ECE 901: Large-scale Machine Learning and Optimization **Fall 2016**

Lecture 0 — 10/24

Lecturer: Dimitris Papailiopoulos *Scribe: Devin Conathan, Ari Biswas*

Note: These lecture notes are still rough, and have only have been mildly proofread.

0.1 Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization

0.1.1 General Idea

The problem in mind here is solving something like an SVM, where you are minimizing a set of scalar convex loss functions. Stochastic Gradient Descent (SGD) has some disadvantages, so instead we're going to consider an alternative called dual coordinate ascent (DCA), which solves the dual problem. The paper proposes a basic "Stochastic Dual Coordinate Ascent" (SDCA) algorithm and a few varients, and proves some results about their convergence.

0.1.2 Problem Setup

We are looking for the solution:

$$w^* = \arg\min_{w\in\mathbb{R}^d} P(w) = \arg\min_{w\in\mathbb{R}^d} \left[\frac{1}{n}\sum_{i=1}^{n}\phi_i(w^T x_i) + \frac{\lambda}{2}||w||^2\right] \tag{1}$$

Where $x_1,\dots,x_n$ are vectors in $\mathbb{R}^d$ and $\phi_1,\dots,\phi_n$ are your scalar convex loss functions. Instead we solve the dual problem:

$$\alpha^* = \arg\max_{\alpha\in\mathbb{R}^n} D(\alpha) = \arg\max_{\alpha\in\mathbb{R}^n} \left[\frac{1}{n}\sum_{i=1}^{n} -\phi_i^*(-\alpha_i) - \frac{\lambda}{2}\left\|\frac{1}{\alpha n}\sum_{i=1}^{n}\alpha_i x_i\right\|^2\right] \tag{2}$$

Where $\phi_i^*(u) = \max_z(zu - \phi_i(z))$, the convex conjugate of ϕ_i. If we have α^*, we can compute $\frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i^* x_i = w^*$, to get the solution for (1).

It is known that $P(w^*) = D(\alpha^*)$ which implies that $P(w) \geq D(\alpha)$ for all $\alpha,\ w$, so we can define the *duality gap* for a given α as:

$$P\left(\frac{1}{\lambda n}\sum_{i=1}^{n}\alpha_i x_i\right) - D(\alpha) \tag{3}$$

The duality gap is 0 at the optimal solution, so we wish to make the duality gap as small as possible.

0.1.3 Results

The paper proposes "Vanilla SDCA", which is essentially running coordinate ascent on the dual problem for a random coordinate direction at each iteration. There are two variations: one is a "two stage" approach that starts out using SGD and switches to SDCA. Another variation is SDCA-Perm, which cycles through random permutations of coordinates rather than sampling a random coordinate each time. The paper proves some results about the "Vanilla SDCA" and does some experiments with the variations.

For Vanilla SDCA, the paper is able to show the following for a achieving a duality gap of ϵ:

1. For L-Lipschitz loss functions, convergence in $O\left(n + \frac{L^2}{\lambda\epsilon}\right)$ steps.

2. An even better rate for loss functions that are "almost everywhere smooth" (e.g. hinge-loss)

3. For $\frac{1}{\gamma}$-smooth loss functions, convergence in $O\left(\left(n + \frac{1}{\lambda\epsilon}\right)\log\frac{1}{\epsilon}\right)$ steps.

0.2 Variance Reduction for Faster Non-Convex Optimization

0.2.1 General Idea

The problem here is minimizing a set of smooth but non-convex loss functions, which is typical in machine learning problems like training neural networks. The issue is that the convergence guarantees using vanilla Stochastic Gradient Descent are pretty bad for the non-convex case because the number of steps depends on the variance of the stochastic gradient, which can be quite large for these sorts of problems. The paper demonstrates (with a proof and some experiments) that the convergence of the Stochastic Variance Reduced Gradients (SVRG) algorithm does does depend on this variance and thus is better in practice.

0.2.2 Problem Setup

We are trying to solve:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{4}$$

Where the f_i are L-smooth but not convex, which is typical for problems like neural networks. Since we don't have complexity we cannot guarantee finding the global optimum, but instead can only find *stationary points*, i.e. points x where $||\nabla f(x)|| = 0$. We say x is an ϵ-approximate stationary point if $||\nabla f(x)||^2 < \epsilon$.

Normal Stochastic Gradient Descent (SGD) can be bad for these problems because you get $\epsilon-$approximate stationary points in $O\left(\frac{L}{\epsilon}+\frac{L\sigma^2}{\epsilon^2}\right)$ steps, where σ^2 is the variance of the stochastic gradient ($\frac{\sigma^2}{\epsilon^2}$ is typically very large!).

