}{2\eta}||\boldsymbol{u}||_2^2 + \sum_{t=1}^{T} \langle \boldsymbol{w}_t - \boldsymbol{w}_{t+1}, \boldsymbol{z}_t \rangle$$

- Using that $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{z}_t$:

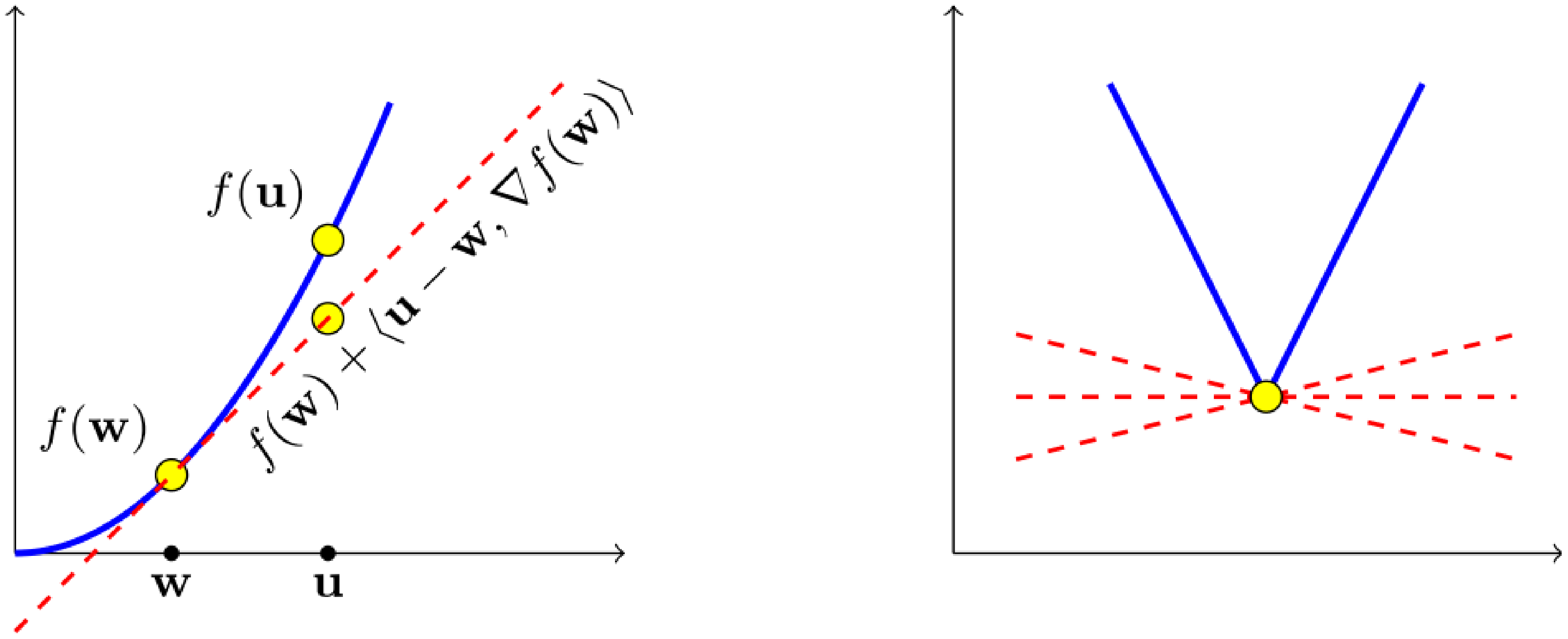
$$= \frac{1}{2\eta}||\boldsymbol{u}||_2^2 + \sum_{t=1}^{T} \langle \eta \boldsymbol{z}_t, \boldsymbol{z}_t \rangle = \frac{1}{2\eta}||\boldsymbol{u}||_2^2 + \eta \sum_{t=1}^{T} ||\boldsymbol{z}_t||_2^2$$

- Thus: $$\boxed{\text{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_2^2}$$

Extending from Linear to Convex Functions: Subgradients

Lemma 2.5. Let S be a convex set. A function $f : S \to \mathbb{R}$ is convex iff for all $\mathbf{w} \in S$ there exists $\mathbf{z}$ such that

$$\forall \mathbf{u} \in S, \quad f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \mathbf{z} \rangle. \tag{2.3}$$



