6.883: Online Methods in Machine Learning

Alexander Rakhlin

LECTURE 4

This lecture is partly based on chapters 14-15 in [SSBD14].

Let us now give a variant of SGD for strongly convex functions.

Algorithm 1 SGD for strongly convex functions

Input: $\sigma > 0$ (strong convexity parameter)
Init: $w_1 = 0$
for t=1,...,T **do**
  $w_{t+1} = w_t - \frac{1}{\sigma t}\nabla_t$, where $\mathbb{E}_t[\nabla_t] \in \partial f(w_t)$
end for

Lemma 1. *If f is strongly convex with parameter σ and $\mathbb{E}\|\nabla_i\|^2 \le G^2$ for all $i \in [T]$, then the average of the trajectory $\widehat{w} = \frac{1}{T}\sum_{t=1}^T w_t$ satisfies*

$$\mathbb{E}[f(\widehat{w})] - f(w^*) \le \frac{G^2}{2\sigma} \cdot \frac{(1+\log T)}{T}.$$

The proof is a small modification of the gradient descent lemma from the previous lecture. We prove the result for non-stochastic gradient, and leave the stochastic version as an exercise.

Proof. Following the proof of the gradient descent lemma, but with time-varying η_t, we get

$$\sum_{t=1}^T \langle \nabla f(w_t), w_t - w^* \rangle = \sum_{t=1}^T \frac{1}{2\eta_t}\left[\|w_t - w^*\|^2 - \|w_{t+1} - w^*\|^2\right] + \sum_{t=1}^T \frac{\eta_t}{2}\|\nabla f(w_t)\|^2. \quad (1)$$

However, there is now additional negative term coming from strong convexity. This term will give us the faster $1/T$ convergence rate:

$$f(w_t) - f(w^*) \le \langle \nabla f(w_t), w_t - w^* \rangle - \frac{\sigma}{2}\|w_t - w^*\|^2$$

Then

$$f(\widehat{w}) - f(w^*) \le \frac{1}{T}\sum_{t=1}^T f(w_t) - f(w^*)$$

which is upper bounded by

$$\frac{1}{T}\sum_{t=1}^T \left[\frac{1}{2\eta_t}\|w_t - w^*\|^2 - \frac{1}{2\eta_t}\|w_{t+1} - w^*\|^2 - \frac{\sigma}{2}\|w_t - w^*\|^2\right] + \frac{1}{T}\sum_{t=1}^T \frac{\eta_t}{2}\|\nabla f(w_t)\|^2. \quad (2)$$

This upper bound is

$$\frac{1}{T}\sum_{t=1}^T \frac{G^2}{2\sigma t}.$$

□

A few remarks:

- The logarithmic factor can be removed by averaging only the second half of the trajectory, or putting some other nonuniform weights on the trajectory.
- In practice, the last iterate w_T is quite good, and possibly better than the average. Analysis for this last iterate was done in [SZ12].
- Once again, we may incorporate a Euclidean projection step onto a convex set after each update. In this case, the guarantee is with respect to w^* in that set.

0.1 Full gradient for empirical objectives

Many offline (or, "batch") problems in machine learning can be written as an empirical objective

$$\min_w \sum_{t=1}^{n} \boldsymbol{\ell}(w, (x_t, y_t)) \tag{3}$$

or a regularized version of it

$$\min_w \sum_{t=1}^{n} \boldsymbol{\ell}(w, (x_t, y_t)) + \lambda R(w) \tag{4}$$

for some penalty R, tradeoff parameter λ, and a function $\boldsymbol{\ell}$ that measures how well w explains the relationship between x and y.

