CS880: Approximation and Online Algorithms	**Scribe:** Lorenzo Najt
Lecture 0: Experts Problem and Mirror Descent	**Date:** LectureDate

The scribe is responsible for any errors in this document, which has not been scrutinized to the degree a published work would be. References are given throughout, and the reader is encouraged to look to them in case of typos.

0.1 Experts and Mirror Descent

In this lecture we study the expert's problem, defined as follows:

Definition 0.1.1 (Experts Problem) *Given: N choices indexed by i. Sequence of cost functions $c_t : [N] \to [0,1]$.*
Goal: Pick i_t for $t \in [T]$ before *observing c_t to minimize $\sum_{t\in[T]} c_t(i_t)$.*
The performance of an algorithm is measured in terms of the regret, $R_T = \sum_{t=1}^{T} c_t(i_t) - \sum_{t=1}^{T} c_t(i^*)$ where $i^* \in argmin_{i\in[N]} \sum_{t=1}^{T} c_t(i)$.

This problem is similar to metrical task systems, but will be easier to solve. We will develop a solution which we will later understand to be mirror descent, a continuous optimization technique. Then we will see how mirror descent extends to the metrical task system problem.

0.1.0.1 Comparison of experts to Metrical Task Systems (MTS)

Comparison against MTS:

1. No movement cost.
2. No look ahead (we choose the expert before observing the cost vector)
3. Bounded cost.
4. Comparing against a fixed expert.

We can think of each choice in $[N]$ as being one point in the metric space, as in MTS. In this problem, we call them "experts." We are given a sequence of cost functions, mapping each expert to a cost. We choose an expert and incur the cost that the expert causes. The major difference between experts and MTS is that we pick the expert before observing the cost. On the other hand, there is no movement cost, we can change the expert we follow without any cost. We will see below that the combination of no movement cost, bounded costs and no look ahead is *almost* equivalent to having movement cost and look ahead.

The bounded cost has to do with us bounding the regret, rather than the competitive ratio. Unlike competitive ratio, regret is an additive notion: it says that the total cost that we incur should be *additively* close to the total cost of the best expert. In the regret setting, the most cost we can incur in any step is 1, so the regret is at most T. Thus, anything regret linear in T would be unsatisfying as answer. Our goal in this situation is to get sublinear regret. We will be able to achieve $\sqrt{T}$ regret in the experts setting, which is best possible. Note : if the optimal cost is linear in T, and our regret is $\sqrt{T}$, then as a fraction of the optimal cost, the regret goes to zero.

Additionally, in experts we are comparing against a fixed expert, but in MTS we are comparing against a moving target (the optimal hindsight algorithm can move the position of the server). The regret guarantees we will achieve for the experts setting will be stronger, because the competing algorithm (the best hindsight expert) doesn't have the same power as our algorithm, which can change expert from round to round.

0.1.1 Randomized Algorithm

The online algorithm will choose some sequence of distributions $p_t \in \Delta^{[N]}$ over the experts, and will choose experts based on it. Thus, the expected regret will be $\sum_{t=1}^{T} \mathbb{E}_{p_t}(c_t) - \sum c_t(i^*)$.

We would like to optimize for p_t. However, we cannot just set p_t based on c_t, because we must determine p_t before observing c_t. We will rewrite the objective function to reveal a connection to the framework where lookaheads are possible but where there is a movement cost. Note that $\mathbb{E}_p(c_t) = \langle p_t, c_t \rangle$, where $\langle , \rangle$ denotes the dot product of two vectors.

$$\sum_{t=1}^{T} \mathbb{E}_{p_t}(c_t) - \sum_{t=1}^{T} c_t(i^*) = \tag{0.1.1}$$

$$\sum_{t=1}^{T} \mathbb{E}_{p_t}(c_t) - \sum_{t=1}^{T} \mathbb{E}_{p_{t+1}}(c_t) + \sum_{t=1}^{T} \mathbb{E}_{p_{t+1}}(c_t) - \sum_{t=1}^{T} c_t(i^*) = \tag{0.1.2}$$

$$\sum_{t=1}^{T} \langle p_t - p_{t+1}, c_t \rangle + \sum_{t=1}^{T} \langle p_{t+1}, c_t - c_t(i^*) \rangle \tag{0.1.3}$$

The first term in the last expression calculates the cost of c_t under the change in p_t, which is like a movement cost with a metric depending on c_t. Since $c_t \leq 1$ this movement cost term is bounded by $\sum_{t=1}^{T} ||p_t - p_{t+1}||_1$.

The second term calculates the cost relative to the optimal expert; however, we have gained the ability to look ahead, since we are comparign the cost vector at time t with the proposed probability distribution at time $t + 1$.

We thus obtain the following bound $R_T \leq \sum_t (|p_t - p_{t+1}|_1 + \langle p_{t+1}, c_t - c_t(i^*) \rangle)$.

We will find that the L_1-metric is not a convenient notion to work with. Instead, we will work with KL-divergence, which we now take a moment to review.

0.1.2 KL-Divergence and Entropy

Recall the definition of entropy of a probability distribution p on $[N]$: $\sum_{i \in [N]} p_i \log(1/p_i)$. This is in the range $[0, \log(N)]$. KL-divergence gives a related way of measuring distances between distributions:

Definition 0.1.2 *Let p and q be distributions on $[N]$, then $D(p\|q) = \sum_i p_i \log(p_i/q_i)$.*

Observe that this is zero if $q = p$. This intuition behind this function has to do with information gain: If you thought the distribution was q, but it was actually p, this will be the amount of information you gain when you observe samples from the distribution.

Fact 0.1.3 *How to remember which is on the top and the bottom? If q is fixed, then this should be convex, since it is a sort of distance function, so it should look like negative entropy rather than entropy.*

The idea will be to set $p_{t+1} = \text{argmin}_{p\in\Delta} D(p\|p_t) + \eta\langle p, c_t\rangle$, which tries to strike a balance between not updating too much and accommodating information from the new cost vector. Here $\eta > 0$ is some parameter called the learning rate, which will be tuned depending on the context. In the next section we will show that the classical exponential weights algorithm is obtained from this minimization. Later on, after developing the framework of mirror descent, we will see that this is a simply projected gradient descent with the right choice of geometry, and that this minimization problem is the proximal description of projected gradient descent.

0.1.2.1 Derivation of Exponential Weights Algorithm

In this section we will solve the minimization problem $\text{argmin}_{p\in\Delta} D(p\|p_t) + \eta\langle p, c_t\rangle$. Let $\phi(p) = D(p\|q) + \langle p, c\rangle$, where q is a nowhere zero pdf on $[N]$, $c : [N] \to [0, 1]$, and p ranges over the probability simplex $\Delta_{[N]}$. By calculating the Hessian to be $\text{Diag}(p_i/q_i)$, one sees that this is convex function over the probability simplex. From here, we will formulate the convex program, and formulate the dual, and derive the exponential weight updates from the KKT conditions. From a review of convex optimization, consult Boyd and Vanderberghe chapter 5.

$$\min \phi(p) \tag{0.1.4}$$
$$-p_i \leq 0 \tag{0.1.5}$$
$$\sum p_i - 1 = 0 \tag{0.1.6}$$
$$\tag{0.1.7}$$

Thus, the Lagrangian is

$$L(\lambda, \nu, p) =\phi(p) - \sum_i \lambda_i p_i + \nu(\sum p_i - 1) \tag{0.1.8}$$
$$=(\sum(\log(p_i/q_i) + c_i - \lambda_i + \nu)p_i) - \nu \tag{0.1.9}$$

We can analytically minimize $(\log(p_i/q_i) + c_i - \lambda_i + \nu)p_i$ with respect to p_i, noting that this is only defined on $p_i \geq 0$ (where it equals 0 when $p_i = 0$ by continuity). We obtain, $\inf_{p_i}(\log(p_i/q_i) + c_i - \lambda_i + \nu) = -q\exp(-1 + \lambda_i - c_i - \nu)$.

