[begin lecture 13] (13)

5 Online learning

5.1 Introduction (Lecture 13)

- Thus far, we have analyzed algorithms (maximum likelihood, ERM) in the *statistical setting*, where we assume the the training and test data are both drawn i.i.d. from some distribution p^*. We even boasted that we need not make any assumption about what p^* is. In this unit, we will weaken the assumptions even more and assume that data can be generated completely *adversarily*. In addition, we will move to the online setting where training and test are interleaved. Thus we make two shifts to the learning setup:
  - Batch to online
  - Statistical to adversarial
- We will first discuss the online learning framework, focusing on prediction. Then, we will cast online learning as online convex optimization and develop several algorithms and prove regret bounds for these algorithms. Finally, we will look at multi-armed bandit problems, where the learner obtains partial feedback. Throughout this section, there will be very little probability (since we will be working in the adversarial setting), but we will draw quite a bit from convex analysis.
- Framework
  - Prediction task: we want to map inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$.
  - The online learning setting can be thought of as the following game between a learner and nature:

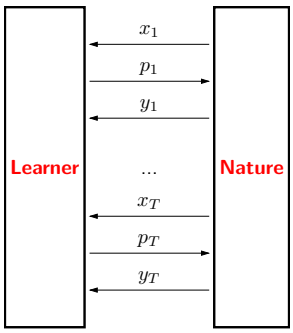

Figure 9: Online learning game.

  - $*$ Iterate $t = 1, \ldots, T$:

· Learner receives input $x_t \in \mathcal{X}$
· Learner outputs prediction $p_t \in \mathcal{Y}$
· Learner receives true label $y_t \in \mathcal{Y}$
· (Learner suffers loss $\ell(y_t, p_t)$)
· (Learner updates model parameters)

– More formally, the learner is a function $\mathcal{A}$ that returns the current prediction given the full history:[15]

$$p_{t+1} = \mathcal{A}(x_{1:t}, p_{1:t}, y_{1:t}, x_{t+1}). \tag{459}$$

Nature can be defined similarly.

- **Example 23 (online binary classification for spam filtering)**
  - Inputs: $\mathcal{X} = \{0, 1\}^d$ are boolean feature vectors (presence or absence of a word).
  - Outputs: $\mathcal{Y} = \{+1, -1\}$: whether a document is spam or not spam.
  - Zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$ is whether the prediction was incorrect.
- Remarks
  - The typical training phase (setting model parameters) and testing phase (making predictions for evaluation) are **interleaved** in online learning.
  - Note that the the online learning setting leaves completely open the time and memory usage of the algorithms that operate in this setting. Technically, we could just train an SVM on all the data that we've seen so far, and predict on the next example. However, the spirit of online learning suggests that the amount of work an algorithm does per iteration should not grow with t. In practice, online learning algorithms update parameters after each example, and hence tend to be **faster** than traditional batch optimization algorithms such as Newton's method.
  - The real world is complex and constantly-changing, but online learning algorithms have the potential to **adapt** (although we will not analyze their adapative capabilities in this course).
  - In some applications such as spam filtering, the inputs could be generated by an **adversary**. In our analyses, we will make no assumptions about the input/output sequence.
- Evaluation
  - Now comes the most important part: How we measure the quality of an online learner $\mathcal{A}$? While seemingly simple and obvious, how we answer this question has a *defining impact* on the behavior of $\mathcal{A}$.

[15] For now, assume deterministic algorithms. Later, we'll consider stochastic algorithms, which will be important against adversaries.

cf) adversarial ML: a research field that lies at the intersection of ML and computer security (e.g., biometric authentication, network intrusion detection, and spam filtering).
cf) adversarial training: adversarial samples can cause any ML algorithm fail to work.
However, they can be leveraged to build a more accurate model.
=> learning with an adversary; a two-player game

ex 1) min-max objective function
ex 2) unified gradient regularization framework

- The first attempt is to just write down the cumulative loss of the learner:

$$\sum_{t=1}^{T} \ell(y_t, p_t). \tag{460}$$

However, if we are in the adversarial setting, no algorithm can do better than the maximum regret T (for the zero-one loss) since the adversary can always choose $y_t \neq p_t$. What do you do when your performance is awful? You show that others are doing even worse than you.

- Less flippantly, let $\mathcal{H}$ be a set of experts, where each **expert** is a function $h : \mathcal{X} \to \mathcal{Y}$ that predicts $h(x)$ on input $x \in \mathcal{X}$. Note that the learner can adapt (change over time) whereas the experts are fixed.
- **Definition 24 (regret)**
  - The regret of a learner with respect to an expert h is the cumulative difference between the loss incurred by the learner and the loss incurred by expert h:

$$\text{Regret}(h) \stackrel{\text{def}}{=} \sum_{t=1}^{T} [\ell(y_t, p_t) - \ell(y_t, h(x_t))]. \tag{461}$$

    Note that $\text{Regret}(h)$ depends on $\mathcal{H}$, T, the sequences $x_{1:T}, y_{1:T}$, and of course the algorithm $\mathcal{A}$ itself; but we're eliding the dependence on all of these things in the notation.
  - The regret with respect to a class of experts $\mathcal{H}$ is the maximum regret

$$\text{Regret} \stackrel{\text{def}}{=} \max_{h \in \mathcal{H}} \text{Regret}(h). \tag{462}$$

    Equivalently, we can write:

$$\text{Regret} = \underbrace{\sum_{t=1}^{T} \ell(y_t, p_t)}_{\text{learner}} - \underbrace{\min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))}_{\text{best expert}}. \tag{463}$$

- The best expert is a role model for the learner. While it is technically possible for the learner to do better than all the experts since it can adapt over time (leading to negative regret), this generally won't happen, and therefore, we will be content with trying to achieve small positive regret.
- We are interested in particular in the maximum possible regret over all sequences $x_{1:T}, y_{1:T}$; in more colorful language, an adversary chooses the inputs and outputs as to maximize regret.
- Having a comparison set of experts gives us some hope to compete against an adversary. Intuitively, if the adversary tries to screw over the learner, it will probably screw over the experts too.

- In the next few lectures, we will prove results of the following form for various online learning algorithms $\mathcal{A}$: for all $\mathcal{H}, T, x_{1:T}, y_{1:T}$, we have:

$$\text{Regret} \leq \text{SomeFunction}(\mathcal{H}, T). \tag{464}$$

  Usually, we want the regret to be sublinear in T, which means that the average regret per example goes to zero.

5.2 Warm-up (Lecture 13)

We will give two examples, one where online learning is impossible and one where it is possible.

- **Example 24 (negative result)**
  - Assume binary classification: $y \in \{-1, +1\}$
  - Assume zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$
  - Assume the learner is fully **deterministic**.
  - Claim: for all learners $\mathcal{A}$, there exists an $\mathcal{H}$ and input/output sequence $x_{1:T}, y_{1:T}$ such that:

$$\boxed{\text{Regret} \geq T/2} \quad \text{[awful!]} \tag{465}$$

  - **Key point**: adversary (having full knowledge of learner) can choose y_t to be always different from p_t.
  - Learner's cumulative loss: T (make mistake on every example).
  - We're not done yet, because remember regret is the difference between learner's cumulative loss and the best expert's, so we have to check how well the experts do in this case.
  - Consider two experts, $\mathcal{H} = \{h_{-1}, h_{+1}\}$, where h_y always predicts y.
  - The sum of the cumulative losses of the experts equals T because $\ell(y_t, h_{-1}(x_t)) + \ell(y_t, h_{+1}(x_t)) = 1$, so one of the experts must achieve loss $\leq T/2$.
  - Therefore, the difference between T and $\leq T/2$ is $\geq T/2$.
  - It is perhaps not surprising that no learner can do well because nature is too powerful here (it can choose any sequence with full knowledge of what the learner will do). Not even measuring regret with respect to only two experts is enough. So we will need assumptions to get positive results.

- **Example 25 (positive result (learning with expert advice))**
  - Assume zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$.

- Clearly, we need to make *some* assumptions to make progress. Here, we will make a fairly strong assumption just to get some intuition.
- **Assumption 3 (realizable)**
  Assume the best expert $h^* \in \mathcal{H}$ obtains zero cumulative loss ($\ell(y_t, h^*(x_t)) = 0$ for all $t = 1, \ldots, T$); or equivalently, since we're dealing with zero-one loss, $y_t = h^*(x_t)$ for all t. This assumption places a joint restriction on $\mathcal{H}, x_{1:T}, y_{1:T}$.
- We will design an algorithm that queries the experts on each example x_t, and try to combine their predictions in some way; this setting is called **learning with expert advice**.
- **Algorithm 1 (majority algorithm)**
  * Maintain a set $V_t \subseteq \mathcal{H}$ of valid experts (those compatible with the first $t-1$ examples).
  * On each iteration t, predict p_t to be the majority vote over predictions of valid experts $\{h(x_t) : h \in V_t\}$.
  * Keep experts which were correct: $V_{t+1} = \{h \in V_t : y_t = h(x_t)\}$.
- Analysis:
  * On each mistake, at least half of the experts are eliminated.
  * So $1 \leq |V_{T+1}| \leq |\mathcal{H}| 2^{-M}$, where M is the number of mistakes (also equal to Regret since the best expert has zero loss).
  * Note that the lower bound is due to the realizability assumption.
  * M is the exactly the regret here.
  * Take logs and rearrange:

$$\boxed{\text{Regret} \leq \log_2 |\mathcal{H}|.} \tag{466}$$

- Notes:
  * This is a really strong result: note that the regret is constant; after a finite number of iterations, we cease to make any more mistakes forever.
  * However, realizability (that some expert is perfect) is too strong of an assumption.
  * Also, the regret bound is useless here if there are an infinite number of experts ($|\mathcal{H}| = \infty$).

5.3 Online convex optimization (Lecture 13)

- In order to obtain more general results, we move to a framework called online convex optimization. **Convexity** will give us considerable leverage. Afterwards, we'll connect online convex optimization with online learning.
- Let $S \subseteq \mathbb{R}^d$ be a convex set (e.g., representing the set of allowed weight vectors).

- Example: $S = \{u : \|u\|_2 \leq B\}$.

- FIGURE: [convex function with subgradients]

- **Definition 25 (convexity)**

  A function $f : S \to \mathbb{R}$ is convex iff for all points $w \in S$, there is some vector $z \in \mathbb{R}^d$ such that

  $$\boxed{f(u) \geq f(w) + z \cdot (u - w) \quad \text{for all } u \in S.} \tag{467}$$

  This says that at any point $w \in S$, we can find a linear approximation (RHS of (467)) that lower bounds the function f (LHS of (467)).

- **Definition 26 (subgradient)**

  For each $w \in S$, the set of all z satisfying (467) are known as the subgradients at w:

  $$\boxed{\partial f(w) \stackrel{\text{def}}{=} \{z : f(u) \geq f(w) + z \cdot (u - w) \text{ for all } u \in S\}.} \tag{468}$$

  If f is differentiable at w, then there is one subgradient equal to the gradient: $\partial f(w) = \{\nabla f(w)\}$.

- Convexity is remarkable because it allows you to say something (in particular, lower bound) the global behavior (for all $u \in S$) of a function f by something local and simple (a linear function). An important consequence of f being convex is that if $0 \in \partial f(w)$, then w is a global minimum of f.

- Checking convexity

  - How do you know if a function is convex? You can try to work it out from the definition or if the function is twice-differentiable, compute the Hessian and show that it's positive semidefinite.
  - But often, we can check convexity by decomposing the function using a few rules. This is by no means an exhaustive list, but these are essentially all the important cases that we'll need in this class.
    - Linear functions: $f(w) = w \cdot z$ for any $z \in \mathbb{R}^d$
    - Quadratic functions: $f(w) = w^\top A w$ for positive semidefinite $A \in \mathbb{R}^{d \times d}$
    - Negative entropy: $f(w) = \sum_{i=1}^d w_i \log w_i$ on $w \in \Delta_d$.
    - Sum: $f + g$ if f and g are convex
    - Scale: cf where $c \geq 0$ and f is convex
    - Supremum: $\sup_{f \in \mathcal{F}} f$, where $\mathcal{F}$ is a family of convex functions
  - Example: hinge loss (skipped in class)
    - Function: $f(w) = \max\{0, 1 - y(w \cdot x)\}$.

* Subgradient:
  - $\partial f(w) = \{0\}$ if $y(w \cdot x) > 1$ (w achieves margin at least 1)
  - $\partial f(w) = \{-yx\}$ if $y(w \cdot x) < 1$
  - $\partial f(w) = \{-yx\alpha : \alpha \in [0, 1]\}$ if $y(w \cdot x) = 1$

- The setup for online convex optimization is similar to online learning:
  - Iterate $t = 1, \ldots, T$:
    * Learner chooses $w_t \in S$
    * Nature chooses convex loss function $f_t : S \to \mathbb{R}$

    Formally, the learner $\mathcal{A}$ chooses w_{t+1} depending on the past:

$$w_{t+1} = \mathcal{A}(w_{1:t}, f_{1:t}). \tag{469}$$

  - Regret is defined in the way you would expect (cumulative difference of losses):

$$\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)]. \tag{470}$$

$$\text{Regret} \stackrel{\text{def}}{=} \max_{u \in S} \text{Regret}(u). \tag{471}$$

  - The set S plays two roles: it is the set of experts with which we define our regret, and it is also the set of parameters that our learner is going to consider. For simplicity, we assume these two are the same, although in general, they do not have to be.

