


# Adaptive Gradient Methods

## AdaGrad / Adam



Machine Learning for Big Data  
CSE547/STAT548, University of Washington

Sham Kakade

# Announcements:

- HW3 posted
  - Dual coordinate ascent
  - (some review of SGD and random features)
  
- Today:
  - Review: tradeoffs in large scale learning
  - Today: adaptive gradient methods

# Review

Tradeoffs in Large Scale Learning.

# Tradeoffs in Large Scale Learning.

- Many issues sources of “error”
- approximation error: our choice of a hypothesis class
- estimation error: we only have  $n$  samples
- optimization error: computing exact (or near-exact) minimizers can be costly.
- How do we think about these issues?

# The true objective

- hypothesis map  $x \in \mathcal{X}$  to  $y \in \mathcal{Y}$ .
- have  $n$  training examples  $(x_1, y_1), \dots, (x_n, y_n)$  sampled i.i.d. from  $\mathcal{D}$ .
- **Training objective:** have a set of parametric predictors  $\{h(x, w) : w \in \mathcal{W}\}$ ,

$$\min_{w \in \mathcal{W}} \hat{L}_n(w) \text{ where } \hat{L}_n(w) = \frac{1}{n} \sum_{i=1}^n \text{loss}(h(x_i, w), y_i)$$

- **True objective:** to generalize to  $\mathcal{D}$ ,

$$\min_{w \in \mathcal{W}} L(w) \text{ where } L(w) = \mathbb{E}_{(X, Y) \sim \mathcal{D}} \text{loss}(h(X, w), Y)$$

**Optimization:** Can we obtain linear time algorithms to find an  $\epsilon$ -accurate solution? i.e. find  $\hat{h}$  so that

$$L(\hat{w}) - \min_{w \in \mathcal{W}} L(w) \leq \epsilon$$

# Definitions

- Let  $h^*$  is the *Bayes optimal hypothesis*, over all functions from  $\mathcal{X} \rightarrow \mathcal{Y}$ .

$$h^* \in \operatorname{argmin}_h L(h)$$

- Let  $w^*$  is the *best in class hypothesis*

$$w^* \in \operatorname{argmin}_{w \in \mathcal{W}} L(w)$$

- Let  $w_n$  be the *empirical risk minimizer*:

$$w_n \in \operatorname{argmin}_{w \in \mathcal{W}} \hat{L}_n(w)$$

- Let  $\tilde{w}_n$  be what our algorithm returns.

# Loss decomposition

- Observe:

$$\begin{aligned} L(\tilde{w}_n) - L(h^*) &= L(w^*) - L(h^*) && \text{Approximation error} \\ &+ L(w_n) - L(w^*) && \text{Estimation error} \\ &+ L(\tilde{w}_n) - L(w_n) && \text{Optimization error} \end{aligned}$$

- Three parts which determine our performance.
- Optimization algorithms with “best” accuracy dependencies on  $\hat{L}_n$  may not be best.  
Forcing one error to decrease much faster may be wasteful.

# Best in class error

- Fix a class  $\mathcal{W}$ . What is the best estimator of  $w^*$  for this model?
- For a wide class of models (linear regression, logistic regression, etc), the ERM,  $w_n$ , is (in the limit) the best estimator:

$$w_n \in \operatorname{argmin}_{w \in \mathcal{W}} \hat{L}_n(w)$$

- 1 What is the generalization error of best estimator  $w_n$ ?
- 2 How well can we do? Note:

$$\begin{aligned} L(\tilde{w}_n) - L(w^*) &= + L(w_n) - L(w^*) && \text{Estimation error} \\ &+ L(\tilde{w}_n) - L(w_n) && \text{Optimization error} \end{aligned}$$

- **Can we generalize as well as the sample minimizer,  $w_n$ ?**  
(without computing it exactly)



# Statistical Optimality

- Can generalize as well as the sample minimizer,  $w_n$ ? (without computing it exactly)
- For a wide class of models (linear regression, logistic regression, etc), we have that the estimation error is:

$$\mathbb{E}[L(w_n)] - L(w^*) \stackrel{n \rightarrow \infty}{\underset{=}{\sim}} \frac{\sigma_{\text{opt}}^2}{n}$$

where  $\sigma_{\text{opt}}^2$  is an (optimal) problem dependent constant.

- This is the *best* possible statistical rate. (Can quantify the non-asymptotic “burn-in”).
- What is the computational cost of achieving exactly this rate? say for large  $n$ ?