We therefore obtain the following dual problem:

$$\max \sum_i -q_i \exp(-1+\lambda_i - c_i - \nu) \tag{0.1.10}$$
$$\lambda_i \geq 0 \tag{0.1.11}$$
$$\nu \in \mathbb{R} \tag{0.1.12}$$
$$\tag{0.1.13}$$

We observe that at the dual optimal, we have $\lambda_i^* = 0$. Based on this, gradient part of the KKT optimality condition becomes $\nabla\phi(p^*) + \nu^*\nabla(\sum p_i^* - 1) = 0$, which is equivalent to $\log(p_i^*/q_i) + c_i + 1 + \nu^* = 0$ for all i, or $p_i = q_i \exp(-c_i)\exp(-1-\nu^*)$. The term $\exp(-1-\nu^*)$ is the normalizing constant.

With $q_i = p_t(i)$ and $p_i = p_{t+1}(i)$, this gives us the exponential weight update algorithm: $p_{t+1}(i) = p_t(i)\exp(-c_i)C$, where C is the normalizing constant.

We can add a weighting η to the $\langle p, c_t\rangle$ term in $\phi(p)$, which is called the learning rate. One would tune η to obtain optimal performance. If one repeats the above calculation with $\phi(p) = D(p_{t+1}\|p_t) + \eta p \cdot c_t$, the update rule would become $p_{t+1}(i) = p_t(i)\exp(-c_i)C_{t+1}$.

0.1.2.2 Analysis of Regret

In this section, we analyze the regret incurred by following the strategy of the previous section. That is, we have

$$w_1(i) = 1, \forall i \in [N] \tag{0.1.14}$$
$$w_t(i) = \exp(-\eta c_t(i))w_{t-1} = \exp(-\eta \sum_{s=1}^{t-1} c_t(i)), t \geq 2 \tag{0.1.15}$$
$$p_t = \frac{w_t}{||w_t||_1} \tag{0.1.16}$$

Definition 0.1.4 (Regret Function) $R_n = \sum_{t=1}^n p_t \cdot c_t - \inf_{1\leq i\leq N}\sum_{t=1}^n c_t(i)$.

Theorem 0.1.5 *By following the exponential weight update strategy with learning rate η, we have that $R_n \leq \frac{\log(N)}{\eta} + \frac{n\eta}{8}$. By setting $\eta = 2\sqrt{\frac{2\log(N)}{n}}$, we get $R_n \leq \sqrt{\frac{n\log(N)}{2}}$.*

We provide two proofs, although the second appears to be a proof of a slightly different statement. The first is from : `http://sbubeck.com/BubeckLectureNotes.pdf`. The second proof emphasizes the potential function perspective.

Proof: Let $W_t = \sum_{i\in[N]} w_t(i)$. First, we calculate:

$$\log(\frac{W_{n+1}}{W_1}) = \log(\sum_{i\in[N]} W_{n+1}(i)) - \log(N) \tag{0.1.17}$$
$$\geq \log(\max_{i\in[N]} w_{n+1}(i)) - \log(N) \tag{0.1.18}$$
$$= -\eta \min_{i\in[N]}(\sum_{s=1}^{n} c_t(i)) - \log(N) \tag{0.1.19}$$

Additionally, we have

$$\log(\frac{W_{n+1}}{W_1}) = \sum_{t=1}^{n} \log(\frac{W_{t+1}}{W_t}).$$

We let I_t be a random variable on $[N]$ distributed according to $p_t = \frac{w_t}{W_t}$. Using Hoeffdings inequality, we calculate:

$$\log(\frac{W_{t+1}}{W_t}) = \log(\sum_{i=1}^{N} \frac{w_t(i)}{W_t} \exp(-\eta c_t(i))) \tag{0.1.20}$$
$$= \log(\mathbb{E}(\exp(-\eta c_t(I_t)))) \tag{0.1.21}$$
$$- \eta \mathbb{E}(c_t(I_t)) + \frac{\eta^2}{8} \tag{0.1.22}$$
$$= -\eta p_t \cdot c_t + \frac{\eta^2}{8} \tag{0.1.23}$$

Thus,we have:

$$-\eta \min_{i\in[N]}(\sum_{t=1}^{n} c_t(i)) + \log(d) \leq \log(\frac{W_{n+1}}{W_n}) = \sum_{t=1}^{n} \log(\frac{W_{t+1}}{W_1}) \leq \sum_{t=1}^{n}(-\eta p_t \cdot c_t + \frac{\eta^2}{8})$$

Rearranging gives $R_n \leq \frac{n\eta}{8} + \frac{\log(d)}{\eta}$. ■

We now provide a second errro bound, in which the potential function analysis is more clear. Roughly speaking, a potential function is a function of the current state, and not how one arrived at that state, that is bounded below and which has the property that when the algorithm pay the potential function decreases.[1]

[1]A simple example of this kind of analysis is given by the binary experts problem where we are guaranteed that there is an expert that is always right. That is, there are n strings $x_i \in \{0,1\}^{\mathbb{N}}$, and one true string $t \in \{0,1\}^{\mathbb{N}}$. We are guaranteed that there is an i so that $x_i = y$. At each time t, we observe the first t digit of the x_i, and are asked to make a guess about the tth digit of y. The majority vote algorithm makes this decision by taking the majority vote of the set of experts that have made no mistakes by time $t-1$. Note that if the algorithm guesses wrong, the pool of strings that will be considered the next time has decreased by half. Since the pool of experts will never be zero, we see that the algorithm will make at most $\lfloor \log_2(m) \rfloor$ mistakes. The two key features of this algorithm are :

Theorem 0.1.6 $\sum_{t=1}^{n} p_t \cdot c_t \leq \frac{1}{1-\epsilon} \inf_{1 \leq i \leq N} \sum_{t=1}^{n} c_t(i) + \frac{1}{\epsilon} \log(N)$.

Proof:

We follow `http://www.cs.cornell.edu/courses/cs6820/2012fa/handouts/experts.pdf`

We will define the multiplicative update algorithm in terms of multiplying $w_t(i)$ by $(1-\epsilon)^{c_t(i)}$, since this will make the algebra easier. This is of course the same, provided we pick an η with $\exp(-\eta) = (1-\epsilon)$.

Let $W_t = \sum_{i \in N} w_t(i)$ be the total weights at time t, and $C_T(i) = \sum_{t=1}^{T} c_t(i)$ be the total cost incurred by expert i. Let i^* be the optimal expert.

For our analysis, $\ln(W_T)$ will be the potential function. From $W_T > w_T(i^*) = (1-\epsilon)^{C_T(i^*)}$ we have the following lower bound on the potential function:

$$\ln(W_T) > \ln(1-\epsilon) C_T(i^*) \tag{0.1.24}$$

Next, or goal is relate decreases in W_T to the amount that the algorithm pays. We will let x_t be the decision that the algorithm makes at time t; note that this is random variable whose distribution is determined by the $w_t(i)$.