- We now turn to our original goal of doing online learning. We will show some examples of reducing online learning to online convex optimization. In particular, given an input/output sequence input $x_{1:T}, y_{1:T}$, we will construct a sequence of convex functions $f_{1:T}$ such that the low regret incurred by a learner on the online convex optimization problem implies low regret on the online learning problem. Let OL be the online learner who has access to OCO, a online convex optimizer.
  - **Example 26 (linear regression)**
    * FIGURE: [two boxes, OL and OCO with arrows between]
    * Assume we are doing linear regression with the squared loss, $\ell(y_t, p_t) = (p_t - y_t)^2$.
    * On each iteration $t = 1, \ldots, T$:
      - OL receives an input $x_t \in \mathbb{R}^d$ from nature.
      - OL asks OCO for a weight vector $w_t \in \mathbb{R}^d$.
      - OL sends the prediction $p_t = w_t \cdot x_t$ to nature.
      - OL receives the true output y_t from nature.

· OL relays the feedback to OCO via the loss function

$$f_t(w) = (w \cdot x_t - y_t)^2. \tag{472}$$

Note that (x_t, y_t) is baked into f_t, which changes each iteration. Since the squared loss is convex, f_t is convex. One can check easily based on matching definitions that the regret of OCO is exactly the same as regret of OL.

– **Example 27 (learning with expert advice)**

* Now let's turn to a problem where the convexity structure isn't as apparent.
* Assume we have a finite number (this is important) of experts $\mathcal{H}$, which the learner can query. Let

$$\mathcal{H} = \{h_1, \ldots, h_d\}. \tag{473}$$

* Assume we are doing binary classification with the zero-one loss (this is not important—any bounded loss function will do), $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$.
* Note that neither the loss function nor the set of experts $\mathcal{H}$ is convex; in fact, this doesn't really make sense, since the domains are discrete.
* Nonetheless, we can convexify the problem using **randomization**. Specifically, we allow the learner to produce a probability distribution $w_t \in \Delta_d$ ($\Delta_d \subseteq \mathbb{R}^d$ is the $(d-1)$-dimensional simplex)[16] over the d experts $\mathcal{H}$, sample an expert from this distribution, and predict according to that expert. The expected zero-one loss is then a convex (in fact, linear) function of w_t.
* Formally, the reduction is as follows:
  · OL receives an input x_t from nature.
  · OL asks OCO for a weight vector $w_t \in \Delta_d$, which represents a distribution over d experts.
  · OL samples an expert $j_t \sim w_t$ and send prediction $p_t = h_{j_t}(x_t)$ to nature.
  · OL receives the true output y_t from nature.
  · OL relays the feedback to OCO via the loss function

$$f_t(w) = w \cdot z_t, \tag{474}$$

where z_t is the vector of losses incurred by each expert:

$$z_t = [\ell(y_t, h_1(x_t)), \ldots, \ell(y_t, h_d(x_t))] \in \{0, 1\}^d. \tag{475}$$

Again, f_t bakes the loss and the data into the same function, and one can check that the expected regret of OL is the same as the regret of OCO. Note that we assume no structure on x_t and y_t—they could be arbitrarily complicated; we are only exposed to them via the the experts and the loss function.

[16] Formally, think of a probability distribution over a random variable $J \in \{1, \ldots, d\}$, with $\mathbb{P}[J = j] = w_j$.

* This convexify-by-randomization trick applies more generally to any loss function and any output space $\mathcal{Y}$. The key is that the set of experts is finite, and the learner is just choosing a convex combination of those experts.
* Yet another way to convexify non-convex losses without randomization is to use an upper bound (e.g., hinge loss or logistic loss upper bounds the zero-one loss). However, minimizing the convex upper bound does not guarantee minimizing the zero-one loss.

5.4 Follow the leader (FTL) (Lecture 13)

We first start out with a natural algorithm called follow the leader (FTL), which in some sense is the analog of the majority algorithm for online learning. We'll see that it works for quadratic functions but fails for linear functions. This will give us intuition about how to fix up our algorithm.

- **Algorithm 2 (follow the leader (FTL))**
  - Let $f_1, \dots, f_T$ be the sequence of loss functions played by nature.
  - The learner chooses the weight vector $w_t \in S$ that minimizes the cumulative loss so far on the previous $t-1$ iterations:

$$\boxed{w_t \in \arg\min_{w \in S} \sum_{i=1}^{t-1} f_i(w).} \tag{476}$$

    (If there are multiple minima, choose any one of them. This is not important.)
  - Aside: solving this optimization problem at each iteration is expensive in general (which would seem to destroy the spirit of online learning), but we'll consider special cases with analytic solutions.
  - Note: We can think of FTL as an empirical risk minimizer where the training set is the first $t-1$ examples.

- We want to now study the regret of FTL. Regret compares the learner (FTL) with any expert $u \in S$, but this difference can be hard to reason about. So to get started, we will use the following result (Lemma 7) to replace u in the bound by (i) something easier to compare w_t to and (ii) at least as good as any $u \in S$.

- **Lemma 7 (compare FTL with one-step lookahead cheater)**
  - Let $f_1, \dots, f_T$ be any sequence of loss functions.
  - Let $w_1, \dots, w_T$ be produced by FTL according to (476). For any $u \in S$:

$$\boxed{\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)] \leq \sum_{t=1}^{T} [f_t(w_t) - f_t(w_{t+1})].} \tag{477}$$

- Note: This is saying our regret against the best fixed u is no worse than comparing against the one-step lookahead w_{t+1} that peeks at the current function f_t (which is cheating!).
- Note: The RHS terms $f_t(w_t) - f_t(w_{t+1})$ measure how *stable* the algorithm is; smaller is better. Stability is an important intuition to develop.

- Proof of Lemma 7:
  - Subtracting $\sum_t f_t(w_t)$ from both sides, it suffices to show

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(u) \tag{478}$$

    for all $u \in S$. Intuitively, this says that w_{t+1} (which takes the minimum over the first t functions) is better than using a fixed u for all time.
  - Proof by induction:
    * Assume the inductive hypothesis on $T-1$:

$$\sum_{t=1}^{T-1} f_t(w_{t+1}) \leq \sum_{t=1}^{T-1} f_t(u) \quad \text{for all } u \in S. \tag{479}$$

    * Add $f_T(w_{T+1})$ to both sides:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T-1} f_t(u) + f_T(w_{T+1}) \text{ for all } u \in S. \tag{480}$$

    * In particular, this holds for $u = w_{T+1}$, so we have:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(w_{T+1}). \tag{481}$$

    * Since $w_{T+1} \in \arg\min_u \sum_{t=1}^{T} f_t(u)$ by definition of FTL, we have:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(u) \text{ for all } u \in S, \tag{482}$$

      which is the inductive hypothesis for T.

- Note that Lemma 7 doesn't actually require on convexity of f_t, but rather only stability of the iterates $\{w_t\}$ as measured by f_t. As we'll see later, strong convexity is the main way we will achieve this stability. For now, let's consider two examples to gain some intuition: one where the $\{w_t\}$ are stable (Example 28) and one where they are not (Example 29).

- **Example 28 (quadratic optimization: FTL works)**
  - Assume nature always chooses quadratic functions:

$$f_t(w) = \frac{1}{2}\|w - z_t\|_2^2, \tag{483}$$

    where the points are bounded: $\|z_t\|_2 \leq L$ for all $t = 1, \ldots, T$.
  - FTL (minimizing over $S = \mathbb{R}^d$) has a closed form solution, which is just the average of the previous points:

$$w_t = \frac{1}{t-1}\sum_{i=1}^{t-1} z_i. \tag{484}$$

  - Bound one term of the RHS of Lemma 7 (intuitively the difference is only one term):

$$\begin{aligned} f_t(w_t) - f_t(w_{t+1}) &= \frac{1}{2}\|w_t - z_t\|_2^2 - \frac{1}{2}\|(1-1/t)w_t + (1/t)z_t - z_t\|_2^2 && (485)\\ &= \frac{1}{2}(1-(1-1/t)^2)\|w_t - z_t\|_2^2 && (486)\\ &\leq (1/t)\|w_t - z_t\|_2^2 && (487)\\ &\leq (1/t)4L^2. && (488) \end{aligned}$$

  - Side calculation: summing $1/t$ yields $\log T$:

$$\sum_{t=1}^{T}(1/t) \leq 1 + \int_1^T (1/t)dt = \log(T) + 1. \tag{489}$$

  - Summing over t yields:

$$\boxed{\text{Regret} \leq \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})] \leq 4L^2(\log(T)+1).} \tag{490}$$

  - The important thing here is that the difference between w_t and w_{t+1} (measured in terms of loss) is really small (only $1/t$), which means that FTL for quadratic functions is really stable. This makes sense because averages are stable: adding an extra data point should only affect the running average by $O(1/t)$.
- **Example 29 (linear optimization: FTL fails)**
  - We will construct an evil example to make FTL fail.

- Let $S = [-1, 1]$ be FTL's possible predictions. This is a nice bounded one-dimensional convex set, so we're not even trying hard to be pathological.
- Consider linear functions $f_t(w) = wz_t$ in $d = 1$ dimension, where

$$(z_1, z_2, \dots) = (-0.5, 1, -1, 1, -1, 1, -1, \dots). \tag{491}$$

- The minimizer computed by FTL will be attained at an extreme point, causing oscillating behavior.

$$(w_1, w_2, \dots) = (0, 1, -1, 1, -1, 1, -1, 1, \dots). \tag{492}$$

- FTL obtains $T - 1$ cumulative loss (get loss 1 on every single example except the first).
- Expert $u = 0$ obtains 0 cumulative loss (not necessarily even the best).
- Therefore, the regret is pretty depressing:

$$\boxed{\text{Regret} \geq T - 1}. \tag{493}$$

- What's the lesson?
  - For these quadratic functions, w_t and w_{t+1} must get closer (low regret).
  - For these linear functions, w_t and w_{t+1} do not get closer (high regret).
  - It seems then that FTL works when functions f_t offer "stability" (e.g., quadratic) but fail when they do not (e.g., linear).
  - We will reveal the more general principle (strong convexity) later work.

[begin lecture 14] (14)

5.5 Follow the regularized leader (FTRL) (Lecture 14)

- It would be nice if nature just handed us quadratic-looking f_t's, but in general, we're not the ones calling the shots there. But we do control the learner, so the key idea is to *add some regularization* of our own to stablize the learner.
- **Algorithm 3 (follow the regularized leader (FTRL))**
  - Let $\psi : S \to \mathbb{R}$ be a function called a **regularizer** (this defines the learner).
  - Let $f_1, \ldots, f_T$ be the sequence of loss functions played by nature.
  - On iteration t, the learner chooses the weight vector that minimizes the regularizer plus the losses on the first $t-1$ examples:

$$\boxed{w_t \in \arg\min_{w \in S} \psi(w) + \sum_{i=1}^{t-1} f_i(w).} \tag{494}$$

  - Note: FTL is just FTRL with $\psi = 0$.
- Quadratic ψ, linear f_t
  - For the remainder of the section, just to build the right intuition in a transparent way, let's specialize to quadratic regularizers ψ and linear loss functions f_t:

$$\psi(w) = \frac{1}{2\eta}\|w\|_2^2, \qquad f_t(w) = w \cdot z_t. \tag{495}$$

  - Then the FTRL optimization problem (494) is:

$$w_t = \arg\min_{w \in S} \left\{ \frac{1}{2\eta}\|w\|_2^2 - w \cdot \theta_t \right\}, \tag{496}$$

    where

$$\theta_t = -\sum_{i=1}^{t-1} z_i \tag{497}$$

    is the negative sum of the gradients z_t. Interpret θ_t as the direction that we want to move in to reduce the loss, but now, unlike in FTL, we're held back by the quadratic regularizer.

- If $S = \mathbb{R}^d$, then FTRL has a closed form solution:

$$w_t = \eta\theta_t, \tag{498}$$

a scaled down version of θ_t. We can write $w_t = -\eta z_1 - \eta z_2 - \cdots - \eta z_{t-1}$ and equivalently think of the weights as being updated incrementally according to the following recurrence:

$$\boxed{w_{t+1} = w_t - \eta z_t.} \tag{499}$$

From this perspective, the recurrence in (499) looks awfully like an online subgradient update where z_t is the gradient and η is the step size.

- If $S \neq \mathbb{R}^d$, then FTRL requires a projection onto S. We rewrite (496) by completing the square (add $\frac{\eta}{2}\|\theta_t\|_2^2$):

$$\boxed{w_t \in \arg\min_{w \in S} \frac{1}{2\eta}\|w - \eta\theta_t\|_2^2 = \Pi_S(\eta\theta_t),} \tag{500}$$

which is a Euclidean projection of $\eta\theta_t$ onto set S. This is called a **lazy projection** since θ_t still accumulates unprojected gradients, and we only project when we need to obtain a weight vector w_t for prediction. This is also known as Nesterov's **dual averaging** algorithm.

- Regularizers in online and batch learning
  - It's worth pausing to examine the difference in the way regularization enters online learning versus batch learning, with which you're probably more familiar.
  - In batch learning (e.g., in training an SVM or ridge regression), one typically seeks to optimize a function which is the sum of the training loss plus the regularizer. Notably, the regularizer is part of the objective function.
  - In online learning here, our objective in some sense is the regret, which makes no mention of the regularizer. The regularizer lies purely inside the learner's head, and is used to defines the updates. In our example so far, the regularizer (in the context of FTRL) gives birth to the online gradient algorithm in (499).
- Now we will analyze the regret FTRL for quadratic regularizers and linear losses.
- **Theorem 30 (regret of FTRL)**
  - Let $S \subseteq \mathbb{R}^d$ be a convex set (of weight vectors).
  - Let $f_1, \ldots, f_T$ be any sequence of linear loss functions: $f_t(w) = w \cdot z_t$ for some $z_t \in \mathbb{R}^d$.
  - Let $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ be a quadratic regularizer for any $\eta > 0$.