Extending from Linear to Convex Functions: Online Gradient Descent

Online Gradient Descent (OGD)

parameter: $\eta > 0$
initialize: $\mathbf{w}_1 = \mathbf{0}$
update rule: $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{z}_t \quad \text{where } \mathbf{z}_t \in \partial f_t(\mathbf{w}_t)$

- SGD is the special case where loss functions are sampled iid from a distribution

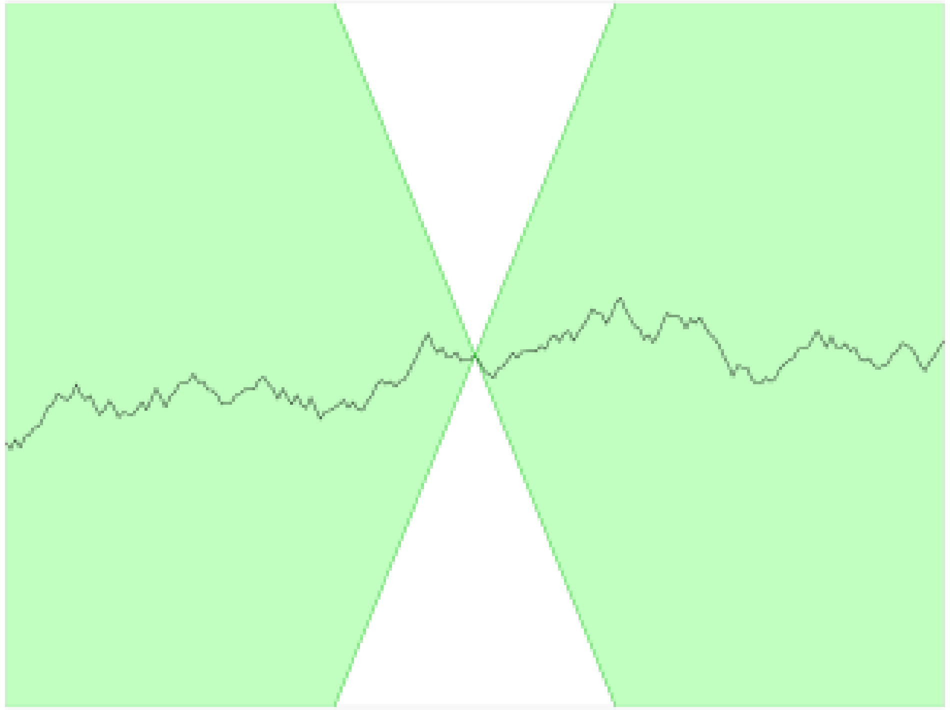
Extending from Linear to Convex Functions: (Unsatisfying) Regret Bound

$$\text{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta \sum_{t=1}^{T} \|\mathbf{z}_t\|_2^2.$$

- We want sublinear regret

Extending from Linear to Convex Functions: Lipschitz-ness to the rescue!

- A function f is L-Lipschitz if for all x, y in the domain of f, we have $|f(x) - f(y)| \le L|x - y|$



Extending from Linear to Convex Functions: Lipschitz-ness to the rescue!

- A function f is L-Lipschitz if for all x, y in the domain of f, we have |f(x) – f(y)| <= L|x – y|

If we further assume that each f_t is L_t-Lipschitz with respect to $\|\cdot\|_2$, and let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \leq L^2$. Then, for all $\mathbf{u}$, the regret of OGD satisfies

$$\mathrm{Regret}_T(\mathbf{u}) \leq \frac{1}{2\eta}\|\mathbf{u}\|_2^2 + \eta T L^2.$$

Extending from Linear to Convex Functions: How to pick step size?

