Advanced Course in Machine Learning Spring 2011

Online Learning

Lecturer: Shai Shalev-Shwartz Scribe: Shai Shalev-Shwartz

In this lecture we describe a different model of learning which is called *online* learning. Online learning takes place in a sequence of consecutive rounds. To demonstrate the online learning model, consider again the papaya tasting problem. On each online round, the learner first receives an instance (the learner buys a papaya and knows its shape and color, which form the instance). Then, the learner is required to predict a label (is the papaya tasty?). At the end of the round, the learner gets the correct label (he tastes the papaya and then knows if it's tasty or not). Finally, the learner uses this information to improve his future predictions.

Previously, we used the batch learning model in which we first use a batch of training examples to learn a hypothesis and only when learning is completed the learned hypothesis is tested. In our papayas learning problem, we should first buy bunch of papayas and taste them all. Then, we use all of this information to learn a prediction rule that determines the taste of new papayas. In contrast, in online learning there is no separation between a training phase and a test phase. The same sequence of examples is used both for training and testing and the distinguish between train and test is through time. In our papaya problem, each time we buy a papaya, it is first considered a *test* example since we should predict if it's going to taste good. But, after we take a byte from the papaya, we know the true label, and the same papaya becomes a *training* example that can help us improve our prediction mechanism for future papayas.

The goal of the online learner is simply to make few prediction mistakes. By now, the reader should know that there are no free lunches – we must have some prior knowledge on the problem in order to be able to make accurate predictions. As in previous lectures, we encode our prior knowledge on the problem using some representation of the instances (e.g. shape and color) and by assuming that there is a class of hypotheses, $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$, and on each online round the learner uses a hypothesis from $\mathcal{H}$ to make his prediction.

To simplify our presentation, we start the lecture by describing online learning algorithms for the case of a finite hypothesis class.

1 Online Learning and Mistake Bounds for Finite Hypothesis Classes

Throughout this section we assume that $\mathcal{H}$ is a finite hypothesis class. On round t, the learner first receives an instance, denoted $\mathbf{x}_t$, and is required to predict a label. After making his prediction, the learner receives the true label, $y_t = h^\star(\mathbf{x}_t)$, where $h^\star \in \mathcal{H}$ is a fixed (but unknown to the learner) hypothesis.

The most natural learning rule is to use a consistent hypothesis at each online round. If there are several consistent hypotheses, the `Consistent` algorithm chooses one of them arbitrarily. This corresponds to the ERM learning rule we discussed in batch learning.

It is trivial to see that the number of mistakes the `Consistent` algorithm will make on any sequence is at most $|\mathcal{H}| - 1$. This follows from the fact that whenever `Consistent` errs, at least one hypothesis is no longer consistent with all previous examples. Therefore, after making $|\mathcal{H}| - 1$ mistakes, the only hypothesis which will remain consistent is $h^\star$, and it will never err again. On the other hand, it is easy to construct a sequence of examples on which `Consistent` indeed makes $|\mathcal{H}| - 1$ mistakes (this is left as an exercise).

Next, we define a variant of `Consistent` which has much better mistake bound. On each round, the `RandConsistent` algorithm choose a consistent hypothesis uniformly at random, as there is no reason to prefer one consistent hypothesis over another. This leads to the following algorithm.

Algorithm 1 `RandConsistent`

INPUT: A finite hypothesis class $\mathcal{H}$

INITIALIZE: $V_1 = \mathcal{H}$

FOR $t = 1, 2, \ldots$

Receive $\mathbf{x}_t$

Choose some h from V_t uniformly at random

Predict $\hat{y}_t = h(\mathbf{x}_t)$

Receive true answer y_t

Update $V_{t+1} = \{h \in V_t : h(\mathbf{x}_t) = y_t\}$

The `RandConsistent` algorithm maintains a set, V_t, of all the hypotheses which are consistent with $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{t-1}, y_{t-1})$. It then chooses a hypothesis uniformly at random from V_t and predicts according to this hypothesis.

Recall that the goal of the learner is to make few mistakes. The following theorem upper bounds the expected number of mistakes `RandConsistent` makes on a sequence of examples. To motivate the bound, consider a round t and let α_t be the fraction of hypotheses in V_t which are going to be correct on the example $(\mathbf{x}_t, y_t)$. Now, if α_t is close to 1, it means we are likely to make a correct prediction. On the other hand, if α_t is close to 0, we are likely to make a prediction error. But, on the next round, after updating the set of consistent hypotheses, we will have $|V_{t+1}| = \alpha_t |V_t|$, and since we now assume that α_t is small, we will have a much smaller set of consistent hypotheses in the next round. To summarize, if we are likely to err on the current example then we are going to learn a lot from this example as well, and therefore be more accurate in later rounds.

Theorem 1 *Let $\mathcal{H}$ be a finite hypothesis class, let $h^\star$ be some hypothesis in $\mathcal{H}$ and let $(\mathbf{x}_1, h^\star(\mathbf{x}_1)), \ldots, (\mathbf{x}_T, h^\star(\mathbf{x}_T))$ be an arbitrary sequence of examples. Then, the expected number of mistakes the `RandConsistent` algorithm makes on this sequence is at most $\ln(|\mathcal{H}|)$, where expectation is with respect to the algorithm's own randomization.*

Proof For each round t, let $\alpha_t = |V_{t+1}|/|V_t|$. Therefore, after T rounds we have

$$1 \;\leq\; |V_{T+1}| \;=\; |\mathcal{H}| \prod_{t=1}^{T} \alpha_t \;.$$

Using the inequality $b \leq e^{-(1-b)}$, which holds for all b, we get that

$$\begin{aligned} 1 \;\leq\; |\mathcal{H}| \prod_{t=1}^{T} e^{-(1-\alpha_t)} \;&=\; |\mathcal{H}|\, e^{-\sum_{t=1}^{T}(1-\alpha_t)} \\ \Rightarrow \quad \sum_{t=1}^{T}(1-\alpha_t) \;&\leq\; \ln(|\mathcal{H}|) \;. \end{aligned} \tag{1}$$

Finally, since we predict $\hat{y}_t$ by choosing $h \in V_t$ uniformly at random, we have that the probability to make a mistake on round t is

$$\mathbb{P}[\hat{y}_t \neq y_t] \;=\; \frac{|\{h \in V_t : h(\mathbf{x}_t) \neq y_t\}|}{|V_t|} \;=\; \frac{|V_t| - |V_{t+1}|}{|V_t|} \;=\; (1-\alpha_t) \;.$$

Therefore, the expected number of mistakes is

$$\sum_{t=1}^{T} \mathbb{E}[\mathbb{1}_{[\hat{y}_t \neq y_t]}] \;=\; \sum_{t=1}^{T} \mathbb{P}[\hat{y}_t \neq y_t] \;=\; \sum_{t=1}^{T}(1-\alpha_t) \;.$$

Combining the above with Eq. (1) we conclude our proof. ■

It is interesting to compare the guarantee in Theorem 1 to guarantees on the generalization error in the PAC model (see Corollary **??**). In the PAC model, we refer to the T examples in the sequence as a training set. Then, Corollary **??** implies that with probability of at least $1-\delta$, our average error on new examples is guaranteed to be at most $\ln(|\mathcal{H}|/\delta)/T$. In contrast, Theorem 1 tells us a much stronger guarantee. We do not need to first train the model on T examples, in order to have error rate of $\ln(|\mathcal{H}|)/T$. We can have this same error rate immediately on the first T examples we observe. In our papayas example, we don't need to first buy T papayas, taste them all, and only then to be able to classify new papayas. We can start making predictions from the first day, each day trying to buy a tasty papaya, and we know that our performance after T days will be the same as our performance had we first trained our model using T papayas !

Another important difference between the online model and the batch model is that in the latter we assume that instances are sampled i.i.d. from some underlying distribution, but in the former there is no such an assumption. In particular, Theorem 1 holds for any sequence of instances. Removing the i.i.d. assumption is a big advantage. Again, in the papayas problem, we are allowed to choose a new papaya every day, which clearly violates the i.i.d. assumption. On the flip side, we only have a guarantee on the total number of mistakes but we have no guarantee that after observing T examples we will identify the 'true' hypothesis. Indeed, if we observe the same example on all the online rounds, we will make few mistakes but we will remain with a large set V_t of hypotheses, all of them are potentially the true hypothesis.

Note that the `RandConsistent` algorithm is a specific variant of the general `Consistent` learning paradigm (i.e., ERM) and that the bound in Theorem 1 relies on the fact that we use this specific variant. This stands in contrast to the results we had before for the PAC model in which it doesn't matter how we break ties, and any consistent hypothesis is guaranteed to perform well. In some situations, it is computationally harder to sample a consistent hypothesis from V_t while it is less demanding to merely find one consistent hypothesis. Moreover, if $\mathcal{H}$ is infinite, it is not well defined how to choose a consistent hypothesis uniformly at random. On the other hand, as mentioned before, the results we obtained for the PAC model assume that the data is i.i.d. while the bound for `RandConsistent` holds for any sequence of instances. If the data is indeed generated i.i.d. then it is possible to obtain a bound for the general `Consistent` paradigm.

Theorem 1 bounds the *expected* number of mistakes. Using martingale measure concentration techniques, one can obtain a bound which holds with extremely high probability. A simpler way is to explicitly derandomize the algorithm. Note that `RandConsistent` predicts 1 with probability greater than $1/2$ if the majority of hypotheses in V_t predicts 1. A simple derandomization is therefore to make a deterministic prediction according to a majority vote of the hypotheses in V_t. The resulting algorithm is called `Halving`.