- Then the regret of FTRL (as defined in Algorithm 3) with respect to any $u \in S$ is as follows:

$$\boxed{\text{Regret}(u) \leq \frac{1}{2\eta}\|u\|_2^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|z_t\|_2^2.} \tag{501}$$

- Interpretation: the step size η allows us to trade off two terms:
  - First term: "squared bias" due to regularization. To compete with u, the learner's iterates w_t must somehow get close to u. Smaller η means more regularization (remember $w_t = \eta\theta_t$), which means it's harder to get there.
  - Second term: "variance" due to changing z_t. A smaller η means that successive weight vectors are closer and thus stabler (recall $w_{t+1} - w_t = -\eta z_t$). With a small η, we can hope to avoid the bad scenario in Example 29.
- FIGURE: [ellipse with u at edge, draw iterates w_t trying to reach u]
- Corollary:
  - To make the bound look cleaner:
    - Let $\|z_t\|_2 \leq L$.
    - Let $\|u\|_2 \leq B$ for all experts $u \in S$.

    Now (501) can be rewritten as:

$$\text{Regret}(u) \leq \frac{B^2}{2\eta} + \frac{\eta T L^2}{2}. \tag{502}$$

  - Side calculation:
    - Suppose we want to minimize some function with the following form: $C(\eta) = a/\eta + b\eta$.
    - Take the derivative: $-a/\eta^2 + b = 0$, resulting in $\eta = \sqrt{a/b}$ and $C(\eta) = 2\sqrt{ab}$, which is just twice the geometric average of a and b.
  - Letting $a = B^2/2$ and $b = TL^2/2$, we get $\eta = \frac{B}{L\sqrt{T}}$ and

$$\boxed{\text{Regret} \leq BL\sqrt{T}.} \tag{503}$$

    Note that the average regret goes to zero as desired, even though not as fast as for quadratic functions ($\log T$).
- Proof of weakened version of Theorem 30

– We will prove Theorem 30 later using Bregman divergences, but just to give some intuition without requiring too much technical machinery, we will instead prove a slightly weaker result (note that the second term is looser by a factor of 2):

$$\boxed{\text{Regret}(u) \leq \frac{1}{2\eta}\|u\|_2^2 + \eta \sum_{t=1}^{T} \|z_t\|_2^2.} \tag{504}$$

– The key idea is to reduce FTRL to FTL. Observe that FTRL is the same as FTL where the first function is the regularizer.

– Let us then apply Lemma 7 to the sequence of functions $\psi, f_1, \ldots, f_T$ (when applying the theorem, note that the indices are shifted by 1). This results in the bound:

$$[\psi(w_0) - \psi(u)] + \sum_{t=1}^{T} [f_t(w_t) - f_t(u)] \leq [\psi(w_0) - \psi(w_1)] + \sum_{t=1}^{T} [f_t(w_t) - f_t(w_{t+1})]. \tag{505}$$

– Canceling $\psi(w_0)$, noting that $\psi(w_1) \geq 0$, and rearranging, we get:

$$\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)] \leq \frac{1}{2\eta}\|u\|_2^2 + \sum_{t=1}^{T} [f_t(w_t) - f_t(w_{t+1})]. \tag{506}$$

– Now let's bound one term of the RHS sum:

$$\begin{aligned}
f_t(w_t) - f_t(w_{t+1}) &= z_t \cdot (w_t - w_{t+1}) \quad [\text{since } f_t(w) = w \cdot z_t] && (507)\\
&\leq \|z_t\|_2 \|w_t - w_{t+1}\|_2 \quad [\text{Cauchy-Schwartz}] && (508)\\
&= \|z_t\|_2 \|\Pi_S(\eta\theta_t) - \Pi_S(\eta\theta_{t+1})\|_2 \quad [\text{since } w_t = \Pi_S(\eta\theta_t)] && (509)\\
&\leq \|z_t\|_2 \|\eta\theta_t - \eta\theta_{t+1}\|_2 \quad [\text{projection decreases distance}] && (510)\\
&= \eta\|z_t\|_2^2 \quad [\text{since } \theta_{t+1} = \theta_t - z_t]. && (511)
\end{aligned}$$

Plugging this bound back into (506) completes the proof.

5.6 Online subgradient descent (OGD) (Lecture 14)

- So far, we have proven regret bounds for FTRL with linear loss functions. However, many loss functions we care about in practice (e.g., squared loss, hinge loss) are not linear.

- Even if did derive a regret for FTRL with more general losses, there would still be a computational problem: FTRL (see (494)) requires minimizing over all the loss functions seen so far, which in general is computationally impractical, especially for an online learning algorithm. For linear loss functions, on the other hand, we could optimize w_t easily by maintaining θ_t as a "sufficient statistics" (by linearity).

- Our strategy to handling general losses efficiently is by appealing to the linear machinery we already have. The key idea is to *run FTRL on a linear approximation of* f_t.
- What linear approximation $w \mapsto w \cdot z_t$ should we use? Let's use the subgradient of f_t at the current weights w_t: take any $z_t \in \partial f_t(w_t)$. Just to highlight the simplicity of the algorithm, here it is:
- **Algorithm 4 (Online subgradient descent (OGD))**
  - Let $w_1 = 0$.
  - For iteration $t = 1, \dots, T$:
    - Predict w_t and receive f_t.
    - Take any subgradient $z_t \in \partial f_t(w_t)$.
    - If $S = \mathbb{R}^d$, perform the update:

$$w_{t+1} = w_t - \eta z_t. \tag{512}$$

    - If $S \subseteq \mathbb{R}^d$, project cumulative gradients onto S:

$$w_{t+1} = \Pi_S(\eta\theta_{t+1}), \quad \theta_{t+1} = \theta_t - z_t. \tag{513}$$

- To emphasize: OGD on f_t is nothing more than FTRL on quadratic regularizers and linear subgradient approximations of f_t.
- Analyzing regret
  - From our earlier analysis of FTRL (Theorem 30), we already have a bound on the regret on the linearized losses:

$$\sum_{t=1}^{T} [w_t \cdot z_t - u \cdot z_t]. \tag{514}$$

  - We are interested on controlling the actual regret with respect to f_t:

$$\sum_{t=1}^{T} [f_t(w_t) - f(u)]. \tag{515}$$

  - How do we relate these two? Here's where *convexity* comes in a crucial way.
  - Since $z_t \in \partial f_t(w_t)$ is a subgradient, we have by the definition of subgradient:

$$f_t(u) \geq f_t(w_t) + z_t \cdot (u - w_t). \tag{516}$$

    Rearranging, we get a direct comparison for each term of the regret:

$$\boxed{f_t(w_t) - f_t(u) \leq (w_t \cdot z_t) - (u \cdot z_t).} \tag{517}$$

- The upshot is that the bounds we got for linear losses apply without modification to general losses! Intuitively, linear functions are the hardest to optimize using online convex optimization.
- FIGURE: [draw convex f_t with linearized]

- Remarks:
  - OGD works for any convex loss function f_t (so does FTRL, but we only analyzed it for linear losses).
  - OGD is in some sense the first practical, non-toy algorithm we've developed.
  - Gradient-based methods are most commonly thought of as a procedure for optimizing a global objective, but this analysis provides a different perspective: that of doing full minimization of linearized losses with quadratic regularization.
  - The minimization viewpoint opens way to many other algorithms, all of which can be thought of as using different regularizers or using better approximations of the loss functions, while maintaining efficient parameter updates.

- **Example 30 (Online SVM)**
  - Let us use our result on OGD to derive a regret bound for learning SVMs in an online manner.
  - We will just apply OGD on the hinge loss for classification ($x_t \in \mathbb{R}^d, y_t \in \{+1, -1\}$):

$$f_t(w) = \max\{0, 1 - y_t(w \cdot x_t)\}. \tag{518}$$

    The algorithm (assume $S = \mathbb{R}^d$, so we don't need to project):
    - If $y_t(w_t \cdot x_t) \geq 1$ (classify correctly with margin 1): do nothing.[17]
    - Else: $w_{t+1} = w_t + \eta y_t x_t$.
  - Analysis:
    - Assume the data points are bounded: $\|x_t\|_2 \leq L$. Then $z_t \in \partial f_t(w_t)$ also satisfies that bound $\|z_t\|_2 \leq L$.
    - Assume that expert weights are bounded: $\|u\|_2 \leq B$.
    - The regret bound from Theorem 30 is as follows:

$$\boxed{\text{Regret} \leq BL\sqrt{T}.} \tag{519}$$

- **Example 31 (Learning with expert advice)**

[17] This algorithm is very similar to the Perceptron algorithm; the only difference is that Perceptron just requires any positive margin, not necessarily 1.

- Now let us consider learning with expert advice.
  - We maintain a distribution $w_t \in \Delta_d$ over d experts and predict by sampling an expert from that distribution.
  - Assume the zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$.
  - The loss function is linear: $f_t(w_t) = w_t \cdot z_t$, where

$$z_t = [\ell(y_t, h_1(x_t)), \cdots, \ell(y_t, h_d(x_t))] \in \{0,1\}^d \tag{520}$$

    is the loss vector.
- Bound on set of experts (B): the experts live in the simplex $S = \Delta_d$, which has its 2-norm bounded by $B = 1$ (attained at a corner).
- Bound on loss gradient (L):
  - The Lipschitz constant is bounded by the norm of the gradient z_t, which is at most $\sqrt{d}$.
  - Therefore, the regret bound we get is

$$\boxed{\text{Regret} \leq BL\sqrt{T} = \sqrt{dT}.} \tag{521}$$

- Note that we are depending on the square root of the number of experts d rather than the log of the number of experts in our first online learning bound for learning with expert advice. Can we obtain a $\log d$ dependence without assuming realizability? We will find out in the next section.

5.7 Online mirror descent (OMD) (Lecture 14)

So far, we have analyzed FTRL for quadratic regularizers, which leads to (lazy projected) gradient-based algorithms. Quadratic regularization is imposing a certain prior knowledge, namely that there is a good parameter vector w in a small L_2 ball. But perhaps we know some dimensions to be more important than others. Then we might want to use a non-spherical regularizer. Or in the case of learning with expert advice, we know that $w \in \Delta_d$ (a probability distribution), so negative entropy might be more appropriate. In this section, we will develop a general way of obtaining regret bounds for general regularizers, and make explicit the glorious role that strong convexity plays. We make make extensive use of **Fenchel duality** and **Bregman divergences**.

- The goal for this lecture is to analyze FTRL (Algorithm 3) for arbitrary convex loss functions and regularizers. The resulting algorithm is this:
- **Algorithm 5 (online mirror descent (OMD))**
  - Let $\psi : \mathbb{R}^d \to \mathbb{R}$ be the regularizer (this defines the learner).
  - Let $f_1, \ldots, f_T$ be the sequence of loss functions played by nature.

- On each iteration $t = 1, \ldots, T$, the learner chooses weights w_t to minimize the regularized (linearized) loss:

$$w_t \in \arg\min_{w \in \mathbb{R}^d} \{\psi(w) - w \cdot \theta_t\}, \tag{522}$$

  where $z_t \in \partial f_t(w_t)$ is the t-th subgradient, and $\theta_t = -\sum_{i=1}^{t-1} z_i$ is the negative sum of the first $t-1$ subgradients.

- Technical note: we will let the domain of weight vectors be unconstrained ($S = \mathbb{R}^d$). We can always fold a constraint into the regularizer by setting $\psi(w) = \infty$ if w violates the constraint.

- To recap the terminology:
  - OMD on f_t is equivalent to FTRL on the linearizations $w \mapsto w \cdot z_t$.
  - OGD is OMD with the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$

- Examples of regularizers:
  - Quadratic regularizer: $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$.
  - Non-spherical quadratic regularizer: $\psi(w) = \frac{1}{2\eta} w^\top A w$ for $A \succeq 0$.
  - Entropic regularizer: $\psi(w) = \frac{1}{\eta}\sum_{j=1}^d w_j \log w_j$ if $w \in \Delta_d$ (this is the negative entropy defined on the probability simplex), and ∞ otherwise.
  - Note: the difference between the two regularizers is that the entropic regularizer slopes up violently when w approaches the boundary of the simplex Δ_d (the function value is finite but the gradient goes to infinity).

- We now need to build up some tools to help us analyze OMD. First, we introduce Fenchel duality, an important tool in optimization:
  - **Definition 27 (Fenchel conjugate)**
    - The **Fenchel conjugate**[18] of a function (not necessarily convex) $\psi : \mathbb{R}^d \to \mathbb{R}$ is

$$\boxed{\psi^*(\theta) \stackrel{\text{def}}{=} \sup_{w \in \mathbb{R}^d} \{w \cdot \theta - \psi(w)\}.} \tag{523}$$

  - Intuition
    - For scalars $w, \theta \in \mathbb{R}$, given a fixed θ (interpreted as a slope), $-\psi^*(\theta)$ is the position where the supporting hyperplane of ψ with slope θ hits the vertical axis.

[18] Also known as the convex conjugate or Legendre-Fenchel transformation.

· FIGURE: [draw ψ]

* One can think of sweeping θ and reading out $\psi^*(\theta)$; this information in some sense encodes the epigraph of ψ if ψ is convex.