# Averaged SGD

- SGD:

$$w_{t+1} \leftarrow w_t - \eta_t \nabla \text{loss}(h(x, w_t), y)$$

- An (asymptotically) optimal algo:
  - Have  $\eta_t$  go to 0 (sufficiently slowly)
  - (**iterate averaging**) Maintain the a running average:

$$\bar{w}_n = \frac{1}{n} \sum_{t \leq n} w_t$$

- (Polyak & Juditsky, 1992) for large enough  $n$  and with **one pass** of SGD over the dataset:

$$\mathbb{E}[L(\bar{w}_n)] - L(w^*) \stackrel{n \rightarrow \infty}{=} \frac{\sigma_{\text{opt}}^2}{n}$$

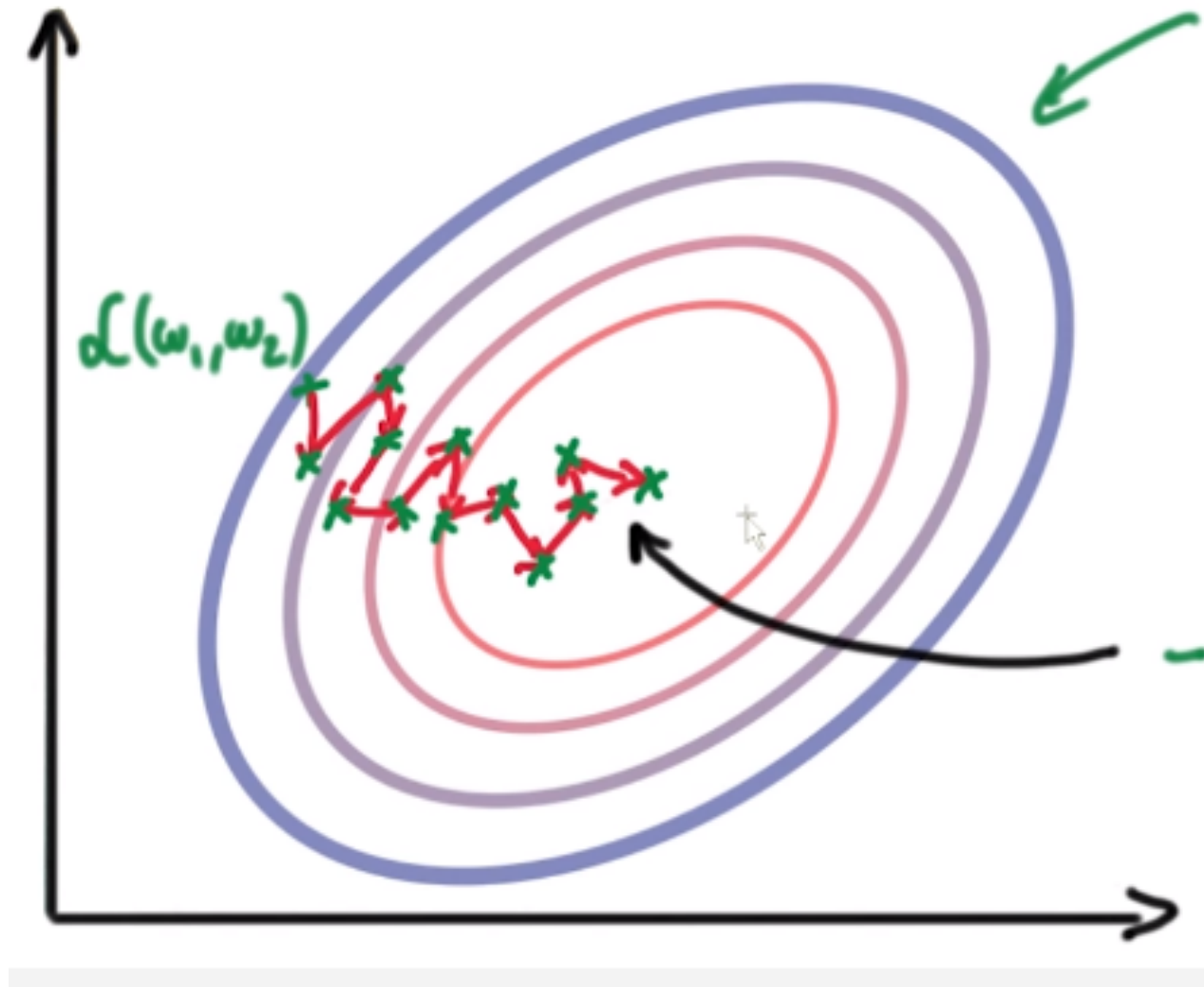
# Adaptive Gradient Methods

## AdaGrad / Adam

Machine Learning for Big Data  
CSE547/STAT548, University of Washington

Sham Kakade

# The Problem with GD (and SGD)



# Adaptive Gradient Methods: Convex Case

- What <sup>do</sup> we want?
- Newton's method:

$$w \leftarrow w - [\nabla^2 L(w)]^{-1} \nabla L(w)$$

- Why is this a good idea?

- Guarantees?

- Stepsize?

$$\eta = 1$$

steps:  $2 \epsilon$

① no dep. on condition #  
②  $\log \log \frac{1}{\epsilon}$

- Related ideas:

- Conjugate Gradient/Acceleration:

- L-BFGS

- Quasi-Newton methods

# Adaptive Gradient Methods: Non-Cvx Case

Descent method  
 $w \leftarrow w - \eta M \nabla \mathcal{L}(w)$

- What do we want?

- Hessian may not be PSD, so is Newton's method a descent method?

