

# Adaptive Subgradient Methods for Stochastic Optimization

Hanxiao Liu  
hanxiaol@cs.cmu.edu

October 28, 2015

- 1 Stochastic (sub)gradient methods
  - Formulations
  - Limitations
- 2 Adaptive stochastic (sub)gradient methods (AdaGrad)
  - Formulations
  - Interpretations
  - Theoretical Guarantees
- 3 Empirical comparison

# Stochastic (Sub)gradient Methods

## Notations

$w_t$  - model parameter at the  $t$ -th step

$g_t$  - (sub)gradient at the  $t$ -th step, i.e.  $g_t \in \partial_w f_t(w)$

$\mathcal{W}$  - feasible domain for  $w$

$\eta_0$  - learning rate

## Stochastic (sub)gradient method (SGD<sup>1</sup>)

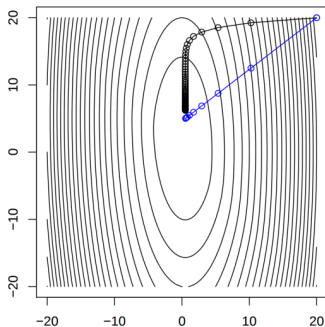
$$w_{t+1} = \Pi_{\mathcal{W}}(w_t - \eta_0 g_t) = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \|w - (w_t - \eta_0 g_t)\|_2^2 \quad (1)$$

- Tricky to find a good learning rate  $\eta_0$
- Implicitly presumes the data geometry to be isotropic

---

<sup>1</sup>strictly speaking, SGD is subsumed by stochastic subgradient methods

# Limitation of SGD



**Figure :** SGD (black) may suffer from inappropriate isotropic assumption about the data geometry

—Is it possible to probe the underlying geometry during stochastic updates, and carry out correction in favor of SGD?

<sup>1</sup><http://www.stat.cmu.edu/~ryantibs/convexopt-S15/lectures/14-newton.pdf>

# AdaGrad - A Geometric Interpretation

AdaGrad [[Duchi et al., 2011](#)]:

Making SGD adaptive to the data geometry

$$w_{t+1} = \Pi_{\mathcal{W}} \left( w_t - \eta_0 G_t^{-\frac{1}{2}} g_t \right) \quad (2)$$

$$= \operatorname{argmin}_{w \in \mathcal{W}} \left\| w - \left( w_t - \eta_0 G_t^{-\frac{1}{2}} g_t \right) \right\|_{G_t^{-\frac{1}{2}}}^2 \quad (3)$$

where  $G_t := \sum_{\tau=0}^t g_{\tau} g_{\tau}^{\top}$  is a covariance matrix estimated from historical subgradients for 2nd-order correction

- Similar in spirit to quasi-Newton SGD [[Bordes et al., 2009](#)]

$$w_{t+1} = \Pi_{\mathcal{W}} \left( w_t - \eta_0 \tilde{H}_t^{-1} g_t \right) \quad (4)$$

- But is more widely applicable, e.g., to nonsmooth problems

# AdaGrad - Approximation

$$\text{Full AdaGrad: } w_{t+1} = \Pi_{\mathcal{W}} \left( w_t - \eta_0 G_t^{-\frac{1}{2}} g_t \right)$$

- Problem:  $G_t^{\frac{1}{2}}$  is computationally impractical for large  $p$ .
- Workaround: replace  $G_t$  with its diagonal:

$$w_{t+1} = \Pi_{\mathcal{W}} \left( w_t - \eta_0 \text{diag}(G_t)^{-\frac{1}{2}} g_t \right) \quad (5)$$

$$= \Pi_{\mathcal{W}} \left( w_t - \eta'_t \odot g_t \right) \quad (6)$$

$\eta'_t \in \mathbb{R}^p$ : learning rates adapted to different dimensions

$$\eta'_{t,j} = \frac{\eta_0}{\sqrt{\sum_{\tau=0}^t g_{\tau,j}^2}} \quad j = 1, 2, \dots, p \quad (7)$$

# AdaGrad - An Alternative Interpretation

Per-feature learning rate:  $\eta'_{t,j} = \eta_0 / \sqrt{\sum_{\tau=0}^t g_{\tau,j}^2}$

AdaGrad takes notice of infrequent features by associating them with relatively larger learning rates during optimization

- The  $j$ -th feature is rare  $\implies g_{\tau,j} = 0$  with a high probability  $\implies \eta'_{t,j} = \frac{\eta_0}{\sqrt{\sum_{\tau=0}^t g_{\tau,j}^2}}$  is large
- Infrequent features are usually more predictive!

