11

Differential Privacy and Machine Learning

One of the most useful tasks in data analysis is machine learning: the problem of automatically finding a simple rule to accurately predict certain unknown characteristics of never before seen data. Many machine learning tasks can be performed under the constraint of differential privacy. In fact, the constraint of privacy is not necessarily at odds with the goals of machine learning, both of which aim to extract information from the distribution from which the data was drawn, rather than from individual data points. In this section, we survey a few of the most basic results on private machine learning, without attempting to cover this large field completely.

The goal in machine learning is very often similar to the goal in private data analysis. The *learner* typically wishes to learn some simple rule that explains a data set. However, she wishes this rule to generalize — that is, it should be that the rule she learns not only correctly describes the data that she has on hand, but that it should also be able to correctly describe *new* data that is drawn from the same distribution. Generally, this means that she wants to learn a rule that captures distributional information about the data set on hand, in a way that does not depend too specifically on any single data point. Of

course, this is exactly the goal of private data analysis — to reveal *distributional information* about the private data set, without revealing too much about any single individual in the dataset. It should come as no surprise then that machine learning and private data analysis are closely linked. In fact, as we will see, we are often able to perform private machine learning *nearly as accurately, with nearly the same number of examples* as we can perform non-private machine learning.

Let us first briefly define the problem of machine learning. Here, we will follow Valiant's *PAC* (Or *Probably Approximately Correct*) model of machine learning. Let $\mathcal{X} = \{0,1\}^d$ be the domain of "unlabeled examples." Think of each $x \in \mathcal{X}$ as a vector containing d boolean attributes. We will think of vectors $x \in \mathcal{X}$ as being paired with *labels* $y \in \{0,1\}$.

Definition 11.1. A *labeled example* is a pair $(x, y) \in \mathcal{X} \times \{0,1\}$: a vector paired with a label.

A learning problem is defined as a distribution $\mathcal{D}$ over labeled examples. The goal will to be to find a function $f : \mathcal{X} \to \{0,1\}$ that correctly labels almost all of the examples drawn from the distribution.

Definition 11.2. Given a function $f : \mathcal{X} \to \{0,1\}$ and a distribution $\mathcal{D}$ over labeled examples, the *error rate* of f on $\mathcal{D}$ is:

$$\mathrm{err}(f, \mathcal{D}) = \Pr_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$$

We can also define the error rate of f over a finite sample D:

$$\mathrm{err}(f, D) = \frac{1}{|D|}|\{(x, y) \in D : f(x) \neq y\}|.$$

A learning *algorithm* gets to observe some number of labeled examples drawn from $\mathcal{D}$, and has the goal of finding a function f with as small an error rate as possible when measured on $\mathcal{D}$. Two parameters in measuring the quality of a learning algorithm are its running time, and the number of examples it needs to see in order to find a good hypothesis.

Definition 11.3. An algorithm A is said to PAC-learn a class of functions C over d dimensions if for every $\alpha, \beta > 0$, there exists an

$m = \text{poly}(d, 1/\alpha, \log(1/\beta))$ such that for every distribution $\mathcal{D}$ over labeled examples, A takes as input m labeled examples drawn from $\mathcal{D}$ and outputs a hypothesis $f \in C$ such that with probability $1 - \beta$:

$$\text{err}(f, \mathcal{D}) \leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + \alpha$$

If $\min_{f^* \in C} \text{err}(f^*, \mathcal{D}) = 0$, the learner is said to operate in the *realizable* setting (i.e., there exists some function in the class which perfectly labels the data). Otherwise, the learner is said to operate in the *agnostic* setting. If A also has run time that is polynomial in $d, 1/\alpha$, and $\log(1/\beta)$, then the learner is said to be *efficient*. If there is an algorithm which PAC-learns C, then C is said to be PAC-learnable.

The above definition of learning allows the learner to have direct access to labeled examples. It is sometimes also useful to consider models of learning in which the algorithm only has oracle access to some noisy information about $\mathcal{D}$.