Algorithm 2 `Halving`

INPUT: A finite hypothesis class $\mathcal{H}$

INITIALIZE: $V_1 = \mathcal{H}$

FOR $t = 1, 2, \ldots$

Receive $\mathbf{x}_t$

Predict $\hat{y}_t = \operatorname{argmax}_{r \in \{\pm 1\}} |\{h \in V_t : h(\mathbf{x}_t) = r\}|$

(In case of a tie predict $\hat{y}_t = 1$)

Receive true answer y_t

Update $V_{t+1} = \{h \in V_t : h(\mathbf{x}_t) = y_t\}$

Theorem 2 *Let $\mathcal{H}$ be a finite hypothesis class, let $h^\star$ be some hypothesis in $\mathcal{H}$ and let $(\mathbf{x}_1, h^\star(\mathbf{x}_1)), \ldots, (\mathbf{x}_T, h^\star(\mathbf{x}_T))$ be an arbitrary sequence of examples. Then, the number of mistakes the `Halving` algorithm makes on this sequence is at most $\log_2(|\mathcal{H}|)$.*

Proof We simply note that whenever the algorithm errs we have $|V_{t+1}| \leq |V_t|/2$. (Hence the name Halving.) Therefore, if M is the total number of mistakes, we have

$$1 \leq |V_{T+1}| \leq |\mathcal{H}|\, 2^{-M} \ .$$

Rearranging the above inequality we conclude our proof. ■

A guarantee of the type given in Theorem 2 is called a *Mistake Bound*. Theorem 2 states that `Halving` enjoys a mistake bound of $\log_2(|\mathcal{H}|)$. In the next section, we relax the assumption that all the labels are generated by a hypothesis $h^\star \in \mathcal{H}$.

2 Weighted Majority and Regret Bounds

In the previous section we presented the `Halving` algorithm and analyze its performance by providing a mistake bound. A crucial assumption we relied on is that the data is realizable, namely, the labels in the sequence are generated by some hypothesis $h^\star \in \mathcal{H}$. This is a rather strong assumption on our data and prior knowledge. In this section we relax this assumption.

Recall that the mistake bounds we derived in the previous section do not require the data to be sampled i.i.d. from some distribution. We allow the sequence to be deterministic, stochastic, or even adversarially adaptive to our own behavior (for example, this is the case in spam email filtering). Clearly, learning is impossible if there is no correlation between past and present examples.

In the realizable case, future and past examples are tied together by the common hypothesis, $h^\star \in \mathcal{H}$, that generates all labels. In the non-realizable case, we analyze the performance of an online learner using the notion of *regret*. The learner's regret is the difference between his number of prediction mistakes and the number of mistakes the optimal fixed hypothesis in $\mathcal{H}$ makes on the same sequence of examples. This is termed 'regret' since it measures how 'sorry' the learner is, in retrospect, not to have followed the predictions of the optimal hypothesis. We again use the notation $h^\star$ to denote the hypothesis in $\mathcal{H}$ that makes the least number of mistakes on the sequence of examples. But, now, the labels are not generated by $h^\star$, meaning that for some examples we might have $h^\star(\mathbf{x}_t) \neq y_t$.

As mentioned before, learning is impossible if there is no correlation between past and future examples. However, even in this case, the algorithm can have low regret. In other words, having low regret does not necessarily mean that we will make few mistakes. It only means that the algorithm will be competitive with an oracle, that knows in advance what the data is going to be, and chooses the optimal $h^\star$. If our prior knowledge is adequate, then $\mathcal{H}$ contains a hypothesis that (more or less) explains the data, and then a low regret algorithm is guaranteed to have few mistakes.

We now present an online learning algorithm for the non-realizable case, also called the agnostic case. As in the previous section, we assume that $\mathcal{H}$ is a finite hypothesis class and denote $\mathcal{H} = \{h_1, \ldots, h_d\}$. Recall that the `RandConsistent` algorithm maintains the set V_t of all hypotheses which are consistent with the examples observed so far. Then, it samples a hypothesis from V_t uniformly at random. We can represent the set V_t as a vector $\mathbf{w}_t \in \mathbb{R}^d$, where $w_{t,i} = 1$ if $h_i \in V_t$ and otherwise $w_{t,i} = 0$. Then, the `RandConsistent` algorithm chooses h_i with probability $w_{t,i}/(\sum_j w_{t,j})$, and the vector $\mathbf{w}$ is updated at the end of the round by zeroing all elements corresponding to hypotheses that err on the current example.

If the data is non-realizable, the weight of $h^\star$ will become zero once we encounter an example on which $h^\star$ errs. From this point on, $h^\star$ will not affect the prediction of the `RandConsistent` algorithm. To overcome this problem, one can be less aggressive and instead of zeroing weights of erroneous hypotheses, one can just diminish their weight by scaling down their weight by some $\beta \in (0, 1)$. The resulting algorithm is called `Weighted-Majority`.

Algorithm 3 `Weighted-Majority`

INPUT: Finite hypothesis class $\mathcal{H} = \{h_1, \ldots, h_d\}$; Number of rounds T

INITIALIZE: $\beta = e^{-\sqrt{2\,\ln(d)/T}}$; $\mathbf{w}_1 = (1, \ldots, 1) \in \mathbb{R}^d$

FOR $t = 1, 2, \ldots, T$

Let $Z_t = \sum_{j=1}^{d} w_{t,j}$

Sample a hypothesis $h \in \mathcal{H}$ at random according to $\left(\frac{w_{t,1}}{Z_t}, \ldots, \frac{w_{t,d}}{Z_t}\right)$

Predict $\hat{y}_t = h(\mathbf{x}_t)$

Receive true answer y_t

Update: $\forall j,\;\; w_{t+1,j} = \begin{cases} w_{t,j}\,\beta & \text{if } h_j(\mathbf{x}_t) \neq y_t \\ w_{t,j} & \text{else} \end{cases}$

The following theorem provides an expected regret bound for the algorithm.

Theorem 3 *Let $\mathcal{H}$ be a finite hypothesis class, and let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be an arbitrary sequence of examples. If we run* `Weighted-Majority` *on this sequence we have the following expected regret bound*

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}_{[\hat{y}_t \neq y_t]}\right] - \min_{h \in \mathcal{H}} \sum_{t=1}^{T} \mathbb{1}_{[h(\mathbf{x}_t) \neq y_t]} \;\leq\; \sqrt{0.5\,\ln(|\mathcal{H}|)\,T}\,,$$

where expectation is with respect to the algorithm own randomization.

Proof Let $\eta = \sqrt{2\,\ln(d)/T}$ and note that $w_{t+1,i} = w_{t,i} e^{-\eta\,\mathbb{1}_{[h_i(\mathbf{x}_t) \neq y_t]}}$. Therefore,

$$\ln \frac{Z_{t+1}}{Z_t} = \ln \sum_i \frac{w_{t,i}}{Z_t}\, e^{-\eta \mathbb{1}_{[h_i(\mathbf{x}_t) \neq y_t]}}\,.$$

Hoeffding inequality tells us that if X is a random variable over $[0, 1]$ then

$$\ln \mathbb{E}[e^{-\eta X}] \leq -\eta\,\mathbb{E}[X] + \frac{\eta^2}{8}\;.$$

Since $\mathbf{w}_t / Z_t$ is a probability vector and $\mathbb{1}_{[h_i(\mathbf{x}_t) \neq y_t]} \in [0, 1]$, we can apply Hoeffding's inequality to obtain:

$$\ln \frac{Z_{t+1}}{Z_t} \;\leq\; -\eta \sum_i \frac{w_{t,i}}{Z_t} \mathbb{1}_{[h_i(\mathbf{x}_t) \neq y_t]} + \frac{\eta^2}{8} \;=\; -\eta\,\mathbb{E}[\mathbb{1}_{[\hat{y}_t \neq y_t]}] + \frac{\eta^2}{8}\,.$$

Summing the above inequality over t we get

$$\ln(Z_{T+1}) - \ln(Z_1) = \sum_{t=1}^{T} \ln \frac{Z_{t+1}}{Z_t} \leq -\eta \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}_{[\hat{y}_t \neq y_t]}] + \frac{T\,\eta^2}{8}\,.$$

Next, we lower bound Z_{T+1}. For each i, let M_i be the number of mistakes h_i makes on the entire sequence of T examples. Therefore, $w_{T+1,i} = e^{-\eta M_i}$ and we get that

$$\ln Z_{T+1} = \ln\left(\sum_i e^{-\eta M_i}\right) \geq \ln\left(\max_i e^{-\eta M_i}\right) = -\eta \min_i M_i\,.$$

Combining the above and using the fact that $\ln(Z_1) = \ln(d)$ we get that

$$-\eta \min_i M_i - \ln(d) \;\leq\; -\eta \sum_{t=1}^{T} \mathbb{E}[\mathbb{1}_{[\hat{y}_t \neq y_t]}] + \frac{T\,\eta^2}{8} \;,$$

which can be rearranged as follows:

$$\sum_{t=1}^{T} \mathbb{E}[\mathbb{1}_{[\hat{y}_t \neq y_t]}] - \min_{h \in \mathcal{H}} M_i \;\leq\; \frac{\ln(d)}{\eta} + \frac{\eta\, T}{8} \;.$$

Plugging the value of η into the above concludes our proof. ■

Comparing the regret bound of `Weighted-Majority` in Theorem 3 to the mistake bound of `RandConsistent` given in Theorem 1 we note several differences. First, the result for `Weighted-Majority` holds for any sequence of examples while the mistake bound of `RandConsistent` assumes that the labels are generated by some $h^\star \in \mathcal{H}$. As a result, the bound of `Weighted-Majority` is relative to the minimal number of mistakes a hypothesis $h \in \mathcal{H}$ makes. Second, dividing both bounds by the number of rounds T, we get that the error rate in Theorem 1 decreases as $\ln(|\mathcal{H}|)/T$ while the error rate in Theorem 3 decreases as $\sqrt{\ln(|\mathcal{H}|)/T}$. This is similar to the results we had for PAC learning in the realizable and non-realizable cases.