In particular, we will prove the following inequality, which tells us that the drop in the potential at time t bounds a constant times the expected cost of the algorithm at time t:

$$\epsilon \mathbb{E}(c_t(x_t)) \leq \mathbb{E}(\ln(W_t)) - \mathbb{E}(\ln(W_{t+1})) \tag{0.1.25}$$

We use 0.1.25 and 0.1.24 in the following way: buy summing over all time steps we obtain $\mathbb{E}(\sum_{t=1}^{T} c_t(x_t)) \leq \frac{1}{\epsilon}(\ln(n) - \mathbb{E}(\ln(W_T))) \leq \frac{1}{\epsilon}(\ln(n) + \ln(\frac{1}{1-\epsilon}) C_T(i^*))$, since $W_0 = n$.

From this, we can apply the identity $(1/y)\ln(1/(1-y)) < 1/(1-y)$ for $y \in (0,1)$ to obtain $\mathbb{E}(\sum_{t=1}^{T} c_t(x_t)) \leq \frac{\ln(n)}{\epsilon} + \frac{1}{1-\epsilon} C_T(i^*)$.

We now show how to derive 0.1.25.

1) it maintains a notion of how credible the set of experts are, the size of the pool of experts that have never made a mistake. 2) Each time the algorithm makes a mistake, the credibility decreases multiplicative, but since we've assumed the existence of an expert that is close to the true value, the credibility never decreases below some lower bound. This algorithm is information theoretically optimal, which means that no algorithm can make fewer guess in the worse case. The scribe convinced himself of the optimality in the following way. First, we recall that the notion of an algorithm in this context is *any* sequence of functions f_t from the strings $x_i[0,t]$ that outputs a bit. In order to be optimal, we note that the adversary can produce any sequence of bits after observing the behavior of the algorithm, so long as it doesn't contradict the assumption that at least one of the experts is always right. This means that if we have a pool of 2^m experts, making their first m prediction in accordance with a unique one of the 2^m elements of $\{0,1\}^m$, then for the first m steps the adversary can also choose the bit other than the bit chosen by the algorithm, forcing any deterministic algorithm to make at least m mistakes.

$$\mathbb{E}(W_{t+1}|w_t) = \sum_{i\in[N]} \mathbb{E}((1-\epsilon)^{c_t(i)} w_t(i)|w_t) \tag{0.1.26}$$
$$\leq \sum_{i\in[N]} \mathbb{E}((1-\epsilon c_t(i)) w_t(i)|w_t) \tag{0.1.27}$$
$$= \sum_{i\in[N]} w_t(i) - \epsilon\mathbb{E}(\sum_{i\in[N]} \mathbb{E}(c_t(i) w_t(i)|w_t) \tag{0.1.28}$$
$$= W_t(1-\epsilon\mathbb{E}(\sum_{i\in[N]} \mathbb{E}(c_t(i) p_t(i)|w_t)) \tag{0.1.29}$$
$$= W_t(1-\epsilon\mathbb{E}((\mathbb{E}_{w_t}(c_t(x_t))|w_t))) \tag{0.1.30}$$
$$\tag{0.1.31}$$

From the last line above, an application of Jensen's inequality along with $\ln(1+x) \leq x$ for $x > -1$ (noting that the costs are in $[0,1]$) yields:

$$\mathbb{E}(\ln(W_{t+1})|w_t) \leq \ln(W_t) + \ln(1-\epsilon\mathbb{E}[(\mathbb{E}_{w_t}(c_t(x_t))|w_t)]) \tag{0.1.32}$$
$$\leq \ln(W_t) - \epsilon\mathbb{E}[(\mathbb{E}_{w_t}(c_t(x_t))|w_t)] \tag{0.1.33}$$
$$\tag{0.1.34}$$

Taking expectation of both sides and rearranging yields 0.1.25.

■

We remark the the learning rate η depends on n, the number of trials we will undergo, sometimes called the horizon. Since we may not know the horizon n , we cannot always expect to pick the optimal learning rate. It is possible to obtain a regret bound of the same order by using the changing learning rate of $\eta_t = 2\sqrt{\frac{\log(d)}{t}}$. See Proposition 2.4 of `http://sbubeck.com/BubeckLectureNotes.pdf`.

Additionally, we remark that if $c_t \in [m, M]$ instead an identical analysis applies, except that the regret scales by a factor of $\frac{1}{M-m}$.

0.1.3 Continuous version, bounding regret

In this section we will examine the continuous time version of the algorithm derived in the previous section.

That is, we have a continuous function $C : [0,T] \times N \to [0,1]$, and a differential strategy $p : [0,T] \to \Delta_N$. Let i^* be the expert minimizing $\int_0^T c_t(i^*)$. Based on our transformation of the discrete time problem into a movement term and a cost term, we consider minimizing the following regret:

Definition 0.1.7 (Continuous Regret) *The regret functional is* $R(p) = \int_0^T |\partial_t p_t|_1 dt + \int_0^T \langle p_t, c_t - c_t(i^*)\rangle dt$.

In the discrete case, our update rule is $p_{t+1}(i) = p_t(i)e^{-\eta c_t(i)}N_t$, where $N_t = \sum_i p_t(i)\exp(-\eta c_t(i))$ is the normalizing constant. This can be rewritten as

$$\log(p_{t+1}(i)) - \log(p_t(i)) = -\eta c_t(i) + \log(\sum_i p_t(i)\exp(-\eta c_t(i)))$$

If we think of the lefthand side as the discrete derivative, by analogy we can describe an ODE for p_t in the continuous time case:

$\partial_t \log(p_t(i)) = -\eta c_t(i) + N_t$, where N_t will be a function designed to ensure that $\sum p_t(i) = 1$ for all t. Below we will derive the necessary formula for N_t. By simple calculus this formula implies $\partial_t p_t(i) = -\eta c_t(i)p_t(i) + N_t p_t(i)$.

We will now compute N_t: Given, $\sum p_t(i) = 1$ it follows that

$$0 = \sum_i \partial_t p_t(i) = \tag{0.1.35}$$

$$\sum_i -\eta c_t(i)p_t(i) + N_t p_t(i) = \tag{0.1.36}$$

$$\sum_i (-\eta c_t(i)p_t(i)) + N_t \tag{0.1.37}$$

Thus, $N_t = \eta \sum_i c_t(i)p_t(i) = \eta\langle p_t, c_t\rangle$. This gives us an ODE describing the evolution of p_t:

$$\begin{cases} \partial_t \log(p_t(i)) = -\eta c_t(i) + \eta p_t \cdot c_t \\ p_0 = \text{Uniform on [N]} \end{cases} \tag{0.1.38}$$

We will now bound the expected regret of using this update rule/ ODE for the probability distributions over the choice of experts. The proof strategy we follow will be to use a potential function: this means that we will define a function such that each time we incur some large regret, the potential function decreases a lot. We will then bound the total decrease in the potential.

We will denote the delta distribution on expert i by δ_i, and the potential function will be KL divergence relative to δ_{i^*}, where i^* is the hindsight optimal expert.

Theorem 0.1.8 *If p_t follows the ODE above, then $R(p) \leq 3(\sqrt{T\log(N)})$.*

Proof: The potential function $P(t) = D(i^*\|p_t) = -\log(p_t(i^*))$. From here, we calculate the rate of change in the potential function: $\partial_t D(i^*\|p_t) = -\partial_t \log(p_t(i^*)) = \eta c_t(i^*) - \eta\langle p_t, c_t\rangle$. The first equality in this last expression is the definition of KL divergence, and the second follows from the ODE for the evolution p_t. Continuing our calculation, we obtain $\langle p_t, c_t - c_t(i^*)\rangle = \frac{-1}{\eta}\partial_t D(i^*|p_t)$. Integrating both sides over t, we get:

$\int_0^T \langle p_t, c_t - c_t(i^*)\rangle dt = \frac{1}{\eta}[D(i^*\|p_0) - D(i^*\|p_T))] \leq \frac{\log(N)}{\eta}$

The last inequality follows since, $D(i^*|p_t) \geq 0$, and p_0 was uniformly distributed.