Definition 11.4. A *statistical query* is some function $\phi : \mathcal{X} \times \{0, 1\} \to [0, 1]$. A *statistical query oracle* for a distribution over labeled examples $\mathcal{D}$ with tolerance τ is an oracle $\mathcal{O}^\tau_\mathcal{D}$ such that for every statistical query ϕ:

$$\left| \mathcal{O}^\tau_\mathcal{D}(\phi) - \mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x, y)] \right| \leq \tau$$

In other words, an SQ oracle takes as input a statistical query ϕ, and outputs some value that is guaranteed to be within $\pm\tau$ of the expected value of ϕ on examples drawn from $\mathcal{D}$.

The statistical query model of learning was introduced to model the problem of learning in the presence of noise.

Definition 11.5. An algorithm A is said to SQ-learn a class of functions C over d dimensions if for every $\alpha, \beta > 0$ there exists an $m = \text{poly}(d, 1/\alpha, \log(1/\beta))$ such that A makes at most m queries of tolerance $\tau = 1/m$ to $\mathcal{O}^\tau_\mathcal{D}$, and with probability $1-\beta$, outputs a hypothesis $f \in C$ such that:

$$\text{err}(f, \mathcal{D}) \leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + \alpha$$

Note that an SQ learning algorithm does not get any access to $\mathcal{D}$ except through the SQ oracle. As with PAC learning, we can talk about an SQ learning algorithm operating in either the realizable or the agnostic setting, and talk about the computational efficiency of the learning algorithm. We say that a class C is SQ learnable if there exists an SQ learning algorithm for C.

11.1 The sample complexity of differentially private machine learning

Perhaps the first question that one might ask, with respect to the relationship between privacy and learning, is "When is it possible to privately perform machine learning"? In other words, you might ask for a PAC learning algorithm that takes as input a dataset (implicitly assumed to be sampled from some distribution $\mathcal{D}$), and then privately output a hypothesis f that with high probability has low error over the distribution. A more nuanced question might be, "How many *additional* samples are required to privately learn, as compared with the number of samples *already required* to learn without the constraint of differential privacy?" Similarly, "How much additional run-time is necessary to privately learn, as compared with the run-time required to learn non-privately?" We will here briefly sketch known results for $(\varepsilon, 0)$-differential privacy. In general, better results for (ε, δ)-differential privacy will follow from using the advanced composition theorem.

A foundational *information theoretic result* in private machine learning is that private PAC learning is possible with a polynomial number of samples if and only if non-private PAC learning is possible with a polynomial number of samples, even in the agnostic setting. In fact, the increase in sample complexity necessary is relatively small — however, this result does not preserve *computational efficiency*. One way to do this is directly via the exponential mechanism. We can instantiate the exponential mechanism with a range $R = C$, equal to the class of queries to be learned. Given a database D, we can use the quality score $q(f, D) = -\frac{1}{|D|}|\{(x, y) \in D : f(x) \neq y\}|$: i.e., we seek to minimize the fraction of misclassified examples in the private dataset. This is clearly

a $1/n$ sensitive function of the private data, and so we have via our utility theorem for the exponential mechanism that with probability $1-\beta$, this mechanism returns a function $f \in C$ that correctly labels an $\text{OPT} - \frac{2(\log|C|+\log\frac{1}{\beta})}{\varepsilon n}$ fraction of the points in the database correctly. Recall, however, that in the learning setting, we view the database D as consisting of n i.i.d. draws from some distribution over labeled examples $\mathcal{D}$. Recall the discussion of sampling bounds in Lemma 4.3. A Chernoff bound combined with a union bound tells us that with high probability, if D consists of n i.i.d. samples drawn from $\mathcal{D}$, then for all $f \in C$: $|\text{err}(f, D) - \text{err}(f, \mathcal{D})| \leq O(\sqrt{\frac{\log|C|}{n}})$. Hence, if we wish to find a hypothesis that has error within α of the optimal error on the distribution $\mathcal{D}$, it suffices to draw a database D consisting of $n \geq \log|C|/\alpha^2$ samples, and learn the best classifier f^* on D.