NO

if  $M$  is PSD.

$\Rightarrow \mathcal{L}(w)$  will  
not increase  
(for small  $\eta$ )

- Other ideas:

- Natural Gradient methods:

- Curvature adaptive:

- Adagrad, AdaDelta, RMS prop, ADAM, I-BFGS, heavy ball gradient, momentum

- Noise injection:

- Simulated annealing, dropout, Langevin methods

- Caveats:

- Batch methods may be poor: "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima"

# Natural Gradient Idea

- Probabilistic models and maximum likelihood estimation:

$$\hat{L}(w) = -\log \Pr(\text{data}|w)$$

- True likelihood function:

$$L(w) = -E_{z \sim D} \log \Pr(z|w)$$

where  $z$  is sampled from the underlying data distribution  $D$ .

- Suppose the model is correct, i.e.  $z \sim \Pr(z|w^*)$  for some  $w^*$ 
  - Let's look at the Hessian at  $w^*$

$$\begin{aligned} \nabla^2 L(w^*) &= \mathbb{E}_{z \sim \Pr(z|w^*)} [-\nabla^2 \log \Pr(z|w^*)] \\ &= \mathbb{E}_{z \sim \Pr(z|w^*)} [\nabla \log \Pr(z|w^*) (\nabla \log \Pr(z|w^*))^\top] \end{aligned}$$

- How do we approximate the Hessian at  $w$ ?

$$D = \Pr(z|w^*)$$

$$\log f(\omega)$$

$$\partial^2 \log f(\omega) = \partial \left[ \frac{f'}{f} \right]$$

$$= \frac{f f'' - (f')^2}{f^2}$$

$$= \frac{f''}{f} - \frac{(f')^2}{f^2}$$

hit with expectation  $\rightarrow$

$$= \left( \frac{f''}{f} - \left[ (\log f)' \right]^2 \right)$$



# Fisher Information Matrix

$$\nabla^2 L(w) \neq F(w)$$

- Define the Fisher matrix:

$$F(w) := \mathbb{E}_{z \sim \Pr(z|w)} [\nabla \log \Pr(z|w) (\nabla \log \Pr(z|w))^\top]$$

- If the model is correct and if  $w \rightarrow w^*$ , then  $F(w) \rightarrow F(w^*)$
- Natural Gradient:** Use the update rule:

$$w \leftarrow w - [F(w)]^{-1} \nabla L(w)$$

and  
 $\nabla^2 L(w) \rightarrow F(w^*)$

- Empirically, use  $\hat{L}(w)$  and

$$\hat{F}(w) := \frac{1}{T} \sum_t g_t(w) g_t(w)^\top$$

where  $g_t(w)$  is the gradient of the log likelihood of the  $t$ -th data point

# Curvature approximation:

- One idea:

$$\nabla^2 \hat{L}(w) \stackrel{?}{\approx} \frac{1}{T} \sum_t g_t(w) g_t(w)^\top$$

where  $g_t(w)$  is the gradient of the  $t$ -th data point

- Many ideas try to use this approximation
  - Quasi-Newton methods, Gauss newton methods
  - Ellipsoid method (sort of)



S.D.

# Motivating AdaGrad (Duchi, Hazan, Singer 2011)

- Assuming  $\mathbf{w} \in \mathbb{R}^d$ , standard stochastic (sub)gradient descent updates are of the form:

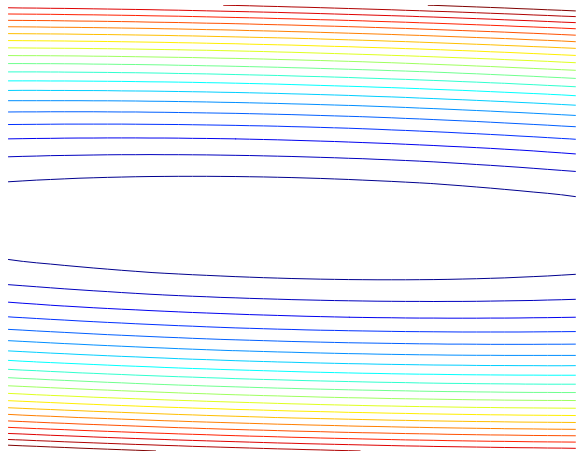
$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_t g_{t,i}$$

- Should all features share the same learning rate?