– Useful facts:

* ψ^* is always convex (even if ψ is not), since it's just a supremum over a collection of linear functions $\theta \mapsto w \cdot \theta - \psi(w)$.
* $\psi^*(\theta) \geq w \cdot \theta - \psi(w)$ for all $w \in \mathbb{R}^d$ (**Fenchel-Young inequality**). This follows directly from the definition of ψ^*.
* If $r(w) = a\psi(w)$ with $a > 0$, then $r^*(\theta) = a\psi^*(\theta/a)$. This is useful because once we've computed the Fenchel conjugate for one function, we know it for different scalings. In fact, Fenchel duality has a very nice algebra that makes computing Fenchel conjugates modular.
* $\psi^{**} = \psi$ iff ψ is convex (and lower semi-continuous). In this case, we have

$$\psi(w) = \psi^{**}(w) = \sup_{\theta \in \mathbb{R}^d} \{w \cdot \theta - \psi^*(\theta)\}. \tag{524}$$

* If ψ is differentiable, then

$$\nabla \psi^*(\theta) = \arg\max_{w \in \mathbb{R}^d} \{w \cdot \theta - \psi(w)\}. \tag{525}$$

This is because the gradient of the supremum of a collection of functions at θ is the gradient of the function that attains the max at θ.

– Mirroring

* Comparing this with the the OMD update (522), we see that $w_t = \nabla \psi^*(\theta_t)$, and $-\psi^*(\theta_t)$ is the corresponding value of the regularized loss.
* Since w_t attains the supremum of the Fenchel conjugate ψ^*, the optimality conditions (differentiate with respect to w) tell us that $\theta_t = \nabla \psi(w_t)$.
* We have a one-to-one mapping between weights w and negative cumulative subgradients θ, linked via the gradients of ψ and ψ^*, which are inverses of one another:

$$\boxed{w_t = \nabla \psi^*(\theta_t), \quad \theta_t = \nabla \psi(w_t).} \tag{526}$$

* This provides a very elegant view of what online mirror descent is doing. OMD takes a series of gradient updates $\theta_{t+1} = \theta_t - z_t$, generating $\theta_1, \theta_2, \ldots, \theta_T$. These steps are *mirrored* via Fenchel duality in the sequence $w_1, w_2, \ldots, w_T$.
* FIGURE: [mapping]

– **Example 32 (Quadratic regularizer)**

* Let $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$.
* Then $\psi^*(\theta) = \frac{\eta}{2}\|\theta\|_2^2$, attained by $w = \nabla \psi^*(\theta) = \eta\theta$.

* In this case, w and θ are simple rescalings of each other.

– **Example 33 (Entropic regularizer)**

* Let $\psi(w) = \frac{1}{\eta} \sum_{j=1}^{d} w_j \log w_j$ for $w \in \Delta_d$ (negative entropy).
* Then $\psi^*(\theta) = \frac{1}{\eta} \log \left(\sum_{j=1}^{d} e^{\eta \theta_j} \right)$ (log partition function), attained by $w_j = \nabla \psi^*(\theta) = \frac{e^{\eta \theta_j}}{\sum_{k=1}^{d} e^{\eta \theta_k}}$.
* Aside: this is maximum entropy duality in exponential families, where w and θ represent the mean and canonical parametrization of a multinomial distribution.

[begin lecture 15] (15)

5.8 Regret bounds with Bregman divergences (Lecture 15)

- Motivation
  - Having reinterpreted OMD as a mapping using a conjugate function, let's turn to proving a regret bound. Recall that in order to prove bounds, we needed to ensure that w_t and w_{t+1} don't change too much according to some criteria.
  - For quadratic regularization, this criteria was based on Euclidean distance.
  - We now generalize this by using Bregman divergences, which generalizes the notion of distance based on the regularizer.
  - **Definition 28 (Bregman divergence)**
    - Let f be a continuously-differentiable convex function.
    - The **Bregman divergence** between w and u is the difference at w between f and a linear approximation around u:

$$\boxed{D_f(w\|u) \stackrel{\text{def}}{=} f(w) - f(u) - \nabla f(u) \cdot (w - u).} \tag{527}$$

    - Intuitively, the divergence captures the the error of the linear approximation of f based on the gradient $\nabla f(u)$.
    - FIGURE: [show gap between f and its linear approximation]
  - Property: $D_f(w\|u) \geq 0$ (by definition of convexity).
  - Note: Bregman divergences are not symmetric and therefore not a distance metric.
  - **Example 34 (Quadratic regularizer)**
    - Let $f(w) = \frac{1}{2}\|w\|_2^2$.
    - Then $D_f(w\|u) = \frac{1}{2}\|w - u\|_2^2$ (squared Euclidean distance).
  - **Example 35 (Entropic regularizer)**
    - Let $f(w) = \sum_{j=1}^d w_j \log w_j$ for $w \in \Delta_d$ (negative entropy).
    - Then $D_f(w\|u) = \mathrm{KL}(w\|u) = \sum_j w_j \log(w_j/u_j)$ for $w \in \Delta_d$ (KL divergence).
  - Property (scaling): $D_{af}(w\|u) = aD_f(w\|u)$.
- **Theorem 31 (regret of OMD using Bregman divergences)**
  - OMD (Algorithm 5) obtains the following regret bound:

$$\boxed{\mathrm{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \sum_{t=1}^T D_{\psi^*}(\theta_{t+1}\|\theta_t).} \tag{528}$$

- Furthermore, if all f_t's are linear, the inequality is actually an equality if u is the best expert. So this bound is pretty air tight.

- Proof of Theorem 31:
  - We assume all the loss functions are linear; recall in our analysis of OGD that linear functions are the worst case.
  - The key step is to find a potential function which allows us to monitor progress. The pivoting quantity is $\psi^*(\theta_{T+1})$, the negative regularized loss of the best fixed expert. This will allow us to relate the learner (w_t) to the expert (u).
  - Recall:
    - Learner's loss is $\sum_{t=1}^T w_t \cdot z_t$.
    - Expert's loss is $\sum_{t=1}^T u \cdot z_t$.
    - The regret is the difference between the two.
  - The expert:

$$\psi^*(\theta_{T+1}) \geq u \cdot \theta_{T+1} - \psi(u) \tag{529}$$

$$= \sum_{t=1}^T [-u \cdot z_t] - \psi(u) \tag{530}$$

    by the Fenchel-Young inequality. Note that we have equality if u is the best expert, by definition of ψ^*.
  - The learner:

$$\psi^*(\theta_{T+1}) = \psi^*(\theta_1) + \sum_{t=1}^T [\psi^*(\theta_{t+1}) - \psi^*(\theta_t)] \quad \text{[telescoping sums]} \tag{531}$$

$$= \psi^*(\theta_1) + \sum_{t=1}^T [\nabla \psi^*(\theta_t) \cdot (\theta_{t+1} - \theta_t) + D_{\psi^*}(\theta_{t+1} \| \theta_t)] \quad \text{[Bregman definition]} \tag{532}$$

$$= \psi^*(\theta_1) + \sum_{t=1}^T [-w_t \cdot z_t + D_{\psi^*}(\theta_{t+1} \| \theta_t)] \quad \text{[definition of OMD (526) and } \theta_t\text{]}. \tag{533}$$

    Note that $\psi^*(\theta_1) = -\psi(w_1)$ since $\theta_1 = 0$.
  - Combining last equations for the expert and learner and rearranging yields the result.
  - The regret bound (528) is a generalization of (501), where $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ is quadratic regularizer, and $D_{\psi^*}(\theta_{t+1} \| \theta_t) = \frac{\eta}{2}\|z_t\|_2^2$.
  - However, a general Bregman divergence usually doesn't have such a nice form, and thus it is useful to bound it using a nicer quantity: a norm of some kind.

5.9 Strong convexity and smoothness (Lecture 15)

- First, let's review some intuition behind norms.
  - L_p norms decrease as p increases:

$$\|w\|_1 \geq \|w\|_2 \geq \|w\|_\infty. \tag{534}$$

  - The difference between the norms can be huge. For example, take $w = (1, \ldots, 1) \in \mathbb{R}^d$. Then $\|w\|_1 = d$, $\|w\|_2 = \sqrt{d}$, $\|w\|_\infty = 1$.
  - Recall that the dual norm of a norm $\|\cdot\|$ is

$$\|x\|_* = \sup_{\|y\| \leq 1} (x \cdot y). \tag{535}$$

    Thinking of y as a linear operator on x, the dual norm measures the gain when we measure perturbations in x using $\|\cdot\|$.
  - The norms $\|\cdot\|_p$ and $\|\cdot\|_q$ are dual to each other when $\frac{1}{p} + \frac{1}{q} = 1$. Two common examples are:

$$\|\cdot\|_1 \quad \text{is dual to} \quad \|\cdot\|_\infty. \tag{536}$$

$$\|\cdot\|_2 \quad \text{is dual to} \quad \|\cdot\|_2. \tag{537}$$

- We will now use these squared norms to control Bregman divergences provided the Bregman divergences have an important special structure called *strong convexity*:
- **Definition 29 (strong convexity/smoothness)**
  - A function f is α-strongly convex with respect to a norm $\|\cdot\|$ iff for all w, u:

$$D_f(w\|u) \geq \frac{\alpha}{2}\|w - u\|^2. \tag{538}$$

  - A function f is α-strongly smooth with respect to a norm $\|\cdot\|$ iff for all w, u:

$$D_f(w\|u) \leq \frac{\alpha}{2}\|w - u\|^2. \tag{539}$$

- Intuitions
  - Strong convexity means that f must be growing at least quadratically.
  - Strong smoothness means that f must be growing slower than some quadratic function.
  - Example: the quadratic regularizer $\psi(w) = \frac{1}{2}\|w\|_2^2$ is both 1-strongly convex and 1-strongly smooth with respect to the L_2 norm, since $D_\psi(w\|u) = \frac{1}{2}\|w - u\|_2^2$.

- Duality links strong convexity and strong smoothness, as the following result shows.
- **Lemma 8 (strong convexity and strong smoothness)**
  - The following two statements are equivalent:
    - $\psi(w)$ is $1/\eta$-strongly convex with respect to a norm $\|\cdot\|$.
    - $\psi^*(\theta)$ is η-strongly smooth with respect to the dual norm $\|\cdot\|_*$.
  - Sanity check the quadratic regularizer: $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ and $\psi^*(\theta) = \frac{\eta}{2}\|\theta\|_2^2$.
- With these tools, we can finally make some progress on our regret bound from Theorem 31, rewriting the Bregman divergences in terms of norms.
- **Theorem 32 (regret of OMD using norms)**
  - Suppose ψ is a $\frac{1}{\eta}$-strongly convex regularizer.
  - Then the regret of online mirror descent is

$$\boxed{\text{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \frac{\eta}{2}\sum_{t=1}^{T}\|z_t\|_*^2.} \tag{540}$$

- Proof of Theorem 32
  - By Lemma 8, since ψ is $1/\eta$-strongly convex, the Fenchel conjugate ψ^* is η-strongly smooth.
  - By definition of strong smoothness and the fact that $\theta_{t+1} = \theta_t - z_t$,

$$\boxed{D_{\psi^*}(\theta_{t+1}\|\theta_t) \leq \frac{\eta}{2}\|z_t\|_*^2.} \tag{541}$$

  - Plugging this bound into (528) gives us the result.
- Remarks:
  - If we plug in the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$, then we get back the original result (501).
  - However, (540) generalizes to other regularizers.
  - We get to measure the size of z_t using the dual norm $\|\cdot\|_*$ rather than the default L_2 norm, which will help us get tighter bounds for the learning from expert advice problem.
- Learning form expert advice
  - Let's now use our tools to improve the regret bound that we got for learning from expert advice.

- Recall that when we used the quadratic regularizer, we got a regret bound of $\sqrt{dT}$ because $\|z_t\|_2$ could be as big as $\sqrt{d}$.
- To reduce this, let's just use another norm: $\|z_t\|_\infty \leq 1$; no dependence on d!
- But this means that the regularizer ψ has to be strongly convex with respect to the L_1 norm, but this is harder because $\|\cdot\|_1 \geq \|\cdot\|_2$.
- Failure: the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ is only $\frac{1}{\eta d}$-strongly convex with respect to the L_1 norm:

$$D_\psi(w\|u) \geq \frac{1}{2\eta}\|w-u\|_2^2 \geq \frac{1}{2\eta d}\|w-u\|_1^2. \tag{542}$$

  Sanity check: $\|(1,\ldots,1)\|_2^2 = d$ but $\|(1,\ldots,1)\|_1^2 = d^2$. So if we try to use this in the regret bound, we get $\frac{1}{2\eta} + \frac{T\eta d}{2}$, which is still $\sqrt{dT}$ (for optimal η).
- Failure: the L_1 regularizer $\psi(w) = \|w\|_1$ is neither strongly convex nor strongly smooth with respect to the any norm for any α: it doesn't grow fast enough for large w and grows too fast for small w.
- Success: entropic regularizer $\psi(w) = \frac{1}{\eta}\sum_j w_j \log w_j$ for $w \in \Delta_d$ is $1/\eta$-strongly convex with respect to the L_1 norm. This requires some algebra and an application of Cauchy-Schwartz (see Example 2.5 in Shai Shalev-Shwartz's online learning tutorial).
- FIGURE: [compare quadratic and entropic regularizer for $d = 2$]

- **Example 36 (exponentiated gradient (EG))**
  - Entropic regularizer: $\psi(w) = \frac{1}{\eta}\sum_{j=1}^d w_j \log w_j$ for $w \in \Delta_d$.
  - Recall $\psi^*(\theta) = \frac{1}{\eta}\log\sum_{j=1}^d e^{\eta\theta_j}$ and $\nabla\psi_j^*(\theta) = \frac{e^{\eta\theta_j}}{\sum_{k=1}^d e^{\eta\theta_k}}$.
  - OMD updates:

$$\boxed{w_{t,j} \propto e^{\eta\theta_{t,j}}.} \tag{543}$$

    The equivalent recursive formula:

$$\boxed{w_{t+1,j} \propto w_{t,j}e^{-\eta z_{t,j}}.} \tag{544}$$

    Interpretation: we maintain a distribution over the d experts. We use the gradient z_t to reweight the experts, normalizing afterwards to ensure a proper distribution.
  - Recap: the exponentiated gradient (EG) algorithm is just online mirror descent using the entropic regularizer.