Additionally, using that $c_t(i) \in [0, 1]$ and the ODE describing the evolution of $\log(p_t(i))$, we have $\sum_i |\partial_t p_t(i)| = \sum_i |\eta(p_t(i)c_t(i) - \langle p_t, c_t\rangle p_t(i)| \leq \eta((\sum_i p_t(i)c_t(i)) + \langle p_t, c_t\rangle) \leq 2\eta$

From the definition of the regret we get $R(p) \leq 2\eta T + \log(N)/\eta$. We optimize this inequality by choosing $\eta = \sqrt{\log(N)/T}$, from which we obtain $R(p) \leq 3\sqrt{T\log(N)}$.

■

0.2 Mirror Descent

In this section we will put the algorithm above into the general framework of mirror descent. First, we will review projected gradient descent in the Euclidean case.

0.2.1 Projected Gradient Descent

In order to state the optimality conditions for convex optimization over a convex set, we will need the notion of the normal cone at a point:

Definition 0.2.1 (Normal Cone) *Let K be a convex set, and $y \in K$. Then $N_K(y) = \{\theta \in \mathbb{R}^n : \forall x \in K, \theta \cdot (x-y) \leq 0\}$ is called the normal cone to K at y. It consists of all the directions that would take you out of K.*

Lemma 0.2.2 *Let K be a convex set, and f a differentiable convex function. Then $y \in argmin_{x\in K} f(x)$ if and only if $-\nabla f(y) \in N_K(y)$.*

Proof: Let $y \in K$ satisfy $-\nabla f(y) \in N_K(y)$. Let $z \in K$. Since $f(z) \geq f(y) + \nabla f(y)(z-y)$ by convexity, and $\nabla f(y)(z-y) \geq 0$ by $-\nabla f(y) \in N_K(y)$, it follows that $f(z) \geq f(y)$. On the other hand, suppose that $y \in \text{argmin}_{x\in K} f(x)$. If $(-\nabla f)(y) \notin N_K(y)$, then there is $x \in K$ so that $(-\nabla f(y))(x-y) > 0$ so by definition of the directional derivative there is some $\epsilon > 0$ so that $f(y + (x-y)\epsilon) < f(y)$. ■

The following lemma shows that Euclidean projection onto any convex set only decreases the distance to other points in that set. This will be relevant when showing that the projection step of projected gradient descent does not lose any relevant information.

Definition 0.2.3 (Euclidean Projection) *Define the function $\Pi_K(z) = argmin_{y\in K} ||z-y||_2$.*

Lemma 0.2.4 (Properties of Euclidean Projection) *For any closed convex set K, there is a unique solution to $argmin_{y\in K} ||z-y||_2$. Thus, $\Pi_K(z)$ exists and is well defined. If $y = \Pi_K(z)$, then for any $w \in K$, $||y-w||_2 \leq ||z-w||$.*

Proof: The existence of the minimum follows by a compactness argument. The uniqueness of the minimum then follows by strict convexity of the distance function. The inequality follows by writing $z = y + p$, where p is in the normal cone to K at z; this can be done because of the optimality conditions for optimizing the function $f(x) = ||x-z||^2$, which has $-\nabla f(x) = z - x$, so $z - y \in N_K(y)$ if $y = \text{argmin}_{x\in K} f(x)$. The claim now follows by expanding $\langle y+p-w, y+p-w\rangle = ||y||^2 + ||w||^2 - 2\langle y, w\rangle + ||p||^2 - 2\langle p, y-w\rangle \geq \langle y-w, y-w\rangle$, using that $\langle p, w-y\rangle < 0$. ■

We note that in some cases, such as for the case of K being the probability simplex, the problem of finding the nearest point in Euclidean distance can be solved exactly.

Definition 0.2.5 (Projected gradient descent) *We fix $\eta \in \mathbb{R}_{>0}$, called the learning rate. The algorithm starts with a point x_0. The algorithm is defined by the update rule: $x_{t+1} = \Pi_K(x_t -$*

$\eta\nabla f(x_t))$.

In order to analyze the performance of projected gradient descent, either as an optimization algorithm or an online learning algorithm, the notion of a potential function is important. We already met potential functions in the second analysis of the experts problem given above. Roughly speaking, a potential function is some function of the distance to the optimal point and the difference in optimal values, which can be guaranteed to decrease during each step of the algorithm, and which will never become negative. By analyzing how the potential function changes, one gains insight into how the optimization algorithm approaches the optimum value.

Definition 0.2.6 (Potential function for projected gradient descent) *Suppose that x^* is the minimum of f. The function $||x_t - x^*||$ is called the potential function.*

The following theorem shows how these tools fit together in the optimization context. Note that the online learning context is a strict generalization, in which the function f might change from round to round.

Theorem 0.2.7 *Suppose that f is L-Lipschitz. If projected gradient descent runs for T steps with $\eta = \frac{||x^*-x_0||}{L\sqrt{T}}$, there is a $t \leq T$ such that $f(x_t) - f(x^*) \leq \frac{L||x^*-x_0||}{\sqrt{T}}$*

Proof: It is a simple calculation to verify that $||a||^2 - ||a-b||2 = 2a \cdot b - ||b||^2$. Thus, taking $a = x^* - x_t$ and $b = x_{t+1} - x_t = -\eta\nabla f(x_t)$, we have that:

$$||x^* - x_t||_2^2 - ||x^* - x_{t+1}||_2^2 = 2(x^* - x_t)\cdot(-\eta\nabla f(x_t)) - \eta^2||\nabla f(x_t)||_2^2 \tag{0.2.39}$$

$$\geq 2\eta(f(x_t) - f(x^*)) - \eta^2||\nabla f(x_t)||_2^2 \tag{0.2.40}$$

$$\geq 2\eta(f(x_t) - f(x^*)) - \eta^2 L^2 \tag{0.2.41}$$

0.2.40 followed because from convexity we have that $f(x_t) - f(x^*) \leq \nabla f(x_t)(x_t - x^*)$.

Summing over all steps, we obtain:

$$\sum_{t=1}^{T}(f(x_t) - f(x^*)) \leq \frac{||x^* - x_1|| - ||x^* - x_T||}{2\eta} + \frac{\eta}{2}TL^2 \leq \frac{||x^* - x_1||}{2\eta} + \frac{\eta}{2}TL^2$$

The result follows by making the optimal choice of η, as $q(\eta) = \frac{A}{2\eta} + \frac{\eta B}{2}$ is minimized when $\eta = \sqrt{\frac{A}{B}}$.
■

A key idea in the previous argument is that the reduction in the potential is related to the suboptimality gap.

We remark that you may not know the optimal learning rate in advance, so one may have to make a guess for a bound $||x^* - x_0|| \leq D$. From this the calculation of the previous theorem will provide $||x^* - x_1||^2/(2\eta) + (\eta/2)TL^2 \leq D/(2\eta) + (\eta/2)TL^2 \leq \sqrt{DT}L$. Note that if f is L' Lipschitz, and $L \geq L'$, then it is also L-Lipschitz, hence bounds on the Lipschitz constant are also acceptable for the analysis.

In order to understand the minimization problem from earlier that lead to the exponential update algorithm, the following optimization interpretation of the gradient descent step will be useful:

Proposition 0.2.8 *Let K be any convex set.* $\Pi_K(y-v) = argmin_{xinK} v \cdot x + \frac{1}{2}||x-y||^2$.

