Online Learning and Mirror Descent

Optimization and Machine Learning Seminar, NTUA December 2018

These notes are intended as reference notes for participants of the Optimization and Machine Learning Seminar in NTUA. Thus, they are in no way complete or free of errors. Please exercise critical thinking and don't take anything for granted.

1. Online Learning

In the Online Learning problem (also called Regret minimization problem) we wish to minimize the empirical loss, given that the data is provided to us sequentially (in an online fashion). More specifically, given some loss function $l : \mathcal{A} \times \mathcal{Z}$ where $\mathcal{A}$ is a set of *actions* (which can also be thought of as a hypothesis space in the context of statistical learning) and $\mathcal{Z}$ the set of possible data points, we are asked to come up with a sequence of actions $(a_1, \ldots, a_T) \in \mathcal{A}^T$ that minimize the *regret* of our strategy (which is essentially an online empirical loss):

$$\sum_t l(a^t, z^t) - \min_{a \in \mathcal{A}} \sum_t l(a, z^t)$$

Moreover, each action a^t has to be decided by the algorithm *before* the corresponding data point z^t is revealed. Essentially we are asked to provide an online sequence of actions whose total loss is comparable with that of the best *fixed* action in hindsight.

Of course, as stated, the Online Learning problem is extremely general. In order for us to have any chance of solving it, we should make some assumptions on the structure of the sets $\mathcal{A}$ and $\mathcal{Z}$ as well as the loss function l. There are numerous such special cases that are very interesting from a Machine Learning point of view.

1.1. Learning from experts.

One particular well known special case of the Online Learning problem is the *Learning from experts* framework. In this framework, we assume to have n *experts*, each with their own sequence of actions and we wish to predict binary outcomes over a sequence of rounds. Specifically, in each round t, each player's action $b^t(i) \in \mathcal{A}$ for that round is revealed, and depending on those, we are asked to decide our action a^t (probabilistically). Let's assume binary actions and outcomes $\mathcal{A} = \mathcal{Z} = \{0, 1\}$ and a loss function $l(x, y) = 1_{x \neq y}$.

More concretely, in each round we will keep track of a convex combination p^t of the expert actions (essentially a probability distribution over experts) and let $l^t(i) \in \{0, 1\}$ be the loss of the i-th expert in the t-th round. The goal is to minimize the expected regret:

$$R_T = \sum_{t \leq T} \sum_{i=1}^{n} p^t(i) l^t(i) - \min_{1 \leq i \leq n} \sum_{t \leq T} l^t(i)$$

A natural way to try to update p^t is to scale down the weights of experts i that made a mistake, i.e. $l^t(i) = 1$. Indeed, the following strategy, called *Multiplicative Weight Update*, gives asymptotically optimal regret:

$$p_t(i) = \frac{e^{-\eta \sum\limits_{t' < t} l_t(i)}}{\sum\limits_{i=1}^{n} e^{-\eta \sum\limits_{t' < t} l_t(i)}}$$

Here $\eta > 0$ is a parameter to be chosen.

It is easy to notice that this strategy can be implemented as follows (hence its name): After every round t, we scale the weight of each expert that mispredicted by a factor of $e^{-\eta}$ and then normalize the weights to yield a probability distribution.

The following theorem can be proved:

Theorem 1.1. For any $0 < \eta < 1/2$ and any expert i the Mutliplicative Weight Update as described above achieves a total loss of

$$\sum_{t \leq T} \sum_{i=1}^{n} p^t(i) l^t(i) \leq \sum_{t \leq T} l^t(i) + \eta \sum_{t \leq T} l^t(i) + \frac{\log n}{\eta}$$

By upper bounding the above quantity by $\sum\limits_{t \leq T} l^t(i) + \eta T + \frac{\log n}{\eta}$ and setting $\eta = \sqrt{\frac{\log n}{T}}$, we immediately get

$$R_T \leq \sqrt{T \log n}$$

This is pretty impressive, since it means that, up to a sublinear term in the number of rounds, we will perform as well as the best expert in hindsight! This also shows that if we need the *average* regret to be at most ϵ, then after $\Theta\left(\frac{\log n}{\epsilon^2}\right)$ rounds this is automatically guaranteed.

It should be mentioned that the Multiplicative Weights Update algorithm is an extremely versatile tool in algorithm design as well. Some of its uses include approximately solving Linear Programs, efficiently computing spectral sparsifiers, etc. Interestingly, as we will see, this seemingly ad-hoc algorithm can be seen as a special case of a more general optimization algorithm called Mirror Descent.

