6.883 Learning with Combinatorial Structure
Note for Lecture 19
Author: Michael Cohen

1 Other submodular problems

We briefly mention some additional problems on submodular functions.

1.1 Submodular minimization under constraints

Submodular minimization finds the exact overall minimum of a submodular function F, but we might want to minimize the function with respect to additional constraints.

In general, this can be hard to solve exactly. However, relaxing the problem or approximating the function F can lead to efficient algorithms. One approach is to use linear approximation–replacing F with a linear function that dominates it. Another approach, from [Goemans et al, 2004], starts with the representation of a submodular function F in terms of its base polytope B:

$$F(S) = f(\mathbf{1}_S) = \max_{y \in B} y^T \mathbf{1}_S$$

The idea is to approximate the base polytope B with another region, leading to a more tractable function approximating F. Specifically, this approach replaced B with an ellipsoid.

1.2 Learning submodular functions

Another problem is to (approximately) learn a submodular function, given only limited access to the function. There are several different variants of this problem, depending the available information and the desired guarantee.

One relatively strong model is "active" learning, in which one is given oracle access to the function, but can make only a limited number of queries. Even in this model, however, it has been shown to be hard to learn a submodular function up to *multiplicative* error.

A weaker model provides only observations of evaluations of the function on specific inputs: that is, one is given a set of ordered pairs $(S, F(S))$, rather than being able to choose

S. Since this model is strictly weaker than the active one, it is still hard to achieve multiplicative error. However, even this weaker model can learn F up to *additive* error: that is, it can produce a function G such that $|F(S) - G(S)| \leq \epsilon$ for all S.

Another model is an instance of structured prediction. Here, one is given access to pairs (V_i, S_i) such that

$$V_i = \arg\max_{S \subseteq V_i} F(S)$$

Here, it is not necessarily possible to learn the entire submodular function, but one can obtain a function $\hat{F}$ for which these structured prediction queries will produce similar results.

2 Online learning

We move on to a new topic: *online learning*. There are several different problems in this setting. Here, "online" means that one receives information over time, and must make decisions as it arrives (as opposed to e.g. first being given all the examples and then processing all of them as a batch).

In this lecture we focused on a specific type of problem: online optimization or regret minimization.

2.1 Online optimization

The basic setting of online optimization is a game between a player and an "adversary", running for T rounds. The goal is to design a strategy for the player that achieve guarantees that are valid, *regardless of the choices of the adversary*. It is therefore valid even in adversarial conditions–thus the choice of name. The guarantees will be in terms of the player's "regret", which will be defined below.

The player has an allowable set of "actions", $\mathcal{K}$, for each round. Round t proceeds as follows:

- The player chooses any action w_t from $\mathcal{K}$.
- The adversary chooses a loss function ℓ_t, mapping $\mathcal{K}$ to real numbers.
- The player incurs a loss of $\ell_t(w_t)$, and receives some feedback.

The specifics of the problem still depend on the set of actions available $\mathcal{K}$, constraints on the loss function chosen by the adversary, and the feedback given.

2.2 Prediction from expert advice

A simple instance of this type of online learning problem is "prediction from expert advice". Here, there are n experts. Each round, the player can choose one expert, i, to listen to. They then incur a loss of $\ell_t(i)$, the values of which are constrained to lie in $[0, 1]$.

One might try to model this by setting $\mathcal{K}$ to $\{1, 2, \ldots, n\}$. However, plugging this in to the game described above does not allow the player to achieve strong guarantees (the adversary can always choose to inflict loss on whoever the player chose). Instead, we think of the player as being allowed to randomly choose an expert–but with the adversary not able to observe the random choice before choosing ℓ_t–and track their *expected* loss. This means that our actions are not actually single experts but probability distributions over experts, so $\mathcal{K} = \Delta_n$, the probability simplex.

The loss functions ℓ_t are then *linear* functions whose coefficients lie in $[0, 1]$ (as expected loss is linear in the choice of probability distribution).

2.3 Online shortest paths and structured problems

A more structured example is online shortest paths. Here, there is some graph with a source vertex s and a destination vertex t, and each round the player chooses an s-t path. The adversary chooses weights for each edge, and the loss incurred is the length of the path according to those weights.

One could try to reduce this to the experts scenario by creating an expert for each path. However, there would then be exponentially many experts, making this approach infeasible.

Instead, we can also just directly set $\mathcal{K}$ to the convex hull of all allowable choices (in this case, unit s-t flows), and attempt to solve the problem directly.

Other structured online learning problems, such as picking permutations and subsets, are similar.

2.4 Regret minimization

We are seeking algorithms with guarantees independent of the choices of the adversary, so clearly we do not want to ask for a universal bound on the loss (since the adversary could simply max out all losses each round). Instead, we might wish to compare the loss with the best possible loss achievable in retrospect–however, this clearly also cannot

work, since the adversary could assign the losses with no pattern, making them impossible to "guess".