Proof: $argmin_{xinK}(v \cdot x + \frac{1}{2}||x-y||^2) = argmin_{xinK}(\frac{x^2}{2} + \langle x, v-y\rangle)$. Let $\phi(x) = \frac{x^2}{2} + \langle x, v-y\rangle$, which is a strictly convex function. We know that the minimum occurs at x iff $-\nabla\phi(x) \in N_K(x)$ or $y - v - x \in N_K(x)$. This holds for $x = \Pi_K(y-v)$, so the claim follows. ∎

0.2.1.1 Euclidean Projected Gradient Descent for the experts problem

In this section we will show that one can derive a suboptimal algorithm for the experts problem through Euclidean projected gradient descent. Mirror descent will generalize projected gradient descent, and recover the optimal exponential weights algorithm from before.[2] The value of this section is to demonstrate the use of the correct geometry.

Translating the experts problem in the projected gradient descent setting, and denoting the probability simplex by Δ, we have $x_{t+1} = \Pi_\Delta(x_t - \eta c_t)$.

The regret is $\sum_{t=1}^T(c_t p_t - l_t q)$, where q is the hindsight optimal expert.

The calculation from the analysis of projected gradient descent gave:

$||x^* - x_t||_2^2 - ||x^* - x_{t+1}||_2^2 = 2(x^* - x_t)\cdot(-\eta\nabla f(x_t)) - \eta^2||\nabla f(x_t)||_2^2$, which in this context is:

$||q - p_t||_2^2 - ||q - p_{t+1}||_2^2 = 2(q - p_t)\cdot(-\eta c_t) - \eta^2||c_t||_2^2$.

Summing this over all t, and using that $||c_t||_2^2 \le N$ and $||q-p||_2 \le 1$[3], we get:

$$Regret = \sum_{t=1}^T c_t(p_t - q) = \frac{||q-p_1||^2 - ||q-p_{T+1}||^2}{2\eta} + \frac{\eta}{2}\sum ||c_t||_2^2 \le \frac{1}{2\eta} + \frac{TN\eta}{2}.$$

Tuning, we obtain an error bound of $\sqrt{TN}$, which is suboptimal compared to $\sqrt{T\log(N)}$ obtained earlier from the exponential weights algorithm. However, the idea to use gradient descent is right; we just need to work in a different geometry. This geometry can thought of as incorporating additional information about the problem. For the experts problem, we know that the optimal solution sparse (unless many experts are equally good), and we will modify the geometry so that gradient descent moves faster when far from a sparse point.

0.2.2 Preliminaries for Mirror Descent

These notes were useful in the creation of these notes:

1. `http://sbubeck.com/BubeckLectureNotes.pdf`
2. `https://www.cs.ubc.ca/~nickhar/F18-531/Notes20.pdf`
3. https://homes.cs.washington.edu/ jrl/teaching/cse599swi16/notes/lecture3.pdf

[2] For more on how Mirror descent can recover classical algorithms, see the following talk by Alexander Madry: `https://www.youtube.com/watch?v=noRNcDbqtVY`

[3] $\sum p_i^2 \le (\sum p_i)^2 = 1$

Recall that the idea of projected gradient descent is that you take a step in the gradient direction, and then find the nearest point in the feasible set to project back on to. Usually, by projection we mean to find the closest point in Euclidean distance. The insight of Mirror descent is that in certain contexts Euclidean distance is not as useful as other notions of distance.

We will use KL-divergence instead. Although KL-divergence is not a metric, the convexity inherent will make it amenable to a more general projected gradient descent theory called mirror descent.

We will now describe the basic theory underlying mirror descent.

Definition 0.2.9 *Throughout, D will denote an open convex subset of $\mathbb{R}^n$.*

Definition 0.2.10 (Legendre Function) *A function $F : \overline{D} \to \mathbb{R}$ is Legendre if:*

- *F is strictly convex and has continuous partials on D*
- $\lim_{x \to \overline{D} \setminus D} ||\nabla F(x)|| = \infty$

Recall that if ϕ is convex, we have $\phi(y) \geq \phi(x) + \langle \nabla\phi(x), (y - x)\rangle$. Bregman divergence measures the error of this lower bound.

Definition 0.2.11 (Bregman Divergence) *Say that F is a Legendre function on D. Let $x, y \in D$. We define: $D_F(x, y) = F(x) - (F(y) + (x - y)^T \nabla F(y))$ for any $(x, y) \in \overline{D} \times D$.*

We will only use certain Legendre functions for gradient descent.

Definition 0.2.12 (Mirror Map) *A mirror map will be any Legendre function F defined on a region $D \subseteq \mathbb{R}^n$, such that $\nabla F(D) = (\mathbb{R}^n)^*$.*[4]

The most basic example of a mirror map is the function $||x||_2^2$ defined on $\mathbb{R}^n$, from which we will recover projected gradient descent. An important example of a mirror map will be $\sum_i x_i \log(x_i) + \sum x_i$ on $\mathbb{R}^n_{>0}$.

0.2.2.1 Infinitesimal Bregman Divergence as a Riemannian Metric

In this section we pause to examine the Bregman divergence as a Riemannian metric. Observe that if F has continuous second derivatives, then we obtain via Taylor series expansion: $F(x) = F(y) + \nabla F(y)(x-y) + (x-y)\frac{1}{2}\nabla^2 F(y)(x-y) + O(||y-x||^3)$. Thus, $D_F(x, y) = (x-y)\frac{1}{2}\nabla^2 F(y)(x-y) + O(||y-x||^3)$. In particular, we have that $D_F(y+h, y) = h\frac{1}{2}\nabla^2 F(y)h + O(||h||^3)$.

Recall that the differential df lies in the cotangent bundle; in order to bring it back to a vector in $\mathbb{R}^n$, we need to use the inner product given by the Riemannian metric g. When we do so, we obtain:

$$\text{grad}_g f(x) = (\nabla^2 F(x))^{-1}(\nabla f(x)).$$

In the previous formula ∇f is the usual Euclidean gradient. Unconstrained mirror descent takes steps in the $-\eta\text{grad}_g f$ direction; with constraints these steps will be projected back to the feasible set.

[4]In some sources this last condition is dropped; if $\nabla F(D) \subsetneq \mathbb{R}^n$ some care is needed to make sure that the algorithm is well defined. We discuss this below when we define the algorithm.

In the remainder of this section, we will calculate this Riemannian metric in the case when the underlying set is the interior of the probability simplex on $[n]$, and the mirror map is negative entropy $F(p) = \sum_i p_i \log(p_i)$. In this case, the metric we obtain is called the Fisher information metric. We show that after a change of variables it becomes the usual round metric on the sphere, and recall some intriguing comments about the connection to the Fubini-Studi metric on the space of quantum states.

First, it is straightforward to calculate that $\nabla^2 F(p) = Diag(1/p_i)$. Thus, the Riemannian metric can be written $\sum_I \frac{dp_i dp_i}{p_i}$.

We will now consider the map $y_i = p_i^2$, mapping the probability simplex to the intersection of the unit sphere with the positive orthant. Let $\sum dy_i dy_i$ be the standard Euclidean metric restricted to the sphere. Since $y_i = \sqrt{p_i}$, we can calculate the pullback of this metric along the map $p \to y$ as follows:

$$\sum_i dy_i dy_i = \sum d(\sqrt{p_i}) d(\sqrt{p_i}) = \sum_i \frac{1}{4} \frac{dp_i dp_i}{p_i}$$

In a comment on his blog[5], John Baez points out that this metric suggests the quantum formalism: when think of a probability distribution, we may prefer to think of a point on the sphere that normalizes to that probability distribution. That is, there is a natural section of the probability amplitude map $S^{2n-1} \to \mathbb{CP}^n \to \Delta_n$, whose image lies in the non-negative real orthant, and of course this is the map $(x_1, \ldots, x_n) \to (x_1^2, \ldots, x_n^2)$, which is the map transforming the Fischer metric to the round metric. Working with the Fischer metric is on the simplex like mapping probability distributions to the quantum state in the real positive orthant of $\mathbb{CP}^n$, and using the Fubini-Study metric.