3 Online Learnability and Littlestone's dimension

So far, we focused on specific hypothesis classes – finite classes or the class of linear separators. We derived specific algorithms for each type of class and analyzed their performance. In this section we take a more general approach, and aim at characterizing online learnability. In particular, we target the following question: what is the optimal online learning algorithm for a given class $\mathcal{H}$.

To simplify our derivation, we again consider the realizable case in which all labels are generated by some hypothesis $h^\star \in \mathcal{H}$. We already encountered worst-case mistake bounds. For completeness, we give here a formal definition.

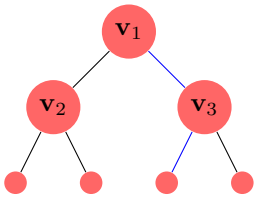
Definition 1 (Mistake bound) *Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\mathcal{Y} = \{\pm 1\}$ and let A be an online algorithm. We say that M is a mistake bound for A, if for any hypothesis $h^\star \in \mathcal{H}$ and a sequence of examples $(\mathbf{x}_1, h^\star(\mathbf{x}_1)), \ldots, (\mathbf{x}_T, h^\star(\mathbf{x}_T))$, the online algorithm A does not make more than M prediction mistakes when running on the sequence of examples.*

Remarks:

- If an algorithm has a mistake bound M it does not necessarily mean that the algorithm will eventually know the identity of $h^\star$, even if we present it with infinite number of examples (e.g. we can feed the online algorithm the same example again and again). The only guarantee that we have is that the total mistakes that the algorithm will make is at most M.
- When the learning algorithm makes randomized predictions, it is also possible to define a mistake bound M as an upper bound on the expected number of mistakes, where expectation is with respect to learner's own randomization. For simplicity, throughout this section we do not analyze expected regret and allow the environment to observe the learner's prediction before providing the label.

In the following we present a dimension of hypothesis classes that characterizes the best possible achievable mistake bound. This measure was proposed by Littleston and we therefore refer to it as $\mathrm{Ldim}(\mathcal{H})$.

To motivate the definition of Ldim it is convenient to view the online learning process as a game between two players: the learner vs. the environment. On round t of the game, the environment picks an instance $\mathbf{x}_t$,



	h_1	h_2	h_3	h_4
$\mathbf{v}_1$	-1	-1	1	1
$\mathbf{v}_2$	-1	1	?	?
$\mathbf{v}_3$	?	?	-1	1

Figure 1: An illustration of a shattered tree of depth 2. The blue path corresponds to the sequence of examples $((\mathbf{v}_1, 1), (\mathbf{v}_3, -1))$. The tree is shattered by $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$, where the predictions of each hypothesis in $\mathcal{H}$ on the instances $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ is given in the table (a question mark means that $h_j(\mathbf{v}_i)$ can be either 1 or -1).

the learner predicts a label $\hat{y}_t \in \{+1, -1\}$, and finally the environment outputs the true label, $y_t \in \{+1, -1\}$. Suppose that the environment wants to make the learner err on the first M rounds of the game. Then, it must output $y_t = -\hat{y}_t$, and the only question is how to choose the instances $\mathbf{x}_t$ in such a way that ensures that for some $h^\star \in \mathcal{H}$ we have $y_t = h^\star(\mathbf{x}_t)$ for all $t \in [M]$.

It makes sense to assume that the environment should pick $\mathbf{x}_t$ based on the previous predictions of the learner, $\hat{y}_1, \ldots, \hat{y}_{t-1}$. Since in our case we have $y_t = -\hat{y}_t$ we can also say that $\mathbf{x}_t$ is a function of $y_1, \ldots, y_{t-1}$. We can represent this dependence using a complete binary tree of depth M (we define the depth of the tree as the number of edges in a path from the root to a leaf). We have $2^{M+1} - 1$ nodes in such a tree, and we attach an instance to each node. Let $\mathbf{v}_1, \ldots, \mathbf{v}_{2^{M+1}-1}$ be these instances. We start from the root of the tree, and set $\mathbf{x}_1 = \mathbf{v}_1$. At round t, we set $\mathbf{x}_t = \mathbf{v}_{i_t}$ where i_t is the current node. At the end of round t, we go to the left child of i_t if $y_t = -1$ or to the right child if $y_t = 1$. That is, $i_{t+1} = 2i_t + \frac{y_t+1}{2}$. Unraveling the recursion we obtain $i_t = 2^{t-1} + \sum_{j=1}^{t-1} \frac{y_j+1}{2} 2^{t-1-j}$.

The above strategy for the environment succeeds only if for any $(y_1, \ldots, y_M)$ there exists $h \in \mathcal{H}$ such that $y_t = h(\mathbf{x}_t)$ for all $t \in [M]$. This leads to the following definition.

Definition 2 ($\mathcal{H}$ Shattered tree) *A shattered tree of depth d is a sequence of instances $\mathbf{v}_1, \ldots, \mathbf{v}_{2^d-1}$ in $\mathcal{X}$ such that for all labeling $(y_1, \ldots, y_d) \in \{\pm 1\}^M$ there exists $h \in \mathcal{H}$ such that for all $t \in [d]$ we have $h(\mathbf{v}_{i_t}) = y_t$ where $i_t = 2^{t-1} + \sum_{j=1}^{t-1} \frac{y_j+1}{2} 2^{t-1-j}$.*

An illustration of a shattered tree of depth 2 is given in Figure 1.

Definition 3 (Littlestone's dimension (Ldim)) $\mathrm{Ldim}(\mathcal{H})$ *is the maximal integer M such that there exist a shattered tree of depth M.*

The definition of Ldim and the discussion above immediately imply the following:

Lemma 1 *If $\mathrm{Ldim}(\mathcal{H}) = M$ then no algorithm can have a mistake bound strictly smaller than M.*

Proof Let $\mathbf{v}_1, \ldots, \mathbf{v}_{2^M-1}$ be a sequence that satisfies the requirements in the definition of Ldim. If the environment sets $\mathbf{x}_t = \mathbf{v}_{i_t}$ and $y_t = -\hat{y}_t$ for all $t \in [M]$, then the learner makes M mistakes while the definition of Ldim implies that there exists a hypothesis $h \in \mathcal{H}$ such that $y_t = h(\mathbf{x}_t)$ for all t. ■

Let us now give several examples.

Example 1 *Let $\mathcal{H}$ be a finite hypothesis class. Clearly, any tree that is shattered by $\mathcal{H}$ has depth of at most $\log_2(|\mathcal{H}|)$. Therefore, $\mathrm{Ldim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$.*

Example 2 *Let $\mathcal{X} = \mathbb{R}$ and $\mathcal{H} = \{x \mapsto \mathrm{sign}(x - a) : a \in \mathbb{R}\}$. Then, $\mathrm{Ldim}(\mathcal{H}) = \infty$. (proof is left as an exercise).*

Example 3 *Let* $X = \{\mathbf{x} \in \{0,1\}^* : \|\mathbf{x}\|_0 \leq r\}$ *and* $\mathcal{H} = \{\mathbf{x} \mapsto \mathit{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) : \|\mathbf{w}\|_0 \leq k\}$. *The size of* $\mathcal{H}$ *is infinite. Nevertheless,* $\mathrm{Ldim}(\mathcal{H}) \leq r\,k$. *(The proof uses the* `Perceptron` *mistake bound and is left as an exercise).*

Lemma 1 states that $\mathrm{Ldim}(\mathcal{H})$ lower bounds the mistake bound of any algorithm. Interestingly, there is a standard algorithm whose mistake bound matches this lower bound. The algorithm is similar to the `Halving` algorithm. Recall that the prediction of `Halving` is according to a majority vote of the hypotheses which are consistent with previous examples. We denoted this set by V_t. Put another way, `Halving` partition V_t into two sets: $V_t^+ = \{h \in V_t : h(\mathbf{x}_t) = 1\}$ and $V_t^- = \{h \in V_t : h(\mathbf{x}_t) = -1\}$. It then predicts according to the larger of the two groups. The rational behind this prediction is that whenever `Halving` makes a mistake it ends up with $|V_{t+1}| \leq 0.5\,|V_t|$.

The optimal algorithm we present below uses the same idea, but instead of predicting according to the larger class, it predicts according to the class with larger Ldim.

Algorithm 4 `Standard Optimal Algorithm (SOA)`

INPUT: A hypothesis class $\mathcal{H}$

INITIALIZE: $V_1 = \mathcal{H}$

FOR $t = 1, 2, \ldots$

Receive $\mathbf{x}_t$

For $r \in \{\pm 1\}$ let $V_t^{(r)} = \{h \in V_t : h(\mathbf{x}_t) = r\}$

Predict $\hat{y}_t = \mathrm{argmax}_r\, \mathrm{Ldim}(V_t^{(r)})$

Receive true answer y_t

Update $V_{t+1} = V_t^{(y_t)}$

The following lemma formally establishes the optimality of the above algorithm.

Lemma 2 `SOA` *enjoys the mistake bound* $M = \mathrm{Ldim}(\mathcal{H})$.

Proof It suffices to prove that whenever the algorithm makes a prediction mistake we have $\mathrm{Ldim}(V_{t+1}) \leq \mathrm{Ldim}(V_t) - 1$. We prove this claim by assuming the contrary, that is, $\mathrm{Ldim}(V_{t+1}) = \mathrm{Ldim}(V_t)$. If this holds true, then the definition of $\hat{y}_t$ implies that $\mathrm{Ldim}(V_t^{(r)}) = \mathrm{Ldim}(V_t)$ for both $r = 1$ and $r = -1$. But, in this case we can construct a tree of depth $\mathrm{Ldim}(V_t) + 1$ that satisfies the requirements given in the definition of Ldim for the class V_t, which leads to the desired contradiction. ■

Combining Lemma 2 and Lemma 1 we obtain:

Corollary 1 *Let* $\mathcal{H}$ *be a hypothesis class with* $\mathrm{Ldim}(\mathcal{H}) = M$. *Then, the standard optimal algorithm enjoys the mistake bound* M *and no other algorithm can have a mistake bound strictly smaller than* M.