2. Mirror Descent

We first consider the proof of subgradient descent for Lipschitz and bounded radius convex functions from the previous lecture, which we reproduce here for completeness. Then, we will generalize it to yield a whole family of algorithms that depends on the choice of the norm $\|\cdot\|$ defining the gradient step.

2.1. **Subgradient Descent.** If our function is not smooth but is L-Lipschitz lower bounding the function value decrease in each iteration as above is not possible. Instead, we use a potential argument in the lines of "While the function value is large, we are moving fast towards the optimum". The algorithm will be (sub)gradient descent (because f might not be differentiable, $\nabla f(x)$ denotes any vector such that $f(y) \geq f(x) + \nabla f(x)^T (y - x)$ for all y), given by $x^{t+1} = x^t - \eta \nabla f(x^t)$, for some still undetermined step size η.

$$\begin{aligned} f(x^t) - f(x^*) \leq& \nabla f(x^t)^T (x^t - x^*) \\ =& \frac{1}{\eta}(x^t - x^{t+1})^T (x^t - x^*) \end{aligned}$$

On the other hand, we have

$$\begin{aligned} \|x^t - x^*\|_2^2 - \|x^{t+1} - x^*\|_2^2 =& \|x^t - x^{t+1}\|_2^2 + 2(x^{t+1} - x^*)^T (x^t - x^{t+1}) \\ =& \|x^t - x^{t+1}\|_2^2 + 2(x^t - x^*)^T (x^t - x^{t+1}) + 2(x^{t+1} - x^t)^T (x^t - x^{t+1}) \\ =& -\|x^t - x^{t+1}\|_2^2 + 2(x^t - x^*)^T (x^t - x^{t+1}) \\ \geq& -\eta^2 \|\nabla f(x^t)\|_2^2 + \eta(f(x^t) - f(x^*)) \end{aligned}$$

so after T iterations this gives

$$\begin{aligned}\frac{1}{T}\sum_t (f(x^t)-f(x^*)) \leq & \frac{1}{\eta T}(\|x^0-x^*\|_2^2-\|x^T-x^*\|_2^2)+\eta L^2\\ \leq & \frac{R^2}{\eta T}+\eta L^2\\ \leq & \frac{RL}{\sqrt{T}}\end{aligned}$$

for $\eta = \frac{R}{L\sqrt{T}}$.

Since f is convex, setting $\bar{x} = \frac{1}{T}\sum\limits_t x^t$ we get

$$f(\bar{x})-f(x^*) \leq \frac{1}{T}\sum_t (f(x^t)-f(x^*)) \leq \frac{RL}{\sqrt{T}} \leq \epsilon$$

so $T = O(\frac{R^2L^2}{\epsilon^2})$ iterations are enough.

2.2. **General norms.** Remember that the gradient descent step $x^{t+1} = x^t - \eta\nabla f(x^t)$ essentially comes from the optimal solution to the following optimization problem:

$$\min_\delta \nabla f(x^t)^T\delta + \frac{1}{2\eta}\|\delta\|_2^2$$

However, why are we using an ℓ_2 norm here? As we saw in the previous lecture it is sometimes possible to capture the geometry of the problem in a much better way by replacing this regularization term by a different norm of δ. However, if we are to try to generalize the subgradient descent analysis for general norms, there is a very serious problem: The cosine law that is crucially used in the proof is only true for the ℓ_2 norm!

To overcome this barrier, we think even more generally and don't even require the regularization term to come from a norm. This motivates the definition of a generalization of the ℓ_2^2 distance, given in the following section.

2.3. **Bregman Divergence.** A Bregman Divergence is a (not necessarily symmetric) distance function $D: X^2 \to \mathbb{R}_+$, determined by a convex function $\Phi: X \to \mathbb{R}$ (called a *mirror map*) as

$$D(y,x) = \Phi(y) - \Phi(x) - \nabla\Phi(x)^T(y-x)$$

It is essentially the error of the linear approximation of Φ at x, evaluated at y. In particular, if we pick $\Phi(x) = \frac{1}{2}\|x\|_2^2$ this sets $D(y,x) = \frac{1}{2}\|y-x\|_2^2$.