Finally, we note that boundary of Δ_n, $\nabla^2 F$ blows up, so $\text{grad}_g f$ is shrunk. We will see that can think of this as slowing down the walk near the boundary of the probability simplex, or (comparatively) speeding it up in the interior. In the context of the experts problem, this can be thought of as incorporating the information that we are competing against a good single expert.

While we can think of the unconstrained case as gradient flow, and thus a setting for classical ODE theory. In `https://arxiv.org/pdf/1711.01085.pdf` one can see how in the constrained case we can obtain a differential inclusion with a good existence and uniqueness theory.

0.2.2.2 Background on Fenchel Duality

We will now collect and prove several fundamental properties of the Bregman divergence. A key concept for these results is the Fenchel dual (Convex Conjugate):

Definition 0.2.13 (Fenchel Dual) *Let $D \subseteq \mathbb{R}^n$, and $f : D \to \mathbb{R}$ be any function. We define $f^*(u) = \sup_{x \in D}(x^T u - f(x))$, called the Fenchel (or convex) dual.*

$f^*(u)$ can be interpreted as the amount needed to shift f down so that $x^T u$ just touches its epigraph. The notion of duality it invokes is that of describing the epigraph by the collection of supporting

[5] `https://johncarlosbaez.wordpress.com/2011/03/02/information-geometry-part-7/`

half-planes. If $f(x) = cx$, then $f^*(u) = \infty\delta_{u\neq c}$, and $(f^*)^*(x) = \sup_u(xu - \infty\delta_{u\neq c}) = cx$. Thus, if h is an affine function, $(h^*)^* = h$.

Lemma 0.2.14 (Properties of the Fenchel Dual) *Suppose that $f : D \to \mathbb{R}$ is some function. Then:*

1. $(f^*)^* \leq f$
2. *If $f \geq h$, then $f^* \leq h^*$*
3. *If f is closed and convex, then $(f^*)^* = f$.*
4. *If f is closed and convex, then $f^*(s) = s \cdot x - f(x)$ iff $s \in \partial f(x)$ iff $x \in \partial f^*(s)$.*

Proof:

1. Follows from:

$$(f^*)^*(x) = \sup_{u\in\mathbb{R}^n}(ux - f^*(u)) = \tag{0.2.42}$$

$$\sup_{u\in\mathbb{R}^n}(ux - \sup_{z\in D}(uz - f(z)) \leq \tag{0.2.43}$$

$$\sup_{u\in\mathbb{R}^n}(ux - (ux - f(x)) = f(x) \tag{0.2.44}$$

2. $f^*(u) = \sup_x(xu - f(x)) \leq \sup_x(xu - h(x)) = h^*(u)$.

3. Since f is a closedproper[6] convex function, we may write f is the supremum of all the affine functions that it majorizes.

4. If $f^*(s) = sx - f(x)$, then for all z we have that $sx - f(x) \geq sz - f(z)$, which we can rearrange into $f(z) \geq f(x) + s(z - x)$. In other words, this implies that $s \in \partial f(x)$. On the other hand, if $s \in \partial f(x)$, then $f(z) \geq f(x) + s(z - x)$ holds by definition of subgradient, and so $f^*(s) = sx - f(x)$.

   For the last iff statement we will use that f is closed and convex, in order to use that $f^{**} = f$. Since $f^*(s) = sx - f(x)$ iff $(f^*)^*(x) = f(x) = xs - f^*(s)$, we get that $x \in \partial f^*(x)$. On the other hand, if $x \in \partial f^*(s)$, we have $f^*(s') \geq f^*(s) + x(s' - s)$ $\forall s'$, so $xs - f^*(s) \geq xs' - f^*(s')$ $\forall s'$, so $xs - f^*(s) = (f^*)^*(x) = f(x)$.

$\blacksquare$

Let $f : D \to \mathbb{R}^n$. We think of ∇f as a function $D \to \mathbb{R}^n$, sending a point x to the gradient of f at that point: $x \to \nabla_x f$. We let D^* be the image of D under this map. With this understood, we have the following lemma:

[6]That is, it is not $+\infty$ eveywhere, and it is nowhere $-\infty$.

Lemma 0.2.15 *If F is Legendre then F^* is also differentiable and $\nabla(F^*) = (\nabla F)^{-1}$ on D^*.*

Proof: If we can show that F^* is also differentiable, then we will be done as we will have that $F^*(u) = ux - f(x)$ iff $u = \nabla F(x)$ iff $x = \nabla F^*(u)$. Putting the last two statements together gives $x = (\nabla F^*)(\nabla F(x))$.

Since F^* is a supremum of affine functions, we know that it has subderivatives everywhere. Suppose that $x, x' \in \partial F^*(u)$, so that $F^*(u) = ux - F(x) = ux' - F(x')$. Rewriting this last equality, we get $F(x) = F(x') + u(x - x')$. Since F is strictly convex, this implies that $x = x'$. Thus F^* has exactly one subdifferential at each u. This implies that F is differentiable everywhere. ■

We will now calculate two key examples of Bregmann divergences. The first will help us relate this to ordinary gradient descent, and the second will be relevant to the our study of the experts problem.

0.2.2.3 Squared Euclidean Distance

We take $F(x) = \frac{1}{2}||x||_2^2$ and $D = \mathbb{R}^n$. Then $D_F(x,y) = \frac{1}{2}||x-y||_2^2$.

0.2.2.4 KL-Divergence

$F(x) = \sum x_i \log(x_i) - \sum x_i$ defines the generalized negative entropy on $D = (0, +\infty)^d$.

We have $D_F(x,y) = \sum x_i \log(x_i/y_i) - \sum(x_i - y_i)$, which is called the generalized KL-divergence.

$\nabla F = (\log(x_i))$ so $D^* = \mathbb{R}^n$.

0.2.3 Further Preliminaries on Mirror Descent

Returning to the general theme, we will now prove several lemmas that are key to the Mirror descent algorithm, and show that the online experts algorithm of the previous section is a special case of Mirror descent.

Lemma 0.2.16 *If F is Legendre, then we have $D_F(x,y) = D_{F^*}(\nabla F(y), \nabla F(x))$.*

Proof: This follows from straightforward algebra, using the definition of the Bregman divergence and the identity $(\nabla F)^{-1} = (\nabla F^*)$. ■

Lemma 0.2.17 (Generalized Pythagorean Theorem) $D_F(x,y) + D_F(y,z) = D_F(x,z) + (x - y)(\nabla F)(z) - \nabla F(y))$

Proof: This follows by straightforward algebraic manipulation. ■

The important thing is that D_F behaves like squared Euclidean distance. To verify this, we need several more lemmas:

Lemma 0.2.18 *If F is Legendre then $D_F(x,y)$ is strictly convex in x.*

Proof: Since $D_F(x,y) = F(x) - (F(y) + (x-y)^T \nabla F(y))$, this follows from the strict convexity of F. ■

Lemma 0.2.19 $\nabla_x D_F(x,y) = \nabla F(x) - \nabla F(y)$

Proof: $D_F(x,y) = F(x) - (F(y) + (x-y)^T \nabla F(y))$, so $\nabla D_F(x,y) = \nabla F(x) + \nabla F(y)$. ■

Scribe Remark: We will assume that F has a continuous extension to $\overline{D}$. I'm not sure if this matters, but it makes the statements easier and is true for the two examples (generalized entropy and Euclidean squared distance).