Heuristics that emphasize rare features: TF-IDF (preprocessing)

# AdaGrad - Guarantees

Let  $\mathcal{W} = \{w : \|w\|_\infty \leq 1\}$  and  $R(t) = \sum_{\tau=0}^t [f_\tau(w_\tau) - f_\tau(w^*)]$ .  
With diagonal approximation<sup>2</sup>

$$R_{ada}(t) \leq 2\sqrt{2} \times \sum_{j=1}^p \left[ \sum_{\tau=0}^t g_{\tau,j}^2 \right]^{\frac{1}{2}} \quad (8)$$

$$\text{while } R_{sgd}(t) \leq 2\sqrt{2} \times p \left[ \sum_{j=1}^p \sum_{\tau=0}^t g_{\tau,j}^2 \right]^{\frac{1}{2}} \quad (9)$$

AdaGrad has a tighter bound over SGD:

$$\sum_{j=1}^p \left[ \sum_{\tau=0}^t g_{\tau,j}^2 \right]^{\frac{1}{2}} \stackrel{\text{Cauchy's}}{\leq} \left( p \sum_{j=1}^p \sum_{\tau=0}^t g_{\tau,j}^2 \right)^{\frac{1}{2}} \leq p \left[ \sum_{j=1}^p \sum_{\tau=0}^t g_{\tau,j}^2 \right]^{\frac{1}{2}} \quad (10)$$

---

<sup>2</sup>See [Duchi et al., 2011] for proof details.



# Other Popular Methods

A non-exhaustive list

- Momentum [Rumelhart et al., 1988]
- NAG: Nesterov's accelerated gradient [Nesterov et al., 1994]
- AdaDelta: AdaGrad refined [Zeiler, 2012]
- Rprop & Rmsprop [Tieleman and Hinton, 2012]: Ignoring the magnitude of gradient

Demos <sup>3</sup>

- Animation 0
- Animation 1
- Animation 2
- Animation 3

---







<sup>3</sup>[https://www.reddit.com/r/MachineLearning/comments/2gopfa/visualizing\\_gradient\\_optimization\\_techniques/cklhott](https://www.reddit.com/r/MachineLearning/comments/2gopfa/visualizing_gradient_optimization_techniques/cklhott)

AdaGrad is a variant of SGD which

- Adaptively adjusts to data geometry
- Adaptively determines learning rates for different dimensions

Advantages of AdaGrad

- Easy to implement and tends to work well in practice.
- Not sensitive to the initial learning rate ( $\eta_0$ ).
- Generic, backed up by nice theoretical guarantees.

-  Bordes, A., Bottou, L., and Gallinari, P. (2009).  
Sgd-qn: Careful quasi-newton stochastic gradient descent.  
*The Journal of Machine Learning Research*, 10:1737–1754.
-  Duchi, J., Hazan, E., and Singer, Y. (2011).  
Adaptive subgradient methods for online learning and stochastic optimization.  
*The Journal of Machine Learning Research*, 12:2121–2159.
-  Nesterov, Y., Nemirovskii, A., and Ye, Y. (1994).  
*Interior-point polynomial algorithms in convex programming*, volume 13.  
SIAM.
-  Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988).  
Learning representations by back-propagating errors.  
*Cognitive modeling*, 5:3.
-  Tieleman, T. and Hinton, G. (2012).  
Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.  
*COURSERA: Neural Networks for Machine Learning*, 4.
-  Zeiler, M. D. (2012).  
Adadelta: An adaptive learning rate method.  
*arXiv preprint arXiv:1212.5701*.