**EE194/290S: ML for Sequential Decision under Uncertainty** **Fall 2018**

# Lecture 9: Online Optimization and Introduction to Games Learning

*Lecturer: Anant Sahai* *Scribes: Ashwin Balakrishna,Ishani Vyas*

## 9.1 Linear Losses in Online Optimization

Consider a state variable that evolves with time, such as the value of a stock, given by $\{z_t\}_{t=1}^{T}$. Suppose one's goal is to update a weight vector $w \in W$ at each timestep in order to minimize losses in hindsight. The linear loss function

$$l_t(w_t) = -z_t w_t \tag{9.1}$$

penalizes situations in which $z_t$ and $w_t$ are of opposite signs. This is akin to penalizing cases where one buys a stock that is doing poorly or sells a stock that is doing well.

Given all values of the state variable of interest up to time $t$, $\{z_i\}_{i=1}^{t}$, a follow the leader approach (FTL) would simply choose a weight vector $w_{t+1}$ such that loss in hindsight is minimized. From the loss function given by (9.1), it is clear that the choice of $w$ that minimizes loss in hindsight is:

$$w_{t+1} = \arg\min_{w} \sum_{i=1}^{t} -z_i w \tag{9.2}$$

As shown in last lecture, the above formulation provides no protection against adversarial examples, since $w$ is not constrained to be small. This motivated a regularized weight update as shown below:

$$w_{t+1} = \arg\min_{w} \sum_{i=1}^{t} -z_i w + \frac{1}{2\eta}||w||^2 \tag{9.3}$$

Optimizing the above expression to minimize $w$ gives the following:

$$0 = \sum_{i=1}^{t} [-z_i] + \frac{1}{\eta} w \tag{9.4}$$

which can be rearranged to obtain:

$$w_{t+1} = \eta \sum_{i=1}^{t} z_i \tag{9.5}$$

By the same logic, we see that:

$$w_t = \eta \sum_{i=1}^{t-1} z_i \tag{9.6}$$

Subtracting (9.6) from (9.5) and rearranging yields a weight-update function:

$$w_{t+1} = w_t + \eta z_t \tag{9.7}$$

In the last lecture we analyzed this weight update in detail and found that by choosing $\eta = \frac{B}{G\sqrt{2T}}$ and assuming that there are bounds $B$ and $G$ such that

$$B \geq ||u|| \qquad \forall u \in W \tag{9.8}$$
$$G \geq |z_t| \tag{9.9}$$

the total regret of this strategy is bounded above by $BG\sqrt{2T}$. This result shows that even in an adversarial environment, under linear losses it is possible to achieve average regret that approaches 0 as $T \to \infty$ subject to bounds on the norm of our weights and the signal strength of state variable $z_t$.

We show a plot of the average hindsight loss under linear loss for a specific sequence in Figure 9.1 below.