Proposition 0.2.20 *If $A \subseteq \overline{D}$ is closed and convex, with $A \cap D \neq \emptyset$, then $\forall x \in D$, $b = argmin_{a\in A} D_F(a, x)$ exists and is unique. Moreover, if $b \in A \cap D$, then $\forall a \in A$, $D_F(a, x) \geq D_F(a, b) + D_F(b, x)$.*

Proof: Uniqueness + Existence follows from the strict convexity of F, and the fact that A is closed and convex. The second condition will follow from the generalized pythagorean theorem plus the normal cone optimality conditions for minimizing a convex function. ■

The previous proposition lets us define the projection onto A using the Bregman divergence:

Definition 0.2.21 (Bregman divergence projection) *Under the conditions of 0.2.3, let $\Pi^F_A(x) = argmin_{a\in A\cap D} D_F(a, x)$.*

0.2.3.1 Mirror Descent

The mirror descent algorithm operates as follows; where l_t is the sequence of (differnetiable) loss functions.

1. Start at $a_1 \in \operatorname{argmin}_{a\in A} F(a)$.
2. Set $w_{t+1} = (\nabla F)^{-1}(\nabla F(a_t) - \eta \nabla l_t(a_t))$. [7]
3. Set $a_{t+1} = \operatorname{argmin}_{a\in A\cap D} D_F(a, w_{t+1}) = \Pi^F_A(w_{t+1})$.

We examine the optimization perspective on these updating steps. As we saw much earlier with the experts problem, this perspective can be useful for deriving explicit formulas for the update steps via convex duality.

Proposition 0.2.22 (Proximal Formulation) [8] *The update step of the Mirror descent algorithm can also be described by: $a_{t+1} = argmin_{x\in A\cap D}(\eta g_t^T x + D_F(x, a_t))$.*

Proof:

$$\begin{aligned}
a_{t+1} =& \operatorname{argmin}_{x\in A\cap D} D_F(x, w_{t+1}) \\
=& \operatorname{argmin}_{x\in A\cap D}(F(x) - \nabla F(w_{t+1}) \cdot x) \\
=& \operatorname{argmin}_{x\in A\cap D}(F(x) - (\nabla F(a_t) - \eta \nabla l_t(a_t)) \cdot x) \\
=& \operatorname{argmin}_{x\in A\cap D}(\eta \nabla l_t(a_t)) \cdot x + (F(x) - \nabla F(a_t) \cdot x)) \\
=& \operatorname{argmin}_{x\in A\cap D}(\eta \nabla l_t(a_t)) \cdot x + (F(x) - (F(a_t) + \nabla F(a_t) \cdot (x - a_t))) \\
=& \operatorname{argmin}_{x\in A\cap D}(\eta \nabla l_t(a_t)) \cdot x + D_F(x, a_t))
\end{aligned}$$

■

[7] This is well defined if $\nabla F(a_t) - \eta \nabla l_t(a_t) \in D^*$. In the case of interest to us, where the mirror map is negative entropy, $D^* = \mathbb{R}^n$, so this condition will always be satisfied.

[8] See `https://blogs.princeton.edu/imabandit/2013/04/16/orf523-mirror-descent-part-iii/`

Below, we will show how the experts problem can be cast as mirror descent. The reader can refer to the given references to see how this provides us with a regret bound that matches our earlier analysis.

Finally, we remark that for $F(x) = \frac{1}{2}||x||_2^2$ mirror descent recovers Projected Gradient Descent.

0.2.3.2 Online Mirror Descent and experts

In this section we verify that the online mirror descent algorithm, with A the probability simplex and mirror map the negative generalized entropy results in the exponential weight update algorithm from before. First, we note that $D^* = \mathbb{R}^n$, so the step updating w_{t+1} in mirror descent is always well defined.

We now examine the projection step. Recall:

$$\begin{aligned}
D_F(x,y) &= F(x) - (F(y) + \nabla F(y)(x-y)) \\
&= (\sum x_i \log(x_i) - \sum x_i) - (\sum y_i \log(y_i) - \sum y_i + (\log(y_i)) \cdot (x-y)) \\
&= \sum (y_i - x_i) + \sum x_i \log(x_i/y_i)
\end{aligned}$$

To compute the projection to the probability simplex using the Bregman divergence D_F, we will need the following lemma:

Lemma 0.2.23 *If f is convex and differentiable on X, then $f(x) \le f(y)$ for all $y \in X$ iff $\nabla f(x)(y-x) \ge 0$ for all $y \in X$.*

Proof: This is a reformulation of the optimality conditions, $\nabla f(x)(y-x) \ge 0$ for all $y \in X$ is equivalent to $-\nabla f(x) \in N_K(x)$. ■

We use this to show that in this context the projection map is just renormalization.

Lemma 0.2.24 *In the setting $A = \Delta_n$ is the probability simplex, and F is generalized negative entropy, we have $\Pi_A^F(y) = \frac{y}{||y||_1}$.*

Proof: Since the minimizer is unique, and specified by the condition of 0.2.23, it suffices to show that $\frac{w}{||w||_1}$ satisfies it. We have $f(x) = \sum(y_i - x_i) + \sum x_i \log(x_i/y_i)$, so $\nabla f(x) = (\log(x_i/y_i)$. We need to show that $\nabla f(x)(z-x) \ge 0$ for all $z \in \Delta_n$. But, we have that $\nabla f(x)(z-x) = \sum_i \log(x_i/y_i)(z_i - x_i) = \sum_i \log(1/||y||_1)(z_i - x_i) = \log(1/||y||_1)(\sum_i z_i - \sum_i y_i/||y||) = 0$. ■

Now we tackle the weight update term. Since $l_t(x) = c_t \cdot x$, by definition we have that $w_{t+1} = (\nabla F)^{-1}(\nabla F(a_t) - \eta \nabla c_t))$

We recall that $(\nabla F)(x) = (\log(x_i))$, so $(\nabla F)^{-1}(u) = (\exp(u_i))$. Thus, we have $w_{t+1}(i) \exp(\log(a_t(i)) - \eta c_t(i)) = a_t(i) \exp(-\eta c_t(i))$. This, along with the calculation of the projection map, shows that online mirror descent recovers the exponential weights algorithm.

0.3 Metrical Task Systems

Finally, we explain a little about an important modern application of Mirror descent to the metrical task system problem. This follows `https://arxiv.org/pdf/1807.04404.pdf` closely.

0.3.1 Metrical Task Systems Problem

First, we recall the metrical task system problem. We have a metric space (X, d), $|X| = n$. The input is a sequence of costs, $c_t : X \to \mathbb{R}_+$, $t \in \mathbb{N}$.

Definition 0.3.1 (Online Algorithm) *An online algorithm is a sequence of maps $\rho = (\rho_1, \rho_2, \ldots)$, where $\rho_t : (\mathbb{R}_+^X)^t \to X$ maps a sequence of costs functions to a state in X. We have a fixed initial state $\rho_0 \in X$. If the maps ρ incorporate some randomness, then the map is said to be a randomized online algorithm.*

Definition 0.3.2 (Total Cost of the Algorithm ρ in a cost sequence) *Let $c = (c_1, \ldots, c_t)$ be a sequence of cost functions, $c_t : X \to \mathbb{R}_+$. The total cost of the algorithm ρ in servicing c is:*

$$cost_\rho(c) = \sum_{t \geq 1} [c_t(\rho_t(c_1, \ldots, c_t)) + d(\rho_{t-1}(c_1, \ldots, c_{t-1}), \rho(c_1, \ldots, c_t))$$

. $c_t(\rho_t)$ is referred to as the service cost, and $d(\rho_{t-1}, \rho_t)$ is the movement cost.