- **Example 37 (EG for learning with expert advice)**

- Consider learning with expert advice (Example 27).
- We use the expected zero-one loss: $f_t(w) = w \cdot z_t$, where z_t is the loss vector.
- We have that the dual norm of the gradients are bounded $\|z_t\|_\infty \leq 1$.
- The minimum and maximum values of the regularizer:
  - $\max_w \psi(w) = 0$ (minimum entropy)
  - $\min_w \psi(w) = \frac{\log(1/d)}{\eta} = \frac{-\log d}{\eta}$ (maximum entropy)
- Then

$$\text{Regret} = \frac{\log(d)}{\eta} + \frac{\eta T}{2}. \tag{545}$$

- Setting $\eta = \sqrt{\frac{2\log d}{T}}$ yields:

$$\boxed{\text{Regret} = \sqrt{2\log(d)T}.} \tag{546}$$

- To compare with quadratic regularization (OGD):
  - Quadratic: $[\max_u \psi(u) - \min_u \psi(u)] \leq \frac{1}{2\eta}$, $\|z_t\|_2 \leq \sqrt{d}$
  - Entropic: $[\max_u \psi(u) - \min_u \psi(u)] \leq \frac{\log d}{\eta}$, $\|z_t\|_\infty \leq 1$
- Discussion
  - Online mirror descent (OMD) allows us to use different regularizers based on our prior knowledge about the expert vector u and the data z_t. As we see with EG, tailoring the regularizer can lead to better bounds.
  - Using the L_2 norm means that we use the bound $\|z_t\|_2 \leq \sqrt{d}$. To get rid of the dependence on d here, we use the L_∞ norm with $\|z_t\|_\infty \leq 1$.
  - However, this means that ψ must be strongly convex with respect to the L_1 norm. The quadratic regularizer isn't strong enough (only $\frac{1}{\eta d}$-strongly convex with respect to L_2), so we need the entropic regularizer, which is 1-strongly convex with respect to L_1.
  - Curvature hurts us because $[\psi(u) - \psi(w_1)]$ is now larger, but the simplex is small, so $[\psi(u) - \psi(w_1)]$ only grows from 1 (with the quadratic regularizer) to $\log d$ (with the entropic regularizer).
  - So the tradeoff was definitely to our advantage.

5.10 Local norms (Lecture 15)

- Recall that that the general online mirror descent (OMD) analysis (Theorem 31) yields:

$$\text{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \sum_{t=1}^{T} D_{\psi^*}(\theta_{t+1} \| \theta_t). \tag{547}$$

  Using the fact that ψ is $1/\eta$-strongly convex with respect to some norm $\|\cdot\|$, we can upper bound the Bregman divergence by the following (540):

$$D_{\psi^*}(\theta_{t+1} \| \theta_t) \leq \frac{\eta}{2} \|z_t\|_*^2. \tag{548}$$

- If we use the entropic regularizer for ψ with norm $\|\cdot\| = \|\cdot\|_1$ and dual norm $\|\cdot\|_* = \|\cdot\|_\infty$, we get our key $\sqrt{2\log(d)T}$ regret bound for EG.
- In this section, we will use the local norms technique to improve (548). This will allow us to do two things:
  - Recover the strong $O(\log d)$ bound for the realizable setting.
  - Allow us to analyze the multi-armed bandit setting.
- Why we should do better:
  - Consider an example where there are two experts: one which is perfect ($z_{t,1} = 0$ for all t) and one which is horrible ($z_{t,2} = 1$ for all t).
  - The EG algorithm will quickly downweight the bad expert exponentially fast:

$$w_{t,1} \propto 1 \qquad w_{t,2} \propto e^{-\eta(t-1)}. \tag{549}$$

  - So as $t \to \infty$, we have basically put all our weight on the first expert, and should be suffering no loss.
  - But $\|z_t\|_\infty^2 = 1$, so in the regret bound we still pay $\frac{\eta T}{2}$, which is simply a travesty.
  - We would hope that $\|z_t\|_\infty^2 = \max_{1 \leq j \leq d} z_{t,j}^2$ should be replaced with something that is sensitive to how much weight we're placing on it, which is $w_{t,j}$. Indeed the following theorem fulfills this dream:
- **Theorem 33 (exponentiated gradient (analysis using local norms))**
  - Assume nature plays a sequence of linear loss functions $f_t(w) = w \cdot z_t$, where $z_{t,j} \geq 0$ for all $t = 1, \ldots, T$ and $j = 1, \ldots, d$.
  - Then the exponentiated gradient (EG) algorithm (Example 37) achieves the following regret bound:

$$\boxed{\text{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \eta \sum_{t=1}^{T} \sum_{j=1}^{d} w_{t,j} z_{t,j}^2.} \tag{550}$$

- This bound (550) is better than the bound in Theorem 32 because we are taking an average (with respect to the distribution $w_t \in \Delta_d$) instead of a max:

$$\sum_{j=1}^{d} w_{t,j} z_{t,j}^2 \leq \max_{1 \leq j \leq d} z_{t,j}^2 \stackrel{\text{def}}{=} \|z_t\|_\infty^2. \tag{551}$$

  This allows some components of the loss subgradients $z_{t,j}$ to be large provided that the corresponding weights $w_{t,j}$ are small.

- **Example 38 (exponentiated gradient in the realizable setting)**
  - Assume the loss vector is bounded: $z_t \in [0,1]^d$.
  - Assume there exists an expert $u \in \Delta_d$ with zero cumulative loss (realizability).
  - Recall the regret bound from using local norms:

$$\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} (w_t \cdot z_t) - \underbrace{\sum_{t=1}^{T} (u \cdot z_t)}_{=0} \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \underbrace{\sum_{j=1}^{d} w_{t,j} z_{t,j}^2}_{\leq w_t \cdot z_t} \tag{552}$$

    .
  - Note that $\sum_{j=1}^{d} w_{t,j} z_{t,j}^2 \leq w_t \cdot z_t$, because all quantities are non-negative and $a^2 \leq a$ for $a \in [0,1]$.
  - Rearranging, we get:

$$\text{Regret}(u) \leq \frac{\log d}{\eta(1-\eta)}. \tag{553}$$

  - Minimize the bound with $\eta = \frac{1}{2}$ to obtain:

$$\boxed{\text{Regret}(u) \leq 4 \log d.} \tag{554}$$

  - Recall that the majority algorithm (which aggressively zeros out the weights of components as soon as they err) also obtained the very low $O(\log d)$ regret (see Example 25), so it's really nice to see that EG obtains the same regret guarantee.
  - If the problem isn't realizable, the majority algorithm isn't even correct (it will eliminate all the experts), whereas EG will gracefully fall back to $O(\sqrt{\log(d) T})$ regret.

- Now that you are hopefully somewhat convinced that the theorem is useful, let's prove it.

- Proof of Theorem 33:

– This proof mostly involves starting with the Bregman divergence, and performing some low-level algebraic manipulation. Perhaps the most useful high-level take-away is whenever we're trying to massage some expression with log's and exp's, it's useful to try to approximate the functions using linear and quadratic approximations (think Taylor approximations).

– Specifically, we will use the following two facts:

  * Fact 1: $e^{-a} \leq 1 - a + a^2$ for $a \geq 0$ (this actually holds for smaller a, but let's keep it simple)
  * Fact 2: $\log(1-a) \leq -a$

– Recall the Fenchel conjugate of the entropic regularizer is the log-partition function:

$$\psi^*(\theta) = \frac{1}{\eta} \log \sum_{j=1}^{d} e^{\eta \theta_j}. \tag{555}$$

– By the definition of Bregman divergences (this was used in the proof of Theorem 31), we have:

$$D_{\psi^*}(\theta_{t+1} \| \theta_t) = \psi^*(\theta_{t+1}) - \psi^*(\theta_t) - \underbrace{\nabla \psi^*(\theta_t)}_{w_t} \cdot \underbrace{(\theta_{t+1} - \theta_t)}_{-z_t}. \tag{556}$$

– The rest is just applying the two facts and watching stuff cancel:

$$\begin{aligned}
D_{\psi^*}(\theta_{t+1} \| \theta_t) &= \psi^*(\theta_{t+1}) - \psi^*(\theta_t) + w_t \cdot z_t && (557)\\
&= \frac{1}{\eta} \log \left(\frac{\sum_{j=1}^{d} e^{\eta \theta_{t+1,j}}}{\sum_{j=1}^{d} e^{\eta \theta_{t,j}}} \right) + w_t \cdot z_t \quad \text{[definition of } \psi^*\text{]} && (558)\\
&= \frac{1}{\eta} \log \left(\sum_{j=1}^{d} w_{t,j} e^{-\eta z_{t,j}} \right) + w_t \cdot z_t \quad \text{[definition of } \theta_t\text{]} && (559)\\
&\leq \frac{1}{\eta} \log \left(\sum_{j=1}^{d} w_{t,j} [1 - (\eta z_{t,j} - \eta^2 z_{t,j}^2)] \right) + w_t \cdot z_t \quad \text{[fact 1]} && (560)\\
&= \frac{1}{\eta} \log \left(1 - \sum_{j=1}^{d} w_{t,j} (\eta z_{t,j} - \eta^2 z_{t,j}^2) \right) + w_t \cdot z_t \quad [w_t \in \Delta_d] && (561)\\
&\leq \frac{1}{\eta} \sum_{j=1}^{d} w_{t,j} (-\eta z_{t,j} + \eta^2 z_{t,j}^2) + w_t \cdot z_t \quad \text{[fact 2]} && (562)\\
&= \eta \sum_{j=1}^{d} w_{t,j} z_{t,j}^2 \quad \text{[algebra]}. && (563)
\end{aligned}$$

[begin lecture 16] (16)

5.11 Adaptive optimistic mirror descent (Lecture 16)

- Motivation
  - Recall that the local norm bound technique for exponentiated gradient (EG) yields the following bound (550):

$$\text{Regret}(u) \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \sum_{j=1}^{d} w_{t,j} z_{t,j}^2, \tag{564}$$

    improving the $\|z_t\|_\infty^2$ from the standard mirror descent analysis (Example 37) to the local norm $\|z_t\|_{\text{diag}(w_t)}^2$. (Note that this was just an improvement in the analysis, not the algorithm.)
  - But there are two weaknesses of this regret bound. To see the first weakness, consider the following example.
  - **Example 39 (perfect and confused expert)**
    * We have two experts:
      · Expert 1 is perfect ($z_{t,1} = 0$).
      · Expert 2 is just confused and alternates between loss of -1 and $+1$ ($z_{t,2} = (-1)^{t-1}$).
    * Playing either expert 1 or expert 2 alone is optimal, yielding a cumulative loss of 0 (for expert 2, for even T). But expert 2 has higher variance and is thus intuitively riskier, so maybe we should avoid it.
    * However, the EG algorithm does not:

$$w_t \propto \begin{cases} [1, 1] & \text{if } t \text{ is odd} \\ [1, \exp(-\eta)] & \text{if } t \text{ is even.} \end{cases} \tag{565}$$

      EG doesn't penalize expert 2 on the odd rounds at all and barely penalizes it on the even rounds, because expert 2 redeems itself on the even rounds.
    * On this example, assuming the step size $\eta \leq 1$, EG incurs a regret (not just

a regret bound) of:

$$\text{EG regret} = \frac{T}{2}\left(\frac{1}{2}(1) + \frac{\exp(-\eta)}{1+\exp(-\eta)}(-1)\right) \tag{566}$$

$$= \frac{T}{2}\left(\frac{1}{2} - \frac{1}{1+\exp(\eta)}\right) \tag{567}$$

$$\geq \frac{T}{2}\left(\frac{1}{2} - \frac{1}{2+2\eta}\right) \quad [\exp(\eta) \leq 1+2\eta \text{ for } \eta \leq 1] \tag{568}$$

$$\geq \frac{T}{2}\eta. \tag{569}$$

  - Typically, $\eta = \Theta(\frac{1}{\sqrt{T}})$, in which case, the regret is $\Omega(T)$.
  - If η is too large, then EG ends up swinging back and forth between the two experts, and incurring a lot of regret, just as in the FTL example with linear loss functions (Example 29). On the other hand, η can't be too small either, or else we don't end up being able to switch to a good expert quickly enough (consider the example where expert 2 gets a loss of 1 always).

- The second weakness is that if we add 5 to the losses of all the experts on all T iterations, then the regret bound suffers something like an extra $25\eta T$. But morally, the problem hasn't changed!
- In what follows, we will produce two improvements to the EG algorithm. There are two key ideas:
  - **Adaptivity**: instead of using a fixed regularizer ψ, we will use a regularizer ψ_t that depends on the current iteration. This way, we can adapt the regularizer to the loss functions that we've seen so far.
  - **Optimism**: we will incorporate hints $m_1, \dots, m_T$, which are a guess of the actual loss sequence $z_1, \dots, z_T$. By incorporating these hints, our algorithm can do much better when the hints are correct, but still be robust enough to perform well when the hints are not correct. As a concrete example, think of $m_t = z_{t-1}$ (a recency bias), which hints that the subgradients aren't changing very much.