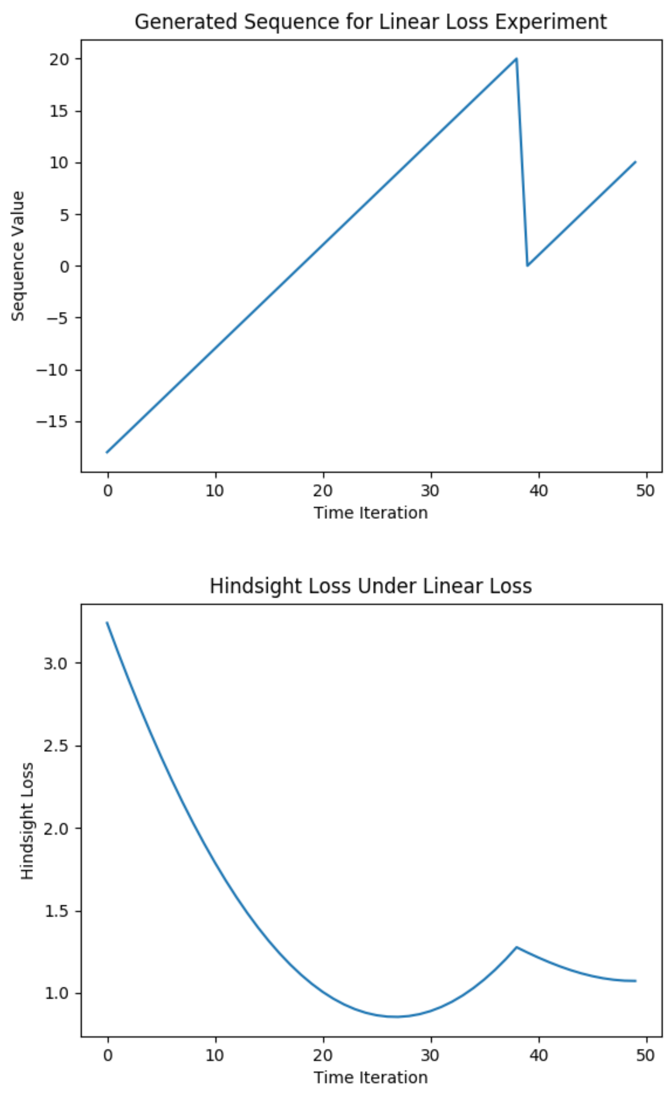

Figure 9.1: Here, G = 20, T = 50, B = 2. Notice that the average regret (average hindsight loss after the full 50 timesteps) is significantly lower than the average regret bound, given by $BG\sqrt{2/T} = 8$. The hindsight loss profile makes sense, since after about just 15 timesteps, the average hindsight loss starts rapidly decaying as expected.

However, we note that we can recast the weight update in terms of the gradient of the loss function as well. By observing that $\nabla_{w_t} l_t(w_t) = -z_t$ we can substitute $z_t$ in the update $w_{t+1} = w_t + \eta z_t$:

$$w_{t+1} = w_t - \eta \nabla_{w_t} l_t(w_t) \tag{9.10}$$

This will be discussed in further detail in the following section.

## 9.2 Convex Losses in Online Optimization

Now consider a *convex* loss function, such as squared loss, given by

$$l_t(w_t) = ||y_t - w_t x_t||^2 \tag{9.11}$$

This is not a linear loss function, so the regret guarantees that we proved previously may not hold. However, if we linearize this loss in the neighborhood of $w_t$, then it may be possible to reuse some of the same techniques.

Suppose we have some $w_t$ and then we draw some $(x_t, y_t)$ from nature. This causes us to incur some loss $l_t(w_t)$, based on which we seek to determine an updated loss $w_{t+1}$. For the loss function (9.11) we can use a first order Taylor expansion to linearly approximate our loss function in the neighborhood of $w$:

$$\hat{l}_t(w) = l_t(w_t) + l'_t(w_t)(w - w_t) \tag{9.12}$$

As long as this approximation and the bounds in (9.9) are reasonable, it is clear that we can achieve the same regret bound as the linearized loss formulated above ($\hat{l}$) as we showed for linear losses last lecture. For squared loss, we have the following loss function:

$$l(w_t) = ||y_t - w_t x_t||^2$$

Computing $\nabla_{w_t} l(w_t)$ yields the following:

$$\nabla_{w_t} l(w_t) = 2(w_t x_t - y_t)x_t \tag{9.13}$$

This only depends on $w_t$ (which we have already assumed is bounded per (9.8)), $x_t$, and $y_t$. It is also reasonable to assume that $x_t$ and $y_t$ are bounded, since this basically says that nature cannot provide us with any individual samples that have such high SNR that there would be no way to avoid having our predictions be heavily skewed. Thus, we see that under the above assumptions, our linearized loss function can achieve the same regret bound as truly linear loss functions. This bound is as follows:

$$\sum_{t=1}^{T} \left[\hat{l}_t(w_t) - \hat{l}_t(u)\right] \leq O(\sqrt{T}) \tag{9.14}$$

where $u \in W$ is the $u$ such that $u = \arg\min_{u'} \sum_{t=1}^{T} \hat{l}_t(u')$.

This is a good regret bound for linearized loss $\hat{l}$, but we still need to translate this into the appropriate regret bound for our original loss function $l$. Since we constructed $\hat{l}_t$ as follows:

$$\hat{l}_t(w) = l_t(w_t) + l'_t|_{w_t}(w - w_t)$$

If we let $w = w_t$ in (9.12) then

$$\hat{l}_t(w_t) = l_t(w)$$

This allows us to replace $\hat{l}_t(w_t)$ in (9.14) with $l_t(w_t)$ to obtain:

$$\sum_{t=1}^{T} \left[l_t(w_t) - \hat{l}_t(u)\right] \leq O(\sqrt{T})$$

Furthermore, since we constructed $\hat{l}_t(w)$ as a first-order Taylor approximation to $l_t(w)$, at $w_t$, since $l_t(w)$ is a convex loss function, we know that since convex functions are always above their supporting hyperplane, $l_t(w) \geq \hat{l}_t(w)\ \forall w$. Thus, we can simply replace $\hat{l}_t(u)$ with $l_t(u)$ in the above regret bound and still have it hold because $l_t(u) \geq \hat{l}_t(u)\ \forall u$. This gives us the following regret bound for convex losses, which is exactly the same as the bound we obtained for linear losses in the last lecture:

$$\sum_{t=1}^{T} [l_t(w_t) - l_t(u)] \leq O(\sqrt{T})$$

We show a plot of the average hindsight loss under squared loss for a specific sequence in Figure 9.2 below.

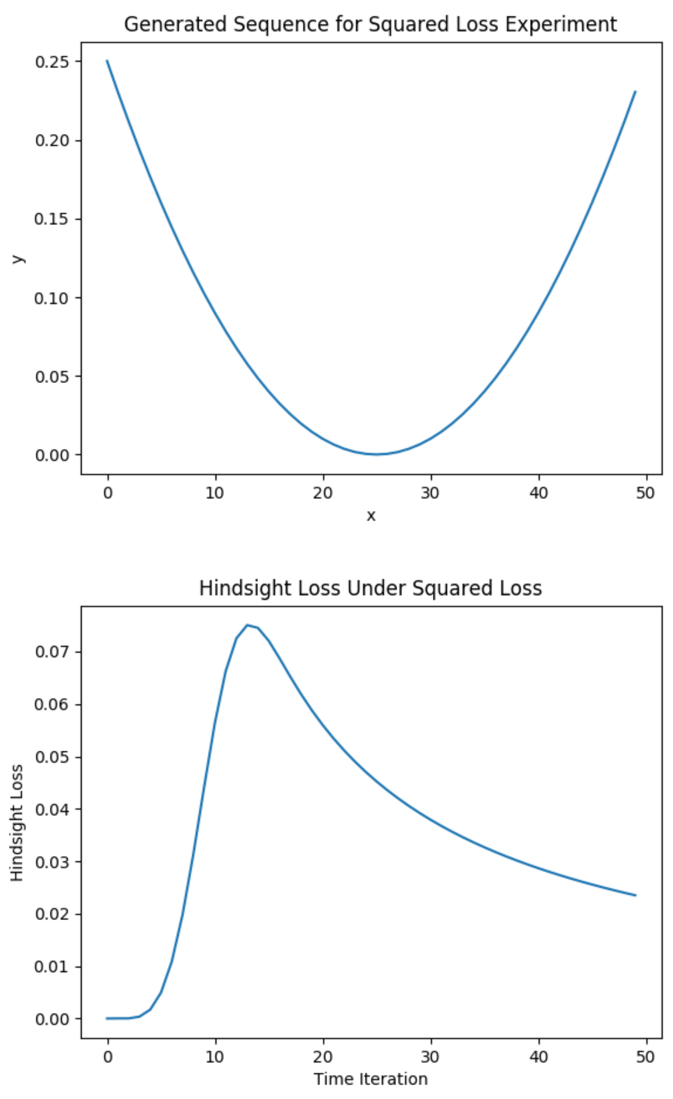

Figure 9.2: Here, G = 25, T = 50, B = 0.25. Notice that the average regret (average hindsight loss after the full 50 timesteps) is significantly lower than the average regret bound, given by $BG\sqrt{2/T} = 1.25$. The hindsight loss profile makes sense, since the abrupt change in the sequence values results in a controlled change in the hindsight loss, which then begins to decrease thereafter once the sequence reverts to its previous pattern.

Note that throughout the above discussion, we made no assumptions regarding the data distribution, but only assumed that our weights and loss gradients were bounded.

In showing that we can achieve low regret against convex loss functions, we utilized the fact that we had already shown that we can achieve low regret against linear loss functions and our prior knowledge about linear approximators to convex loss functions. However, even for the sequence prediction setting we implicitly utilized this convenient property of linear loss functions to achieve low regret.