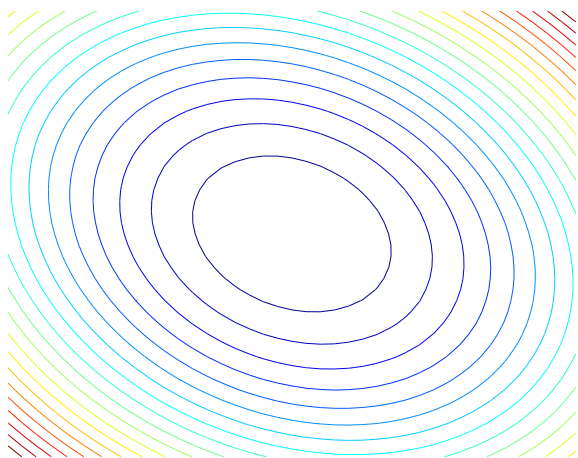
*can try feature specific learning rates?*

- Motivating AdaGrad (Duchi, Hazan, Singer 2011):  
Often have high-dimensional feature spaces
  - Many features are irrelevant
  - Rare features are often very informative
- Adagrad provides a feature-specific adaptive learning rate by incorporating knowledge of the geometry of past observations

# Why Adapt to Geometry?



Hard



Nice

$y_t$	$\mathcal{X}_{t,1}$	$\mathcal{X}_{t,2}$	$\mathcal{X}_{t,3}$
1	1	0	0
-1	.5	0	1
1	-.5	1	0
-1	0	0	0
1	.5	0	0
-1	1	0	0
1	-1	1	0
-1	-.5	0	1

*Examples from  
Duchi et al.  
ISMP 2012  
slides*

- 1 Frequent, irrelevant
- 2 Infrequent, predictive
- 3 Infrequent, predictive

# Not All Features are Created Equal

- Examples:

Text data:

The most unsung birthday in American business and technological history this year may be the 50th anniversary of the **Xerox** 914 photocopier.<sup>a</sup>

---

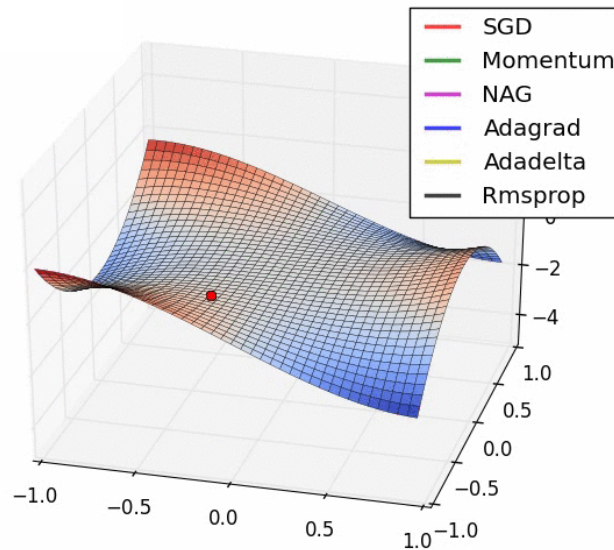
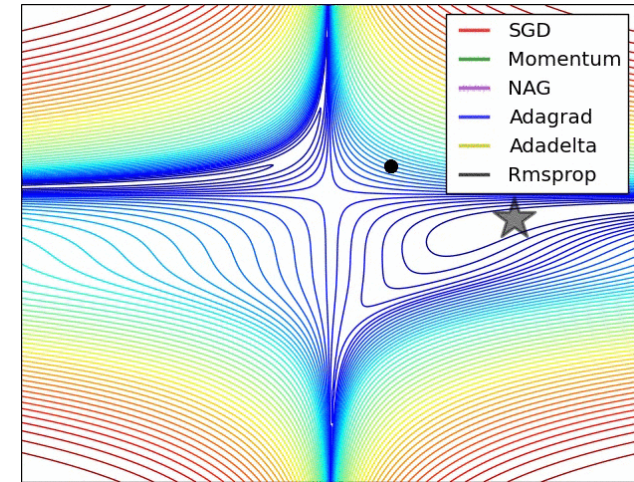
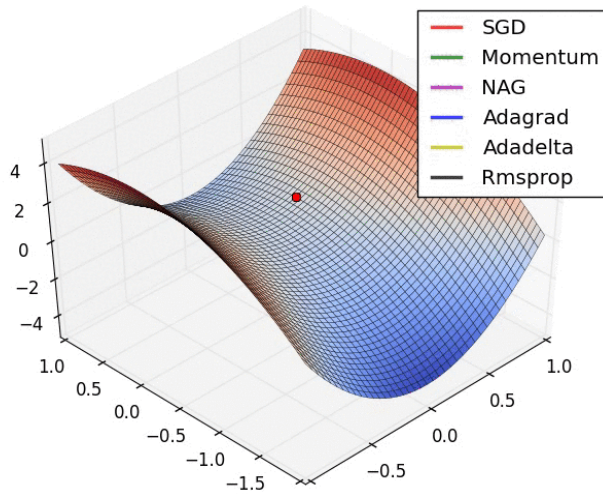
<sup>a</sup> *The Atlantic*, July/August 2010.

High-dimensional image features



*Images from Duchi et al. ISMP 2012 slides*

# Visualizing Effect



**Credit:**  
<http://imgur.com/a/Hqolp>

# Regret Minimization *streaming setting*

- How do we assess the performance of an online algorithm?