0.2.3 Results

The paper proposes Stochatic Variance Reduced Gradients (SVRG) and shows that the number of iterations does not depend on σ^2 and has an overall computational cost of:

$$O\left(\frac{n^{2/3}L}{\epsilon}\right) \tag{5}$$

which is cheaper than gradient descent.

0.2.4 General Approach

The complete proof is really complex. Refer to slides and paper for a more detailed explanation.

The general idea is instead of minimizing $G = \frac{1}{n}\sum_{i=1}^{n} g(x_i)$, we minimize $\hat{G}$, where $\mathbb{E}G = \mathbb{E}\hat{G}$ and $Var(G) \geq Var(\hat{G})$. That is, $\hat{G}$ is an unbiased estimator of G with a smaller variance.

The convergence is divided into epochs $s = 1, \ldots, S$. For each epoch, you have x_0^s, the average from previous epoch and compute $\tilde{\mu} = \nabla f(x_0^s)$, the full gradient.

Then for this epoch for your updates become:

$$x_{k+1}^s = x_k^s - \eta\tilde{\nabla}_k^s \tag{6}$$

Where $\nabla_k^s = \nabla f_i(x_k^s) - \nabla f_i(x_0^s) + \tilde{\mu}$. This asymptotically reduces the variance of the gradient.

Then the basic idea is to bound $\mathbb{E}(\sigma_k^s)^2$. This is done by showing that $\mathbb{E}(\sigma_k^s)^2 \leq O(||x_k^s - x_0^s||^2)$. and then showing $||x_k^s - x_0^s||^2$ is bounded by a constant times $f(x_k^s) - f(x_0^s)$. The former follows from L-smoothness. The latter is derived through a complicated telescoping trick that involves dividing the epochs into subepochs.

The overall convergence result follows from being able to achieve this bound on on $\mathbb{E}(\sigma_k^s)^2$.

0.3 Second Order Stochastic Optimization in Linear Time

0.3.1 Big picture

The goal is solve the following unconstrained optimization:

$$\min_{w\in\mathbb{R}^d} \sum_{i=1}^{m} f_i(w) + R(w)$$

NOTE: f is α strongly convex and β smooth, $\nabla^2 f(x)$ is M-lipschitz. Pretty much all the good properties you want a function to have we assume.

This kind of optimization shows up all the time in Machine learning in cases like sparse logistic regression, ridge regression etc. In fact most of the theorems proved in this paper only apply to Generalized Linear Models

Popular first order methods like gradient descent or stochastic gradient descent can be used to solve this optimization but the those algorithms may not as quickly as desired. In this case it is common to use second order information of the function to approximate the optimal solution. A classical second order method, known as Newtons method, is an example of algorithm that has super linear convergence under certain assumptions, can be used to solve the above problem.

The update step looks of Newtons method like the following:

$$w^{t+1} = w^t - \nabla^2 f\left(w^t\right)^{-1} \nabla f\left(w^t\right)$$

Though we may have faster convergence i.e take less iterations to get to optimal solution; the cost per iteration is very expensive as hessians are expensive to calculate $\Omega(md^2+d^2)$

To handle this computational cost issue, the authors of this paper have come up with LiSSA which claims that the hessian can be updated in linear time $\Omega(md)$

0.3.2 LiSSA- Linear Stochastic second order algorithm

Key idea: The bottle neck is computing the hessian inverse. If we could do that easily we have solved the problem. They use taylor expansion to construct an unbiased estimator of the inverse of the hessian.

For any matrix A s.t $||A|| \leq 1$ and $A \succeq 0$ we have

$$A^{-1} = \sum_{i=1}^{\infty} (I - A)^i$$

Equivalently this can be written as a recursive equation:

$$A_j^{-1} = I + (I - A)A_{j-1}^{-1}$$

where A_j^{-1} is the approximation of A^{-1} using only j summands from the above equation. Note as j goes to infinity, we get a perfect estimate.

Now if we are given j unbiased samples $\{X_1, X_2, ..., X_j\}$ we can approximate the hessian inverse by the following recursive formula:

$$\tilde{\nabla}^2 f_0^{-1} = I \text{ and } \tilde{\nabla}^2 f_t^{-1} = I + (I - X_j)\tilde{\nabla}^2 f_{t-1}$$

It is easy to show that the above approximate of the hessian is asymptotically unbiased.

This little math trick is the essence of the whole paper, now you can approximate the hessian inverse recursively and this is what gives huge savings in their algorithm.

0.3.3 Major claim:

For GLM's LiSSA gets epsilon close to the optimal with high probability in total time complexity $O(m + \kappa)d\log(\frac{1}{\epsilon})$ where κ is a function of the conditional number of the hessian and the cost per iteration $\tilde{O}(md + \kappa^2 d)$

For a more detailed description of the algorithm refer to the slides or the paper