3.1 Comparison to VC dimension

In this section we compare the online learning dimension Ldim with the VC dimension we used for determining the learnability of a class in the batch learning model.

To remind the reader, the VC dimension of a class $\mathcal{H}$, denoted $\mathrm{VCdim}(\mathcal{H})$, is the maximal number d such that there are instances $\mathbf{x}_1, \ldots, \mathbf{x}_d$ that are shattered by $\mathcal{H}$. That is, for any sequence of labels $(y_1, \ldots, y_d) \in \{+1, -1\}^d$ there exists a hypothesis $h \in \mathcal{H}$ that gives exactly this sequence of labels.

Theorem 4 *For any class* $\mathcal{H}$ *we have* $\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{Ldim}(\mathcal{H})$.

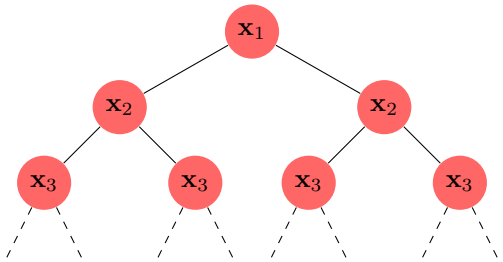


Figure 2: How to construct a shattered tree from a shattered sequence $\mathbf{x}_1, \ldots, \mathbf{x}_d$.

Proof Suppose $\mathrm{VCdim}(\mathcal{H}) = d$ and let $\mathbf{x}_1, \ldots, \mathbf{x}_d$ be a shattered set. We now construct a complete binary tree of instances $\mathbf{v}_1, \ldots, \mathbf{v}_{2^d-1}$, where all nodes at depth i are set to be $\mathbf{x}_i$ (see the illustration in Figure 2). Now, the definition of shattered sample clearly implies that we got a valid shattered tree of depth d, and our proof is completed. ■

Corollary 2 *For a finite hypothesis class, we have*

$$\mathrm{VCdim}(\mathcal{H}) \leq \mathrm{Ldim}(\mathcal{H}) \leq \log(|H|) .$$

Both inequalities in Corollary 2 can be strict as the following examples show.

Example 4 *Consider again the class of initial segments on the real numbers. That is, $\mathcal{X} = \mathbb{R}$ and $\mathcal{H} = \{x \mapsto sign(x - a) : a \in \mathbb{R}\}$. We have shown that the VC dimension of $\mathcal{H}$ is 1 while the Littlestone dimension is ∞.*

Example 5 *Let $\mathcal{X} = \{1, \ldots, d\}$ and $\mathcal{H} = \{h_1, \ldots, h_d\}$ where $h_d(x) = 1$ iff $x = d$. Then, it is easy to show that $\mathrm{Ldim}(\mathcal{H}) = 1$ while $|\mathcal{H}| = d$ can be arbitrarily large.*

Advanced Course in Machine Learning Spring 2011

Online Convex Optimization

Lecturer: Shai Shalev-Shwartz Scribe: Shai Shalev-Shwartz

A convex repeated game is a two players game that is performed in a sequence of consecutive rounds. On round t of the repeated game, the first player chooses a vector $\mathbf{w}_t$ from a convex set A. Next, the second player responds with a convex function $g_t : A \to \mathbb{R}$. Finally, the first player suffers an instantaneous loss $g_t(\mathbf{w}_t)$. We study the game from the viewpoint of the first player.

In offline convex optimization, the goal is to find a vector $\mathbf{w}$ within a convex set A that minimizes a convex objective function, $g : A \to \mathbb{R}$. In online convex optimization, the set A is known in advance, but the objective function may change along the online process. The goal of the online optimizer, which we call the learner, is to minimize the averaged objective value $\frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{w}_t)$, where T is the total number of rounds.

Low regret: Naturally, an adversary can make the cumulative loss of our online learning algorithm arbitrarily large. For example, the second player can always set $g_t(\mathbf{w}) = 1$ and then no matter what the learner will predict, the cumulative loss will be T. To overcome this deficiency, we restate the learner's goal based on the notion of *regret*. The learner's regret is the difference between his cumulative loss and the cumulative loss of the optimal offline minimizer. This is termed 'regret' since it measures how 'sorry' the learner is, in retrospect, not to use the optimal offline minimizer. That is, the regret is

$$R(T) = \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \min_{\mathbf{w}\in A} \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{w}) \ .$$

We call an online algorithm a *low regret* algorithm if $R(T) = o(1)$. In this lecture we will study low regret algorithms for online convex optimization. We will also show how several familiar algorithms, like the Perceptron and Weighted Majority, can be derived from an online convex optimizer.

We start with a brief overview of basic notions form convex analysis.

4 Convexity

A set A is convex if for any two vectors $\mathbf{w}_1, \mathbf{w}_2$ in A, all the line between $\mathbf{w}_1$ and $\mathbf{w}_2$ is also within A. That is, for any $\alpha \in [0,1]$ we have that $\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2 \in A$. A function $f : A \to \mathbb{R}$ is convex if for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ and $\alpha \in [0,1]$ we have

$$f(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) \ \leq \ \alpha f(\mathbf{u}) + (1-\alpha) f(\mathbf{v}) \ .$$

It is easy to verify that f is convex iff its *epigraph* is a convex set, where $\mathrm{epigraph(f)} = \{(\mathbf{x}, \alpha) : f(\mathbf{x}) \leq \alpha\}$.

We allow f to output ∞ for some inputs x. This is a convenient way to restrict the domain of A to a proper subset of $\mathcal{X}$. So, in this section we use $\mathbb{R}$ to denote the reals number and the special symbol ∞. The domain of $f : \mathcal{X} \to \mathbb{R}$ is defined as $\mathrm{dom}(f) = \{x \ : \ f(x) < \infty\}$.

A set A is open if every point in A has a neighborhood lying in A. A set A is closed if its complement is an open set. A function f is closed if for any finite scalar α, the level set $\{\mathbf{w} \ : \ f(\mathbf{w}) \leq \alpha\}$ is closed. Throughout, we focus on closed and convex functions.

4.1 Sub-gradients

A vector $\boldsymbol{\lambda}$ is a *sub-gradient* of a function f at $\mathbf{w}$ if for all $\mathbf{u} \in A$ we have that

$$f(\mathbf{u}) - f(\mathbf{w}) \geq \langle \mathbf{u} - \mathbf{w}, \boldsymbol{\lambda} \rangle \ .$$

The *differential set* of f at $\mathbf{w}$, denoted $\partial f(\mathbf{w})$, is the set of all sub-gradients of f at $\mathbf{w}$. For scalar functions, a sub-gradient of a convex function f at x is a slope of a line that touches f at x and is not above f everywhere.

Two useful properties of subgradients are given below:

1. If f is differentiable at $\mathbf{w}$ then $\partial f(\mathbf{w})$ consists of a single vector which amounts to the *gradient* of f at $\mathbf{w}$ and is denoted by $\nabla f(\mathbf{w})$. In finite dimensional spaces, the gradient of f is the vector of partial derivatives of f.

2. If $g(\mathbf{w}) = \max_{i\in[r]} g_i(\mathbf{w})$ for r differentiable functions $g_1, \ldots, g_r$, and $j = \arg\max_i g_i(\mathbf{u})$, then the gradient of g_j at $\mathbf{u}$ is a subgradient of g at $\mathbf{u}$.

Example 6 (Sub-gradients of the logistic-loss) *Recall that the logistic-loss is defined as* $\ell(\mathbf{w};\mathbf{x},y) = \log(1+\exp(-y\langle\mathbf{w},\mathbf{x}\rangle))$. *Since this function is differentiable, a sub-gradient at* $\mathbf{w}$ *is the gradient at* $\mathbf{w}$*, which using the chain rule equals to*

$$\nabla\ell(\mathbf{w};\mathbf{x},y) = \frac{-\exp(-y\langle\mathbf{w},\mathbf{x}\rangle)}{1+\exp(-y\langle\mathbf{w},\mathbf{x}\rangle)}\, y\,\mathbf{x} = \frac{-1}{1+\exp(y\langle\mathbf{w},\mathbf{x}\rangle)}\, y\,\mathbf{x}\,.$$

Example 7 (Sub-gradients of the hinge-loss) *Recall that the hinge-loss is defined as* $\ell(\mathbf{w};\mathbf{x},y) = \max\{0, 1-y\langle\mathbf{w},\mathbf{x}\rangle\}$. *This is the maximum of two linear functions. Therefore, using the two propoerties above we have that if* $1-y\langle\mathbf{w},\mathbf{x}\rangle > 0$ *then* $-y\,\mathbf{x} \in \partial\ell(\mathbf{w};\mathbf{x},y)$ *and if* $1-y\langle\mathbf{w},\mathbf{x}\rangle < 0$ *then* $\mathbf{0} \in \partial\ell(\mathbf{w};\mathbf{x},y)$. *Furthermore, it is easy to verify that*

$$\partial\ell(\mathbf{w};\mathbf{x},y) = \begin{cases} \{-y\mathbf{x}\} & \text{if } 1-y\langle\mathbf{w},\mathbf{x}\rangle > 0 \\ \{\mathbf{0}\} & \text{if } 1-y\langle\mathbf{w},\mathbf{x}\rangle < 0 \\ \{-\alpha y\mathbf{x} : \alpha\in[0,1]\} & \text{if } 1-y\langle\mathbf{w},\mathbf{x}\rangle = 0 \end{cases}$$

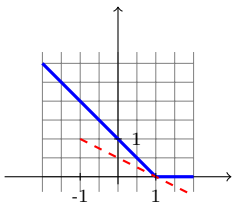


Figure 3: An illustration of the hinge-loss function $f(x) = \max\{0, 1-x\}$ and one of its sub-gradients at $x = 1$.