- Algorithm iteratively predicts  $\mathbf{w}^{(t)}$

- Incur **loss**  $\ell_t(\mathbf{w}^{(t)})$

- **Regret:**

What is the total incurred loss of algorithm relative to the best choice of  $\mathcal{W}$  that could have been made **retrospectively**

$$R(T) = \sum_{t=1}^T \ell_t(\mathbf{w}^{(t)}) - \inf_{\mathbf{w} \in \mathcal{W}} \sum_{t=1}^T \ell_t(\mathbf{w})$$



# Regret Bounds for Standard SGD

- Standard ~~projected~~ gradient stochastic updates:

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w} - (\mathbf{w}^{(t)} - \eta g_t)\|_2^2$$

suppose  $\|g_t\| \leq B$

- Standard regret bound:

$$\sum_{t=1}^T \left[ \ell_t(\mathbf{w}^{(t)}) - \ell_t(\mathbf{w}^*) \right] \leq \frac{1}{2\eta} \|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 + \frac{\eta}{2} \sum_{t=1}^T \|g_t\|_2^2$$

if  $\eta = O\left(\frac{1}{\sqrt{T}}\right)$

$O\left(\frac{1}{\eta} + \eta T\right)$

Regret =  $O(\sqrt{T})$

$\frac{\text{Regret}}{T} = \frac{1}{\sqrt{T}} \rightarrow 0$



# Projected Gradient using Mahalanobis

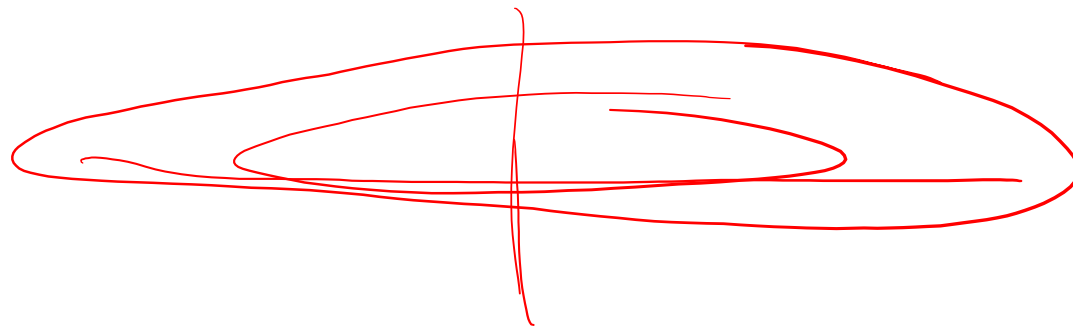
- Standard projected gradient stochastic updates:

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w} - (\mathbf{w}^{(t)} - \eta g_t)\|_2^2$$

- What if instead of an  $L_2$  metric for projection, we considered the **Mahalanobis** norm

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta A^{-1} g_t$$

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w} - (\mathbf{w}^{(t)} - \eta A^{-1} g_t)\|_A^2$$



$$\|x\|_M^2 = x^T M x$$

# Mahalanobis Regret Bounds

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w} - (\mathbf{w}^{(t)} - \eta A^{-1} g_t)\|_A^2$$

- **What  $A$  to choose?**
- Regret bound now:

$$\sum_{t=1}^T \ell_t(\mathbf{w}^{(t)}) - \ell_t(\mathbf{w}^*) \leq \frac{1}{2\eta} \|\mathbf{w}^{(1)} - \mathbf{w}^*\|_A^2 + \frac{\eta}{2} \sum_{t=1}^T \|g_t\|_{A^{-1}}^2$$

- What if we minimize upper bound on regret ~~w.r.t.~~  $A$  in hindsight?

$$\min_A \sum_{t=1}^T g_t^T A^{-1} g_t$$

# Mahalanobis Regret Minimization

- Objective:

$$\min_A \sum_{t=1}^T g_t^T A^{-1} g_t \quad \text{subject to } A \succeq 0, \text{tr}(A) \leq C$$

- Solution:

$$A = c \left( \sum_{t=1}^T g_t g_t^T \right)^{\frac{1}{2}}$$

For proof, see Appendix E, Lemma 15 of Duchi et al. 2011.  
Uses “trace trick” and Lagrangian.

- A defines the norm of the metric space we should be operating in

# AdaGrad Algorithm

- At time  $t$ , estimate optimal (sub)gradient modification  $A$  by

$$A_t = \left( \sum_{\tau=1}^t g_{\tau} g_{\tau}^T \right)^{\frac{1}{2}}$$

- For  $d$  large,  $A_t$  is computationally intensive to compute. Instead,

$$\text{diag}(A_t) = \begin{pmatrix} A_{11} & & 0 \\ & \dots & \\ 0 & & A_{dd} \end{pmatrix} \quad (A_t)_{ii} = \sqrt{\sum_{\tau=1}^t g_{\tau,i}^2}$$

- Then, algorithm is a simple modification of normal updates:

$$\mathbf{w}^{(t+1)} = \arg \min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w} - (\mathbf{w}^{(t)} - \eta \text{diag}(A_t)^{-1} g_t)\|_{\text{diag}(A_t)}^2$$

