Privacy in Statistics and Machine Learning Spring 2021
Lecture 12: Private Gradient Descent

Adam Smith and Jonathan Ullman

1 Gradient Descent

An important family of optimization algorithms are based on *gradient descent*. The idea is to iteratively improve a candidate solution w by taking a small step in the direction of opposite to the gradient at w.

When the loss L is differetiable at w, the gradient $\nabla L(w)$ gives the coefficients of the best linear approximation to L in the area right around w. Moving in the direction opposite to the gradient is thus the same as moving in the direction in which the function drops most steeply. If we think of the graph of the function as a surface, then the gradient tells us which direction a drop of water would trickle in if it were placed at the point corresponding to w. Water finds minima of a landsacpe, and so can gradient descent.

Of course, gradient descent methods can get stuck in *local minima*. These are points where the function cannot be decreased by taking small steps. A global minimum, in contrast, is an input where the function takes its smallest possible value. On the surface of the earth, lakes and ponds occur at local minima; the global minimum is the bottom of the Mariana Trench.

Even though gradient descent won't always work, we can prove that it converges—and get a very good handle on how quickly—if we focus on convex functions. Appendix **??** gives a quick recap of relevant definitions.

There are many variations on gradient descent and its analysis. We will see on variant, called *projected gradient descent* (Algorithm 1), which aims to minimize a loss L on a set C. For simplicity, we will assume that C is a convex set for which *projection*—that is, finding the nearest point in C—is computationally easy. For example, if C is a d-dimensional ℓ_2 ball, or a d-dimensional hypercube $[-1,+1]^d$, then projection is simple.

```
Algorithm 1: Projected Gradient Descent($L, C, \eta$)
Input: Set $C \subseteq \mathbb{R}^d$ for which the projection $\Pi_C$ is easy to compute. Loss function $L : C \to \mathbb{R}$ to
       be minimized. Learning rate $\eta > 0$.
$w_0 \leftarrow$ arbitrary point in $C$ ;
for $t = 1, 2, \ldots, T$ do
    $g_t \leftarrow \nabla L(w_{t-1})$ (if $f$ not differentiable, any valid subgradient $g_t \in \partial L(w_{t-1})$ will do);
    $u_t \leftarrow w_{t-1} - \eta g_t$ ;
    $w_t \leftarrow \Pi_C(u_t)$ (the nearest point in $C$ to $u_t$) ;
return $\hat{w} = \frac{1}{T}\sum_{t=0}^{T-1} w_t$ ;
```

Running Time As written, PGD requires us to compute the gradient of all n individual losses for each time step T, and then to perform T summations of n vectors in $\mathbb{R}^d$, for a total running time of $O(nT(d + \mathsf{Time}(\nabla\ell))$, where $\mathsf{Time}(\nabla\ell)$ is the time of a gradient computation.

Convergence? Gradient descent will not always converge to a global minimizer. However, when the loss L is convex, then we can actually prove convergence.

1.1 Gradient Descent with Random Estimates

Gradient descent is remarkably robust to variation. One of its notable properties is that in most settings, it continues to work even when all we can get at each step is an *estimate* $\tilde{g}_t$ of the true gradient $g_t = \nabla L(w_{t-1})$. When the loss L we seek to minimize is decomposable, meaning that $L(w) = L(w; \mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \ell(w; x_i)$ (see Equation (**??**)), it's natural to get noisy estimates for a few reasons:

Privacy: Noisy Gradient Descent A simple way to make PGD differentially private is to add i.i.d. Gaussian noise to the gradient estimates obtained at each step, e.g.,

$$\tilde{g}_t = g_t + N(0, \sigma^2 I_d)$$

The expectation of $\tilde{g}_t$ is always g_t.

Efficiency: Stochastic Gradient Descent Rather than evaluating the full gradient of L at each step of the algorithm we can save a factor of n in the running time by evaluating the gradient at a single, randomly chosen point. For each t,

$$\tilde{g}_t = \ell(w_{t-1}; x_i) \quad \text{for } i \sim \text{Uniform}(\{1, \dots, n\})$$

Again, the expectation of $\tilde{g}_t$ is exactly g_t, since

$$\mathbb{E}\left(\tilde{g}_t\right) = \frac{1}{n}\sum_i \nabla_w \ell(w_{t-1}; x_i) \underbrace{=}_{\text{by linearity of } \nabla} \nabla_w \left(\frac{1}{n} \sum_i \ell(w_{t-1}; x_i) \right) = \nabla L(w_{t-1})\,.$$

A useful variation on this idea is *batch* gradient descent, in which we select more than one random record at each step to estimate the gradient. This essentially trades off running time against variance.