For instance, for finding a low-error linear separator in the non-separable case, we may try to perform gradient descent on

$$f(w) = \sum_{t=1}^{n} \max\{0, 1 - y_t \langle w, x_t \rangle\} \tag{5}$$

This would be a non-stochastic gradient descent, but each iteration requires one to compute an element of $\partial f(w_t)$. We may take $\sum_{t=1}^{n} \nabla_t$ where ∇_t is a subgradient of the t-th loss. For the hinge loss case, a subgradient (with respect to example i) can be written as

$$-y_i x_i \cdot \mathbf{1}\{y_i \langle w, x_i \rangle < 1\} \tag{6}$$

The procedure amounts to running through the whole dataset to calculate the full gradient, and then make one step. Time complexity, in terms of gradient evaluations, to obtain ϵ-accurate solution is then

$$n \times \frac{R^2 \|w^*\|^2}{\epsilon^2}$$

where $R = \max \|x_i\|$. Check that R comes from the bound on the gradient of the loss.

Unfortunately, the bound scales with the size of our dataset, and one opts for the SGD procedure. Technically speaking, there are better analyses of SGD that may even get the $\log(1/\epsilon)$ dependence on target accuracy under additional assumptions on the functions. However, it has been argued in the last decade (both empirically and theoretically) that the ability to process larger n is more important than attaining high accuracy for a limited n. That is, if the constraint is computation time, rather than amount of data, one should opt for stochastic gradient descent [BB08, SSSSC11].

0.2 SGD for empirical objectives

In applying SGD to batch learning problems, one views the objective

$$f(w) = \frac{1}{n}\sum_{t=1}^{n} \boldsymbol{\ell}(w, (x_t, y_t)) \tag{7}$$

(or the regularized version) as an empirical distribution. If index I is sampled uniformly at random from $[n]$, then any $\nabla_I \in \partial\boldsymbol{\ell}(w, (x_I, y_I))$ has the property that

$$\mathbb{E}_{I\sim\mathrm{Unif}}[\nabla_I] \in \partial f(w). \tag{8}$$

The time complexity of SGD for attaining an ϵ-minimizer of the objective is independent of n (!!), and the dependence on ϵ, of course, varies according to the properties of $\boldsymbol{\ell}$.

We remark that we proved convergence of SGD for general random unbiased subgradients, but we are applying it to a distribution of a very specific form. This has been exploited to improve the analysis and the dependence on ϵ.

Instead of sampling from $[n]$, it is common to permute the data and run over it in order, possibly several times. There have been several works trying to understand how different the random sampling is from cycling through the data.

0.3 SGD for Support Vector Machines

Recall that the hinge loss penalizes data points close to the boundary, and thus pushes the hyperplane to have a large margin. This is not an entirely precise statement, since the very notion of margin of size 1 was tied to the fact that w^* is a minimal-norm vector. Hence, the objective (5) is not what we want to minimize. Instead, we need a bi-criterion form

$$\min_{w} \qquad \frac{1}{n}\sum_{t=1}^{n} \max\{0, 1 - y_t \langle w, x_t\rangle\}, \qquad \|w\|^2. \tag{9}$$

That is, we want to minimize loss and the norm at the same time. There are several ways to combine the bi-criteria into a single one. Here is one:

$$\min_{w} \ \frac{1}{n}\sum_{t=1}^{n} \max\{0, 1 - y_t \langle w, x_t\rangle\} + \frac{\lambda}{2}\|w\|^2. \tag{10}$$

This is known as the Support Vector Machine (a fancy name is a must in machine learning!) for the case of linear kernels (more on this later). One more caveat is that SVM does not penalize the scalar shift of the nonhomogeneous hyperplane; in the formulation (10), however, this shift is absorbed in w and the norm penalizes it.

Before the large-scale problems came about, the SVMs were solved as a constrained quadratic programming problem. Pegasos, the SGD solution to this objective, proposed by [SSSSC11] (see also [Zha04]) has been very influential in practical applications with large datasets.