Note now that

$$\begin{aligned}D(c,a)-D(c,b) =& \Phi(c)-\Phi(a)-\nabla\Phi(a)^T(c-a)-\Phi(c)+\Phi(b)+\nabla\Phi(b)^T(c-b)\\ =& \Phi(b)-\Phi(a)-\nabla\Phi(a)^T(b-a)+(\nabla\Phi(b)-\nabla\Phi(a))^T(c-b)\\ =& D(b,a)+(\nabla\Phi(b)-\nabla\Phi(a))^T(c-b)\end{aligned}$$

which already looks quite similar to what we needed for the subgradient descent analysis to work.

2.4. **Mirror Descent.** Motivated by the above, we define the step of our generalized algorithm to be

$$\min_{x^{t+1}} \nabla f(x^t)^T(x^{t+1}-x^t) + \frac{1}{\eta}D(x^{t+1},x^t)$$

where D is the Bregman Divergence defined by the mirror map Φ. By taking optimality conditions to find the optimal step, we get that

$$0 = \nabla f(x^t) + \frac{1}{\eta}\nabla_{x^{t+1}}\left[D(x^{t+1},x^t)\right] = \nabla f(x^t) + \frac{1}{\eta}(\nabla\Phi(x^{t+1})-\nabla\Phi(x^t))$$

therefore it should be true that

$$\nabla\Phi(x^{t+1}) = \nabla\Phi(x^t) - \eta\nabla f(x^t)$$

This looks a lot like a gradient descent step, only that it is applied on a special function of the iterates x^t instead of the iterates themselves. In order to get an actual iteration, we make sure that the map $\nabla\Phi$ is invertible, its inverse being $\nabla\Phi^*$ (Here Φ^* is called the *Fenchel dual* of Φ, but we will not go more into that). One condition to ensure we can apply the inverse is that its domain is $\mathbb{R}^m$, or in other words that $\nabla\Phi$ is onto $\mathbb{R}^m$.

Now we can conclude with our step, which is

$$x^{t+1} = \nabla\Phi^* \left(\nabla\Phi(x^t) - \eta\nabla f(x^t)\right)$$

Let's also pick $x^0 = \underset{x}{\operatorname{argmin}}\ \Phi(x)$. In our analysis we will also need the property that the mirror map is strongly convex and has bounded values, and that f is Lipschitz. Therefore we assume that Φ is ℓ-strongly convex with respect to some norm $\|\cdot\|$, as well as that f is G-Lipschitz with respect to that norm and that $\sup_x \Phi(x) - \Phi(x^0) \leq D^2$. Note that we are ignoring here the fact that f might be defined in a restricted domain X, in which case we would need to project (using the distance defined by D) back to X. This is just for ease of presentation. The property that lets us deal with this in the analysis is

$$D(c,a) \geq D(c,b) + D(b,a)$$

which is a generalization of the Pythagorean theorem.

2.5. **Analysis of Mirror Descent.** As we menioned before, the analysis of Mirror Descent is in the lines of Subgradient Descent, with the potential being $D(x^*, x^t)$. In particular, we know that

$$f(x^t) - f(x^*) \leq \nabla f(x^t)^T(x^t - x^*) = \frac{1}{\eta}(\nabla\Phi(x^t) - \nabla\Phi(x^{t+1}))^T(x^t - x^*)$$

and that

$$\begin{aligned}
&D(x^*, x^t) - D(x^*, x^{t+1}) \\
=&D(x^{t+1}, x^t) + (\nabla\Phi(x^{t+1}) - \nabla\Phi(x^t))^T(x^* - x^{t+1}) \\
=&D(x^{t+1}, x^t) + (\nabla\Phi(x^t) - \nabla\Phi(x^{t+1}))^T(x^t - x^*) + (\nabla\Phi(x^t) - \nabla\Phi(x^{t+1}))^T(x^{t+1} - x^t) \\
\geq& - D(x^t, x^{t+1}) + \eta(f(x^t) - f(x^*))
\end{aligned}$$

Therefore

$$\sum_t (f(x^t) - f(x^*)) \leq \frac{1}{\eta}(D(x^*, x^0) - D(x^*, x^T) + \sum_t D(x^t, x^{t+1}))$$

