10-725: Optimization **Fall 2012**

Lecture 27: December 4

Lecturer: Kevin Waugh *Scribes: Andrew Hsi, Carlton Downey*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

27.1 Overview

This lecture covers Dual Averaging, which is a subgradient method which can be thought of as an extension to subgradient descent. It was invented by Nesterov around 2007, and unlike his acceleration techniques there is some nice insight.

27.2 Review

We begin with a review of the subgradient method. Let f be a convex, not necessarily differentiable function; let g_k be a valid subgradient at point x_k. Then the update for x_{k+1} is:

$$x_{k+1} = x_k - t_k g_k$$

where t_k is the step size at iteration k.

Recall that for f convex and Lipschitz continuous, we can achieve convergence rate of $O(\frac{logk}{\sqrt{k}})$ if we pick a divergent step size correctly. Specifically, let

$$t_i > 0$$

$$t_i \to 0 \text{ as i} \to \infty$$

$$\sum_{i=1}^{\infty} t_i = \infty$$

Note that you can potentially be clever about setting the step sizes by taking into account the Lipschitz constant as well as any information you may have regarding how far the current iterate is from the optimum value.

27.3 Dual Averaging

27.3.1 Motivation

One problem with the subgradient method is that it does not weight new subgradients as much as old subgradients. Let us demonstrate this by rewriting the subgradient update into an equivalent form.

Note that the update can be written as a minimization of the quadratic approximation as f; this gives us the following equation for x_{k+1}

$$x_{k+1} = \operatorname*{argmin}_x f(x_k) + g_k * (x - x_k) + \frac{\|x - x_k\|^2}{2t_k}$$

Rearranging terms gives us the following results:

$$x_{k+1} = \operatorname*{argmin}_x f(x_k) + g_k * (x - x_k) - \frac{x^T x_k}{t_k} + \frac{x^T x}{2t_k}$$

$$x_{k+1} = \operatorname*{argmin}_x t_k[f(x_k) + g_k * (x - x_k)] - x^T x_k + \frac{x^T x}{2}$$

$$x_{k+1} = \operatorname*{argmin}_x t_k[f(x_k) + g_k * (x - x_k)] - x^T (x_{k-1} - t_{k-1} g_{k-1}) + \frac{x^T x}{2}$$

$$x_{k+1} = \operatorname*{argmin}_x \frac{\sum_{i=1}^k t_i}{\sum_{i=1}^k t_i}[f(x_i) + g_i * (x - x_i)] + \frac{\|x\|^2}{2\sum_{i=1}^k t_i}$$

$$x_{k+1} = \operatorname*{argmin}_x \sum_{i=1}^k \frac{t_i}{Z_k}[f(x_i) + g_i * (x - x_i)] + \frac{\|x\|^2}{2Z_k}, \text{ where } Z_k = \sum_{i=1}^k t_i$$

Analyzing this last result indicates that the older subgradients are weighted more strongly than the newer subgradients; however, there is little reason to believe that recent subgradients are less important or useful. In the following sections, we will consider an alternative to this manner of weighting subgradients.

27.3.2 Dual Averaging Algorithm

Let us instead use the following update for x_{k+1}

$$x_{k+1} = \operatorname*{argmin}_x \hat{g_k} * x + \frac{\mu_k \|x\|^2}{2k}, \text{ where } \hat{g_k} = \frac{1}{k}\sum_{i=1}^k g_i$$

Looking at the definition of $\hat{g_k}$, we can clearly see that in this formulation, we weight all subgradients equally. The new term μ serves as a step size control.

The basic algorithm is then as follows: given a sequence of μ_k and an initial gradient $g_1 = 0$, for each iteration, we do the following:

$$x_k = \frac{-k\hat{g_k}}{\mu_k}$$

$$g_{k+1} \in \partial f(x_k)$$

Given f convex and L-Lipschitz, with $\mu_k \in O(\sqrt{k})$, the dual averaging algorithm achieves a convergence rate of $O(\frac{1}{\sqrt{k}})$. Note that this gives a better convergence rate than the subgradient method.