Therefore, we instead define the *regret* R_t as the difference between the loss incurred and the best loss achievable by a *fixed* strategy (that is, playing the same action every round). In the case of experts, this means comparing the loss incurred to the loss of the single best expert.

$$R_T = \sum_{t=1}^{T} \ell_t(w_t) - \min_{u \in \mathcal{K}} \sum_{t=1}^{T} \ell_t(u)$$

The goal will be to design a strategy for which this regret is uniformly bounded over all choices of the adversary. Furthermore, we would like these bounds to show that the average regret $\frac{R_T}{T}$ can go to 0 as $T \to \infty$.

2.5 Full information vs bandit

The remaining important piece in defining these online optimization problems is specifying the nature of the feedback given to the player. The two major possibilities are:

- Full information: the player is given the complete function ℓ_t after each round.
- Bandit: the player is only told their actual loss incurred–only one value of the function ℓ_t.

For the rest of this lecture, we will be focusing on the full information case only.

2.6 Linear and convex loss functions

We will only be considering *linear* loss functions (and convex regions $\mathcal{K}$).

Note, however, that if the loss functions ℓ_t were *convex* rather than linear, we could replace them with a local linear approximation $(\nabla \ell_t(w_t))^\top w$. One then can see that by the convexity of the loss functions, this can only overestimate the regret, meaning that regret guarantees for the linear case imply them for the convex case (as long as the gradients or subgradients of the convex function lie in the domain we allow for the linear loss functions). In fact, this means that these online optimization algorithms automatically imply algorithms for minimizing convex functions using gradients, just by applying this argument to a *fixed* convex function (which will still have varying linear approximations between rounds). That is one way to derive the convex optimization algorithm *mirror descent*.

3 Strategies for online learning/optimization

We will now discuss actual strategies for these problems.

3.1 Follow-the-leader

A very simple and natural strategy is "follow-the-leader": choosing the action that minimizes the loss function from the rounds seen so far:

$$w_t = \arg\min_{w \in \mathcal{K}} \sum_{\tau=1}^{T-1} \ell_\tau(w)$$

Note that in the case of experts, this will always choose a single fixed expert rather than a probability distribution. This type of strategy can never achieve vanishing regret bounds. Intuitively, picking the single best action is too unstable: a tiny change in the losses can make a big change to the leader, giving the adversary too much power to manipulate.

3.2 Stabilizing

We then can try to modify this strategy to make it more stable. In general, we can look at strategies of the form

$$w_t = \arg\min_{w \in \mathcal{K}} \eta \sum_{\tau=1}^{T-1} \ell_\tau(w) + \mathcal{R}(w)$$

One idea–follow-the-*perturbed*-leader–is to make $\mathcal{R}$ a *random* linear function–in other words, follow-the-leader with extra noise added to the loss function. This causes a random choice of w and makes the algorithm more stable.

If the coefficients of this linear function are chosen from a *uniform* distribution, this gives an expected regret bound of at most

$$m\sqrt{2dT}$$

where m is maximum ℓ_1 norm inside $\mathcal{K}$. This is from [Kalai, Vempala 2005].

We will focus on a different approach: follow-the-*regularized*-leader. Here, $\mathcal{R}$ is set to a deterministic but nonlinear function which acts as a regularizer. Recall that this still corresponds to a randomized strategy in most instances, since we interpret the result as a probability distribution.

3.3 Follow-the-regularized-leader

In this approach, we specifically want $\mathcal{R}$ to be a strongly convex regularizer. Important special cases are:

- The squared ℓ_2 norm:
$$\mathcal{R}(w) = \|w\|_2^2$$
This leads to an algorithm resembling (sub)gradient descent.
- The entropy regularizer:
$$\mathcal{R}(w) = \sum_i w_i \log w_i - w_i$$
On the simplex, this leads to the "multiplicative weights update method".

3.4 Bounds on follow-the-regularized-leader

We now state the generic bounds on this method. Let $\nabla^2\mathcal{R}$ indicate the Hessian (matrix of second partial derivatives) of $\mathcal{R}$. Then, for any sequence of loss vectors ℓ_t define
$$G = \max_t \max_{w \in \mathcal{K}} \ell_t^\top (\nabla^2 \mathcal{R}(w))^{-1} \ell_t$$
This essentially measures the (squared) max norm of any loss vector according to the strong convexity of $\mathcal{R}$. Additionally, define the diameter-like
$$D = \left(\max_{u \in K} R(u)\right) - \left(\min_{w \in K} R(w)\right)$$
Then setting η optimally one can achieve a regret bound of
$$R_T \leq 2\sqrt{2GDT}$$
For the special case of the d-dimensional simplex with the entropy regularizer, we can see that D is $\log d$ (since entropy is always positive and maximized at $\log d$). If the coefficients of the loss vector at at most 1, then we can show that $G \leq 1$. This results in a regret bound of
$$R_T \leq 2\sqrt{2\log(d)T}$$