In particular, if $U = \{\mathbf{u} : \|\mathbf{u}\|_2 \leq B\}$ and $\eta = \frac{B}{L\sqrt{2T}}$ then

$$\text{Regret}_T(U) \;\leq\; BL\sqrt{2T}.$$

Special Cases:
Specific Convex Functions

- *Logarithmic Regret Algorithms (Hazan)*
  - If the loss functions have bounded first and second gradients, OGD has regret $O(\log T)$
  - Other $O(\log T)$ algorithms exist for weaker assumptions:
    - Newton Step/FTApproximateL

      (for bounded gradient and concave exponential)
    - Exponentially Weighted Optimization

      (for concave exponential)

Special Case: Smoothed OCO

Andrew, L., Barman, S., Ligett, K., Lin, M., Meyerson, A., Roytman, A., & Wierman, A. (2013, June). A tale of two metrics: Simultaneous bounds on competitiveness and regret. In ACM SIGMETRICS Performance Evaluation Review (Vol. 41, No. 1, pp. 329-330). ACM.

Chen, N., Agarwal, A., Wierman, A., Barman, S., & Andrew, L. L. (2015, June). Online convex optimization using predictions. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (pp. 191-204). ACM.

Special Case: Smoothed OCO
The cost model

$$\text{cost}(ALG) = \mathbb{E}_x\left[\sum_{t=1}^{T} c_t(x_t) + \beta||x_t - x_{t-1}||\right]$$

Applications: Geographical Load Balancing, Dynamic Capacity Provisioning

Special Case: Smoothed OCO
Some performance metrics

dynamic solution. Specifically, the optimal offline *static* solution, is[1]

(3) $$STA = \operatorname{argmin}_{x \in G} \sum_{t=1}^{T} c_t(x) + \beta ||x||,$$

and the optimal *dynamic* solution is

(4) $$OPT = \operatorname{argmin}_{(x_1,\dots,x_T) \in G^T} \sum_{t=1}^{T} c_t(x_t) + \beta ||(x_t - x_{t-1})||.$$

Definition 1. *The* ***regret*** *of an online algorithm, ALG, is less than $\rho(T)$ if the following holds:*

(5) $$\sup_{(c_1,\dots,c_T) \in \mathcal{C}^T} \operatorname{cost}(ALG) - \operatorname{cost}(STA) \le \rho(T).$$

Definition 2. *An online algorithm ALG is said to be $\rho(T)$-**competitive** if the following holds:*

(6) $$\sup_{(c^1,\dots,c^T) \in \mathcal{C}^T} \frac{\operatorname{cost}(ALG)}{\operatorname{cost}(OPT)} \le \rho(T)$$

Special Case: Smoothed OCO
Result from first paper

- You can’t have both! (in the worst case)

Special Case: Smoothed OCO
Result from second paper

- … but in practice, it doesn’t matter!
- Most noise is not adversarial
- Often have access to “noisy predictions”
- => Propose an algorithm with good performance in both metrics

Special Case: Smoothed OCO
Averaging Fixed Horizon Control

- Fixed Horizon Control: Given predictions over timesteps t … t + w, just play whatever is optimal over these steps.

- AFHC: average several FHC algorithms starting at different time steps

Special Case: Iterative Soft-Thresholding Algorithm

- Assume L can be decomposed into differentiable and non-differentiable components:
- $L(\boldsymbol{w}) = G(\boldsymbol{w}) + H(\boldsymbol{w})$

Differentiable

Non-differentiable

- Example:
- $L(\boldsymbol{w}) = \underbrace{\sum_{(x_i,y_i)\in S} l(x_i, y_i, w)}_{G} + \underbrace{\lambda||\boldsymbol{w}||_1}_{H}$

Special Case: Iterative Soft-Thresholding Algorithm

- $L(\boldsymbol{w}) = G(\boldsymbol{w}) + H(\boldsymbol{w}) = \sum_{(x_i,y_i) \in S} l(x_i, y_i, w) + \lambda ||\boldsymbol{w}||_1$
- Solve the differentiable part *G* using gradient descent:
- $\boldsymbol{v}_t = \boldsymbol{w}_{t-1} - \eta_t \nabla_w G(\boldsymbol{w} = \boldsymbol{w}_{t-1})$
- For the non-differentiable part *H*, add a regularization term:

$$w_t \leftarrow \mathrm{argmin}_{w'} H(w') + \frac{1}{2\eta_t} \|w' - v_t\|_2^2$$

- Perform the minimization:

$$0 = \nabla_w H(w = w') + \frac{1}{\eta_t}(w' - v_t) \Rightarrow \nabla_w \left[\eta_t H(w = w')\right] + w' = v_t$$