Now consider the problem of privately PAC learning, using the exponential mechanism as described above. Recall that, by Theorem 3.11, it is highly unlikely that the exponential mechanism will return a function f with utility score that is inferior to that of the optimal f^* by more than an additive factor of $O((\Delta u/\varepsilon)\log|C|)$, where in this case Δu, the sensitivity of the utility function, is $1/n$. That is, with high probability the exponential mechanism will return a function $f \in C$ such that:

$$\begin{aligned}\text{err}(f, D) &\leq \min_{f^* \in C} \text{err}(f^*, D) + O\left(\frac{(\log|C|)}{\varepsilon n}\right) \\ &\leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + O\left(\sqrt{\frac{\log|C|}{n}}\right) + O\left(\frac{(\log|C|)}{\varepsilon n}\right).\end{aligned}$$

Hence, if we wish to find a hypothesis that has error within α of the optimal error on the distribution $\mathcal{D}$, it suffices to draw a database D consisting of:

$$n \geq O\left(\max\left(\frac{\log|C|}{\varepsilon\alpha}, \frac{\log|C|}{\alpha^2}\right)\right),$$

which is not asymptotically any more than the database size that is required for non-private learning, whenever $\varepsilon \geq \alpha$.

A corollary of this simple calculation[1] is that (ignoring computational efficiency), a class of functions C is PAC learnable if and only if it is privately PAC learnable.

Can we say something stronger about a concept class C that is SQ learnable? Observe that if C is efficiently SQ learnable, then the learning algorithm for C need only access the data through an SQ oracle, which is very amenable to differential privacy: note that an SQ oracle answers an expectation query defined over a predicate $\phi(x, y) \in [0, 1]$, $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x, y)]$, which is only $1/n$ sensitive when estimated on a database D which is a sample of size n from $\mathcal{D}$. Moreover, the learning algorithm does not need to receive the answer exactly, but can be run with any answer a that has the property that: $|\mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x, y)]-a| \leq \tau$: that is, the algorithm can be run using *noisy answers* on *low sensitivity queries*. The benefit of this is that we can answer such queries computationally efficiently, using the Laplace mechanism — but at the expense of requiring a potentially large sample size. Recall that the Laplace mechanism can answer m $1/n$ sensitive queries with $(\varepsilon, 0)$-differential privacy and with expected worst-case error $\alpha = O(\frac{m \log m}{\varepsilon n})$. Therefore, an *SQ* learning algorithm which requires the answers to m queries with accuracy α can be run with a sample size of $n = O(\max(\frac{m \log m}{\varepsilon\alpha}, \frac{\log m}{\alpha^2}))$. Let us compare this to the sample size required for a non-private *SQ* learner. If the *SQ* learner needs to make m queries to tolerance α, then by a Chernoff bound and a union bound, a sample size of $O(\log m/\alpha^2)$ suffices. Note that for $\varepsilon = O(1)$ and error $\alpha = O(1)$, the non-private algorithm potentially requires exponentially fewer samples. However, at the error tolerance $\alpha \leq 1/m$ as allowed in the definition of SQ learning, the sample complexity for private SQ learning is no worse than the sample complexity for non-private SQ learning, for $\epsilon = \Theta(1)$.

The upshot is that *information theoretically*, privacy poses very little hinderance to machine learning. Moreover, for any algorithm that accesses the data only though an SQ oracle,[2] then the reduction to

[1] Together with corresponding lower bounds that show that for general C, it is not possible to non-privately PAC learn using a sample with $o(\log |C|/\alpha^2)$ points.

[2] And in fact, almost every class (with the lone exception of *parity functions*) of functions known to be PAC learnable is also learnable using only an SQ oracle.

private learning is immediate via the Laplace mechanism, and preserves computational efficiency!

11.2 Differentially private online learning

In this section, we consider a slightly different learning problem, known as the problem of *learning from expert advice*. This problem will appear somewhat different from the classification problems that we discussed in the previous section, but in fact, the simple algorithm presented here is extremely versatile, and can be used to perform classification among many other tasks which we will not discuss here.

Imagine that you are betting on horse races, but unfortunately know nothing about horses! Nevertheless, you have access to the opinions of some k *experts*, who every day make a prediction about which horse is going to win. Each day you can choose one of the experts whose advice you will follow, and each day, following your bet, you learn which horse actually won. How should you decide which expert to follow each day, and how should you evaluate your performance? The experts are not perfect (in fact they might not even be any good!), and so it is not reasonable to expect you to make the correct bet all of the time, or even most of the time if none of the experts do so. However, you might have a weaker goal: can you bet on horses in such a way so that you do almost as well as *the best expert, in hindsight*?