3.5 Proving the bounds

To prove this bound, we begin by defining

$$\phi_t(w) = \eta \sum_{\tau=1}^{t-1} \ell_\tau^\top w + \mathcal{R}(w)$$

We can then prove the following lemma:

Lemma 1. *For any $u \in \mathcal{K}$,*

$$\sum_{t=1}^{T} \ell_t^\top (w_t - u) \leq \sum_{t=1}^{T} \ell_t^\top (w_t - w_{t+1}) + \frac{D}{\eta}$$

Proof. First, by the definition of our strategy, w_t is the minimizer of ϕ_t. Thus, we have

$$\begin{aligned} \phi_{T+1}(w_{T+1}) &\leq \phi_{T+1}(u) \\ &= \eta \sum_{t=1}^{T-1} \ell_t^\top u + \mathcal{R}(u) \end{aligned}$$

Now, again using the definition of w,

$$\begin{aligned} \phi_{t+1}(w_{t+1}) - \phi_t(w_t) &\geq \phi_{t+1}(w_{t+1}) - \phi_t(w_{t+1}) \\ &= \eta \ell_t^\top w_{t+1} \\ &= \eta \ell_t^\top w_t + \eta \ell_t^\top (w_{t+1} - w_t) \end{aligned}$$

Summing this, we get

$$\phi_{T+1}(w_{T+1}) \geq \phi_1(w_1) + \eta \sum_{t=1}^{T} (\ell_t^\top w_t + \ell_t^\top (w_{t+1} - w_t))$$

Combining the upper and lower bounds, we get

$$\begin{aligned}
\phi_1(w_1) + \eta \sum_{t=1}^{T} (\ell_t^\top w_t + \ell_t^\top (w_{t+1} - w_t)) &\leq \eta \sum_{t=1}^{T-1} \ell_t^\top u + \mathcal{R}(u) \\
\eta \sum_{t=1}^{T} \ell_t^\top (w_t - u) &\leq \eta \sum_{t=1}^{T} \ell_t^\top (w_t - w_{t+1}) + (\mathcal{R}(u) - \phi_1(w_1)) \\
\sum_{t=1}^{T} \ell_t^\top (w_t - u) &\leq \sum_{t=1}^{T} \ell_t^\top (w_t - w_{t+1}) + \frac{1}{\eta}(\mathcal{R}(u) - \mathcal{R}(w_1)) \\
&\leq \sum_{t=1}^{T} \ell_t^\top (w_t - w_{t+1}) + \frac{D}{\eta}
\end{aligned}$$

□

Now we wish to bound $\ell_t^\top (w_t - w_{t+1})$. First, define the W norm.

$$\|x\|_X^2 = \max_{w \in \mathcal{K}} x^\top (\nabla^2 \mathcal{R}(w))^{-1} x$$

Define the Z norm as the dual norm to this. By construction, $\mathcal{R}$, and thus the ϕ_t, are 1-strongly convex with respect to the Z norm (since their Hessian norms dominate it). In particular, we can write

$$\phi_{t+1}(w_{t+1} + w') - \phi_{t+1}(w_{t+1}) \geq \frac{1}{2}\|w'\|_Z^2$$

Note that by definition, $\phi_t(w_t) \leq \phi_t(w_{t+1})$, or

$$\begin{aligned}
\phi_{t+1}(w_{t+1} + (w_t - w_{t+1})) - \eta \ell_t^\top w_t &\leq \phi_{t+1}(w_{t+1}) - \eta \ell_t^\top w_{t+1} \\
\phi_{t+1}(w_{t+1} + (w_t - w_{t+1})) - \phi_{t+1}(w_{t+1}) &\leq \eta \ell_t^\top (w_t - w_{t+1}) \\
\frac{1}{2}\|w_t - w_{t+1}\|_Z^2 &\leq \eta \|\ell_t\|_X \|w_t - w_{t+1}\|_Z \text{ (by duality of norms)} \\
\|w_t - w_{t+1}\|_Z &\leq 2\eta \|\ell_t\|_X \\
&= 2\eta\sqrt{G}
\end{aligned}$$

Applying the dual norm inequality again, we get

$$\ell_t^\top (w_t - w_{t+1}) \leq 2\eta G$$

Plugging this into 1, we get a regret bound of

$$2\eta G T + \frac{D}{\eta}$$

Setting $\eta = \sqrt{2GDT}$ gives $2\sqrt{2GDT}$, as desired.

4 Other variants

There is an alternative version of this strategy that uses only a saved w_t, rather than saving the sum of the ℓ_t. This achieves the same bounds, and can be formulated in terms of the "Bregman divergence."