While doing sequence prediction with FTL, we had a discrete loss function given by $l_t = \{l_{t,0}, l_{t,1}\}$. However, under multiplicative weights we instead learned probabilities $w_{t,0}$ and $w_{t,1}$ to determine whether to predict 1 or 0 at timestep $t$. This gave rise to an expected loss at timestep $t$ given by $l_t(w) = [l_{t,0}, l_{t,1}]\,\overrightarrow{w}$. Notice that this loss function is linear. As we will see, using the $w_{t+1} = w_t + \eta z_t$ update scheme in a special way will give us exponential weights again using a technique called mirror descent.

## 9.3 Mirror Descent

Regularized minimization for convex losses can be interpreted as simply taking a gradient step in dual space, a process known as Mirror Descent. Surprisingly, regularization leads to Mirror Descent, and this is a very general framework subsuming many known online algorithms. The intuition behind mirror descent is to adjust gradient updates based on the problem geometry.

Precisely, for some regularizer $R(a)$, for decisions given by

$$\hat{a_{t+1}} = \arg\min_{a \in \mathbb{R}^d} \left( \eta \sum_{s=1}^{t} \nabla l_s(a) \cdot a + R(a) \right) \tag{9.15}$$

it can be shown that:

$$\nabla R(a_{t+1}) = \nabla R(a_t) - \eta \nabla l_t(a_t) \tag{9.16}$$

which can be written as:

$$\hat{a_{t+1}} = (\nabla R)^{-1} (\nabla R(a_t) - \eta \nabla l_t(a_t) \tag{9.17}$$

This corresponds to

1. mapping $a_t$ in primal space $S$ to the corresponding point in dual space $\nabla R(a_t)$,
2. performing a gradient step $\eta \nabla l_t(a_t)$ to arrive at $\nabla R(a_{t+1})$ in dual space,
3. mapping $\nabla R(a_{t+1})$ back to point $\hat{a_{t+1}}$ in the primal space $S$,
4. projecting $\hat{a_{t+1}}$ back to to the original convex set $K$.

This gives rise to the Mirror Descent algorithm, which is given as follows:

**Algorithm 1** Mirror Descent

```
procedure MirrorDescent
    Select a regularizer                         ▷ This defines the dual space mapping $R(a)$
    loop until convergence criteria are met:
        Observe $l_t(a_t)$ from choosing action $a_t$.
        $w_{t+1} \leftarrow (\nabla R)^{-1} (\nabla R(a_t) - \eta \nabla l_t(a_t)$
        $a_{t+1} \leftarrow \arg\min_{a \in \mathcal{A}} D_R(a, w_{t+1})$
```

Here $D_R(a, b)$ defines the Bregman divergence of the mirror map given by regularizer $R$.

Now that we have established the general principle of Mirror Descent, we can show that exponential weights follow directly from Mirror Descent with the appropriate choice of regularizer $R(a)$. Suppose we pick a regularizer given by:

$$R(a) = \sum_{i=1}^{k} (a_i \log a_i - a_i) \tag{9.18}$$

It is easy to see that $\nabla R(u)_i = log(u_i)$. Furthermore, for Mirror Descent, we saw earlier that

$$\nabla R(a_{t+1,i}) = \nabla R(a_{t,i}) - \eta \nabla l_t(a_t)_i \tag{9.19}$$

However, since $\nabla R(u)_i = \log(u_i)$, we can see that $\nabla R(a_{t+1})_i = \log(a_{t+1,i})$ and $\nabla R(a_{t,i}) = \log(a_t)$. Therefore we can rewrite (9.19) as

$$\log(a_{t+1,i}) = \log(a_t) - \eta \nabla l_t(a_t)_i \tag{9.20}$$

which can be rearranged to:

$$a_{t+1,i} = a_{t,i} \; e^{-\eta \nabla l_t(a_t)_i} \tag{9.21}$$

Finally, it can be shown that the projection with respect to the Bregman divergence on the probability simplex $\{a \in \mathbb{R}^n_+ : \sum_{i=1}^n a_i = 1\}$ amounts to a simple renormalization. Thus, the functional form of the above expression for $a_{t+1}$ does not change.