27.3.3 Primal/Dual Problem

One advantage to this setup is that we can directly evaluate the duality gap between the primal and dual problems. We analyze this by expressing the solution in both its primal and dual forms:

$$v^* = \min_{\|x\| \leq D} f(x)$$

$$v^* = \min_{\|x\| \leq D} \max_g g^T x - f^*(g)$$

$$v^* = \max_g \min_{\|x\| \leq D} g^T x - f^*(g)$$

$$v^* = \max_g -f^*(g) - D \max_{\|z\| \leq 1} -z^T y$$

$$v^* = \max_g -f^*(g) - D\| - g\|_*$$

Thus, then:

$$0 = v^* - v^*$$

$$0 = \min_x f(x) - \max_g -f^*(g) - D\| - g\|_*$$

$$0 = \min_{x,g} f(x) + f^*(g) + D\| - g\|_*$$

27.3.4 Usefulness of Dual Averaging

The dual averaging algorithm can potentially offer benefits over using the subgradient method. Note that this is dependent on the particular application/problem being addressed and may not necessarily give improvements for all scenarios.

The key benefits to this approach are:

1. The duality gap can be evaluated
2. The convergence rate beats that of the subgradient method by a log factor
3. The constants within the convergence rate are better – original author claims that even the worst case settings are no worse than subgradient method

One particular danger to note however is that there is a greater chance for making precision errors when implementing the update for the dual averaging algorithm. Recall that the update to the subgradient method adds a term to the previous update – in the worst case, the update is unchanged. In the dual averaging approach however, x_k is determined via multiplication, so if there are precision errors, the results could be far from what was intended. Caution must be used when implementing this algorithm.

27.3.5 Example

Let us consider an example. Using dual averaging on this particular dataset, we can learn a very reasonable classifier for this dataset, as seen in the figure. What is most noteworthy is the rate at which convergence is achieved; while both the subgradient method and dual averaging perform well, it is clear that dual averaging has a much faster convergence rate on this data.