1.2 Private Gradient Descent

Let's look at the first version of the idea above, encapsulated in Algorithm 2, Noisy PGD.

Algorithm 2: Noisy Projected Gradient Descent(L, C, η, σ)

```
Input: $\sigma > 0$ is a noise parameter. The other inputs are the same as in Algorithm 1.
Differences from Alg. 1 are in red.
$w_0 \leftarrow$ arbitrary point in $C$ ;
for $t = 1, 2, \dots, T$ do
    $g_t \leftarrow \nabla L(w_{t-1})$ (if $f$ not differentiable, any valid subgradient $g_t \in \partial L(w_{t-1})$ will do);
    $\tilde{g}_t \leftarrow g_t + N(0, \sigma^2 I_d)$ ;
    $u_t \leftarrow w_{t-1} - \eta \tilde{g}_t$ ;
    $w_t \leftarrow \Pi_C(u_t)$ (the nearest point in $C$ to $u_t$) ;
return $\hat{w} = \frac{1}{T}\sum_{t=0}^{T-1} w_t$ ;
```

To see that Noisy PGD is differentially private, observe that if all the intermediate values w_t were released, it would be an application of adaptive Gaussian Mechanism. When the individual losses ℓ are Lipschitz, the senstivity of each query is bounded. What is actually released is less than all of the iterates, but by postprocessing we know this won't affect privacy.

Lemma 1.1. *If the individual losses ℓ are G-Lipschitz, then Noisy PGD is (ε, δ)-DP for $\sigma \geq \dfrac{2G\sqrt{2T\ln(1/\delta)}}{n\varepsilon}$.*

Proof. NoisyPGD is an instance of the Gaussian noise with adaptively selected queries. The adaptivity comes in because the query point w_t depends on the noisy graidents from previous steps.

Recall that, by the linearity of the gradient, $\nabla L(w)$ can be written as the average of individual loss gradients $\nabla \ell(w; x_i)$. It therefore has ℓ_2 sensitivity at most $2G/n$. To analyze the mechanism as a whole, we can apply Strong Composition or, better yet, the tighter bound for Gaussian noise from the Lecture 10 In-class Exercises. That latter bound shows that even in the adaptive setting, what matters is the overall ℓ_2 sensitivity of the collection of queries. In our case the overall sensitivity is $\sqrt{T} \cdot \frac{2G}{n}$. It is enough for standard deviation σ to exceed this sensitivity by $\frac{\sqrt{2\ln(1/\delta)}}{\varepsilon}$ for the mechanism to be private; this completes the proof. *Note:* Applying generic Strong Composition Lemma would yield a slightly weaker bound with an extra factor of $\sqrt{\ln(1/\delta)}$, but wouldn't change the qualitative flavor of the result. □

Running Time As written, noisy PGD requires us to compute the gradient of all n individual losses for each time step T, and then to perform T summations of n vectors in $\mathbb{R}^d$, for a total running time of $O(nT(d + \mathsf{Time}(\nabla \ell))$, where $\mathsf{Time}(\nabla \ell)$ is the time of a gradient computation.

1.3 Stochastic Gradient Descent and Amplification by Subsampling

We can improve the run time of gradient descent dramatically if we use the second idea above for estimating the gradient, namely we sample only one record at each step and use its gradient as a proxy for the gradient of L. Remarkably, for convex loss functions the convergence rate is essentially the same with the proxy as it is with the full gradient!

We can combine the two ideas to obtain a faster differentially private algorithm known as DP-SGD. Specifically, at each step we set

$$\tilde{g}_t \leftarrow \nabla \ell(w_{t-1}; x_{i_t}) + N(0, \sigma^2 I_d) \quad \text{for } i_t \sim \text{Uniform}(\{1, \dots, n\}) \,. \tag{1}$$

Analyzing the privacy of this algorithm is trickier than in the case of the full gradient. If we released all the intermediate iterates as well as the indices i_t that we used to estimate gradients, we would have an instance of the Gaussian mechanism. But the sensitivity of the gradient at each step would be n times higher than in the previous algorithm—it would be $2G$ instead of $2G/n$. We would have to add far more noise for the same privacy level!