# AdaGrad in Euclidean Space

- For  $\mathcal{W} = \mathbb{R}^d$ ,

- For each feature dimension,

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \eta_{t,i} g_{t,i}$$

where

$$\eta_{t,i} =$$

- That is,

$$w_i^{(t+1)} \leftarrow w_i^{(t)} - \frac{\eta}{\sqrt{\sum_{\tau=1}^t g_{\tau,i}^2}} g_{t,i}$$

- Each feature dimension has it's own learning rate!
  - Adapts with  $t$
  - Takes geometry of the past observations into account
  - Primary role of  $\eta$  is determining rate the first time a feature is encountered

# AdaGrad Theoretical Guarantees

- AdaGrad regret bound:

$$\sum_{t=1}^T \ell_t(\mathbf{w}^{(t)}) - \ell_t(\mathbf{w}^*) \leq 2R_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2$$

$R_\infty := \max_t \|\mathbf{w}^{(t)} - \mathbf{w}^*\|_\infty$

- In stochastic setting:

$$\mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^T w^{(t)} \right) \right] - \ell(\mathbf{w}^*) \leq \frac{2R_\infty}{T} \sum_{i=1}^d \mathbb{E}[\|g_{1:T,i}\|_2]$$

*with online scaling.*

- This is used in practice.
- Many cool examples. Let's just examine one...

# AdaGrad Theoretical Example

- Expect to out-perform when gradient vectors are *sparse*
- SVM hinge loss example:

$$\ell_t(\mathbf{w}) = [1 - y^t \langle \mathbf{x}^t, \mathbf{w} \rangle]_+$$

$$\mathbf{x}^t \in \{-1, 0, 1\}^d$$

- If  $x_j^t \neq 0$  with probability  $\propto j^{-\alpha}$ ,  $\alpha > 1$

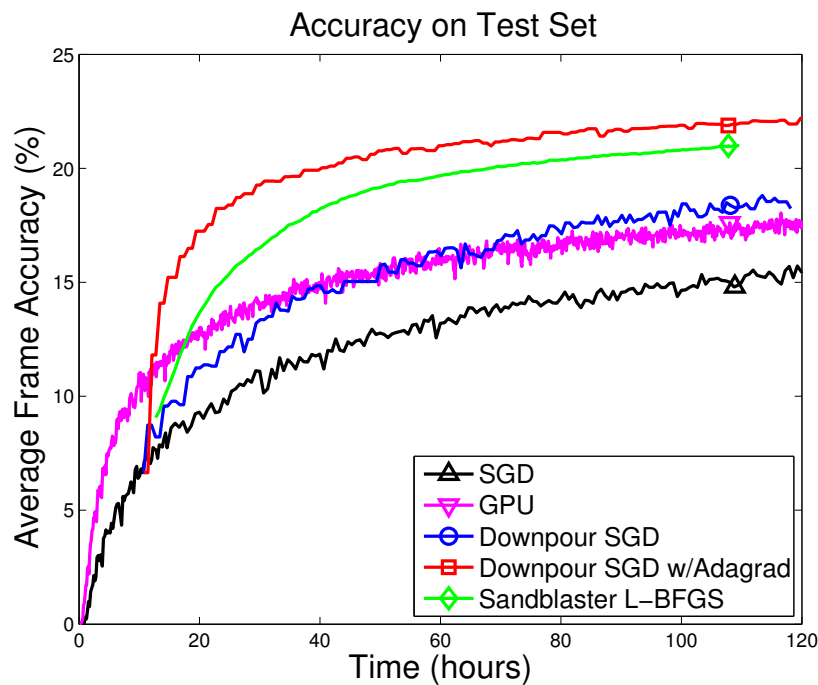
$$\mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^T \mathbf{w}^{(t)} \right) \right] - \ell(\mathbf{w}^*) = \mathcal{O} \left( \frac{\|\mathbf{w}^*\|_\infty}{\sqrt{T}} \cdot \max\{\log d, d^{1-\alpha/2}\} \right)$$

- (sort of) previously bound: 
$$\mathbb{E} \left[ \ell \left( \frac{1}{T} \sum_{t=1}^T \mathbf{w}^{(t)} \right) \right] - \ell(\mathbf{w}^*) = \mathcal{O} \left( \frac{\|\mathbf{w}^*\|_\infty}{\sqrt{T}} \cdot \sqrt{d} \right)$$