Formally, an online learning algorithm A operates in the following environment:

1. Each day $t = 1, \ldots, T$:
   (a) A chooses an expert $a_t \in \{1, \ldots, k\}$
   (b) A observes a loss $\ell_i^t \in [0, 1]$ for each expert $i \in \{1, \ldots, k\}$ and experiences loss $\ell_{a_t}^t$.

For a sequence of losses $\ell^{\leq T} \equiv \{\ell^t\}_{t=1}^T$, we write:

$$L_i(\ell^{\leq T}) = \frac{1}{T} \sum_{t=1}^{T} \ell_i^t$$

to denote the total average loss of expert i over all T rounds, and write

$$L_A(\ell^{\leq T}) = \frac{1}{T}\sum_{t=1}^{T} \ell_{a_t}^t$$

to denote the total average loss of the algorithm.

The *regret* of the algorithm is defined to be the difference between the loss that it actually incurred, and the loss of the *best* expert in hindsight:

$$\text{Regret}(A, \ell^{\leq T}) = L_A(\ell^{\leq T}) - \min_i L_i(\ell^{\leq T}).$$

The goal in online learning is to design algorithms that have the guarantee that for *all possible loss sequences* $\ell^{\leq T}$, even adversarialy chosen, the regret is guaranteed to tend to zero as $T \to \infty$. In fact, this is possible using the multiplicative weights algorithm (known also by many names, e.g., the Randomized Weighted Majority Algorithm, Hedge, Exponentiated Gradient Descent, and multiplicative weights being among the most popular).

Remark 11.1. We have already seen this algorithm before in Section 4 — this is just the multiplicative weights update rule in another guise! In fact, it would have been possible to derive all of the results about the private multiplicative weights mechanism directly from the regret bound we state in Theorem 11.1.

Algorithm 15 The Multiplicative Weights (or Randomized Weighted Majority (RWM)) algorithm, version 1. It takes as input a stream of losses $\ell^1, \ell^2, \ldots$ and outputs a stream of actions $a_1, a_2, \ldots$. It is parameterized by an update parameter η.

RWM(η):

For each $i \in \{1, \ldots, k\}$, let $w_i \leftarrow 1$.
for $t = 1, \ldots$ **do**
  Choose action $a_t = i$ with probability proportional to w_i
  Observe ℓ^t and set $w_i \leftarrow w_i \cdot \exp(-\eta \ell_i^t)$, for each $i \in [k]$
end for

It turns out that this simple algorithm already has a remarkable regret bound.

Theorem 11.1. For any adversarially chosen sequence of losses of length T, $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$ the Randomized Weighted Majority algorithm with update parameter η has the guarantee that:

$$\mathbb{E}[\text{Regret}(\text{RWM}(\eta), \ell^{\leq T})] \leq \eta + \frac{\ln(k)}{\eta T}, \tag{11.1}$$

where k is the number of experts. Choosing $\eta = \sqrt{\frac{\ln k}{T}}$ gives:

$$\mathbb{E}[\text{Regret}(\text{RWM}(\eta), \ell^{\leq T})] \leq 2\sqrt{\frac{\ln k}{T}}.$$

This remarkable theorem states that even faced with an adversarial sequence of losses, the Randomized Weighted Majority algorithm can do as well, on average, as the best expert among k in hindsight, minus only an additional additive term that goes to zero at a rate of $O(\sqrt{\frac{\ln k}{T}})$. In other words, after at most $T \leq 4\frac{\ln k}{\alpha^2}$ rounds, the regret of the randomized weighted majority algorithm is guaranteed to be at most α! Moreover, this bound is the best possible.

Can we achieve something similar, but under the constraint of differential privacy? Before we can ask this question, we must decide *what is the input database*, and at what granularity we would like to protect privacy? Since the input is the collection of loss vectors $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$, it is natural to view $\ell^{\leq T}$ as the database, and to view a neighboring database $\hat{\ell^{\leq T}}$ as one that differs in the entire loss vector in any single timestep: i.e., one in which for some fixed timestep t, $\hat{\ell}^i = \ell^i$ for all $i \neq t$, but in which ℓ^t and $\hat{\ell}^t$ can differ arbitrarily. The output of the algorithm is the sequence of actions that it chooses, $a_1, \ldots, a_T$, and it is this that we wish to be output in a differentially private manner.