  Using these two ideas, we define an algorithm called **adaptive optimistic mirror descent (AOMD)**, which obtain the following regret bound for learning with expert advice (for $\eta \leq \frac{1}{4}$):

$$\text{Regret}(u) \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T}\sum_{j=1}^{d} u_j(z_{t,j} - m_{t,j})^2. \tag{570}$$

  Compared to (564), (570) depends on the expert u_j rather than the learner $w_{t,j}$; and it also depends on the squared errors $(z_{t,j} - m_{t,j})^2$ rather than $z_{t,j}^2$. Unlike local norms, this requires an actual change to the algorithm—a change which is inspired by the theory.

- Recall that the normal mirror descent updates are given as follows:

$$w_t = \arg\min_{w \in \mathbb{R}^d} \left\{ \psi(w) + \sum_{i=1}^{t-1} w \cdot z_i \right\} = \nabla \psi^*(\theta_t). \tag{571}$$

- Define the following **adaptive** mirror descent updates:

$$w_t = \arg\min_{w \in \mathbb{R}^d} \left\{ \psi_t(w) + \sum_{i=1}^{t-1} w \cdot z_i \right\} = \nabla \psi_t^*(\theta_t), \tag{572}$$

  where all we've done is replaced ψ with ψ_t.

- To put **optimism** in, suppose we have a sequence of **hints**

$$m_1, \dots, m_T, \tag{573}$$

  which are guesses of $z_1, \dots, z_T$. These have the property that m_t only depends on $z_{1:t-1}$. For example, the simplest one could be $m_t = z_{t-1}$ (recency bias), which means that we believe the subgradients do not change very much. Note that if $m_t = z_t$, then we would be cheating and using the one-step lookahead. The **adaptive optimistic mirror descent** updates are as follows:

$$\boxed{w_t = \arg\min_{w \in \mathbb{R}^d} \left\{ \psi(w) + \sum_{i=1}^{t-1} w \cdot z_i + w \cdot m_t \right\} = \nabla \psi^*(\tilde{\theta}_t),} \tag{574}$$

  where $\tilde{\theta}_t \stackrel{\text{def}}{=} \theta_t - m_t$ is the guess of θ_{t+1}.

- **Theorem 34 (regret of adaptive optimistic mirror descent)**
  - Let $m_1, \dots, m_T$ be any sequence of hints. Assume $m_1 = 0$.
  - Suppose we have a sequence of regularizers $\psi_1, \dots, \psi_T$ satisfying the following loss-bounding property:

$$\psi_{t+1}^*(\theta_{t+1}) \le \psi_t^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t). \tag{575}$$

  - Then

$$\boxed{\text{Regret}(u) \le \psi_{T+1}(u) - \psi(w_1).} \tag{576}$$

- Note: in our previous mirror descent analysis, we had two terms, one coming from the regularizer, and one coming from the stability of our estimates. In contrast, (575) "pushes the loss into the regularizer", which gives us a bit more flexibility.

- Proof of Theorem 34:

$$u \cdot \theta_{T+1} - \psi_{T+1}(u) \leq \psi^*_{T+1}(\theta_{T+1}) \quad \text{[Fenchel-Young]} \tag{577}$$

$$= \psi^*_1(\theta_1) + \sum_{t=1}^{T} [\psi^*_{t+1}(\theta_{t+1}) - \psi^*_t(\theta_t)] \quad \text{[telescoping]} \tag{578}$$

$$= \psi^*_1(\theta_1) + \sum_{t=1}^{T} [\psi^*_{t+1}(\theta_{t+1}) - \psi^*_t(\tilde{\theta}_t) - \underbrace{\nabla\psi^*(\tilde{\theta}_t)}_{w_t} \cdot m_t] \quad \text{[convexity of } \psi^*_t\text{]} \tag{579}$$

$$= \psi^*_1(\theta_1) + \sum_{t=1}^{T} -w_t \cdot z_t \quad \text{[loss-bounding property]} \tag{580}$$

$$= -\psi_1(w_1) + \sum_{t=1}^{T} -w_t \cdot z_t \quad \text{[since } m_1 = 0\text{]}. \tag{581}$$

  Recall that $\theta_{T+1} = -\sum_{t=1}^{T} z_t$. Algebra completes the proof.

- Now what kind of regularizer should we use to satisfy the loss-bounding property?
  - First attempt: let's try just using a fixed regularizer $\psi_t(w) = \psi(w)$. This means that we need

$$\psi^*(\theta_{t+1}) \leq \psi^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t). \tag{582}$$

    But in fact, since ψ^* is convex, and recalling $w_t = \nabla\psi^*(\tilde{\theta}_t)$ and $\theta_{t+1} - \tilde{\theta}_t = -(z_t - m_t)$ we actually have:

$$\psi^*(\theta_{t+1}) \geq \psi^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t), \tag{583}$$

    which is the opposite of what we want!
  - FIGURE: [graph of ψ^* $\theta_t, \tilde{\theta}_t, \theta_{t+1}$]
  - Second attempt: let's try to provide a second-order correction:

$$\psi^*(\theta_{t+1} - \eta(z_t - m_t)^2) \leq \psi^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t). \tag{584}$$

    The $\eta(z_t - m_t)^2$ term makes the LHS smaller. However, condition doesn't look like the loss-bounding precondition we need for Theorem 34. To make the condition match, the key is to *fold the second-order term into the regularizer.*
  - To that end, define the vector a_t to be the squared magnitude deviations between z_t and m_t:

$$a_t \stackrel{\text{def}}{=} (a_{t,1}, \ldots, a_{t,d}), \quad \boxed{a_{t,j} \stackrel{\text{def}}{=} (z_{t,j} - m_{t,j})^2.} \tag{585}$$

This motivates us to consider the following adaptive regularizer:

$$\boxed{\psi_t(w) \stackrel{\text{def}}{=} \psi(w) + \eta \sum_{i=1}^{t-1} w \cdot a_i.} \tag{586}$$

In the expert advice setting where w is a distribution over experts, we are penalizing experts that have large deviations from the hints, which is in line with our intuitions that we want experts which match our hints (kind of like our prior).

– Algorithmically, AOMD with this choice of ψ_t is:

$$w_t = \arg\min_w \left\{ \psi(w) + \eta \sum_{i=1}^{t-1} w \cdot a_i - w \cdot \tilde{\theta}_t \right\} \tag{587}$$

$$= \arg\min_w \left\{ \psi(w) - w \cdot \tilde{\beta}_t \right\} \tag{588}$$

$$= \nabla \psi^*(\tilde{\beta}_t), \tag{589}$$

where we define the second-corrected vectors (without and with m_t-lookahead):

$$\beta_t \stackrel{\text{def}}{=} \theta_t - \eta \sum_{i=1}^{t-1} a_i, \quad \tilde{\beta}_t \stackrel{\text{def}}{=} \beta_t - m_t. \tag{590}$$

Recall that $\tilde{\theta}_t = \theta_t - m_t$.

- **Theorem 35 (second-order corrections)**

  – Consider running AOMD using the adaptive regularizer ψ_t defined in (586).

  – Assume the second-order-corrected loss-bounding property holds:

$$\psi^*(\beta_{t+1}) \leq \psi^*(\tilde{\beta}_t) - w_t \cdot (z_t - m_t). \tag{591}$$

  – Then we have the following regret bound:

$$\boxed{\text{Regret}(u) \leq \psi(u) - \psi(w_1) + \eta \sum_{t=1}^{T} u \cdot a_t.} \tag{592}$$

- Proof of Theorem 35

  – We need to translate the second-order-corrected loss-bounding property on ψ the loss-bounding property on ψ_t.

    * Since $\psi_t(w) = \psi(w) + w \cdot (\eta \sum_{i=1}^{t-1} a_i)$ by definition, Fenchel conjugacy yields:

$$\psi_t^*(x) = \psi^* \left(x - \eta \sum_{i=1}^{t-1} a_i \right). \tag{593}$$

In other words, ψ_t^* is just ψ^* shifted by the second-order corrections.

* Therefore,

$$\psi^*_{t+1}(\theta_{t+1}) = \psi^*(\beta_{t+1}), \quad , \psi^*_t(\tilde{\theta}_t) = \psi^*(\tilde{\beta}_t). \tag{594}$$

- Note that $\beta_1 = 0$, so $\psi^*(\beta_1) = -\psi(w_1)$.
- Then:

$$\text{Regret}(u) \leq \psi_{T+1}(u) - \psi(w_1) \quad \text{[from Theorem 34]} \tag{595}$$

$$\leq \psi(u) + \eta \sum_{t=1}^{T} u \cdot a_t - \psi(w_1) \quad \text{[definition of } \psi_t\text{]}. \tag{596}$$

- Now let us apply this result to the expert advice setting.
- **Theorem 36 (adaptive optimistic gradient descent (expert advice))**
  - Consider $\|z_t\|_\infty \leq 1$ and $\|m_t\|_\infty \leq 1$.
  - Let $\psi(w) = \frac{1}{\eta}\sum_{j=1}^d w_j \log w_j$ be the standard entropic regularizer, which corresponds to the following algorithm:

$$w_{t,j} \propto \exp(\tilde{\beta}_{t,j}) = \exp\left(\eta\theta_{t,j} - \underbrace{\eta m_{t,j}}_{\text{optimism}} \underbrace{- \eta^2 \sum_{i=1}^{t-1}(z_{i,j} - m_{i,j})^2}_{\text{adaptivity}}\right). \tag{597}$$

  - Assume $0 < \eta \leq \frac{1}{4}$.
  - Then adaptive optimistic mirror descent (AOMD) obtains the following regret bound:

$$\boxed{\text{Regret}(u) \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T}\sum_{j=1}^{d} u_j (z_{t,j} - m_{t,j})^2.} \tag{598}$$

- Proof sketch: do algebra on $\psi^*(\beta_{t+1})$ to show the condition of Theorem 35; see Steinhardt and Liang (2014).
- **Example 40 (path length bound)**
  - Set $m_t = z_{t-1}$ be the average subgradients in the past.
  - Algorithmically, this corresponds to counting the last z_{t-1} twice in addition to penalizing deviations.
  - FIGURE: [plot expert paths over time]

- We obtain

$$\text{Regret} \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} (z_{t,j^*} - z_{t-1,j^*})^2, \tag{599}$$

  where $j^* \in [d]$ is best expert (the one with the lowest cumulative loss).

- On Example 39, expert 1 is a best expert (obtains cumulative loss 0). We have that the path length for expert 1 is $(z_{t,1} - z_{t-1,1})^2 = 0$, so we obtain:

$$\text{Regret} \leq \frac{\log d}{\eta}. \tag{600}$$

  The bound only holds for $\eta \leq \frac{1}{4}$. By setting $\eta = \frac{1}{4}$, we that the regret is a mere constant:

$$\text{Regret} \leq 4 \log d. \tag{601}$$

- Note that unlike EG (which had $\Omega(\sqrt{T})$ regret), we don't get distracted by expert 2, who has very low loss as well, but has longer path length and is thus discounted.

- At a high-level, what we're doing with the hints m_t is penalizing experts whose losses $z_{t,j}$ don't agree with our "prior" $m_{t,j}$. Of course, the hints have only a secondary $\Theta(\eta)$ effect, whereas the actual losses have a primary $\Theta(1)$ effect.

- This section is mostly based on Steinhardt and Liang (2014), who combines optimistic mirror descent (Rakhlin and Sridharan, 2013) and adaptive mirror descent (Orabona et al., 2015).

5.12 Online-to-batch conversion (Lecture 16)

- So far, we have been focusing on the **online** setting, where the learner receives one example at a time and is asked to make a prediction on each one. Good online learners have low **regret**.

- Sometimes it is more natural to operate in the **batch** setting, where the learner gets a set of training examples, learns a model, and then is asked to predict on new test examples. Good batch learners have low **expected risk**.

- In this section, we will show that low regret implies low expected risk by explicitly reducing the online setting to the batch setting.

- Batch setting

  - Assume we have a unknown data-generating distribution $p^*(z)$, where we use $z = (x, y) \in \mathcal{Z}$ to denote an input-output pair.

- Assume our hypothesis class is a convex set of weight vectors $S \subseteq \mathbb{R}^d$.
- As in online learning, we define a convex loss function $\ell(z, w) \in \mathbb{R}$; for example, the squared loss for linear regression would be $\ell((x, y), w) = (w \cdot x - y)^2$. Assume $\ell(z, w)$ is convex in w for each $z \in \mathcal{Z}$.
- The **expected risk** of a weight vector $w \in S$ is defined as follows:

$$\boxed{L(w) = \mathbb{E}_{z \sim p^*}[\ell(z, w)].} \tag{602}$$

- Define the weight vector that minimizes the expected risk:

$$w^* \in \arg\min_{w \in S} L(w). \tag{603}$$

- The batch learner gets T i.i.d. training examples $(z_1, \ldots, z_T)$, where each $z_t \sim p^*$. The goal is to output some estimate $w \in S$ that hopefully has low $L(w)$.