# Neural Network Learning

- Very non-convex problem, but use SGD methods anyway

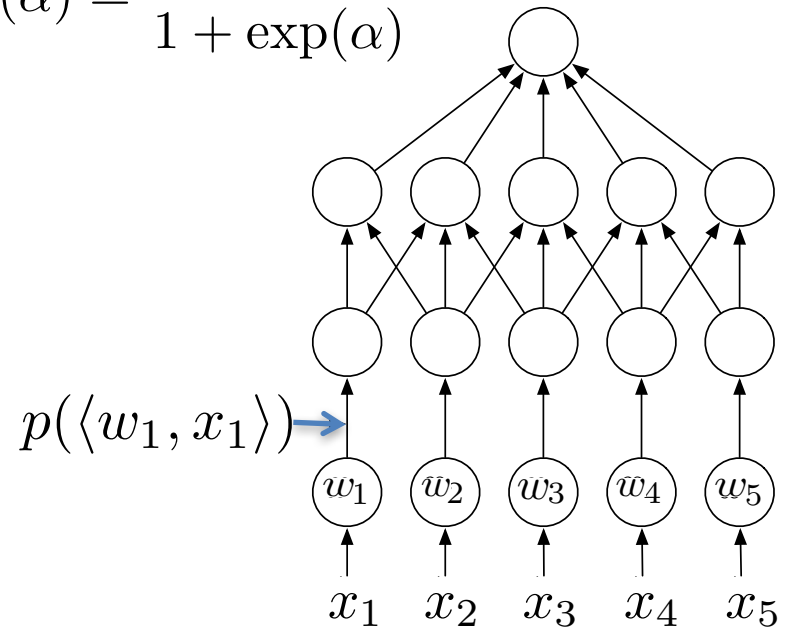
$$\ell(w, x) = \log(1 + \exp(\langle [p(\langle w_1, x_1 \rangle) \cdots p(\langle w_k, x_k \rangle)], x_0 \rangle))$$



(Dean et al. 2012)

Distributed,  $d = 1.7 \cdot 10^9$  parameters. SGD and AdaGrad use 80 machines (1000 cores), L-BFGS uses 800 (10000 cores)

$$p(\alpha) = \frac{1}{1 + \exp(\alpha)}$$



Images from Duchi et al. ISMP 2012 slides



# ADAM

- Like AdaGrad but with “forgetting”
- The algo has component-wise updates

Adam update rule consists of the following steps

- Compute gradient  $g_t$  at current time  $t$
- Update biased first moment estimate

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

- Update biased second raw moment estimate

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

- Compute bias-corrected first moment estimate

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

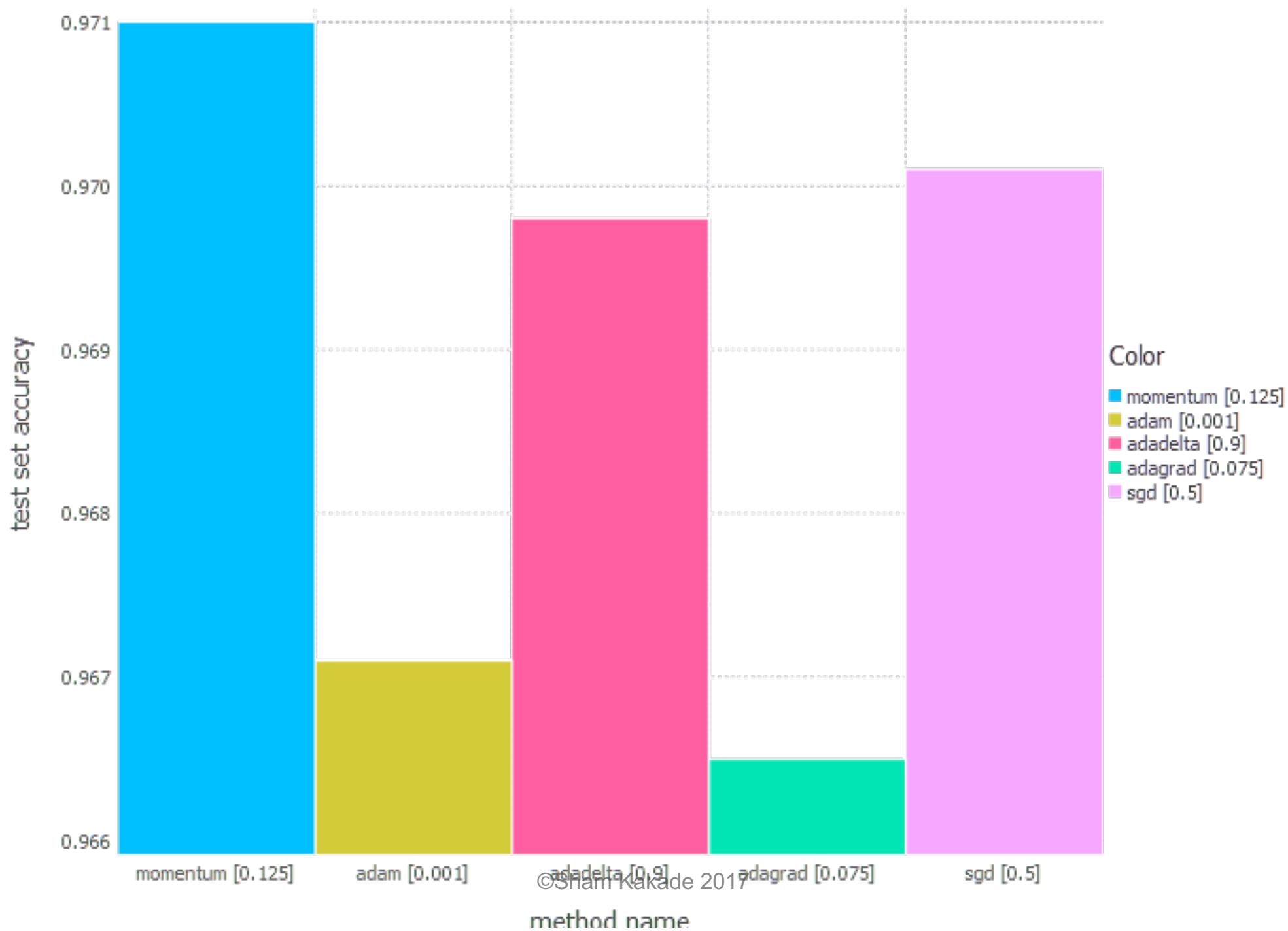
- Compute bias-corrected second raw moment estimate