4.2 Lipschitz functions

We say that $f : A \to \mathbb{R}$ is ρ-Lipschitz if for all $\mathbf{u}, \mathbf{v} \in A$

$$|f(\mathbf{u}) - f(\mathbf{v})| \le \rho\,\|\mathbf{u}-\mathbf{v}\|\,.$$

An equivalent definition is that the ℓ_2 norm of all sub-gradients of f at points in A is bounded by ρ.

More generally, we say that a convex function is V-Lipschitz w.r.t. a norm $\|\cdot\|$ if for all $x \in \mathcal{X}$ exists $v \in \partial f(x)$ with $\|v\|_\star \le V$. Of particular interest are p-norms, $\|x\|_p = (\sum_i |x_i|^p)^{1/p}$.

4.3 Dual norms

Given a norm $\|\cdot\|$, its dual norm is defined by

$$\|y\|_\star = \sup_{x:\|x\|\le 1} \langle x, y\rangle \ .$$

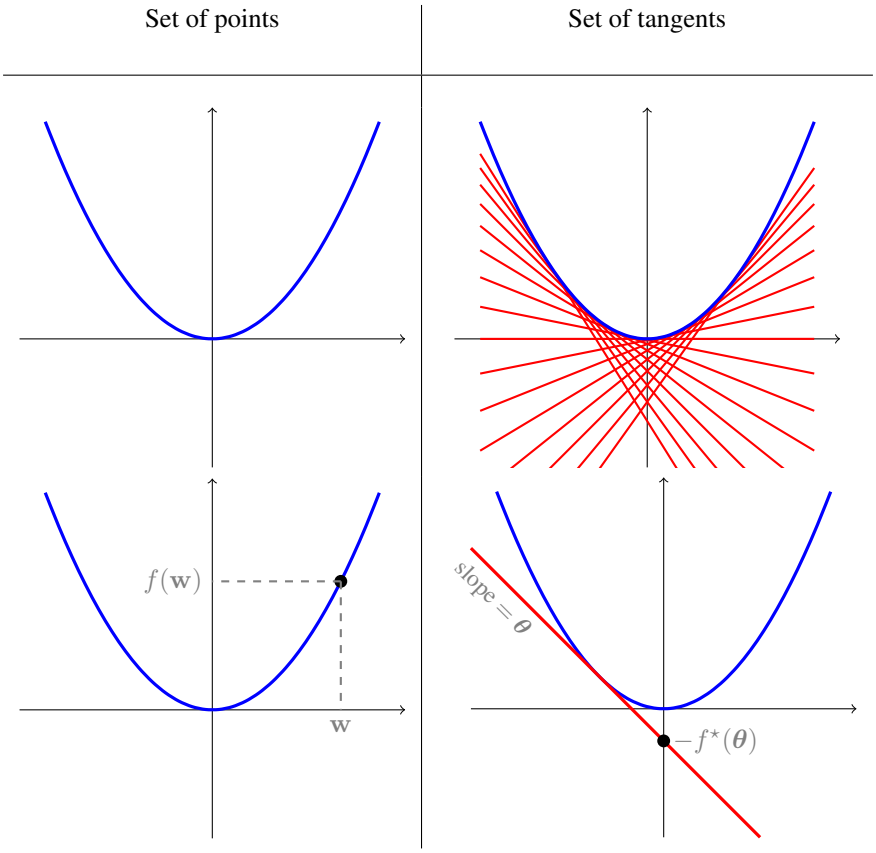
For example, the Euclidean norm is dual to itself. More generally, for any $p, q \ge 1$ such that $1/p + 1/q = 1$, the norms

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{1/p} \quad ; \quad \|x\|_q = \left(\sum_i |x_i|^q\right)^{1/q}$$

are dual norms. The above also holds for $\|x\|_1$ and $\|y\|_\infty = \max_i |y_i|$.

4.4 Fenchel Conjugate

There are two equivalent representations of a convex function. Either as pairs $(x, f(x))$ or as the set of tangents of f, namely pairs (slope,intersection-with-y-axis). The function that relates slopes of tangents to their intersection with the y axis is called the Fenchel conjugate of f.



Formally, the Fenchel conjugate of f is defined as

$$f^\star(\theta) \ = \ \max_x \ \langle x, \theta\rangle - f(x) \ .$$

Few remarks:

- The definition immediately implies **Fenchel-Young inequality**:

$$\begin{aligned} \forall \mathbf{u}, \quad f^\star(\theta) &= \max_x \ \langle x, \theta \rangle - f(x) \\ &\geq \ \langle u, \theta \rangle - f(u) \end{aligned}$$

- If f is closed and convex then $f^{\star\star} = f$
- By the way, this implies Jensen's inequality:

$$\begin{aligned} f(\mathbb{E}[x]) &= \max_\theta \ \langle \theta, \mathbb{E}[x] \rangle - f^\star(\theta) \\ &= \max_\theta \ \mathbb{E}\left[\langle \theta, x \rangle - f^\star(\theta)\right] \\ &\leq \ \mathbb{E}[\ \max_\theta \ \langle \theta, x \rangle - f^\star(\theta)\ ] = \mathbb{E}[f(x)] \end{aligned}$$

Several examples of Fenchel conjugate functions are given below.

$f(x)$	$f^\star(\theta)$
$\frac{1}{2}\|x\|^2$	$\frac{1}{2}\|\theta\|_\star^2$
$\|x\|$	Indicator of unit $\|\cdot\|_\star$ ball
$\sum_i w_i \log(w_i)$	$\log\left(\sum_i e^{\theta_i}\right)$
Indicator of prob. simplex	$\max_i \theta_i$
$c\, g(x)$ for $c > 0$	$c\, g^\star(\theta/c)$
$\inf_{\mathbf{x}} g_1(x) + g_2(x - \mathbf{x})$	$g_1^\star(\theta) + g_2^\star(\theta)$

4.5 Strong Convexity–Strong Smoothness Duality

Recall that the domain of $f : \mathcal{X} \to \mathbb{R}$ is $\{x \ : \ f(x) < \infty\}$. We first define strong convexity.

Definition 4 *A function $f : \mathcal{X} \to \mathbb{R}$ is β-strongly convex w.r.t. a norm $\|\cdot\|$ if for all x, y in the relative interior of the domain of f and $\alpha \in (0,1)$ we have*

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) - \tfrac{1}{2}\beta\alpha(1-\alpha)\|x - y\|^2$$

We now define strong smoothness. Note that a strongly smooth function f is always finite.

Definition 5 *A function $f : \mathcal{X} \to \mathbb{R}$ is β-strongly smooth w.r.t. a norm $\|\cdot\|$ if f is everywhere differentiable and if for all x, y we have*

$$f(x + y) \leq f(x) + \langle \nabla f(x), y \rangle + \tfrac{1}{2}\beta\|y\|^2$$

The following theorem states that strong convexity and strong smoothness are dual properties. Recall that the biconjugate $f^{\star\star}$ equals f if and only if f is closed and convex.

Theorem 5 *(Strong/Smooth Duality) Assume that f is a closed and convex function. Then f is β-strongly convex w.r.t. a norm $\|\cdot\|$ if and only if $f^\star$ is $\frac{1}{\beta}$-strongly smooth w.r.t. the dual norm $\|\cdot\|_\star$.*

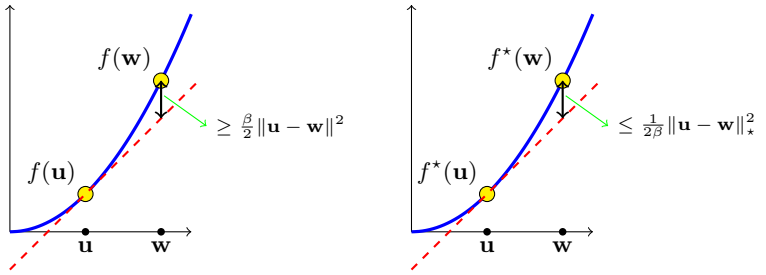


Figure 4: Illustrations of strong convexity (left) and strong smoothness (right).

Subtly, note that while the domain of a strongly convex function f may be a proper subset of $\mathcal{X}$ (important for a number of settings), its conjugate $f^\star$ always has a domain which is $\mathcal{X}$ (since if $f^\star$ is strongly smooth then it is finite and everywhere differentiable). The above theorem can be found, for instance, in Zalinescu 2002 (see Corollary 3.5.11 on p. 217 and Remark 3.5.3 on p. 218).

The following direct corollary of Theorem 5 is central in proving regret bounds. As we will show later, it is also and generalization bounds.

Corollary 3 *If f is β strongly convex w.r.t. $\|\cdot\|$ and $f^\star(\mathbf{0}) = 0$, then, denoting the partial sum $\sum_{j\leq i} v_j$ by $v_{1:i}$, we have, for any sequence $v_1, \ldots, v_n$ and for any u,*

$$\sum_{i=1}^n \langle v_i, u\rangle - f(u) \leq f^\star(v_{1:n}) \leq \sum_{i=1}^n \langle \nabla f^\star(v_{1:i-1}), v_i\rangle + \frac{1}{2\beta}\sum_{i=1}^n \|v_i\|_\star^2 \,.$$

Proof The 1st inequality is Fenchel-Young and the 2nd is from the definition of smoothness by induction. ■

Examples of strongly convex functions

Lemma 3 *Let $q \in [1,2]$. The function $f : \mathbb{R}^d \to \mathbb{R}$ defined as $f(w) = \frac{1}{2}\|w\|_q^2$ is $(q-1)$-strongly convex with respect to $\|\cdot\|_q$ over $\mathbb{R}^d$.*

A proof can be found in Shalev-Shwartz 2007.