Definition 0.3.3 (Service and movement costs.) *If ρ is a randomized online algorithm, define $S_\rho(c) = \mathbb{E} \sum c_t(\rho_t))$ to be the expected service cost and $M_\rho(c) = \mathbb{E}(\sum d(\rho_{t-1}, \rho_t))$ to be the expected movement cost.*

Definition 0.3.4 (Offline optimum) *For a sequence of cost functions c, the offline optimum, $cost^*(c)$ is the infimum of $\sum(c_t(x_t) + d(x_{t-1}, x_t))$, where x is any sequence of states (of the same length as c).*

Definition 0.3.5 (α-competative) *Let ρ be a randomized online algorithm. It is said to be competative if for every starting position $\rho_0 \in X$[9], there is a constant $\beta > 0$ such that for all cost sequence c,*

$$\mathbb{E}[cost_\rho(c)] \leq \alpha cost^*(c) + \beta$$

.

0.3.2 Tree Metrics

Definition 0.3.6 (Tree Metric) *Let (x, d) be a metric space with a distinguished point $r \in X$. (X, d) is a tree metric if it is the shortest path distance on a metric tree. The combinatorial depth of X is the depth of the tree rooted at the point r.*

Definition 0.3.7 (Hierarchically separated tree) *A hierarchical separated tree (HST) with separation $\tau > 1$ is a tree metric (X, d, r) such that if $T = (V, E, d)$ is the corresponding metric tree, and if T_e denotes the subtree rooted at the vertex of $e = u, v$ furthest from r, then $diam(T_e) \leq (1/\tau) d(u, v)$.[10] These are also called τ-HST metrics.*

[9] It suffices to make this definition for any deterministic starting position.

[10] That is, traversing the subtree is within a factor τ as expensive as crossing the edge to enter it.

Fact 0.3.8 *Any metric space can be probabilistically embedded in a weighted HST of depth $O(\log(n))$ and separation τ with distortion $O(\tau \log(n))$. An $O(f(n))$-competative algorithm for metrical task systems on τ-HSTs implies an $O(f(n)\tau \log(n))$ competative algorithm for any metric space.*

In the next section we will survey how to produce a $O(\log(n))$-competative randomized algorithm for metrical task systems on HST metrics.

0.3.3 Continuous Time Mirror Descent

For this application of mirror descent, we will work with a continuous-time model. (The scribe is not sure why.)

Definition 0.3.9 (Continuous Time Online Algorithm) *An online algorithm is a family of maps $\rho(T)$, $T \in \mathbb{R}_+$, from piece-wise continuous paths of costs functions $c(t) : X \to \mathbb{R}_+$, $t \in [0, T]$ to random variables $\rho(T) \in X$.*

Definition 0.3.10 (Service Cost) $S = \mathbb{E}(\int_{\mathbb{R}_+} c(t)(\rho(t))dt$.

It appears that we assume that for any path of costs functions, c, $\rho(T)$ makes only finitely many jumps in each interval. With this assumption, we can assume without loss that ρ is right-continuous, and make the following definition:

Definition 0.3.11 (Movement Cost) $M = \mathbb{E}(\sum_{t:\rho(t^-)\neq\rho(t)} d(\rho(t^-), \rho(t))$.

(Scribe notes: The definitions in the source text are a little hard to interpret. This is what I think is going on. We note that the goal is to prove something in the discrete time case, after all, and that when we examine their algorithm... this is perhaps the kind of process that we will get. Note that while their path of probability distributions given by mirror descent will be a continuous, it's not implausible that the stochastic process that one gets out of it only makes finitely many jumps.)

Theorem 0.3.12 *The existence of an α-competative algorithm for the continuous-time model with piecewise continuous costs implies the existence of an α-competative algorithm for the discrete-time model.*

Proof: Suppose that C is the discrete time cost vector. We let $T = \max_{x\in X} C(x)$. $(c(t))_{t\in[0,1]}$ is a waterfilling continuous time version of c, which is $c(t)(x) = 1_{C(x)\geq t}$. (Note that this is a step function that takes the values 0 or 1...!). Let $(\rho(t))_{t\in[0,T]}$ be the "path" taken by the α-compatative continuous time algorithm on this cost function path, starting with $\rho(0) = \rho_0$. If we let $s = \text{argmin}_{t\in[0,T]} C(\rho(t))$, the cheapest location that ρ lands on during this time interval, then we have that $\int_0^T c(t)(\rho(t))dt \geq C(\rho(s))$.

We also have that the movement of the continuous-time algorithm is at least $d(\rho(0), \rho(s)) + d(\rho(s), \rho(T))$.

Thus, the discrete time algorithm can move to $\rho(s)$ during that interval. When it comes time to move during the next time interval, then movement cost of the continuous time algorithm will be $d(\rho(T), \rho(s')) + d(\rho(s'), \rho(T'))$. Since $d(\rho(s), \rho(s')) \leq d(\rho(s), \rho(T)) + d(\rho(T), \rho(s'))$ by the triangle inequality, it follows that the total movement cost of the discrete time algorithm is less than that of the continuous time algorithm.

The offline optimium in the continuous time case is smaller than the discrete-time case, because the continuous time model can at the very least do exactly the same moves as the discrete time case. ■

0.3.4 High Level Overview of what is left over

At this point the scribe is pretty confused. It seems like the authors of this paper do two things:

1) They transform the problem into one about following a path of probability distributions. For a sequence of probability distributions, it is clear how to use coupling to get a stochastic process in X, and one where the expected movement cost is the same as the total earth-movers distance cost for the sequence. For a continuous family of probability distributions, this is less clear. The scribe opened a question: `https://mathoverflow.net/questions/347344/jump-process-with-marginals-given-by-a-curve-of-probability-distributions`. If you know the answer (or maybe where the scribe went wrong in interpreting the paper) an answer there would be appreciated. The scribe gave up understanding the details after this.

2) They find a polytopal lift K of probability simplex on X such that there is a norm in the lifted space that agrees with the W_1 distance after the projection. This is something special about tree metrics.

3) Then they pick the right mirror map to for doing mirror descent on K. It turns out that the right thing to do is to build a weighted and shifted entropy.

4) Here are some sketchy notes on the HST case: Break up the overall problem in hierarchical manner given by the HST, where we are making a decision at the top level of whether to go down a subtree or not. They use a notion of an unfair competitor – This says that OPT pays service cost + movement cost and the algorithm pays α service cost + β movement cost, for $\alpha, \beta \geq 1$. This is relevant in the caseof HSTs because we recursively run MST inside of each component (subtree). Each of these algorithms send up some information that just works on the weighted star of the root, which just works on weighted star of its children. When it decides how much probability to assign to a child, the sub algorithm computes the algorithm on that. Each of the subtree algorithm is suboptimal – they incur some competitive ratios. These are going to be exactly the unfairness ratios that are used in this unfair MTS. It could be a problem that the competitive ratios could multiply as we move up the tree. We handle this by working with a potential function that has learning rate, weights and barrier that depend on the subtree in some clever way. Somehow this works by tuning all these parameters and choose the right averaging of the components of the cost. It's interesting to the scribe that the log sum exp soft max appears in this.