Our first observation is that the randomized weighted majority algorithm chooses an action at each day t in a familiar manner! We here rephrase the algorithm in an equivalent way:

It chooses an action a_t with probability proportional to: $\exp(-\eta \sum_{j=1}^{t-1} \ell_i^j)$, which is simply the exponential mechanism with quality score $q(i, \ell^{<T}) = \sum_{j=1}^{t-1} \ell_i^j$, and privacy parameter $\varepsilon = 2\eta$. Note that because each $\ell_i^t \in [0, 1]$, the quality function has sensitivity 1. Thus,

Algorithm 16 The Multiplicative Weights (or Randomized Weighted Majority (RWM)) algorithm, rephrased. It takes as input a stream of losses $\ell^1, \ell^2, \ldots$ and outputs a stream of actions $a_1, a_2, \ldots$. It is parameterized by an update parameter η.

RWM(η):
  for $t = 1, \ldots$ **do**
    Choose action $a_t = i$ with probability proportional to $\exp(-\eta \sum_{j=1}^{t-1} \ell_i^j)$
    Observe ℓ^t
  end for

each round t, the randomized weighted majority algorithm chooses an action a_t in a way that preserves 2η differential privacy, so to achieve privacy ε it suffices to set $\eta = \varepsilon/2$.

Moreover, over the course of the run of the algorithm, it will choose an action T times. If we want the *entire* run of the algorithm to be (ε, δ)-differentially private for some ε and δ, we can thus simply apply our composition theorems. Recall that by Theorem 3.20, since there are T steps in total, if each step of the algorithm is $(\varepsilon', 0)$-differentially private for $\varepsilon' = \varepsilon/\sqrt{8T\ln(1/\delta)}$, then the entire algorithm will be (ε, δ) differentially private. Thus, the following theorem is immediate by setting $\eta = \varepsilon'/2$:

Theorem 11.2. For a sequence of losses of length T, the algorithm RWM(η) with $\eta = \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$ is (ε, δ)-differentially private.

Remarkably, we get this theorem *without modifying the original randomized weighted majority algorithm at all*, but rather just by setting η appropriately. In some sense, we are getting privacy for free! We can therefore use Theorem 11.1, the utility theorem for the RWM algorithm, without modification as well:

Theorem 11.3. For any adversarially chosen sequence of losses of length T, $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$ the Randomized Weighted Majority

algorithm with update parameter $\eta = \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$ has the guarantee that:

$$\begin{aligned}\mathbb{E}[\text{Regret}(\text{RWM}(\eta), \ell^{\leq T})] &\leq \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}} + \frac{\sqrt{32\ln(1/\delta)}\ln k}{\varepsilon\sqrt{T}} \\ &\leq \frac{\sqrt{128\ln(1/\delta)}\ln k}{\varepsilon\sqrt{T}},\end{aligned}$$

where k is the number of experts.

Since the per-round loss at each time step t is an independently chosen random variable (over the choices of a_t) with values bounded in $[-1, 1]$, we can also apply a Chernoff bound to get a high probability guarantee:

Theorem 11.4. For any adversarially chosen sequence of losses of length T, $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$ the Randomized Weighted Majority algorithm with update parameter $\eta = \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$ produces a sequence of actions such that with probability at least $1 - \beta$:

$$\begin{aligned}\text{Regret}(\text{RWM}(\eta), \ell^{\leq T}) &\leq \frac{\sqrt{128\ln(1/\delta)}\ln k}{\varepsilon\sqrt{T}} + \sqrt{\frac{\ln k/\beta}{T}} \\ &= O\left(\frac{\sqrt{\ln(1/\delta)}\ln(k/\beta)}{\varepsilon\sqrt{T}}\right).\end{aligned}$$

This bound is nearly as good as the best possible bound achievable even without privacy (i.e., the RWM bound) — the regret bound is larger only by a factor of $\Omega(\frac{\sqrt{\ln(k)\ln(1/\delta)}}{\varepsilon})$. (We note that by using a different algorithm with a more careful analysis, we can remove this extra factor of $\sqrt{\ln k}$). Since we are in fact using the same algorithm, efficiency is of course preserved as well. Here we have a powerful example in machine learning where privacy is nearly "free." Notably, just as with the non-private algorithm, our utility bound only gets better the longer we run the algorithm, while keeping the privacy guarantee the same.[3]

[3]Of course, we have to set the update parameter appropriately, just as we have to do with the non-private algorithm. This is easy when the number of rounds T is known ahead of time, but can also be done adaptively when the number of rounds is not known ahead of time.