We mainly use the above lemma to obtain results with respect to the norms $\|\cdot\|_2$ and $\|\cdot\|_1$. The case $q = 2$ is straightforward. Obtaining results with respect to $\|\cdot\|_1$ is slightly more tricky since for $q = 1$ the strong convexity parameter is 0 (meaning that the function is not strongly convex). To overcome this problem, we shall set q to be slightly more than 1, e.g. $q = \frac{\ln(d)}{\ln(d)-1}$. For this choice of q, the strong convexity parameter becomes $q - 1 = 1/(\ln(d) - 1) \geq 1/\ln(d)$ and the value of p corresponds to the dual norm is $p = (1 - 1/q)^{-1} = \ln(d)$. Note that for any $x \in \mathbb{R}^d$ we have

$$\|x\|_\infty \leq \|x\|_p \leq (d\|x\|_\infty^p)^{1/p} = d^{1/p}\|x\|_\infty = e\,\|x\|_\infty \leq 3\,\|x\|_\infty \,.$$

Hence the dual norms are also equivalent up to a factor of 3: $\|w\|_1 \geq \|w\|_q \geq \|w\|_1/3$. The above lemma therefore implies the following corollary.

Corollary 4 *The function $f : \mathbb{R}^d \to \mathbb{R}$ defined as $f(w) = \frac{1}{2}\|w\|_q^2$ for $q = \frac{\ln(d)}{\ln(d)-1}$ is $1/(9\ln(d))$-strongly convex with respect to $\|\cdot\|_1$ over $\mathbb{R}^d$.*

Another important example is given in the following lemma.

Lemma 4 *The function $f : \mathbb{R}^d \to \mathbb{R}$ defined as $f(x) = \sum_i x_i \log(x_i)$ is 1-strongly convex with respect to $\|\cdot\|_1$ over the domain $\{x : \|x\|_1 = 1, x \geq 0\}$.*

The proof can also be found in Shalev-Shwartz 2007.

5 An algorithmic framework for Online Convex Optimization

Algorithm 5 provides one common algorithm which achieves the following regret bound. It is one of a family of algorithms that enjoy the same regret bound (see Shalev-Shwartz 2007).

Algorithm 5 Online Mirror Descent

Initialize: $w_1 \leftarrow \nabla f^\star(\mathbf{0})$
for $t = 1$ to T
  Play $w_t \in A$
  Receive g_t and pick $v_t \in \partial g_t(w_t)$
  Update $w_{t+1} \leftarrow \nabla f^\star\left(-\eta \sum_{s=1}^t v_t\right)$
end for

Theorem 6 *Suppose Algorithm 5 is used with a function f that is β-strongly convex w.r.t. a norm $\|\cdot\|$ on A and has $f^\star(\mathbf{0}) = 0$. Suppose the loss functions g_t are convex and V-Lipschitz w.r.t. the norm $\|\cdot\|$. Then, the algorithm run with any positive η enjoys the regret bound,*

$$\sum_{t=1}^T g_t(w_t) - \min_{u \in A} \sum_{t=1}^T g_t(u) \leq \frac{\max_{u \in A} f(u)}{\eta} + \frac{\eta V^2 T}{2\beta} \,.$$

In particular, choosing $\eta = \sqrt{\frac{2\beta \max_u f(u)}{V^2 T}}$ we obtain the regret bound

$$\sum_{t=1}^T g_t(w_t) - \min_{u \in A} \sum_{t=1}^T g_t(u) \leq V \sqrt{\frac{2 \max_{u \in A} f(u)\, T}{\beta}} \,.$$

Proof Apply Corollary 3 to the sequence $-\eta v_1, \ldots, -\eta v_T$ to get, for all u,

$$-\eta \sum_{t=1}^T \langle v_t, u \rangle - f(u) \leq -\eta \sum_{t=1}^T \langle v_t, w_t \rangle + \frac{1}{2\beta} \sum_{t=1}^T \|\eta v_t\|_\star^2 \,.$$

Using the fact that g_t is V-Lipschitz, we get $\|v_t\|_\star \leq V$. Plugging this into the inequality above and rearranging gives,

$$\sum_{t=1}^T \langle v_t, w_t - u \rangle \leq \frac{f(u)}{\eta} + \frac{\eta V^2 T}{2\beta} \,.$$

By convexity of g_t, $g_t(w_t) - g_t(u) \leq \langle v_t, w_t - u \rangle$. Therefore,

$$\sum_{t=1}^T g_t(w_t) - \sum_{t=1}^T g_t(u) \leq \frac{f(u)}{\eta} + \frac{\eta V^2 T}{2\beta} \,.$$

Since the above holds for all $u \in A$ the result follows. ■

6 Tightness of regret bounds

In the previous sections we presented algorithmic frameworks for online convex programming with regret bounds that depend on $\sqrt{T}$ and on the complexity of the competing vector as measured by $f(\mathbf{u})$. In this section we show that without imposing additional conditions our bounds are tight, in a sense that is articulated below.

First, we study the dependence of the regret bounds on $\sqrt{T}$.

Theorem 7 *For any online convex programming algorithm, there exists a sequence of* 1*-Lipschitz convex functions of length* T *such that the regret of the algorithm on this sequence with respect to a vector* $\mathbf{u}$ *with* $\|\mathbf{u}\|_2 \leq 1$ *is* $\Omega(\sqrt{T})$.

Proof The proof is based on the probabilistic method. Let $S = [-1, 1]$ and $f(w) = \frac{1}{2}w^2$. We clearly have that $f(w) \leq 1/2$ for all $w \in S$. Consider a sequence of linear functions $g_t(w) = \sigma_t\, w$ where $\sigma_t \in \{+1, -1\}$. Note that g_t is 1-Lipschitz for all t. Suppose that the sequence $\sigma_1, \ldots, \sigma_T$ is chosen in advance, independently with equal probability. Since w_t only depends on $\sigma_1, \ldots, \sigma_{t-1}$ it is independent of σ_t. Thus, $\mathbb{E}_{\sigma_t}[g_t(w_t) \,|\, \sigma_1, \ldots, \sigma_{t-1}] = 0$. On the other hand, for any $\sigma_1, \ldots, \sigma_T$ we have that

$$\min_{u \in S} \sum_{t=1}^{T} g_t(u) \;\leq\; -\left|\sum_{t=1}^{T} \sigma_t\right| \quad .$$

Therefore,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_T}[\text{Regret}] &= \mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_T}[\sum_t g_t(w_t)] - \mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_T}[\min_u \sum_t g_t(u)] \\
&\geq 0 + \mathop{\mathbb{E}}_{\sigma_1,\ldots,\sigma_T}\left[\,|\sum_t \sigma_t|\,\right] \quad .
\end{aligned}$$

The right-hand side above is the expected distance after T steps of a random walk and is thus equal approximately to $\sqrt{2T/\pi} = \Omega(\sqrt{T})$. Our proof is concluded by noting that if the expected value of the regret is greater than $\Omega(\sqrt{T})$, then there must exist at least one specific sequence for which the regret is greater than $\Omega(\sqrt{T})$. ■

Next, we study the dependence on the complexity function $f(\mathbf{w})$.

Theorem 8 *For any online convex programming algorithm, there exists a strongly convex function* f *with respect to a norm* $\|\cdot\|$*, a sequence of* 1*-Lipschitz convex functions with respect to* $\|\cdot\|_\star$*, and a vector* $\mathbf{u}$*, such that the regret of the algorithm on this sequence is* $\Omega(f(\mathbf{u}))$.

Proof Let $S = \mathbb{R}^T$ and $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. Consider the sequence of functions in which $g_t(\mathbf{w}) = [1 - y_t\langle \mathbf{w}, \mathbf{e}_t\rangle]_+$ where $y_t \in \{+1, -1\}$ and $\mathbf{e}_t$ is the tth vector of the standard base, namely, $e_{t,i} = 0$ for all $i \neq t$ and $e_{t,t} = 1$. Note that $g_t(\mathbf{w})$ is 1-Lipschitz with respect to the Euclidean norm. Consider any online learning algorithm. If on round t we have $w_{t,i} \geq 0$ then we set $y_t = -1$ and otherwise we set $y_t = 1$. Thus, $g_t(\mathbf{w}_t) \geq 1$. On the other hand, the vector $\mathbf{u} = (y_1, \ldots, y_T)$ satisfies $f(\mathbf{u}) \leq T/2$ and $g_t(\mathbf{u}) = 0$ for all t. Thus, Regret $\geq T = \Omega(f(\mathbf{u}))$. ■

7 Convexification

The algorithms we derived previously are based on the assumption that for each round t, the loss function $g_t(\mathbf{w})$ is convex with respect to $\mathbf{w}$. A well known example in which this assumption does not hold is online classification with the 0-1 loss function. In this section we discuss to what extent online convex optimization can be used for online learning with the 0-1 loss. In particular, we will show how the Perceptron and Weighted Majority algorithms can be derived from the general online mirror descent framework.

In online binary classification problems, at each round we receive an instance $x \in \mathcal{X} \subset \mathbb{R}^n$ and we need to predict a label $\hat{y}_t \in \mathcal{Y} = \{+1, -1\}$. Then, we receive the correct label $y_t \in \mathcal{Y}$ and suffer the 0-1 loss: $\mathbb{1}_{[y_t \neq \hat{y}_t]}$.

We first show that no algorithm can obtain a sub-linear regret bound for the 0-1 loss function. To do so, let $\mathcal{X} = \{1\}$ so our problem boils down to finding the bias of a coin in an online manner. An adversary can

make the number of mistakes of any online algorithm to be equal to T, by simply waiting for the learner's prediction and then providing the opposite answer as the true answer. In contrast, the number of mistakes of the constant prediction $u = \text{sign}(\sum_t y_t)$ is at most $T/2$. Therefore, the regret of any online algorithm with respect to the 0-1 loss function will be at least $T/2$. This impossibility result is attributed to Cover.