We'd like to somehow take advantage of the fact that the sample i_t we use to estimate the gradient in (1) is secret. We can do this! In fact, *every* differentially private algorithm has a much lower privacy parameter ε when it is run on a secret sample than when it is run on a sample whose identities are known to the attacker. This phenomenon is known as *amplification by subsampling*.

Roughly speaking, the privacy parameter gets reduced by a factor of $1/n$ when we run the algorithm on a random data entry from a data set $\mathbf{x}$. Let's try to state the result precisely. Given a data set $\mathbf{x} \in \mathcal{U}^n$ and a set of indicies $S \subseteq [n]$, let $\mathbf{x}|_S$ denote the data set obtained by keeping only entries x_i for $i \in S$.

Lemma 1.2 (Amplification by Subsampling)**.** *Let m, n be integers with $m \leq n$. Let A be any algorithm taking inputs in $\mathcal{U}^m$, and define A' on $\mathcal{U}^n$ as follows:*

> *On input $\mathbf{x}$, $A'(\mathbf{x})$ selects a uniformly random set S of size m from $[n]$ (without replacement), and returns $A(\mathbf{x}|_S)$.*

If A is (ε, δ)-differentially private, then A' is (ε', δ')-differentially private for $\varepsilon' = \ln\left(1 + (e^\varepsilon - 1) \cdot \frac{m}{n}\right)$ and $\delta' = \delta \cdot \frac{m}{n}$.

When the original algorithm satisfies $\varepsilon \leq 1$, the lemma shows that the new algorithm has privacy parameter $\varepsilon' \approx \varepsilon \cdot \frac{m}{n}$. For example, if we start from a $(1, \delta)$-differentially private algorithm, then we can get an $(\varepsilon', \varepsilon'\delta)$-DP algorithm by running the original one on a subsample of roughly ε' times the size of the original.

Exercise 1.3. Suppose a national statistical agency samples 0.001% of its country's population in a survey. It runs a (0.5,0)-DP algorithm on the results and publishes them. Assuming that the list of survey participants is not made public, for which ε' is the resulting process $(\varepsilon', 0)$-DP? How does the interpretation change if we know that a particular person of interest did participate in the survey?

Exercise 1.4. Prove Lemma 1.2. A good place to start is to set $\delta = 0$, and aim to prove a bound of the form $\varepsilon' = O(\varepsilon \cdot \frac{m}{n})$ for $\varepsilon \leq 1$.

[*Hint:* Suppose data sets $\mathbf{x}$ and $\mathbf{x}'$ differ only in position i. Break up the probability of event E into $E \cap F$ and $E \cap \bar{F}$, where F is the event that $i \in S$. The probability of F is exactly $\frac{m}{n}$ under both $\mathbf{x}$ and $\mathbf{x}'$. Furthermore, $\mathbb{P}\left(E \mid F, \mathbf{x}\right) \leq e^{\varepsilon}\mathbb{P}\left(E \mid F, \mathbf{x}'\right)$ and $\mathbb{P}\left(E \mid F, \mathbf{x}\right) \leq e^{\varepsilon}\mathbb{P}\left(E \mid \bar{F}, \mathbf{x}\right)$.]

How does this helps us analyze DP-SGD? For a given level of noise σ, we can proceed in two steps:

1. use amplification by subsampling to show that each step of the DP-SGD algorithm satisfies roughly the same privacy guarantee as it would have without subsampling, at least when the noise is fairly large.

2. We can then use strong composition to analyze the entire sequence.

For example, suppose we set $\sigma = \frac{2G\ln(1\delta)}{\varepsilon}$, so that adding noise to a single gradient $\nabla\ell(w; x_i)$ whose index i is *known* is (ε, δ)-DP for some $\varepsilon \leq 1$. The amplification by subsampling lemma shows that this step is DP with $\varepsilon' \approx \varepsilon/n$. Strong composition then allows one to repeat this basic step T times with a blow-up of about $\sqrt{T}$ in the privacy parameter, resulting in a final guarantee of $\varepsilon'' \approx \varepsilon\sqrt{T}/n$. One can thus execute the algorithm for as many as $T = n^2$ steps and get an overall privacy guarantee of about ε.