Special Case:
Iterative Soft-Thresholding Algorithm

$$\nabla_w \alpha\lambda \|w\|_1 = \begin{cases} -\alpha\lambda & \text{if} \quad w \leq -\alpha\lambda \\ [-\alpha\lambda, \alpha\lambda] & \text{if} \quad -\alpha\lambda < w < \alpha\lambda \\ \alpha\lambda & \text{if} \quad w \geq \alpha\lambda \end{cases}$$

$$w_t = \begin{cases} v_t + \eta_t\lambda & \text{if} \quad v_t \leq -\eta_t\lambda \\ 0 & \text{if} \quad -\eta_t\lambda < v_t < \eta_t\lambda \\ v_t - \eta_t\lambda & \text{if} \quad v_t \geq -\eta_t\lambda \end{cases}$$

Generalizing OCO

Generalizations

What if our $\mathbf{w}_t$ *can break the rules sometimes?*

If the weight vectors need only be convex on average in the long run, lower-regret algorithms exist. *(Jennaton, et al.)*

Generalizations

What if our w_t *can't change too quickly?*

Not only is the regret worse under these 'ramp constraints', but learners must be designed not to constrain their future actions.
(Badiei, Li, Wierman)

Generalizations

What if our loss functions aren't convex?

Often OCO techniques still work, especially when training deep learners. *(Balduzzi)*

Generalizations

What if we, or our experts, don't have to play each round?

Often OCO techniques still work. *(Balduzzi)*

Generalizations

What if we have a whole network of online learners that can communicate?

Even when communication is local, low regret-algorithms exist. *(Koppel, et al.)*

Generalizations: Dynamically-Varying Environment

- What if the underlying environment varies over time?
- To improve performance, dynamically model the environment (Hall 2013, Hall 2014)
- *Dynamic Mirror Descent*: incorporates dynamical model state updates
- *Dynamic Fixed Share*: selects a dynamical model from a family of candidates at each time step
- DMD tracking regret bound: Φ is the dynamical system, and we take the regret with respect to the sequence of moves $\{\theta_t\}$

$$R(\theta_T) = O(\sqrt{T}[1 + \sum_t ||\theta_{t+1} - \Phi_t(\theta_t)||])$$

- Note: regret scales with deviation of $\{\theta_t\}$ from dynamical system Φ

Generalizations: Bandit Setting (Agarwal 2010)

- Bandit setting: at each time t, we only find out $l_t(x_t)$, not all of l_t
- Completely adaptive adversary: chooses l_t knowing $x_1, \dots, x_t$
  - Regret is $\Omega(T)$: *at least* order T
- Adaptive adversary: chooses l_t knowing $x_1, \dots, x_{t-1}$
  - Regret is $\Omega(\sqrt{T})$
  - Regret is $\tilde{O}(\sqrt{T})$ with linear loss functions (i.e. $O(\sqrt{T})$ with high probability)
- Compare with regret for full information case
  - $O(\sqrt{T})$ for convex Lipschitz and smooth loss functions
  - $O(\log(T))$ for strongly convex and smooth loss functions

Generalizations: Multi-Point Bandit Setting

- Player queries each loss function at k randomized points
- Take expected regret WRT the player's randomness:

$$Regret = \mathbb{E}\left[\frac{1}{k}\sum_{t=1}^{T}\sum_{i=1}^{k} l_t(y_{t,i})\right] - \min_{x\in\mathcal{K}} \mathbb{E}\left[\sum_{t=1}^{T} l_t(x)\right]$$

Setting	Regret for convex Lipschitz and smooth loss functions	Regret for strongly convex and smooth loss functions
Multi-Point, $k = 2$; adaptive adversary	$\tilde{O}(\sqrt{T})$ (with high probability)	$O(\log(T))$ (expected)
Multi-Point, $k = d + 1$; completely adaptive adversary	$O(\sqrt{T})$ (deterministic)	$O(\log(T))$ (deterministic)
Full Information	$O(\sqrt{T})$ (deterministic)	$O(\log(T))$ (deterministic)

Summary

- Online convex optimization framework captures a huge part of online learning
  - Lots of algorithms we already know are OCO, for instance SGD
- Follow-the-regularized leader, and variants, all perform well in OCO
- Generalizations of OCO are widespread, and rely on OCO algorithms

Questions?