To overcome the above impossibility result, two solutions have been proposed.

7.1 Surrogate convex loss and the Perceptron

The first solution is to find a convex loss function that upper bounds the original non-convex loss function. We describe this solution for the problem of online learning halfspaces. Recall that the set of linear hypotheses is:

$$\mathcal{H} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \|\mathbf{w}\|_2 \leq U\} \ .$$

In the context of binary classification, the actual prediction is $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$ and the $0-1$ loss function of $\mathbf{w}$ on an example $(\mathbf{x}, y)$ is $\ell_{\text{0-1}}(\mathbf{w}, (\mathbf{x}, y)) = \mathbb{1}_{[\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) \neq y]}$.

A popular surrogate convex loss function is the hinge-loss, defined as

$$\ell_{\text{hinge}}(\mathbf{w}, (\mathbf{x}, y)) = [1 - y\langle \mathbf{w}, \mathbf{x} \rangle]_+ \ ,$$

where $[a]_+ = \max\{0, a\}$. It is straightforward to verify that $\ell_{\text{hinge}}(\mathbf{w}, (\mathbf{x}, y))$ is a convex function (w.r.t. $\mathbf{w}$) and that $\ell_{\text{hinge}}(\mathbf{w}, (\mathbf{x}, y)) \geq \ell_{\text{0-1}}(\mathbf{w}, (\mathbf{x}, y))$. Therefore, for any $\mathbf{u} \in A$ we have

$$\begin{aligned} \text{Regret}(T) &= \sum_{t=1}^{T} \ell_{\text{hinge}}(\mathbf{w}_t, (\mathbf{x}_t, y_t)) - \sum_{t=1}^{T} \ell_{\text{hinge}}(\mathbf{u}, (\mathbf{x}_t, y_t)) \\ &\geq \sum_{t=1}^{T} \ell_{\text{0-1}}(\mathbf{w}_t, (\mathbf{x}_t, y_t)) - \sum_{t=1}^{T} \ell_{\text{hinge}}(\mathbf{u}, (\mathbf{x}_t, y_t)) \ . \end{aligned}$$

As a direct corollary from the above inequality we get that a low regret algorithm for the (convex) hinge-loss function can be utilized for deriving an online learning algorithm for the 0-1 loss with the bound

$$\sum_{t=1}^{T} \ell_{\text{0-1}}(\mathbf{w}_t, (\mathbf{x}_t, y_t)) \ \leq \ \sum_{t=1}^{T} \ell_{\text{hinge}}(\mathbf{u}, (\mathbf{x}_t, y_t)) + \text{Regret}(T) \ .$$

Furthermore, denote by $\mathcal{M}$ the set of rounds in which $\text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t \rangle) \neq y_t$ and note that the left-hand side of the above is equal to $|\mathcal{M}|$. We can remove the examples not in $\mathcal{M}$ from the sequence of examples, and run an online convex optimization algorithm only on the sequence of examples in $\mathcal{M}$. In particular, applying Algorithm 5 with $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ we obtain the well known Perceptron algorithm:

Algorithm 6 Perceptron

Initialize: $w_1 \leftarrow \mathbf{0}$
for $t = 1$ to T
  Receive $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(\langle \mathbf{w}_t, \mathbf{x}_t \rangle)$
  Receive y_t
  If $\hat{y}_t \neq y_t$
    Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$
  Else
    Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
  End if
end for

To analyze the Perceptron, we note that an update of the form $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_t \mathbf{x}_t$ will yield the same algorithm, no matter what the value of η is. Therefore, we can use the regret bound given in Theorem 6 on the sequence of round in $\mathcal{M}$ and the set $A = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq U\}$ and get that for any $\mathbf{u}$ with $\|\mathbf{u}\|_2 \leq U$ we have

$$|\mathcal{M}| \;\leq\; \sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(\mathbf{u}, (\mathbf{x}_t, y_t)) + X\, U \sqrt{|\mathcal{M}|}\,, \tag{2}$$

where $X = (\max_{t \in \mathcal{M}} \|\mathbf{x}_t\|_2)$. It is easy to verify that this implies

$$|\mathcal{M}| \;\leq\; \sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(\mathbf{u}, (\mathbf{x}_t, y_t)) + X\, U \sqrt{\sum_{t \in \mathcal{M}} \ell_{\text{hinge}}(\mathbf{u}, (\mathbf{x}_t, y_t))} + X^2 \|\mathbf{u}\|^2 \;\;.$$

Such a bound is called a relative mistake bound.

7.2 Randomization and Weighted Majority

Another way to circumvent Cover's impossibility result is by relying on randomization. We demonstrate this idea using the setting of prediction with expert advice. In this setting, at each round the online learning algorithm receives the advice of d experts, denoted $(f_1^t, \ldots, f_d^t) \in \{0,1\}^d$. Based on the predictions of the experts, the algorithm should predict $\hat{y}_t$. To simplify notation, we assume in this subsection that $\mathcal{Y} = \{0, 1\}$ and not $\{-1, +1\}$.

The following algorithm for prediction with expert advice is due to Littlestone and Warmuth.

Algorithm 7 Learning with Expert Advice (Weighted Majority)

input: Number of experts d ; Learning rate $\eta > 0$
initialize: $\boldsymbol{\theta}^0 = (0, \ldots, 0) \in \mathbb{R}^d$; $Z_0 = d$
for $t = 1, 2, \ldots, T$
receive expert advice $(f_1^t, f_2^t, \ldots, f_d^t) \in \{0,1\}^d$
environment determines y_t without revealing it to learner
Choose i_t at random according to the distribution defined by $w_i^{t-1} = \exp(\theta_i^{t-1})/Z_{t-1}$
predict $\hat{y}_t = f_{i_t}^t$
receive label y_t
update: $\forall i,\ \theta_i^t = \theta_i^{t-1} - \eta |f_i^t - y_t|$; $Z_t = \sum_{i=1}^d \exp(\theta_i^t)$

To analyze the Weighted Majority algorithm we first note that the definition of $\hat{y}_t$ clearly implies:

$$\mathbb{E}[|\hat{y}_t - y_t|] = \sum_{i=1}^{d} w_i^{t-1} |f_i^t - y_t| = \langle \mathbf{w}^{t-1}, \mathbf{v}^t \rangle\,, \tag{3}$$

where $\mathbf{v}_t$ is the vector whose i'th element is $|f_i^t - y_t|$. Based on this presentation, the update rule is equivalent to the update of Algorithm 5 on the sequence of functions $g_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{v}^t \rangle$ with the strongly convex function $f(\mathbf{w}) = \sum_i w_i \log(w_i) + \log(d)$. Letting A be the probabilistic simplex and applying Theorem 6 we obtain that

$$\sum_{t=1}^{T} g_t(\mathbf{w}_t) - \min_{\mathbf{u} \in A} \sum_{t=1}^{T} g_t(\mathbf{u}) \leq \sqrt{2 \log(d) T}\,.$$

In particular, the above holds for any $\mathbf{u} = \mathbf{e}^i$. Combining the above with Eq. (3) we obtain

$$\mathbb{E}\left[\sum_{t=1}^{T} |\hat{y}_t - y_t|\right] \leq \min_i \sum_{t=1}^{T} |f_i^t - y_t| + \sqrt{2 \log(d) T}\,.$$

Remark: Seemingly, we obtained a result w.r.t. the 0-1 loss function, which contradicts Cover's impossibility result. There is no contradiction here because we force the environment to determine y_t before observing the random coins flipped by the learner. In contrast, in Cover's impossibility result, the environment can choose y_t after observing $\hat{y}_t$.

8 Follow the Regularized Leader

In this section we derive a different algorithmic approach in which the weight vector we use in round $t+1$ is defined as

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in A} \sum_{s \le t} g_t(\mathbf{w}) + \frac{1}{\eta} f(\mathbf{w}) \ .$$

This rule is called Follow the Regularized Leader (FTRL). It is also convenient to define an unconstrained version

$$\tilde{\mathbf{w}}_{t+1} = \operatorname*{argmin}_{\mathbf{w}} \sum_{s \le t} g_t(\mathbf{w}) + \frac{1}{\eta} f(\mathbf{w}) \ .$$

The following lemma bounds the regret of FTRL in terms of a stability term.

Lemma 5 *For all $\mathbf{u} \in A$, the regret of FTRL satisfies*

$$\sum_{t=1}^{T} (g_t(\mathbf{w}_t) - g_t(\mathbf{u})) \le \sum_{t=1}^{T} (g_t(\mathbf{w}_t) - g_t(\mathbf{w}_{t+1})) + \eta^{-1}(f(\mathbf{u}) - f(\mathbf{w}_1)) \ .$$

Proof We have

$$g_t(\mathbf{w}_t) - g_t(\mathbf{u}) = g_t(\mathbf{w}_t) - g_t(\mathbf{w}_{t+1}) + g_t(\mathbf{w}_{t+1}) - g_t(\mathbf{u}) \ .$$

Therefore, we need to show that

$$\sum_{t=1}^{T} (g_t(\mathbf{w}_{t+1}) - g_t(\mathbf{u})) \le \eta^{-1}(f(\mathbf{u}) - f(\mathbf{w}_1)) \ .$$

We prove the last inequality by induction. The base case of $T = 0$ holds by the definition of $\mathbf{w}_1$. Now, suppose the statement holds for $T-1$, i.e.

$$\sum_{t=1}^{T-1} g_t(\mathbf{w}_{t+1}) + \eta^{-1} f(\mathbf{w}_1) \le \sum_{t=1}^{T-1} g_t(\mathbf{u}) + \eta^{-1} f(\mathbf{u})$$

Since it holds for any $\mathbf{u} \in A$, it holds for $\mathbf{u} = \mathbf{w}_{T+1}$

$$\sum_{t=1}^{T-1} g_t(\mathbf{w}_{t+1}) + \eta^{-1} f(\mathbf{w}_1) \le \sum_{t=1}^{T-1} g_t(\mathbf{w}_{T+1}) + \eta^{-1} f(\mathbf{w}_{T+1})$$

Adding $g_T(\mathbf{w}_{T+1})$ to both sides,

$$\sum_{t=1}^{T} g_t(\mathbf{w}_{t+1}) + \eta^{-1} f(\mathbf{w}_1) \le \sum_{t=1}^{T} g_t(\mathbf{w}_{T+1}) + \eta^{-1} f(\mathbf{w}_{T+1})$$

The induction step follows from the above since $\mathbf{w}_{T+1}$ is the minimizer of the right-hand side over all $\mathbf{w} \in A$. ■

To derive a concrete bound from the above, one needs to show that the term $\sum_t g_t(\mathbf{w}_{t+1}) - g_t(\mathbf{w}_t)$, which measures the stability of the algorithm, is small. This can be done assuming that $f(\mathbf{w})$ is strongly convex and g_t is Lipschitz, and one can derive bounds which are similar to the bounds for online mirror descent. In the next section, we derive another family of bounds which involves local norms.