Finally, since $a_{t+1,i} \propto w_{t+1,i}$, we can formulate our update scheme as follows, giving the familiar weight update formula for exponential weights:

$$w_{t+1,i} = a_{t,i} \; e^{-\eta \nabla l_t(a_t)_i} \tag{9.22}$$

## 9.4 Games and Learning Introduction

In 2 player games, we model strategic interactions of both players in some environment. For now, we focus on single-step games. Here, each player (denoted as Player $A$ and Player $B$) must perform an action, drawn from possibly different alphabets (denote these $\mathcal{A}$ and $\mathcal{B}$). The game is defined by a loss matrix for each player, which defines the loss that that player incurs when player A plays performs action $a \in \mathcal{A}$ and player B performs action $b \in \mathcal{B}$. Thus, we define the loss matrix for player $A$ as $P^A(a, b)\ \forall a \in \mathcal{A}, b \in \mathcal{B}$ and loss matrix for player $B$ as $P^B(a, b)\ \forall a \in \mathcal{A}, b \in \mathcal{B}$.

We illustrate the implications of certain loss matrices in single-step two-player games below. For simplification, we assume that player $A$ and player $B$ each draw actions from the alphabet $\{1, 2\}$. Furthermore, for compactness, we combine the loss matrices of both players into a single matrix. Each element of the matrix has a tuple $(a, b)$, where $a$ represents the loss of player $A$ and $b$ the loss of player $B$ for the combination of actions corresponding to that matrix entry. Each row of the tables below represents the actions that player $A$ can perform, while each column represents the actions that player $B$ can perform.

**Table 9.1** Dominant Strategy

| | | Player $B$ | |
|---|---|---|---|
| | | 1 | 2 |
| Player $A$ | 1 | $(+1, -)$ | $(+1, -)$ |
| | 2 | $(0, -)$ | $(0, -)$ |

For the game corresponding to the loss matrix in Table 9.1, we see that since player $A$ will always take a lower loss by performing action 2, regardless of the action chosen by player $B$, it does not matter what action player $B$ chooses. In situations like this, we call always playing action 2 player $A$'s dominant strategy. Thus, player $B$'s losses are just denoted by $-$ since player $B$'s decision making process is irrelevant for player $A$.

**Table 9.2** Natural Cooperation

| | | Player $B$ | |
|---|---|---|---|
| | | 1 | 2 |
| Player $A$ | 1 | $(+1, +1)$ | $(-1, -1)$ |
| | 2 | $(-1, -1)$ | $(+1, +1)$ |

For the game corresponding to the loss matrix in Table 9.2, we see that since player $A$ and player $B$ have the exact same loss matrix, and both obviously want to get a loss of $-1$, it is advantageous for the players to simply make opposite action choices. This is clear from the above loss matrix since when player $A$ chooses action 1 and player $B$ chooses action 2 or vice-versa, both take a loss of $-1$, while when they both choose the same action, both take a loss of $+1$. Thus, they have a natural incentive to cooperate to ensure that they make opposite decisions, since doing so will benefit both parties.

**Table 9.3** Natural Competition

| | | Player $B$ | |
|---|---|---|---|
| | | 1 | 2 |
| Player $A$ | 1 | $(+1, -1)$ | $(-1, +1)$ |
| | 2 | $(-1, +1)$ | $(+1, -1)$ |

For the game corresponding to the loss matrix in Table 9.3, we see that since player $A$ and player $B$ have the exact opposite loss matrix (sum of losses for any given action choice is 0), any deterministic strategy will be a poor choice, since if either player knows their opponent's strategy, then they can simply choose an action that will minimize their loss at the expense of the opponent. Thus, this situation lends itself to natural competition.

Inspired by this example of natural competition, we define a **Nash Equilibrium** for a 2 player strategic game as follows. A strategy profile (set of strategies for both players) is considered a Nash Equilibrium if, given the strategy of the opponent, no player can achieve a lower loss by changing their strategy. Here, a player's strategy is defined as a probability distribution over action choices.

In the natural competition example shown above, $\{(\frac{1}{2}, \frac{1}{2}), (\frac{1}{2}, \frac{1}{2})\}$ would be a Nash equilibrium since the strategy $(\frac{1}{2}, \frac{1}{2})$ over action set $\{1, 2\}$ makes you indifferent to your opponent's actions on average, as long as they too are following the same strategy, since both of you will end up with an average loss of 0.

In general, finding Nash equilibria can be shown to be computationally hard, but this is not the case for two player, zero-sum games, in which both players have loss matrices that are exactly in opposition to each other (sum of losses for any given set of action choices is 0), as seen in the natural competition example above. In the next lecture, we will explore the minimax theorem, which gives a convenient way to efficiently find Nash equilibria for zero-sum games.