For the randomly chosen example i, the subgradient of the SVM objective is

$$\nabla_t = -y_i x_i \cdot \mathbf{1}\{y_i \langle w_t, x_i\rangle < 1\} + \lambda w_t. \tag{11}$$

To apply SGD it remains to decide on the step size. Since the objective is λ-strongly convex due to the regularization term, we choose the step-size

$$\eta_t = \frac{1}{\lambda t} \tag{12}$$

and apply SGD for strongly convex objectives.

Suppose we substitue a potentially suboptimal choice $w = 0$ in (10). The value of the objective is then at most 1. Hence, the optimal solution w^* should give an objective value no greater than that. That implies

$$\frac{\lambda}{2}\|w^*\|^2 \leq 1, \tag{13}$$

or $\|w^*\| \leq \sqrt{2/\lambda}$ (with a bit more work, 2 can be replaced by 1). The SGD algorithm may add the projection step onto a unit ball of this radius to help guide the search with the extra information about the location of w^*. [SSSSC11] reports that the projection step makes little difference in the experiments they performed.

We summarize the Pegasos algorithm below.

Algorithm 2 SGD for SVM objective (Pegasos)

Input: $\lambda > 0$ (regularization parameter)
Init: $w_1 = 0$
for t=1,...,T **do**
  Set $\eta_t = \frac{1}{\lambda t}$
  Sample $i \sim \text{Unif}[n]$
  if $y_i \langle w_t, x_i \rangle < 1$ **then**
    $w_{t+1} = (1 - \eta_t \lambda) w_t + \eta_t y_i x_i$
  else
    $w_{t+1} = (1 - \eta_t \lambda) w_t$
  end if
  Optionally, rescale w_{t+1} to have norm at most $\sqrt{2/\lambda}$
end for

To apply Lemma on convergence of SGD for strongly convex functions, we need to calculate bounds on the gradients and w^*. Observe that the hinge loss is R-Lipschitz, where $R = \max_i \|x_i\|$. Furthermore, the update of SGD can be written succinctly as

$$w_{t+1} = \frac{1}{\lambda t} \sum_{s=1}^{t} y_{i_s} x_{i_s} \mathbf{1}\{y_{i_s} \langle w_s, x_{i_s} \rangle < 1\} \tag{14}$$

(prove this by unwinding the recursion), where i_s is the index chosen at step s. In particular, this implies that $\|w_{t+1}\| \leq R/\lambda$ for all iterates. The Lipschitz constant of the overall function is then upper bounded by $2R$, and the convergence guarantee of Pegasos for the average $\widehat{w}$ of the trajectory is

$$\mathbb{E} f(\widehat{w}) - f(w^*) \leq \frac{4R^2(1 + \log T)}{\lambda T} \tag{15}$$

where f is the SVM objective in (10).

0.4 Mini-batching

A common practice (including in SGD for deep neural nets) is to take a small batch of data, evaluate the average gradient with respect to these data, and then update the parameter. Mini-batching presents a natural interpolation between the full gradient (all gradients at once) and the single-gradient SGD as stated above. This has the effect of reducing variance of the gradients while still being computationally cheap (and independent of n).

0.5 Sparse updates

Examining (14), we only need to keep track of the sum of $y_{i_s} x_{i_s}$ that led to a correction of the hyperplane. Hence, if x's are s-sparse, the update can be implemented in time $O(s)$ rather than $O(d)$. This becomes handy, for instance, in document classification with the bag-of-words (or related) sparse representation.

0.6 Equivalent form of SVM objective

A form that one may encounter in the literature is

$$\min_{w,b,\xi} \quad \frac{1}{m}\sum_{t=1}^{n} \xi_i + \frac{\lambda}{2}\|w\|^2 \tag{16}$$

$$\text{subj to} \qquad y_t(\langle w, x_t\rangle + b) \geq 1 - \xi_t \tag{17}$$

$$\xi_t \geq 0 \tag{18}$$

References

[BB08] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[SZ12] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *arXiv preprint arXiv:1212.1824*, 2012.

[Zha04] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.