We won't work out an exact statement here, for two reasons. First, the parameters one gets by taking this route are somewhat messy. Second, there are other, more involved techniques for analyzing the privacy cost of DP-SGD that are substantially tighter. However, their rough interpretation is fairly simple:

In many parameter regimes, DP-SGD has approximately the same privacy cost and accuracy as Noisy PGD, at approximately $1/n$ fraction of the run time.

2 Analyzing Gradient Descent and Its Variants

We can analyze convergence when the set C and loss function L are convex. See the Appendix in Lecture 11 for background on convex sets and functions.

Theorem 2.1. *Let $L : C \to \mathbb{R}$ be G-Lipschitz and convex. Suppose C is closed and convex and has diameter R. Let $w^* \in \arg\min_{w \in C} L(w)$ be a minimizer of L on C. If we set $\eta = \frac{R}{G\sqrt{T}}$, then*

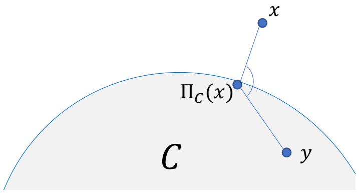
$$L(\hat{w}) \leq L(w^*) + \frac{RG}{\sqrt{T}}.$$

To prove this, we'll need a basic fact about projections onto convex sets.

Lemma 2.2. *Let $C \subseteq \mathbb{R}^d$ be a closed, nonempty convex set. Let $\Pi_C(x)$ be the nearest point in C to x. For every x, $\Pi_C(x)$ is closer than x to every point in C. That is, for every $x \in \mathbb{R}^d$ and $y \in C$,*

$$\|\Pi_C(x) - y\| \leq \|x - y\| .$$

Proof. If $x \in C$, the two distances are equal. If $x \notin C$, then consider the angle formed by the line segments from $\Pi_C(x)$ to x and $\Pi_C(x)$ to y.



If the angle is 90° or greater, then the statement is correct since the segment $\overline{xy}$—the oppposite side of the triangle—is longer than both the other two sides. But the convexity of C means the angle cannot be less than 90°. If it were, we would get a contradiction: we could slide $\Pi_C(x)$ towards y and reduce the distance from $\Pi_C(x)$ to x; this move would not leave the set C, so $\Pi_C(x)$ would not be nearest point in C to x. □

Exercise 2.3. Write down an algorithm to compute the projection onto the following convex sets in $\mathbb{R}^d$: (*a*) the ℓ_2 ball of radius 1, (*b*) the cube $[-1, 1]^d$, and (*c*) the ℓ_1 ball $\left\{x \in \mathbb{R}^d : \sum_i |x(i)| \leq 1\right\}$.

We can now prove our main convergence result.

Proof of Theorem 2.1 . The basic idea of the gradient descent analysis is look at how two key quantities evolve as the algorithm progresses:

- Excess Risk: $L(w_t) - L(w^*)$
- Squared distance to minimizer: $\|w_t - w^*\|^2$

These two quantities are intertwined. Specifically, we'll show that if the excess risk at stage t is large, then the distance to the minimizer will decrease significantly in the next step of the algorithm. Since the distance from the minimizer is at most R to begin with, there won't be too many steps during which the excess risk becomes small.

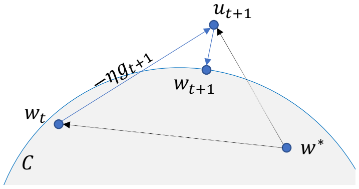
Claim 2.4. *For each $t \in \{0, ..., T-1\}$,*

$$L(w_t) - L(w^*) \leq \frac{\eta\|g_t\|^2}{2} + \frac{1}{2\eta}\left(\|w_t - w^*\|_2^2 - \|w_{t+1} - w^*\|_2^2\right) \tag{2}$$

Proof. For each iterate w_t, the fact that $g_{t+1} \in \partial L(w_t)$ tells us that $L(w^*) \geq L(w_t) + \langle g_t, w^* - w_t\rangle$. Rearranging, we get

$$L(w_t) - L(w^*) \leq \frac{1}{\eta}\left\langle \eta g_t, w_t - w^*\right\rangle . \tag{3}$$

How can we relate this to the change in the squared distance? Consider the triangle formed by w^*, w_t and the unprojected update u_{t+1}.