11.3 Empirical risk minimization

In this section, we apply the randomized weighted majority algorithm discussed in the previous section to a special case of the problem of empirical risk minimization to learn a linear function. Rather than assuming an adversarial model, we will assume that *examples* are drawn from some known distribution, and we wish to learn a classifier from some finite number of samples from this distribution so that our loss will be low on *new* samples drawn from the same distribution.

Suppose that we have a distribution $\mathcal{D}$ over *examples* $x \in [-1,1]^d$, and for each such vector $x \in [-1,1]^d$, and for each vector $\theta \in [0,1]^d$ with $\|\theta\|_1 = 1$, we define the loss of θ on example x to be $\text{Loss}(\theta, x) = \langle \theta, x \rangle$. We wish to find a vector θ^* to minimize the *expected* loss over examples drawn from $\mathcal{D}$:

$$\theta^* = \arg \min_{\theta \in [0,1]^d : \|\theta\|_1 = 1} \mathbb{E}_{x \sim \mathcal{D}}[\langle \theta, x \rangle].$$

This problem can be used to model the task of finding a low error linear classifier. Typically our only access to the distribution $\mathcal{D}$ is through some collection of examples $S \subset [-1,1]^d$ drawn i.i.d. from $\mathcal{D}$, which serves as the input to our learning algorithm. We will here think of this sample S as our private database, and will be interested in how well we can privately approximate the error of θ^* as a function of $|S|$ (the *sample complexity* of the learning algorithm).

Our approach will be to reduce the problem to that of learning with expert advice, and apply the private version of the randomized weighted majority algorithm as discussed in the last section:

1. The *experts* will be the d standard basis vectors $\{e_1, \ldots, e_d\}$, where $e_i = (0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, 0)$.
2. Given an example $x \in [-1,1]^d$, we define a loss vector $\ell(x) \in [-1,1]^d$ by setting $\ell(x)_i = \langle e_i, x \rangle$ for each $i \in \{1, \ldots, d\}$. In other words, we simply set $\ell(x)_i = x_i$.
3. At time t, we choose a loss function ℓ^t by sampling $x \sim \mathcal{D}$ and setting $\ell^t = \ell(x)$.

Note that if we have a sample S from $\mathcal{D}$ of size $|S| = T$, then we can run the RWM algorithm on the sequence of losses as described above for a total of T rounds. This will produce a sequence of outputs $a_1, \ldots, a_T$, and we will define our final classifier to be $\theta^T \equiv \frac{1}{T}\sum_{i=1}^{T} a_i$. (Recall that each a_i is a standard basis vector $a_i \in \{e_1, \ldots, e_d\}$, and so we have $\|\theta^T\|_1 = 1$).

We summarize the algorithm below:

Algorithm 17 An algorithm for learning linear functions. It takes as input a private database of examples $S \subset [-1,1]^d$, $S = (x_1, \ldots, x_T)$, and privacy parameters ε and δ.

LinearLearner(S, ε, δ):

Let $\eta \leftarrow \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$
for $t = 1$ to $T = |S|$ **do**
Choose vector $a_t = e_i$ with probability proportional to $\exp(-\eta\sum_{j=1}^{t-1}\ell_i^j)$
Let loss vector $\ell^t = (\langle e_1, x_t\rangle, \langle e_2, x_t\rangle, \ldots, \langle e_d, x_t\rangle)$.
end for
Output $\theta^T = \frac{1}{T}\sum_{t=1}^{T} a_t$.

We have already seen that LinearLearner is private, since it is simply an instantiation of the randomized weighted majority algorithm with the correct update parameter η:

Theorem 11.5. **LinearLearner**(S, ε, δ) is (ε, δ)-differentially private.

It remains to analyze the classification accuracy of LinearLearner, which amounts to considering the regret bound of the private RWM algorithm.