Assuming we have an online learner as a black box, we will construct a batch learner as follows:

- **Algorithm 6 (online-to-batch conversion)**
  - Input: T training examples $z_1, \ldots, z_T$.
  - Iterate $t = 1, \ldots, T$:
    - Receive w_t from the online learner.
    - Send loss function $f_t(w) = \ell(z_t, w)$ to the online learner.
  - Return the average of the weight vectors: $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$.
- Remarks about randomness
  - In contrast to the online learning (adversarial) setting, many of the quantities we are working with now have distributions associated with them.
  - For example, f_t is a random function that depends on z_t.
  - Each w_t is a random variable which depends on (i.e., is in the sigma-algebra of) $z_{1:t-1}$.
  - Note that $L(w^*)$ is not random.
- **Theorem 37 (Online-to-batch conversion)**
  - Recall the usual definition of regret (which depends on $z_{1:T}$):

$$\text{Regret}(u) = \sum_{t=1}^{T} [f_t(w_t) - f_t(u)]. \tag{604}$$

– Online-to-batch conversion obtains the following expected expected risk:

$$\boxed{\mathbb{E}[L(\bar{w})] \leq L(w^*) + \frac{\mathbb{E}[\text{Regret}(w^*)]}{T}.} \tag{605}$$

- Interpretation: the expected expected risk of the online-to-batch conversion $L(\bar{w})$ is bounded by the best possible expected risk (attained by $w^* \in S$) plus the online regret.
- Note that $\mathbb{E}[L(\bar{w})]$ has two expectations here: the $\mathbb{E}[\cdot]$ is an expectation over possible training datasets, and L contains an expectation over test examples.
- Proof:
  - The first key insight is that $f_t(w_t)$ provides an unbiased estimate of the expected risk of w_t.

$$\mathbb{E}[f_t(w_t) \mid w_t] = L(w_t). \tag{606}$$

    This is true because all the examples are independent, so w_t (deterministic function of $z_{1:t-1}$) is independent of f_t (deterministic function of z_t).
  - The second key insight is that averaging can only reduce loss. Since $\ell(z, \cdot)$ is convex, and an average of convex functions is convex, L is convex. By Jensen's inequality:

$$L(\bar{w}) \leq \frac{1}{T}\sum_{t=1}^{T} L(w_t). \tag{607}$$

  - Now, the rest is just putting the pieces together. Putting (607) and (606) together:

$$L(\bar{w}) \leq \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[f_t(w_t) \mid w_t]. \tag{608}$$

    Adding and subtracting $L(w^*)$ to the RHS, noting that $L(w^*) = \mathbb{E}[f_t(w^*)]$:

$$L(\bar{w}) \leq L(w^*) + \frac{1}{T}\sum_{t=1}^{T} (\mathbb{E}[f_t(w_t) \mid w_t] - \mathbb{E}[f_t(w^*)]). \tag{609}$$

  - Taking expectations on both sides:

$$\mathbb{E}[L(\bar{w})] \leq L(w^*) + \frac{1}{T}\sum_{t=1}^{T} (\mathbb{E}[f_t(w_t) - f_t(w^*)]). \tag{610}$$

  - The second term of the RHS is upper bounded by the regret, so we have:

$$\mathbb{E}[L(\bar{w})] \leq L(w^*) + \frac{\mathbb{E}[\text{Regret}(w^*)]}{T}. \tag{611}$$

- Remarks
  - If you run online subgradient descent once over your training data,[19] you should expect $O(\sqrt{T})$ regret by our previous analyses.
  - The resulting average weight vector will, *in expectation*,[20] have an expected risk which is within $O(1/\sqrt{T})$ of the best weight vector.
  - If the batch learner returns the last weight vector w_{T+1}, then our analysis above doesn't apply.
  - In general, averaging is a useful concept for stabilizing learning algorithms.

5.13 Adversarial bandits: expert advice (Lecture 16)

- In online learning with expert advice, on each iteration, after we choose one of the d experts/actions, nature reveals the loss vector z_t for every single action.
- However, in many applications such as clinical trials or advertisement placement, packet routing, we only get to observe the loss of the action we took, not the losses of the actions we didn't take.
- This setting is known as the multi-armed bandit problem, which is a type of online learning problem with **partial feedback**.
- This problem is much more difficult. Intuitively, the learner should choose actions that we expect to yield low loss, but it must choose which actions to explore to get more information about the losses. Thus, the multi-armed bandit problem exposes the challenges of the classic exploration/exploitation tradeoff.[21]
- Setup
  - FIGURE: [matrix with loss functions]
  - There are d possible actions (corresponding to the arms of a row of slot machines).
  - Let $z_t \in [0,1]^d$ denote the vector of losses of the d actions at iteration t.
  - For each iteration $t = 1, \ldots, T$:
    - Learner chooses a distribution $w_t \in \Delta_d$ over actions, and samples an action $a_t \sim w_t$.
    - Learner observes the loss of *only that particular action* z_{t,a_t} and no others.

[19] In practice, it helps to run the online learning algorithm multiple times over the training data.
[20] You can get high probability bounds too using martingales, but we won't discuss those here.
[21] Reinforcement learning requires managing the exploration/exploitation tradeoff, but is even more difficult because actions take the learner between different states in a way that is unknown to the learner.

- The expected regret with respect to an expert $u \in \Delta_d$ is defined in the same way as it was for online learning with expert advice:

$$\mathbb{E}[\text{Regret}(u)] = \sum_{t=1}^{T} [w_t \cdot z_t - u \cdot z_t]. \tag{612}$$

  Note that taking the max over randomized experts $u \in \Delta_d$ is equivalent to taking the max over single actions $a \in [d]$, since the maximum over a linear function is attained at one of the vertices.

- Estimating the loss vector
  - Recall that in online learning, we would observe the entire loss vector z_t. In that case, we could use the exponentiated gradient (EG) algorithm ($w_{t+1,j} \propto w_{t,j} e^{-\eta z_{t,j}}$) to obtain a regret bound of Regret $\leq \sqrt{2 \log(d) T}$ (see Example 37).
  - In the bandit setting, we don't observe z_t, so what can we do? Let's try to estimate it with some $\hat{z}_t$ that (i) we can observe and (ii) is equal to z_t in expectation (unbiased) in that for all $a = 1, \ldots, d$:

$$\mathbb{E}_{a_t \sim w_t}[\hat{z}_{t,a} \mid w_t] = z_{t,a}. \tag{613}$$

  - Given these two constraints, it's not hard to see that the only choice for $\hat{z}_t$ is:

$$\boxed{\hat{z}_{t,a} = \begin{cases} \frac{z_{t,a}}{w_{t,a}} & \text{if } a = a_t \\ 0 & \text{otherwise.} \end{cases}} \tag{614}$$

  Note that dividing $w_{t,a}$ compensates for sampling $a_t \sim w_t$.

- **Algorithm 7 (Bandit-EG)**
  - Run EG with the unbiased esimate $\hat{z}_t$ rather than z_t. Really, that's it.

- **Theorem 38 (Bandit-EG analysis)**
  - Bandit-EG obtains expected regret (expectation taken over learner's randomness) of

$$\boxed{\mathbb{E}[\text{Regret}(u)] \leq 2\sqrt{d \log(d) T}.} \tag{615}$$

- Comparison of Bandit-EG (bandit setting) and EG (online learning setting):
  - Compared with the bound for EG in the full information online learning setting (546), we see that this bound (615) is worse by a factor of $\sqrt{d}$.

- In other words, the number of iterations T for Bandit-EG needs to be d times larger in order to obtain the same average regret as EG.
- This is intuitive since in the bandit setting, each iteration reveals $(1/d)$-th the amount of information compared with the online learning setting.

- Proof of Theorem 38:
  - If EG is run with the unbiased estimate $\hat{z}_t$ ($z_t = \mathbb{E}[\hat{z}_t \mid w_t]$), then the expected regret is simply:

$$\mathbb{E}[\text{Regret}(u)] \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \mathbb{E}_{a_t \sim w_t} \left[\sum_{a=1}^{d} w_{t,a} \hat{z}_{t,a}^2 \mid w_t \right]. \tag{616}$$

    Note that the random variable $\hat{z}_t$ only depends on the previous actions through the random variable w_t.
  - So now, we just have to compute the expected local norm:

$$\mathbb{E}_{a_t \sim w_t} \left[\sum_{a=1}^{d} w_{t,a} \hat{z}_{t,a}^2 \mid w_t \right] = \sum_{a=1}^{d} w_{t,a}^2 \left(\frac{z_{t,a}^2}{w_{t,a}^2} \right) \leq d, \tag{617}$$

    since $z_t \in [0,1]^d$.
  - Note that it is crucial that we use local norms here; the generic bound (540) of $\|z_t\|_\infty^2$ is not good enough because $\|z_t\|_\infty$ is unbounded.
  - Minimizing the regret bound with respect to η, we get $2\sqrt{d \log(d) T}$ with $\eta = \sqrt{\log(d)/(dT)}$.

- Note that we have only gotten results in expectation (over randomness of the learner $a_t \sim w_t$). To get high probability results, we also need to control the variance of $\hat{z}_t$. To do this, we modify the EG algorithm by smoothing w_t with the uniform distribution over actions. This results in the standard Exp3 algorithm.

5.14 Adversarial bandits: online gradient descent (Lecture 16)

- In the online convex optimization setting, we are presented with a sequence of functions $f_1, \ldots, f_T$. Importantly, we could compute subgradients $z_t \in \partial f_t(w_t)$, which allowed us to update our weight vector using online gradient descent (OGD). In the bandit setting, suppose we can only query function values of f_t. Can we still minimize regret?

- Intuitively, something should be possible. In one dimension, the gradient of f_t at a point w can be approximated by $\frac{f_t(w+\delta) - f_t(w-\delta)}{2\delta}$. But this requires two function evaluations, which is not permitted in the bandit setting: you only get to administer one treatment to a patient, not two! Can we do it with one function evaluation? Can we generalize to high dimensions? The answer to both questions is yes. The original idea is from Flaxman et al. (2005); we follow the exposition in Shalev-Shwartz (2011).

- For a function f, define a smoothed version of f as follows:

$$\tilde{f}(w) = \mathbb{E}_{v \sim U_{\leq 1}}[f(w + \delta v)], \tag{618}$$

where $U_{\leq 1}$ is the uniform distribution over the unit ball (the set of vectors v with $\|v\|_2 = 1$). There are two important properties:

  - We can compute the gradient of $\tilde{f}$ (this is the key step):

$$\boxed{\nabla \tilde{f}(w) = \mathbb{E}_{v \sim U_1}\left[\frac{d}{\delta} f(w + \delta v) v\right].} \tag{619}$$

    This is due to Stokes' theorem, but the intuition can be seen in 1D due to the Fundamental Theorem of calculus:

$$\tilde{f}(w) = \mathbb{E}[f(w + \delta v)] = \frac{\int_{-\delta}^{\delta} f(w + t) dt}{2\delta} = \frac{F(w + \delta) - F(w - \delta)}{2\delta}, \tag{620}$$

    where F is the antiderivative of f and thus

$$\tilde{f}'(w) = \frac{f(w + \delta) - f(w - \delta)}{2\delta}. \tag{621}$$

    In the general setting, if we draw $v \sim U_1$ from the unit sphere, then $\frac{d}{\delta} f(w + \delta v) v$ is an unbiased estimate of $\nabla \tilde{f}(w)$.

  - We have that $\tilde{f}$ well approximates f:

$$|\tilde{f}(w) - f(w)| \leq |\mathbb{E}[f(w + \delta v) - f(w)]| \leq L\delta, \tag{622}$$

    where L is the Lipschitz constant of f.

Now we are ready to define the algorithm:

- **Algorithm 8 (Bandit-OGD)**

  - For $t = 1, \ldots, T$:
    * Set $w_t = \arg\min_{w \in S} \|w - \eta \theta_t\|_2$ (same as OGD)
    * Pick random noise $v_t \sim U_1$
    * Predict $\tilde{w}_t = w_t + \delta v_t$
    * Compute $z_t = \frac{d}{\delta} f_t(\tilde{w}_t) v_t$
    * Update $\theta_{t+1} = \theta_t - z_t$

- **Theorem 39 (regret bound bandit online gradient descent)**

  - Let $f_1, \ldots, f_T$ be a sequence of convex functions.

- Let $B = \max_{u \in S} \|u\|_2$ be the magnitude of an expert.
- Let $F = \max_{u \in S, t \in [T]} f_t(u)$ be the maximum function value.
- Then Bandit-OGD obtains the following regret bound:

$$\mathbb{E}[\text{Regret}(u)] = \mathbb{E}\left[\sum_{t=1}^{T}[f_t(\tilde{w}_t) - f_t(u)]\right] \tag{623}$$

$$\leq 3L\delta T + \frac{B^2}{2\eta} + \frac{\eta}{2}Td^2(F/\delta + L)^2, \tag{624}$$

where the expectation is over the learner's random choice (nature is still adversarial).

- Let us set δ and η to optimize the bound.
  - First, we optimize the step size η: setting $\eta = \frac{B}{d(F/\delta+L)\sqrt{T}}$ yields a regret bound of

$$3L\delta T + Bd(F/\delta + L)\sqrt{T}. \tag{625}$$

  - Second, we optimize the amount of smoothing δ: setting $\delta = \sqrt{\frac{BdF}{3L}}T^{-1/4}$ yields a final regret bound of

$$\mathbb{E}[\text{Regret}] \leq \sqrt{12BdFLT^{3/4} + BdL\sqrt{T}} \tag{626}$$

$$= \boxed{O(\sqrt{BdFL}T^{3/4}),} \tag{627}$$

where in the last equality, we're thinking of the dependence on T as primary.
  - Note that our regret bound of $O(T^{3/4})$ has a worse dependence than the $O(T^{1/4})$ that we're used to seeing. This comes from the fact that we're only getting an unbiased estimate of the gradient of the a *smoothed function* $\hat{f}$, not f. And as usual, we have a $\sqrt{d}$ dependence on the dimension since we are in the bandit setting and only get $1/d$-the the information of a full gradient.