Recall that for any two vectors a, b, we have $\langle a, b\rangle = \frac{1}{2}\big(\|a\|^2 + \|b\|^2 - \|b-a\|^2\big)$. Applying this to (3):

$$\begin{aligned} L(w_t) - L(w^*) &\leq \frac{1}{2\eta}\Big(\|\eta g_t\|^2 + \|w_t - w^*\|^2 - \|\underbrace{w_t - w^* - \eta g_t}_{u_{t+1}-w^*}\|^2\Big) && (4)\\ &\leq \frac{\eta\|g_t\|^2}{2} + \frac{1}{2\eta}\Big(\|w_t - w^*\|^2 - \|u_{t+1} - w^*\|^2\Big) && (5)\\ &\leq \frac{\eta\|g_t\|^2}{2} + \frac{1}{2\eta}\Big(\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2\Big) && (6)\end{aligned}$$

For the last step, we used the fact that projecting u_{t+1} onto C only brings it closer to w^* (Lemma 2.2). □

We can now use the Claim to prove our main result. Because L is convex, the loss at $\hat{w} = \frac{1}{T}\sum_t w_t$ is at most the average of the losses of the intermediate iterates:

$$L(\hat{w}) \leq \frac{1}{T}\sum_{t=0}^{T-1} L(w_t)$$

This last fact follows from Jensen's Inequality (see exercises from Lecture 11). We can bound the excess risk by averaging the excess risk at all times t. Applying the Claim,

$$L(\hat{w}) - L(w^*) \leq \frac{\eta}{2}\max_t \|g_t\|^2 + \frac{1}{2\eta T}\sum_{t=0}^{T-1}\Big(\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2\Big). \qquad (7)$$

Now the sum of the decreases in the squared distance is exactly $\|w_0 - w^*\|^2 - \|w_{T+1} - w^*\|^2$, which is at most R^2. The function is G-Lipschitz, so $\|g_t\| \leq G$. Thus,

$$L(\hat{w}) - L(w^*) \leq \frac{\eta G^2}{2} + \frac{R^2}{2\eta T} \qquad (8)$$

Setting $\eta = \frac{R}{G\sqrt{T}}$ gives us the desired bound. □

There we go! We now have an algorithm to optimize a convex function on a convex set as long as we can compute gradients of the function, and also project onto the set.

It's worth pausing to see what assumptions we used to prove that Projected Gradient Descent works. We used that L is convex in two places. First, and most importantly, we used it to get a lower bound on the excess risk in terms of the gradient at w_t (3). We also used it to connect the loss at $\hat{w}$ to the average of the losses along the path we took, but that step isn't crucial—we could just have easily output the best of iterates we saw by computing L at each step. We also use that C was convex to argue that the projection step did not mess things up.

Even when functions and constraint sets are not actually convex, gradient descent will still work as long as the conditions that help it improve hold at enough of the points along its trajectory. Thus, as we develop differentially private versions of the algorithm, we'll aim to get an algorithm that makes sense to run on any function, even if we will only prove convergence when the function is

Exercise 2.5. Over the course of gradient descent, will the distance $\|w_t - w^*\|_2$ always decrease? Either prove the statement, or give a counterexample.

2.1 Analyzing GD with estimates

The analysis technique of the previous section is remarkably robust to variations. In particular, it will aloow us to analyze the expected excess risk of NoisyPGD (and DP-SGD). In particular, we can derive the following main result:

Theorem 2.6. *Let $L : C \to \mathbb{R}$ be G-Lipschitz and convex. Suppose C is closed and convex and has diameter R. Let $w^* \in \arg\min_{w \in C} L(w)$ be a minimizer of L on C.*

Running Noisy PGD (Algorithm 2) with σ as in Lemma 1.1 for $T = \frac{\varepsilon^2 n^2}{d}$ time steps using $\eta = \frac{1}{T} \cdot \frac{R\varepsilon n}{G\sqrt{d \ln(1/\delta)}}$ leads to expected excess risk

$$\mathbb{E}\left(L(\hat{w})\right) - \min_{w \in C} L(w) = O\left(\frac{RG\sqrt{d\ln(1/\delta)}}{\varepsilon n}\right).$$

Compare this to the bound for the exponential mechanism algorithm from last lecture. We've replaced the d term with the familiar $\sqrt{d\ln(1/\delta)}$—the same sort of change we saw when switching from Laplace noise to Gaussian noise for answering statistical queries. However, we see another big change: we now have an algorithm that runs in reasonable time on any kind of input. While we only analyze its convergence here for convex functions, the analysis gives us some confidence that the algorithm will perform well on "well-behaved" nonconvex functions. In particular, its variations are currently the state of the art for training deep neural networks.