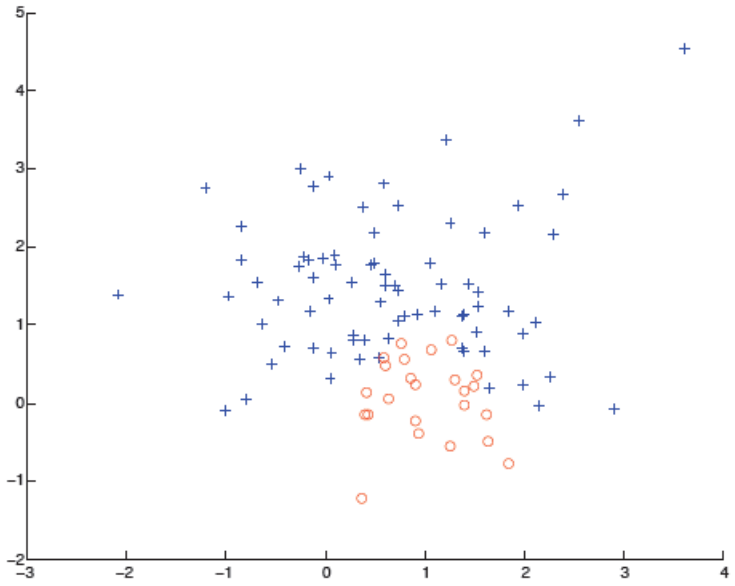

Figure 27.1: Example dataset

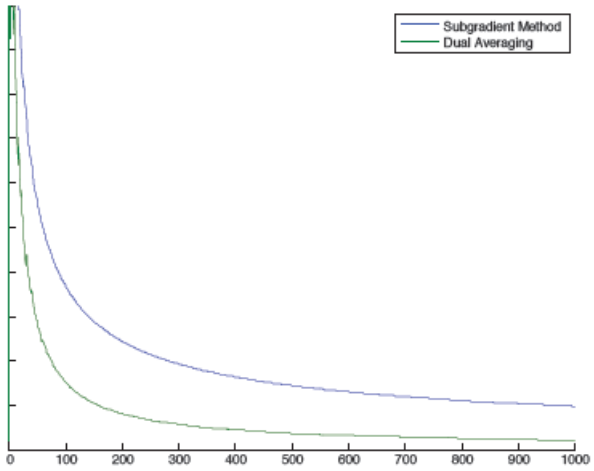

Figure 27.2: Subgradient vs Dual Averaging

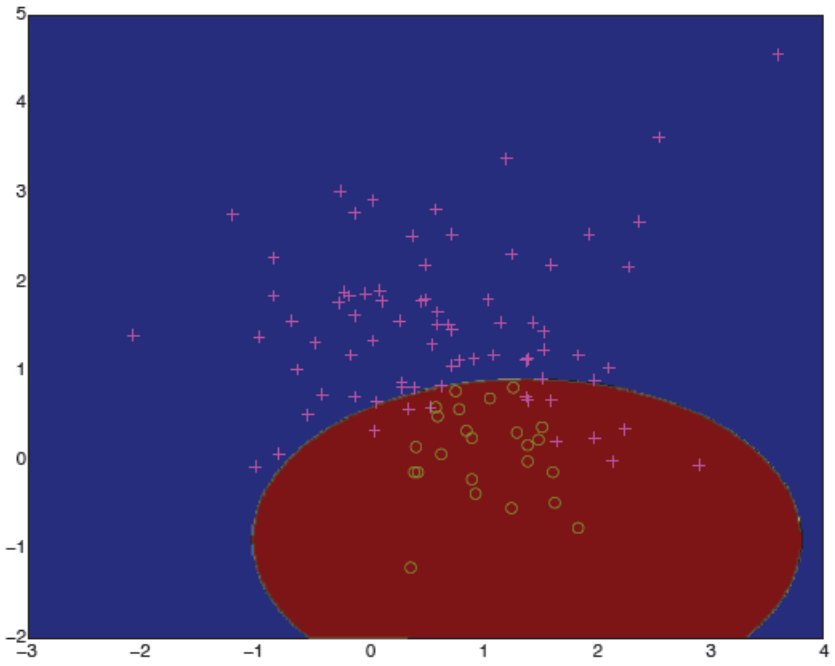

Figure 27.3: Classifier

27.4 Additional Properties

If we happen to know that the function f is strongly convex, then by using $\mu_k = 1 + logk$, we can achieve an even better convergence rate: $O(\frac{logk}{k})$. Additionally, by averaging differently, it is even possible to drop the $logk$ term.

Note that many of the techniques that were previously applied to improving first-order methods can also be applied to dual averaging.

27.4.1 Composite Objectives

We can rewrite f in terms of g and h, where g is convex and differentiable and h is convex but not necessarily differentiable. Proceeding with this gives the following results:

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} [f(x_i) + g_i * (x - x_i)] + h(x) + \frac{\mu_k \|x\|^2}{2k}$$

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \hat{g_k} * x + h(x) + \frac{\mu_k \|x\|^2}{2k}$$

$$x_{k+1} = prox_{k/\mu_k}(\frac{k\hat{g_k}}{\mu_k})$$

By approaching the problem in this manner, we replicate the same ideas used in ISTA.

27.4.2 Bregman Divergences

We can also choose to use a different function in place of the 2-norm:

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} [f(x_i) + g_i * (x - x_i)] + \frac{\mu_k \Psi(x)}{k}$$

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \hat{g_k} * x + \frac{\mu_k \Psi(x)}{k}$$

where $\Psi(x)$ is a strongly convex function. For example, let $\Psi(x) = xlogx - 1 * x$ for the unit simplex. Then:

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} [f(x_i) + g_i * (x - x_i)] + \frac{\mu_k [xlogx - 1 * x]}{k}$$

$$\text{subject to: } \sum_{i=1}^{n} x_i = 1,\ x \geq 0$$

$$x_{k+1} \propto exp(\frac{-k\hat{g_k}}{\mu_k})$$

27.4.3 Acceleration

We can also apply the use of acceleration to dual averaging – conceptually, this is similar to FISTA. This is achieved using the following updates:

$$y_k = (1-\theta_k)x_k + \theta_k v_k$$
$$\mu_k = \frac{4(L+(k+1)^{3/2}}{k(k+1}$$
$$\theta_k = \frac{2}{k+1}$$
$$\tilde{g}_{k+1} = (1-\theta_k)\tilde{g}_k + \theta_k \bigtriangledown f(y_k)$$
$$v_{k+1} = \underset{x}{\operatorname{argmin}}\, \tilde{g}_{k+1} * x + \frac{\mu_k \|x\|^2}{2}$$
$$x_{k+1} = (1-\theta_k)x_k + \theta_k v_{k+1}$$

Doing so gives a convergence rate of $O(\frac{L}{k^2})$

27.4.4 Stochastic Objective

We can also use dual averaging when we have a stochastic objective function:

$$\min_x \mathbb{E}_\xi[f(x|\xi)]$$
$$\mathbb{E}_\xi[g_k] \in \delta\mathbb{E}_\xi[f(x|\xi)]$$

For example:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}(\theta_i \cdot x - y_i)^2$$
$$g_k = 2(\theta_{i_k} \cdot x_k - y_{i_k})\theta_{i_k}$$
$$i_k \sim Uniform(\{1,2,...,n\})$$

27.4.5 Distributed Objective

Suppose we have a function $f(x)$ such that:

$$f(x) = f_1(x) + f_2(x) + \cdots + f_n(x)$$

Then we can rapidly minimize $f(x)$ by evaluating the component functions $f_i(x)$ in parallel on a network of computers using the Distributed Dual Averaging algorithm.

In Distributed DA we have a network of nodes connected by edges, where each node i has a single component function f_i, a primal average $\hat{x}_k^i$, and a dual average $\tilde{g}_k^i$. Each iteration, node i will evaluate f_i based on the

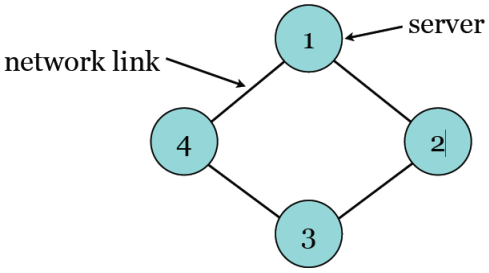

Figure 27.1: A computer network with 4 nodes. We could use this network and and the Distributed DA algorithm for problems of the form $f(x) = f_1(x) + f_2(x) + f_3(x) + f_4(x)$

current value of $\hat{x}_k^i$ and $\tilde{g}_k^i$. It will then share its new dual average with each of its neighbors $N(i)$. Over time each of the primal averages will approach optimal. Formally:

$$f(x) = \sum_{i=1}^{n} f_i(x)$$

$$\tilde{g}_{k+1}^i = \frac{1}{k+1}\left[\sum_{j \in N(i)} \frac{k\tilde{g}_k^j}{|N(i)|} + g_k^i\right]$$

Convergence:

$$f(\hat{x}_k^i) - f(x*) \in O\left(\frac{1}{\sqrt{\gamma_G k}}\right)$$

Where γ_G is the spectral gap of the network (the distance between the two largest eigenvalues of the graph laplacian). It is important to note that the rate of convergence will vary significantly depending on the topology of the network. For example:

- m-paths and cycles: $O\left(\frac{n}{m\sqrt{k}}\right)$
- m by m grid: $O\left(\frac{n^{1/4}}{m\sqrt{k}}\right)$
- expander graphs: $O\left(\frac{1}{\sqrt{k}}\right)$

References

[1] Y. Nesterov. Primal-dual subgradient methods for convex problems, 2005.

[2] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization, 2011.

[3] J. Duchi, A. Agarwal, M. Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling, 2010.

[4] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization, 2008.