9 Another Mirror Descent Algorithm

We now present a slightly different mirror descent algorithm. We will show bounds that depend on local norms, which will later on help us in deriving bounds for online learning in the limited feedback setting. Throughout, we consider the problem of online linear optimization — this is the special case of online convex optimization in which the functions are linear,[1] , and take the form $g_t(\mathbf{w}) = \langle \mathbf{w}, \mathbf{x}_t \rangle$.

The algorithm depends on a regularization function $f : A \to \mathbb{R}$ and the definition of the update is based on the Bregman divergence associated with f.

Definition 6 (Bregman divergence) *Given a convex differentiable function f, the Bregman divergence associated with f is defined as*

$$D_f(\mathbf{u}, \mathbf{v}) = f(\mathbf{u}) - f(\mathbf{v}) - \langle \nabla f(v), \mathbf{u} - \mathbf{v} \rangle \ .$$

Algorithm 8 Online Mirror Descent II

Initialize: $\mathbf{w}_1 = \tilde{\mathbf{w}}_1 = \operatorname{argmin}_{\mathbf{w} \in A} f(\mathbf{w})$
for $t = 1$ to T
  Play $\mathbf{w}_t$
  Get $\mathbf{x}_t$ and pay $\langle \mathbf{w}_t, \mathbf{x}_t \rangle$
  Let $\tilde{\mathbf{w}}_{t+1} = \operatorname{argmin}_{\mathbf{w}} \eta \langle \mathbf{x}_t, \mathbf{w} \rangle + D_f(\mathbf{w}, \mathbf{w}_t)$
  Update $\mathbf{w}_{t+1} = \operatorname{argmin}_{\mathbf{w} \in A} D_f(\mathbf{w}, \tilde{\mathbf{w}}_{t+1})$
end for

To analyze the algorithm we start with the following lemma.

Lemma 6 *For all $\mathbf{u} \in A$:*

$$\sum_{t=1}^{T} \langle \mathbf{x}_t, \mathbf{w}_t - \mathbf{u} \rangle \leq \sum_{t=1}^{T} \langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1} \rangle + \frac{f(\mathbf{u}) - f(\tilde{\mathbf{w}}_1)}{\eta} \ .$$

Proof By optimality condition and the definition of Bregman divergence we have

$$\eta \mathbf{x}_t + \nabla f(\tilde{\mathbf{w}}_{t+1}) - \nabla f(\mathbf{w}_t) = \mathbf{0} \ \ \Rightarrow \ \ \nabla f(\mathbf{w}_t) - \nabla f(\tilde{\mathbf{w}}_{t+1}) = \eta \mathbf{x}_t \ .$$

Therefore

$$\begin{aligned} \eta \langle \mathbf{x}_t, \mathbf{w}_t - \mathbf{u} \rangle &= \langle \nabla f(\mathbf{w}_t) - \nabla f(\tilde{\mathbf{w}}_{t+1}), \mathbf{w}_t - \mathbf{u} \rangle \\ &= D_f(\mathbf{u}, \mathbf{w}_t) - D_f(\mathbf{u}, \tilde{\mathbf{w}}_{t+1}) + D_f(\mathbf{w}_t, \tilde{\mathbf{w}}_{t+1}) \ , \end{aligned}$$

where the last equality follows from the definition of the Bregman divergence. Since $\mathbf{w}_t$ is the projection of $\tilde{\mathbf{w}}_t$ on A and since $\mathbf{u} \in A$ we know from the Bregman projection lemma (see Censor and Lent book or Cesa-Bianchi and Lugosi) that $D_f(\mathbf{u}, \mathbf{w}_t) \leq D_f(\mathbf{u}, \tilde{\mathbf{w}}_{t+1})$. Thus,

$$\eta \langle \mathbf{x}_t, \mathbf{w}_t - \mathbf{u} \rangle \leq D_f(\mathbf{u}, \tilde{\mathbf{w}}_t) - D_f(\mathbf{u}, \tilde{\mathbf{w}}_{t+1}) + D_f(\mathbf{w}_t, \tilde{\mathbf{w}}_{t+1}) \ .$$

Furthermore, from the non-negativity of the Bregman divergence we have

$$\begin{aligned} D_f(\mathbf{w}_t, \tilde{\mathbf{w}}_{t+1}) &\leq D_f(\mathbf{w}_t, \tilde{\mathbf{w}}_{t+1}) + D_f(\tilde{\mathbf{w}}_{t+1}, \mathbf{w}_t) \\ &= \langle \nabla f(\mathbf{w}_t) - \nabla f(\tilde{\mathbf{w}}_{t+1}), \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1} \rangle \\ &= \langle \eta \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1} \rangle \ . \end{aligned}$$

[1] The linearity assumption is not critical since we can reduce a general convex function to a linear function (this is left as an exercise).

Combining the above two inequalities and summing over t we obtain

$$\begin{aligned}\sum_{t=1}^{T}\eta\langle \mathbf{x}_t, \mathbf{w}_t - \mathbf{u}\rangle &\le \sum_{t=1}^{T}(D_f(\mathbf{u},\tilde{\mathbf{w}}_t) - D_f(\mathbf{u},\tilde{\mathbf{w}}_{t+1})) + \sum_{t=1}^{T}\eta\langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1}\rangle \\ &= D_f(\mathbf{u},\tilde{\mathbf{w}}_1) - D_f(\mathbf{u},\tilde{\mathbf{w}}_{T+1}) + \sum_{t=1}^{T}\eta\langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1}\rangle \\ &\le f(\mathbf{u}) - f(\tilde{\mathbf{w}}_1) + \sum_{t=1}^{T}\eta\langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1}\rangle \ .\end{aligned}$$

Dividing the above by η we conclude our proof. ■

To derive concrete bounds from the above we need to further upper bound the sum over $\langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1}\rangle$. We start with a simple example.

Example 8 *Let $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$. Then $\nabla f(\mathbf{w}) = \mathbf{w}$ and $D_f(\mathbf{w},\mathbf{u}) = \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2$. Therefore, $\tilde{\mathbf{w}}_{t+1} = \mathbf{w}_t - \eta\mathbf{x}_t$ and using Cauchy-Schwartz inequality we obtain*

$$\langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1}\rangle = \langle \mathbf{x}_t, \eta\mathbf{x}_t\rangle = \eta\|\mathbf{x}_t\|_2^2 \ .$$

Thus, if $\|\mathbf{x}_t\|_2 \le B_x$ for all t and if A is contained in the ℓ_2 ball of radius B_w we obtain the regret bound $\eta T B_x^2 + \frac{B_w^2}{2\eta}$. Optimizing over η gives

$$\sum_{t=1}^{T}\langle \mathbf{x}_t, \mathbf{w}_t - \mathbf{u}\rangle \le B_w B_x\sqrt{2T} \ .$$

Next, we derive an improved bound for the entropy regularization.

Example 9 *Let $f(\mathbf{w}) = \sum_i w_i\log(w_i)$ and $A = \{\mathbf{w} \ge \mathbf{0} : \|\mathbf{w}\|_1 = 1\}$. Then $\tilde{\mathbf{w}}_1 = (1/d,\ldots,1/d)$ and $f(\tilde{\mathbf{w}}_1) = -\log(d)$. In addition, for all $\mathbf{u} \in A$ we have $f(\mathbf{u}) \le 0$. By optimality condition we have*

$$\nabla f(\tilde{\mathbf{w}}_{t+1}) = \nabla f(\mathbf{w}_t) - \eta\mathbf{x}_t \ .$$

Since $\nabla_i f(\mathbf{w}) = \log(w_i) + 1$ the above implies:

$$\log(\tilde{w}_{t+1,i}) = \log(w_{t,i}) - \eta x_{t,i} \ \ \Rightarrow \ \ \tilde{w}_{t+1,i} = w_{t,i}\,\exp(-\eta x_{t,i}) \ .$$

In addition, it is easy to verify that $w_{t+1,i} = \tilde{w}_{t,i}/\|\tilde{\mathbf{w}}\|_1$. Therefore, we obtained the Weighted Majority algorithm. To analyze the algorithm let us assume that $x_{t,i} \ge 0$ for all i. Using the inequality $1-\exp(-a) \le a$ we obtain

$$\begin{aligned}\langle \mathbf{x}_t, \mathbf{w}_t - \tilde{\mathbf{w}}_{t+1}\rangle &= \sum_i x_{t,i}w_{t,i}(1-\exp(-\eta x_{t,i})) \\ &\le \eta\sum_i w_{t,i}x_{t,i}^2 \ .\end{aligned}$$

We can further bound the above by $\max_i x_{t,i}^2$, but if $x_{t,i}$ also depends on $w_{t,i}$ we will obtain an improved bound. This will be the topic of the next lecture.