To understand why Noisy PGD and DP-SGD do well, it actually helps to analyze the slightly more general setting of PGD with some abstract estimate $\tilde{g}_t$. Let's call the abstract algorithm "PGD with Estimates". Suppose that each each step of PGD with Estimates we obtain an estimate $\tilde{g}_t$ that is

- **Unbiased**: for every $w_0, w_1, \ldots, w_{t-1}$, $\mathbb{E}\left(\tilde{g}_t\right) = g_t = \nabla L(w_{t-1})$
- **Low variance:** for every $w_0, w_1, \ldots, w_{t-1}$, $\mathbb{E}\left(\|\tilde{g}_t\|^2\right)$ is bounded.

Lemma 2.7. *Let $L : C \to \mathbb{R}$ be G-Lipschitz and convex. Suppose C is closed and convex and has diameter R. Let $w^* \in \arg\min_{w \in C} L(w)$ be a minimizer of L on C. Then the output $\hat{w}$ of PGD with Gradient Estimates satisfies*

$$\mathbb{E}\left(L(\hat{w})\right) \le L(w^*) + \frac{\eta}{2}\max_t \mathbb{E}\left(\|\tilde{g}_t\|^2\right) + \frac{R^2}{2T\eta}\,.$$

The expectation is taken over the random choices made by the algorithm in estimating the gradients.

Proof. The proof follows more or less directly in the footprints of Theorem 2.1. The main idea is to take expectations on both sides of the inequality in Claim 2.4. For every $t \in \{1, \ldots, T\}$,

$$\mathbb{E}\left(L(w_t) - L(w^*)\right) \le \frac{\eta}{2}\mathbb{E}\left(\|\tilde{g}_t\|^2\right) + \frac{1}{2\eta}\mathbb{E}\left(\|w_t - w^*\|_2^2 - \|w_{t+1} - w^*\|_2^2\right) \tag{9}$$

Averaging over the iterates, we can gain apply Jensen's inequality to get $L(\hat{w}) = \frac{1}{T}\sum_t L(w_t)$. Taking expectations, we get

$$\begin{aligned}
\mathbb{E}\left(L(\hat{w}) - L(w^*)\right) &\le \frac{1}{T}\sum_t \mathbb{E}\left(L(w_t) - L(w^*)\right) && (10)\\
&\le \frac{\eta}{2}\left(\frac{1}{T}\sum_t \mathbb{E}\left(\|\tilde{g}_t\|^2\right)\right) + \frac{1}{2T\eta}\left(\|w_0 - w^*\|^2 - \mathbb{E}\left(\|w_{T+1} - w^*\|^2\right)\right) && (11)\\
&\le \frac{\eta}{2}\max_t \mathbb{E}\left(\|\tilde{g}_t\|^2\right) + \frac{R^2}{2T\eta} && (12)
\end{aligned}$$

This completes the proof of the lemma. □

We can apply this lemma to analyze Noisy PGD. The main thing we need is a bound on $\mathbb{E}\left(\|\tilde{g}_t\|_2\right)$. The squared ℓ_2 norm is a sum of the squares of the entries. By linearity of expectation, it suffices to see how adding Gaussian noise increases the expected square of each entry. Because the noise has zero mean, it increases the expected square by exactly σ^2 per entry (since $\mathbb{E}\left((g+Z)^2\right) = g^2 + \mathbb{E}\left(Z^2\right) + 2g\mathbb{E}\left(Z\right)$). Thus,

$$\mathbb{E}\left(\|\tilde{g}_t\|_2\right) = \|\nabla L(w_{t-1})\|^2 + d\sigma^2 \leq G^2 + d\sigma^2 \,. \tag{13}$$

To get privacy, we set $\sigma^2 \approx G^2 T \frac{\ln(1/\delta)}{(\varepsilon n)^2}$. Thus, Lemma 2.7 gives un an excess risk bound of (dropping constant factors)

$$\frac{R^2}{T\eta} + \eta G^2\left(1 + \frac{\ln(1/\delta)}{(\varepsilon n)^2} T d\right).$$

Plugging in T and η as in Theorem 2.6 completes the proof of the theorem.