Theorem 11.6. If S consists of T i.i.d. samples $x \sim \mathcal{D}$, then with probability at least $1-\beta$, LinearLearner outputs a vector θ^T such that:

$$\mathbb{E}_{x\sim\mathcal{D}}[\langle\theta^T, x\rangle] \leq \min_{\theta^*} \mathbb{E}_{x\sim\mathcal{D}}[\langle\theta^*, x\rangle] + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right),$$

where d is the number of experts.

Proof. By Theorem 11.4, we have the following guarantee with probability at least $1 - \beta/2$:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\langle a_t, x_t\rangle &\leq \min_{i\in\{1,\dots,d\}}\left\langle e_i, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right)\\ &= \min_{\theta^*\in[0,1]^d:\|\theta^*\|_1=1}\left\langle \theta^*, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right).\end{aligned}$$

In the first equality, we use the fact that the minimum of a linear function over the simplex is achieved at a vertex of the simplex. Noting that each $x_t \sim \mathcal{D}$ independently and that each $\langle x_t, e_i\rangle$ is bounded in $[-1, 1]$, we can apply Azuma's inequality twice to bound the two quantities with probability at least $1 - \beta/2$:

$$\begin{aligned}&\left|\frac{1}{T}\sum_{t=1}^{T}\langle a_t, x_t\rangle - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{x\sim\mathcal{D}}\langle a_t, x\rangle\right|\\ &\quad= \left|\frac{1}{T}\sum_{t=1}^{T}\langle a_t, x_t\rangle - \mathbb{E}_{x\sim\mathcal{D}}\langle \theta^T, x\rangle\right| \leq O\left(\sqrt{\frac{\ln(1/\beta)}{T}}\right)\end{aligned}$$

and

$$\max_{i\in\{1,\dots,d\}}\left|\left\langle e_i, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle - \mathbb{E}_{x\sim\mathcal{D}}\langle e_i, x\rangle\right| \leq O\left(\sqrt{\frac{\ln(d/\beta)}{T}}\right).$$

Hence we also have:

$$\max_{\theta^*\in[0,1]^d:\|\theta^*\|_1=1}\left|\left\langle \theta^*, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle - \mathbb{E}_{x\sim\mathcal{D}}\langle \theta^*, x\rangle\right| \leq O\left(\sqrt{\frac{\ln d/\beta}{T}}\right).$$

Combining these inequalities gives us our final result about the output of the algorithm θ^T:

$$\mathbb{E}_{x\sim\mathcal{D}}\langle \theta^T, x\rangle \leq \min_{\theta^*\in[0,1]^d:\|\theta^*\|_1=1}\mathbb{E}_{x\sim\mathcal{D}}\langle \theta^*, x\rangle + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right).$$

□

11.4 Bibliographical notes

The PAC model of machine learning was introduced by Valiant in 1984 [83], and the SQ model was introduced by Kearns [53]. The randomized weighted majority algorithm is originally due to Littlestone and Warmuth [57], and has been studied in many forms. See Blum and Mansour [9] or Arora et al. [1] for a survey. The regret bound that we use for the randomized weighted majority algorithm is given in [1].

Machine learning was one of the first topics studied in differential privacy, beginning with the work of Blum et al. [7], who showed that algorithms that operate in the SQ-learning framework could be converted into privacy preserving algorithms. The sample complexity of differentially private learning was first considered by Kasiviswanathan, Lee, Nissim, Raskhodnikova, and Smith, "What can we Learn Privately?" [52], which characterize the sample complexity of private learning up to polynomial factors. For more refined analysis of the sample complexity of private learning, see [3, 4, 12, 19].

There is also extensive work on efficient machine learning algorithms, including the well known frameworks of SVMs and empirical risk minimizers [13, 55, 76]. Spectral learning techniques, including PCA and low rank matrix approximation have also been studied [7, 14, 33, 42, 43, 51].

Private learning from expert advice was first considered by Dwork et al. [26]. The fact that the randomized weighted majority algorithm is privacy preserving without modification (when the update parameter is set appropriately) is folklore (following from advanced composition [32]) and has been widely used; for example, in [48]. For a more general study of private online learning, see [50], and for a more general study of empirical risk minimization, see [50, 13].