- Proof of Theorem 39
  - Our analysis of plain OGD Theorem 30 yields the following regret bound when applied to w_t and linear functions $x \mapsto x \cdot z_t$:

$$\sum_{t=1}^{T}[w_t \cdot z_t - u \cdot z_t] \leq \frac{1}{2\eta}\|u\|_2^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|z_t\|_2^2. \tag{628}$$

  - First, taking expectations on both sides, using the fact that $\tilde{f}_t$ is convex with $\nabla\tilde{f}_t(w_t) = \mathbb{E}[z_t]$ for the LHS, expanding the definition of z_t on the RHS:

$$\sum_{t=1}^{T}\mathbb{E}[\tilde{f}_t(w_t) - \tilde{f}_t(u)] \leq \frac{1}{2\eta}\|u\|_2^2 + \frac{\eta}{2}\sum_{t=1}^{T}\mathbb{E}[\|(d/\delta)f_t(\tilde{w}_t)v_t\|_2^2]. \tag{629}$$

– Second, let's handle the gradient: Since f_t is L-Lipschitz, we have that

$$f_t(\tilde{w}_t) \leq f_t(w_t) + L\delta \leq F + L\delta. \tag{630}$$

Plugging this inequality into (629):

$$\sum_{t=1}^{T} \mathbb{E}[\tilde{f}_t(w_t) - \tilde{f}_t(u)] \leq \frac{B^2}{2\eta} + \frac{\eta}{2}\frac{d^2}{\delta^2}(F + L\delta)^2. \tag{631}$$

– Finally, we need to relate $\tilde{f}_t(w_t)$ to $f_t(\tilde{w}_t)$. Using the Lipschitz property of f_t, we have:

$$f_t(\tilde{w}_t) - f_t(u) \leq f_t(w_t) - f_t(u) + L\delta \leq \tilde{f}_t(w_t) - \tilde{f}_t(u) + 3L\delta. \tag{632}$$

Plugging this into (631) completes the proof.

5.15 Stochastic bandits: upper confidence bound (UCB) (Lecture 16)

- Let us return to the learning from expert advice setting where at each iteration, we are choosing an arm $a_t \sim w_t$ and suffering loss z_{t,a_t}. So far, we have assumed that the losses z_t were adversarial. Now, let us now consider the **stochastic setting**, where z_t's are drawn from an (unknown) distribution. By leveraging the probabilistic structure of the problem, we'll show in this section that we can improve the regret bounds from $O(\sqrt{T})$ to $O(\log T)$.

- We will show an algorithm called upper confidence bound (UCB) (Lai and Robbins, 1985; Auer et al., 2002) that is based on the principle of *optimism in the face of uncertainty*: we maintain upper bounds on the reward of each action and choose the action that maximizes the best possible payoff. The idea is that we either get very low loss (which is great), or we learn something.

- Let us define the notation for the stochastic bandits setting:

  – Let $r_1, \ldots, r_T$ be an i.i.d. sequence of reward vectors,[22] where each $r_t \in [0,1]^d$

  – Define $\mu \stackrel{\text{def}}{=} \mathbb{E}[r_t]$ to be the average rewards (μ doesn't depend on t).

  – Define $j^* \stackrel{\text{def}}{=} \arg\max_{j=1}^{d} \mu_j$ be the best action.

  – Define $\Delta_j \stackrel{\text{def}}{=} \mu_{j^*} - \mu_j > 0$ be the gap between action j and the best action j^*. Let $\Delta \stackrel{\text{def}}{=} \min_{j \neq j^*} \Delta_j$ the smallest gap.

[22] In the stochastic bandits literature, we typically maximize reward rather than minimize loss, so think of $r_t = -z_t$.

- Let $a_1, \ldots, a_T \in [d]$ be the sequence of actions taken by the learner.
- Let $S_j(t) \stackrel{\text{def}}{=} \{i \in [t] : a_i = j\}$ be the set of time steps i up to time t where we took action j.
- Let $N_j(t) = |S_j(t)|$ be the number of times we took action j.
- We are interested in minimizing the expected regret:

$$\mathbb{E}[\text{Regret}] = \sum_{t=1}^{T} \mathbb{E}[r_{t,j^*} - r_{t,a_t}] \tag{633}$$

$$= \sum_{j=1}^{d} \mathbb{E}[N_j(T)]\Delta_j, \tag{634}$$

  where in the last inequality, we've rewritten the regret to sum over all possible actions; each of the $N_j(T)$ times we take action j, we add Δ_j to the regret. Note that at each time step t, r_t is independent of a_t, which is a function of the past.

- Now let's design an algorithm for solving the stochastic bandits problem. The way to think about it is that we're just estimating the mean μ in an online manner. At each time step t, we have $N_j(t)$ i.i.d. draws of action j, from which we can form the empirical estimate

$$\hat{\mu}_j(t) = \frac{1}{|S_j(t)|} \sum_{i \in S_j(t)} r_{i,j}. \tag{635}$$

  Provided that $S_j(t)$ grows to infinity, we can estimate the means: $\hat{\mu}_j(t) \xrightarrow{P} \mu_j$.

- The difference from mean estimation is that we are also trying to *exploit* and will choose actions j that are more likely to have higher reward.

- By Hoeffding's inequality,

$$\mathbb{P}[\hat{\mu}_j(t) \leq \mu_j - \epsilon] \leq \exp\left(-2N_j(t)\epsilon^2\right). \tag{636}$$

  In other words, with probability at least $1 - \delta$,

$$\mu_j \leq \hat{\mu}_j(t) + \sqrt{\frac{\log(1/\delta)}{2N_j(t)}}. \tag{637}$$

  This motivates the following algorithm:

- **Algorithm 9 (upper confidence bound (UCB))**
  - Play each of the d actions once.

- For each subsequent time step $t = d+1, \ldots, T$:
  - Choose action

$$a_t = \arg\max_{1 \le j \le d} \left\{ \hat{\mu}_j(t-1) + \sqrt{\frac{2 \log t}{N_j(t-1)}} \right\}, \tag{638}$$

- **Theorem 40 (regret of upper confidence bound (UCB))**
  - The UCB algorithm obtains the following expected regret bound:

$$\boxed{\mathbb{E}[\text{Regret}] \le \frac{8d \log T}{\Delta} + 5d.} \tag{639}$$

- Proof sketch:
  - The main idea is to upper bound the number of times an suboptimal action $j \neq j^*$ was taken. This happens when the following event A happens:

$$\hat{\mu}_j + \epsilon_j \ge \hat{\mu}_{j^*} + \epsilon_{j^*}. \tag{640}$$

  - FIGURE: [$\hat{\mu}_j, \mu_j, \hat{\mu}_{j^*}, \mu_{j^*}$ on a line with Δ_j gap]
  - Recall that the separation between μ_j and μ_{j^*} is Δ_j. If the actual values $\hat{\mu}_j$ and $\hat{\mu}_{j^*}$ are within $O(\Delta_j)$ of their means, then we're all set.
  - The probability of that not happening after m rounds is $e^{-O(m\Delta^2)}$ as given by Hoeffding's inequality. If we wait m rounds and look at the expected number of times j was chosen:

$$m + \sum_{t=1}^{T} t^2 e^{-O(m\Delta_j^2)}, \tag{641}$$

    where the t^2 comes from taking a sloppy union bound over pairs of time steps (because we don't know which time steps we're comparing so just try all pairs). If we set $m = O\left(\frac{\log T}{\Delta_j^2}\right)$, then we can make the second term be a constant. Summing the first term over j and multiplying Δ_j yields the result.
  - Note that we're being pretty sloppy here. For the full rigorous proof, see Auer et al. (2002).

- Interpretation: that we have a logarithmic dependence on T, and a necessary linear dependence on the number of actions d. There is also a dependence on Δ^{-1}; a smaller Δ means that it's harder to distinguish the best action from the others.

5.16 Stochastic bandits: Thompson sampling (Lecture 16)

- In UCB, our principle was optimism in the face of uncertainty. We now present an alternative method called Thompson sampling, which dates back to Thompson (1933). The principle is *choose an action with probability proportional to it being optimal*, according to a Bayesian model of how the world works.

- Setup
  - Prior $p(\theta)$ over parameters θ
  - Probability over rewards given actions: $p(r \mid \theta, a)$

  The Thompson sampling principle leads to a simple, efficient algorithm:

- Algorithm
  - For each time $t = 1, \ldots, T$:
    - Sample a model from the posterior: $\theta_t \sim p(\theta \mid a_{1:t-1}, r_{1:t-1})$
    - Choose the best action under that model: $a_t = \arg\max_a p(r \mid \theta_t, a)$.

- Example for Bernoulli rewards
  - Assume each reward $r_j \sim \text{Bernoulli}(\mu_j)$ (here the parameters θ are the means μ).
  - For each action $j = 1, \ldots, d$
    - Let the prior be $\theta_j \sim \text{Beta}(\alpha, \beta)$
    - Keep track of the number of successes s_j and failures f_j.
  - For each time $t = 1, \ldots, T$:
    - Sample $\theta_j \sim \text{Beta}(\alpha + s_j, \beta + f_j)$ for each $j = 1, \ldots, d$
    - Choose $a_t = \arg\max_j \theta_j$ and observe reward r_t.
    - If $r_t = 1$ then increment s_{a_t}; else increment f_{a_t}.

  Despite its long history, Thompson sampling was not very popular until recently, where rigorous regret bounds have appeared and people realize that it outperforms UCB in many settings. Agrawal and Goyal (2012) prove the following:

- **Theorem 41 (regret for Thompson sampling)**
  - For any $\epsilon > 0$, the expected regret of Thompson sampling us upper bounded by:

$$\mathbb{E}[\text{Regret}] \leq (1 + \epsilon) \sum_{j:\Delta_j > 0} \frac{\Delta_j \log T}{\text{KL}\left(\mu_j \| \mu^*\right)} + O\left(\frac{d}{\epsilon^2}\right). \tag{642}$$

- One strength of Thompson sampling is that it's a principle that allows one to easily generalize the algorithm to structured settings such as reinforcement learning.

- One weakness of Thompson sampling is that it does not choose actions that takes into account the value of information gained (especially important for structured settings). As an example (from Ian Osband), consider $\theta \in \{e_1, \ldots, e_d\}$ be an unknown indicator vector, and suppose $a \in \{0,1\}^d$ and the reward is is $a \cdot \theta - 0.0001\|a\|_1$. The optimal strategy would be to do binary search to find θ in $O(\log d)$ time, but Thompson sampling would just choose one a at a time, which would take $O(d)$ time. More expensive methods such as Gittins index performs the actual dynamic programming and have better guarantees but are computationally more expensive.

5.17 Summary (Lecture 16)

- This concludes our tour of online learning and its extensions to the bandit setting.
- Our measure of success is getting low **regret**, the difference between the learner's cumulative losses and the best *fixed* expert's cumulative losses. In particular, we hope for sublinear regret: Regret $= o(T)$.
- Without no additional assumptions (even with two experts), any deterministic algorithm must have Regret $= \Omega(T)$ (fail).
- In the realizable setting (some expert is perfect), the majority algorithm achieves $O(\log d)$ regret (constant in T).
- We started with the **follow the leader (FTL)**, and saw that it worked for quadratic loss functions (Regret $= O(\log T)$), but failed for linear loss functions (Regret $= \Omega(T)$).
- Inspecting the regret bounds reveals that in order to get low regret, we need to have a *stable* learner, in the sense that w_t and w_{t+1} should be close (according to some notion of proximity).
- This motivated us to look at **follow the regularized leader (FTRL)**, where we add a quadratic regularizer, which gave us a regret bound with two terms: (i) a bias-like term (value of regularizer applied to the best expert) and (ii) a variance-like term (sum of the norm of the subgradients). Balancing the two by controlling the amount of regularization (inverse step size) yields Regret $= O(\sqrt{T})$.
- If our loss functions were non-linear, we could linearize using the subgradient. Coupled with a quadratic regularizer, we get the **online subgradient descent (OGD)** algorithm. We also showed that it suffices to analyze linear functions, since they result in the most regret.
- We looked at learning with expert advice, and got regret bounds of $O(\sqrt{dT})$, where d is the number of experts. Inspired by the logarithmic dependence on d in the majority algorithm, this motivated us to consider different regularizers.

- The general algorithm that deals with different regularizers is **online mirror descent (OMD)**. We analyzed this algorithm using Fenchel duality and Bregman divergences, which allowed us to look at arbitrary regularizers through the lens of the mapping between w_t and θ_t. Specializing to learning with expert advice, we get a $O(\sqrt{\log(d)T})$ regret bound with the entropic regularizer, resulting in the **exponentiated gradient (EG)** algorithm.

- By exploiting properties of exponentiated gradient, we can perform a refined analysis again using **local norms** to achieve stronger results, matching the $O(\log d)$ regret in the realizable setting.

- In the bandit setting with partial feedback, we get $O(\sqrt{d \log(d)T})$ regret again using local norms to control the size of the now unbounded subgradients. The general principle is to find an estimate of the loss vector z_t and run existing online learning algorithms (which are quite tolerant of noise).

- Finally, we showed that learners with low regret in the online setting directly lead to learners with low **expected risk** in the batch setting via an online-to-batch conversion.

5.18 References

- Shalev-Shwartz, 2012: Online Learning and Online Convex Optimization (survey paper)

  - This is a nice introduction to online learning, on which much of these online learning notes are based.