References

- Agarwal, Alekh, Ofer Dekel, and Lin Xiao. "Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback." *COLT*. 2010.
- Andrew, L., Barman, S., Ligett, K., Lin, M., Meyerson, A., Roytman, A., & Wierman, A. (2013, June). A tale of two metrics: Simultaneous bounds on competitiveness and regret. In ACM SIGMETRICS Performance Evaluation Review (Vol. 41, No. 1, pp. 329-330). ACM.
- Badiei, Masoud, Na Li, and Adam Wierman. "Online Convex Optimization with Ramp Constraints."
- Balduzzi, David. "Deep Online Convex Optimization by Putting Forecaster to Sleep." *arXiv preprint arXiv:1509.01851* (2015).
- Chen, N., Agarwal, A., Wierman, A., Barman, S., & Andrew, L. L. (2015, June). Online convex optimization using predictions. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (pp. 191-204). ACM.
- Hall, Eric C., and Rebecca M. Willett. "Online convex optimization in dynamic environments." *Selected Topics in Signal Processing, IEEE Journal of* 9.4 (2015): 647-662.
- Hall, Eric C., and Rebecca M. Willett. "Dynamical models and tracking regret in online convex programming." *Proceedings of the 30th International Conference on Machine Learning*. 2013.
- Hazan, Elad, Amit Agarwal, and Satyen Kale. "Logarithmic regret algorithms for online convex optimization." *Machine Learning* 69.2-3 (2007): 169-192.
- Jenatton, Rodolphe, Jim Huang, and Cédric Archambeau. "Adaptive Algorithms for Online Convex Optimization with Long-term Constraints." *arXiv preprint arXiv:1512.07422* (2015).
- Koppel, Alec, Felicia Y. Jakubiec, and Alejandro Ribeiro. "A saddle point algorithm for networked online convex optimization." *Signal Processing, IEEE Transactions on* 63.19 (2015): 5149-5164.
- Yue, Yisong. "Advanced Optimization Notes." Course notes. http://www.yisongyue.com/courses/cs15/lectures/optimization.pdf

A Couple Back-Up Slides

Strongly Convex Function

Definition 2.4. A function $f : S \to \mathbb{R}$ is σ-strongly-convex over S with respect to a norm $\|\cdot\|$ if for any $\mathbf{w} \in S$ we have

$$\forall \mathbf{z} \in \partial f(\mathbf{w}), \quad \forall \mathbf{u} \in S, \quad f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{z}, \mathbf{u} - \mathbf{w} \rangle + \frac{\sigma}{2} \|\mathbf{u} - \mathbf{w}\|^2 .$$

Eliminating the Time Horizon Dependence: The Doubling Trick

For $m = 0, 1, 2, \ldots, \lceil \log_2 T \rceil$: run algorithm on the 2^m rounds $t = 2^m, \ldots, \min(2^{m+1} - 1,\ T)$

m = 0: run for t = 1

m = 1: run for t = 2, 3

m = 2: run for t = 3, …, 7

Etc.

Eliminating the Time Horizon Dependence: The Doubling Trick

- Assume the algorithm's regret on each 2^m rounds is bounded by $\alpha\sqrt{2^m}$

$$\text{Regret}_T(\boldsymbol{u}) \leq \sum_{m=0}^{\lceil \log_2 T \rceil} \alpha\sqrt{2^m} = \alpha \sum_{m=0}^{\lceil \log_2 T \rceil} \left(\sqrt{2}\right)^m$$

$$= \alpha \frac{1 - \sqrt{2}^{\lceil \log_2 T \rceil + 1}}{1 - \sqrt{2}} \leq \alpha \frac{1 - \sqrt{2}^{\log_2 T + 1}}{1 - \sqrt{2}}$$

$$= \alpha \frac{1 - \sqrt{2T}}{1 - \sqrt{2}} = \frac{\sqrt{2T} - 1}{\sqrt{2} - 1} \alpha \leq \frac{\sqrt{2}}{\sqrt{2} - 1} \alpha\sqrt{T}$$

- So, the regret bound only worsens by a constant multiplicative factor