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- Update parameters

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

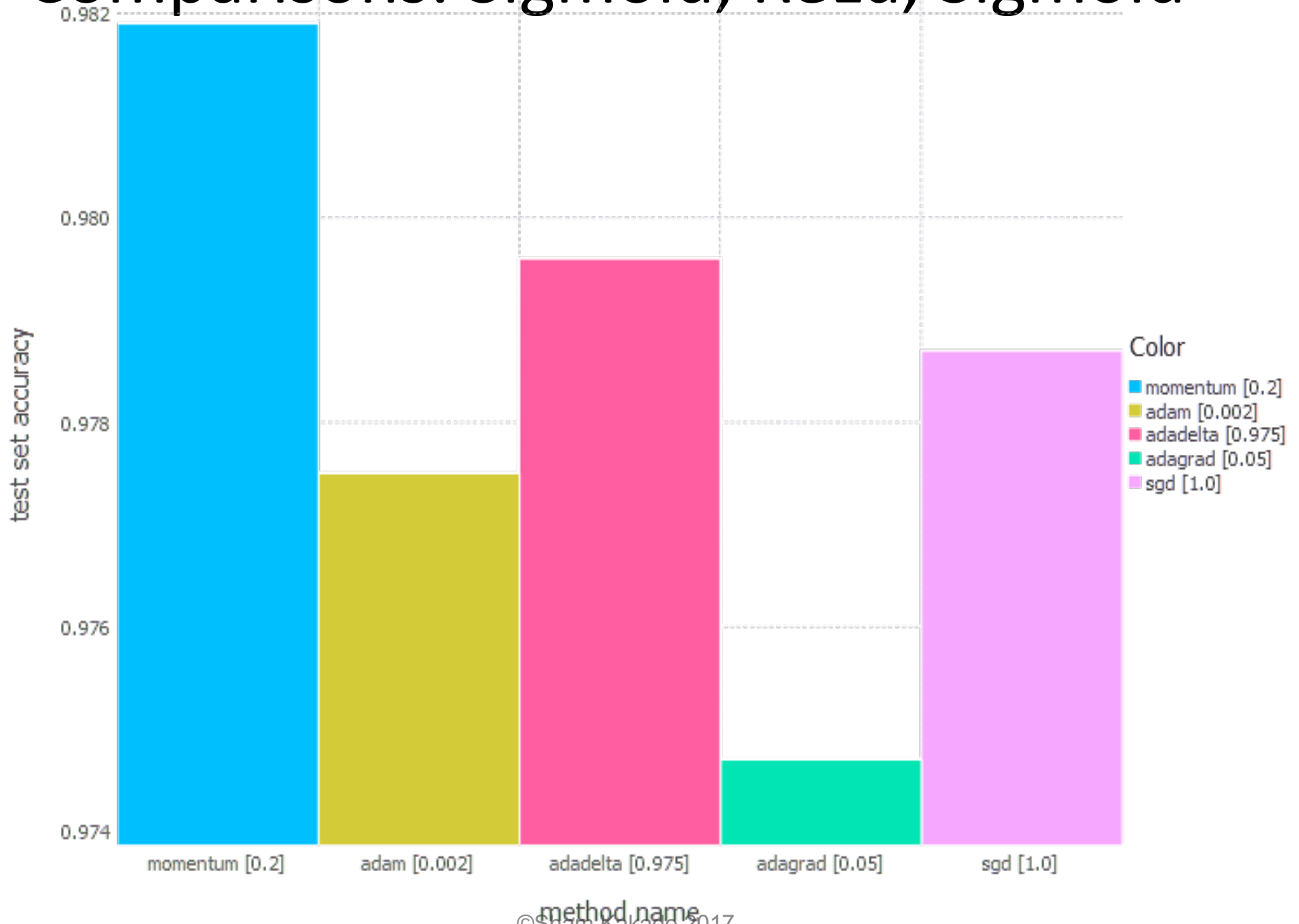
# Comparisons: MNIST, Sigmoid 100 layer



# Comparisons: MNIST, Tanh 100 layer



# Comparisons: Sigmoid, ReLu, Sigmoid



# Acknowledgments

- Some figs taken from: <http://int8.io/comparison-of-optimization-techniques-stochastic-gradient-descent-momentum-adagrad-and-adadelta/>