Notes on AdaGrad

Chris Dyer
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213
cdyer@cs.cmu.edu

Abstract

These are some notes on the adaptive (sub)gradient methods proposed by Duchi et al. (2011), a family of easy-to-implement techniques for online parameter learning with strong theoretical guarantees and widely attested empirical success. These notes are designed to be a quick summary of the technique for those interested in applying it to their own online learning problems.

1 Introduction

Consider the problem of online learning in which a series of guesses (parameter estimates) $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$, where each $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$ (and $\mathcal{X}$ is a convex set), is generated by observing feedback from a series of losses $f_1(\boldsymbol{x}_1), f_2(\boldsymbol{x}_1), \ldots$—namely the series of (sub)gradients $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots$ of the respective unregularized loss $f_t(\boldsymbol{x})$ with respect to $\boldsymbol{x}$. The task is to do as well as possible relative to the best *static* $\boldsymbol{x}^* \in \mathcal{X}$, i.e., to find an algorithm where the regret

$$R(T) = \underbrace{\sum_{t=1}^{T} f_t(\boldsymbol{x}_t)}_{\text{actual incurred loss}} - \underbrace{\inf_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x})}_{\text{best static predictor}}$$

is small.

AdaGrad is an online learning algorithm with asymptotically sublinear regret. We describe the case when $\mathcal{X} = \mathbb{R}^d$ (§2), where a sparsity-inducing ℓ_1 regularizer is desired (§3), and when $\mathcal{X} \subset \mathbb{R}^d$, in which a projection step is necessary to ensure each $\boldsymbol{x}_t \in \mathcal{X}$ (§4).

2 AdaGrad for (sub)gradient optimization

Standard stochastic (sub)gradient methods move $\boldsymbol{x}_t$ in a minimizing direction, given by $-\boldsymbol{g}_t$. When $\mathcal{X} = \mathbb{R}^d$, the familiar stochastic subgradient descent algorithm is simply

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \boldsymbol{g}_t,$$

where $\eta > 0$ is a scalar *learning rate*.

AdaGrad alters this update to adapt based on historical information, so that frequently occurring features in the gradients get small learning rates and infrequent features get higher ones. As Duchi et al. put it, the learner learns slowly from frequent features but "pays attention" to rare but informative features. In practice, this means that infrequently occurring features can be learned effectively along side more frequently occurring features.

AdaGrad provides a per-feature learning rate at each time step t,

$$\eta_{t,i} = \frac{\eta}{\sqrt{G_{t,ii}}},$$

where each $\mathbf{G}_t \in \mathbb{R}^{d \times d}$ is a diagonal matrices where diagonal element i, i is defined to be $\sum_{t'=1}^{t} g_{t',i}^2$, that is, the sum of the squares of the ith dimension of all historical gradients.[1] When implementing this, you will want to keep a d-dimensional vector representing $\mathrm{diag}(\mathbf{G}_t)$ to store a running total of the squares of the gradients.

When $\mathcal{X} = \mathbb{R}^d$, the AdaGrad update per feature is,

$$x_{t+1,i} = x_{t,i} - \frac{\eta}{\sqrt{\sum_{t'=1}^{t} g_{t',i}^2}} g_{t,i},$$

which may be written compactly for the whole vector as

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta \mathbf{G}_t^{-1/2} \odot \boldsymbol{g}_t,$$

where $\odot$ is element-wise multiplication.

Setting η. Since the learning rate for each feature is quickly adapted, the value for η is far less important than it is with SGD. I have used $\eta = 1.0$ for a very large number of different problems. The primary role of η is to determine how much a feature changes the very first time it is encountered, so in problems with large numbers of extremely rare features, some additional care may be warranted.

3 Adding ℓ_1 regularization

Directly applying stochastic subgradient descent to an ℓ_1 regularized objective fails to produce sparse solutions in bounded time, which has motivated several specialized algorithms that target such objectives. We will use the AdaGrad variant of one such learning algorithm, the so-call *regularized dual averaging* algorithm of Xiao (2010), although other approaches are possible.

Xiao's algorithm makes use of the online *average (sub)gradient* at time t,[2]

$$\overline{\boldsymbol{g}}_t = \frac{1}{t} \sum_{t'=1}^{t} \boldsymbol{g}_{t'}.$$

Note that the subgradients $\boldsymbol{g}_t$ do *not* include terms for the regularizer, they are (sub)derivatives of the unregularized objective only—the regularizer is handled separately in the update. And, importantly, the RDA algorithm assumes $\boldsymbol{w}_1 = \overline{\boldsymbol{g}}_0 = \mathbf{0}$.