To bound $D(x^t, x^{t+1})$ we will use the strong convexity and Lipschitz assumptions as follows

$$\begin{aligned}
D(x^t, x^{t+1}) =&\Phi(x^t) - \Phi(x^{t+1}) - \nabla\Phi(x^{t+1})^T(x^t - x^{t+1}) \\
=& - D(x^{t+1}, x^t) + (\nabla\Phi(x^{t+1}) - \nabla\Phi(x^t))^T(x^{t+1} - x^t) \\
\leq& - \frac{\ell}{2}\|x^{t+1} - x^t\|^2 + \eta G\|x^{t+1} - x^t\| \\
\leq&\frac{\eta^2 G^2}{2\ell}
\end{aligned}$$

where the last inequality comes from the fact that $-ax^2 + bx \leq \frac{b^2}{4a}$ for $a > 0$. Putting everything together, we have

$$\begin{aligned} f(\bar{x}) - f(x^*) \leq & \frac{1}{T} \sum_t \left(f(x^t) - f(x^*) \right) \\ \leq & \frac{1}{\eta T} (D(x^*, x^0) - D(x^*, x^T)) + \frac{\eta G^2}{2\ell} \\ \leq & \frac{D^2}{\eta T} + \frac{\eta G^2}{2\ell} \\ \leq & \frac{DG}{\sqrt{2\ell T}} \end{aligned}$$

for $\eta = \frac{D\sqrt{2\ell}}{G\sqrt{T}}$

Therefore after $T = O\left(\frac{D^2G^2}{\ell\epsilon^2}\right)$ iterations we get an ϵ-approximate solution.

To summarize, to optimize a convex function f that is G-Lipschitz with respect to $\|\cdot\|$ up to ϵ accuracy using a Mirror map that is ℓ-strongly convex with respect to $\|\cdot\|$ and has radius D, $O\left(\frac{D^2G^2}{\ell\epsilon^2}\right)$ iterations suffice. It is easy to see that this recovers the Subgradient descent guarantee.

2.6. **Relation to Online Learning.** In the analysis of the previous section we essentially showed that

$$\sum_t \nabla f(x^t)^T (x^t - x^*) \leq \frac{DG\sqrt{T}}{\sqrt{2\ell}}$$

However, the only property of $\nabla f(x^t)$ used for this proof was their Lipschitzness. As a matter of fact, we can replace them with arbitrary vectors g^t of bounded norm, i.e. $\|g^t\|_* \leq G$. In this case, the result becomes

$$\sum_t (g^t)^T (x^t - x^*) \leq \frac{DG\sqrt{T}}{\sqrt{2\ell}}$$

which is essentially a regret bound for online learning with linear loss functions $l(x, g) = g^T x$!

2.7. **Common setups.**

ℓ_2 **norm**. Here we use $\Phi(x) = \frac{1}{2}\|x\|_2^2$ which is obviously 1-strongly convex with respect to the ℓ_2 norm. This exactly recovers the Subgradient Descent algorithm.

ℓ_1 **norm**. By Pinsker's inequality, the KL divergence upper bounds half the square of the ℓ_1 norm of the difference between two probability distributions. Furthermore, the KL divergence is generated as a Bregman divergence by the negative entropy mirror map $\Phi(x) = \sum_i x_i \log x_i$ defined on the unit simplex $X = \Delta_n$ and so $\Phi(x)$ is 1-strongly convex with respect to ℓ_1.

$$D(y, x) = \sum_i y_i \log y_i - \sum_i x_i \log x_i - \sum_i (1 + \log x_i)(y_i - x_i) = \sum_i y_i \log \frac{y_i}{x_i}$$

We also have $[\nabla\Phi(x)]_i = 1 + \log x_i$ and $[\nabla\Phi^*(x)]_i = e^{x_i - 1}$. Therefore our step will be

$$x_i^{t+1} = \nabla\Phi^*(\nabla\Phi(x^t) - \eta g^t) = \nabla\Phi^*(\vec{1} + \log x^t - \eta g^t) = x^t e^{-\eta g^t}$$

but this is exactly the Multiplicative Weights Update method! In fact, if $\|g^t\|_\infty \leq 1$, as is the case in the Learning from experts problem we studied, the guarantee we get is $O(\frac{DG\sqrt{T}}{\sqrt{\ell}})$ with $D^2 = \log n$, $G^2 = 1$, $\ell = 1$ and so we get a regret of $O(\sqrt{T \log n})$, which is exactly the right bound.
Note: As we said before, we have ignored the step of projecting onto the feasible set. Here, this is the simplex Δ_n and we are performing a Bregman projection with the KL Divergence. It is easy

to see that this exactly corresponds to normalizing the entries of the vector so that their sum is 1. Therefore the projection step in this setup is quite simple.