Using the average gradient, the ℓ_1 regularized objective may be optimized with the following update:

$$x_{t+1,i} = \begin{cases} 0 & \text{if } |\overline{g}_{t,i}| \leq \lambda \\ -\mathrm{sgn}(\overline{g}_{t,i}) \eta \sqrt{t} (|\overline{g}_{t,i}| - \lambda) & \text{otherwise} \end{cases}.$$

[1] In the Duchi et al. (2011) paper, this is notated $\mathrm{diag}(G_t)$ and is a diagonal approximation to the outer product matrix of the sequence of gradients up to time t. Since computing this outer product matrix and taking its square root is computationally intractable when d is even moderately large, I assume the diagonal form throughout to keep things simple. However, the outer product form has better guarantees and should be used if d is small; refer to the original paper for details.

[2] The "dual average" in the name comes from the fact that the average of the subgradients exist in a dual space for the original problem. They can be understood in contrast to the *primal average* $\overline{\boldsymbol{x}}$.

This may be adapted to use the AdaGrad's adaptive learning rate as follows:

$$x_{t+1,i} = \begin{cases} 0 & \text{if } |\overline{g}_{t,i}| \leq \lambda \\ -\text{sgn}(\overline{g}_{t,i}) \frac{\eta t}{\sqrt{G_{t,ii}}} (|\overline{g}_{t,i}| - \lambda) & \text{otherwise} \end{cases}.$$

This is particularly simple to implement by keeping track of the unnormalized averaged subgradient $\boldsymbol{u}_t = t\overline{\boldsymbol{g}} = \boldsymbol{u}_{t-1} + \boldsymbol{g}_t$, and then letting

$$x_{t+1,i} = \begin{cases} 0 & \text{if } \frac{|u_{t,i}|}{t} \leq \lambda \\ -\text{sgn}(u_{t,i}) \frac{\eta t}{\sqrt{G_{t,ii}}} (\frac{|u_{t,i}|}{t} - \lambda) & \text{otherwise} \end{cases}.$$

At time t, the $\boldsymbol{u}$ vector must only be updated with the (generally sparse) $\boldsymbol{g}_t$ vector, and the $\boldsymbol{x}$ vector must only be updated with the union of $\boldsymbol{x}_t$'s non-zero components and $\boldsymbol{g}_t$'s non-zero components. This union will be less sparse than the usual update, but when very sparse solutions are sought, it should still be reasonable in practice.

4 AdaGrad with projected (sub)gradients

TODO - This section is not complete. When $\mathcal{X} \neq \mathbb{R}^d$, it is necessary to ensure that updates do not leave $\mathcal{X}$ using a projection operator, i.e.,

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \Pi_{\mathcal{X}}(\boldsymbol{x}_t - \eta \boldsymbol{g}_t) \\ &= \underbrace{\arg\min_{\boldsymbol{x}' \in \mathcal{X}} ||\boldsymbol{x}' - (\boldsymbol{x}_t - \eta \boldsymbol{g}_t)||_2}_{\text{closest point in } \mathcal{X} \text{ to } (\boldsymbol{x}_t - \eta \boldsymbol{g}_t) \text{ under Euclidean distance}}. \end{aligned}$$

AdaGrad, in addition to providing a per-feature learning rate, also alters the projection operator so as to respect the geometry of the learning problem (as revealed through the historical gradients): rather than the Euclidean distance, it uses the Mahalanobis distance with covariance $\mathbf{G}_t$, i.e.,

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \Pi_{\mathcal{X}}^{\mathbf{G}_t}(\boldsymbol{x}_t - \eta \mathbf{G}_t^{-1/2} \odot \boldsymbol{g}_t) \\ \boldsymbol{z}_t &= \boldsymbol{x}_t - \eta \mathbf{G}_t^{-1/2} \odot \boldsymbol{g}_t \\ \boldsymbol{x}_{t+1} &= \underbrace{\arg\min_{\boldsymbol{x}' \in \mathcal{X}} (\boldsymbol{x}' - \boldsymbol{z}_t)^\top \mathbf{G}_t^{-1/2} (\boldsymbol{x}' - \boldsymbol{z}_t)}_{\text{closest point in } \mathcal{X} \text{ to } \boldsymbol{z}_t \text{ under Mahalanobis distance}}. \end{aligned}$$

Note that if $\mathbf{G}_t = \mathbf{I}$, this is equivalent to the standard Euclidean projection operator. And in the more general case when $\mathbf{G}_t$ is diagonal, we may rewrite this as

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}' \in \mathcal{X}} \sum_{i=1}^{d} \frac{(x'_i - z_{t,i})^2}{G_{t,ii}}.$$

Acknowledgements

I thank Brendan O'Connor, Waleed Ammar, Ankur Parikh, and Sam Thompson for comments on these notes. Any errors are my own.

References

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Lin Xiao. 2010. Dual averaging methods for regularized stochastic learning and online optimization. *JMLR*, 11:2543–2596.