Applied Deep Learning

Optimization
Mar 12th, 2019

SHANG-YU SU

HTTP://ADL.MIULAB.TW

國立臺灣大學
National Taiwan University

Beyond SGD

Vanilla Gradient Descent

- Computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset.
- As we need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that don't fit in memory.
- Batch gradient descent also doesn't allow us to update our model *online*, i.e. with new examples on-the-fly.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Stochastic Gradient Descent

- Stochastic gradient descent (SGD) in contrast performs a parameter update for *each* training example.
- It is therefore usually much faster and can also be used to learn online.

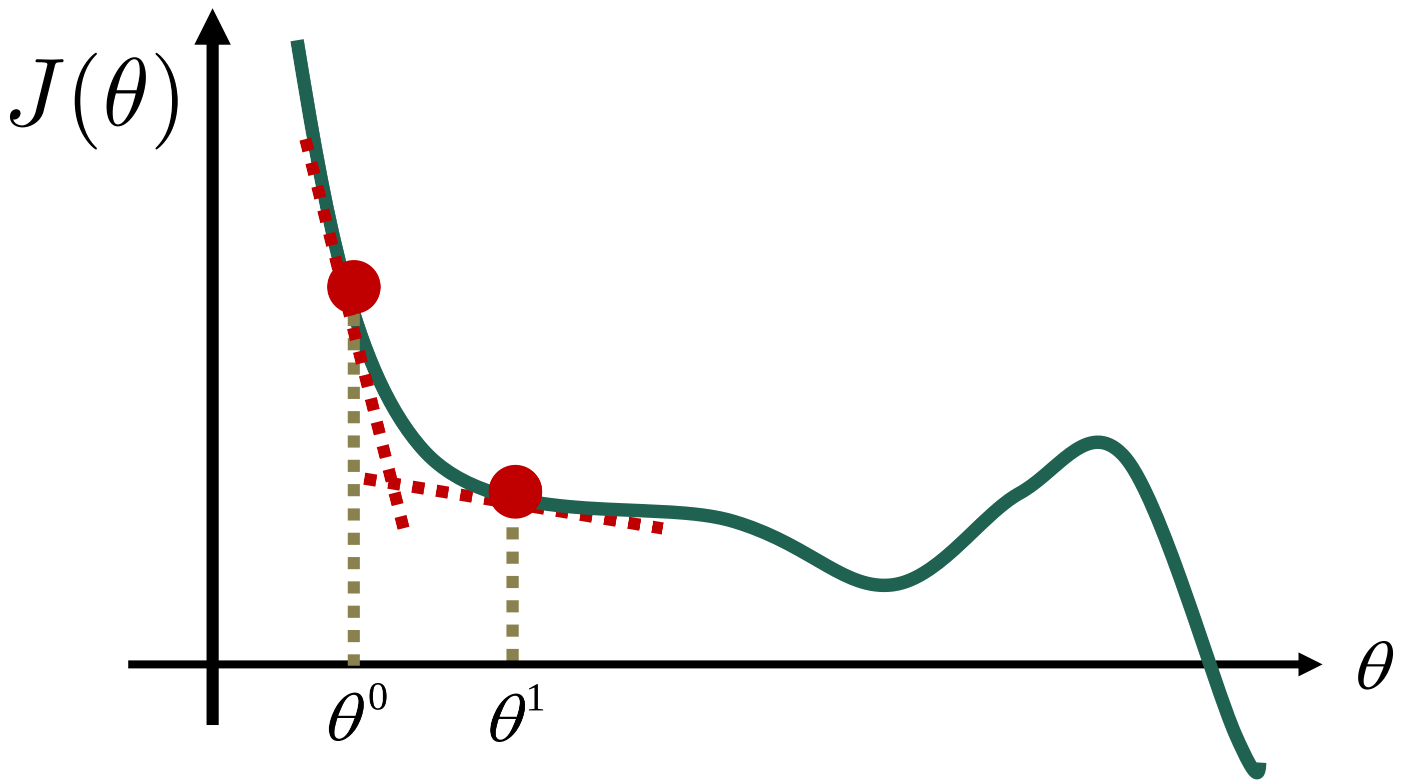
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

Mini-batch Stochastic Gradient Descent

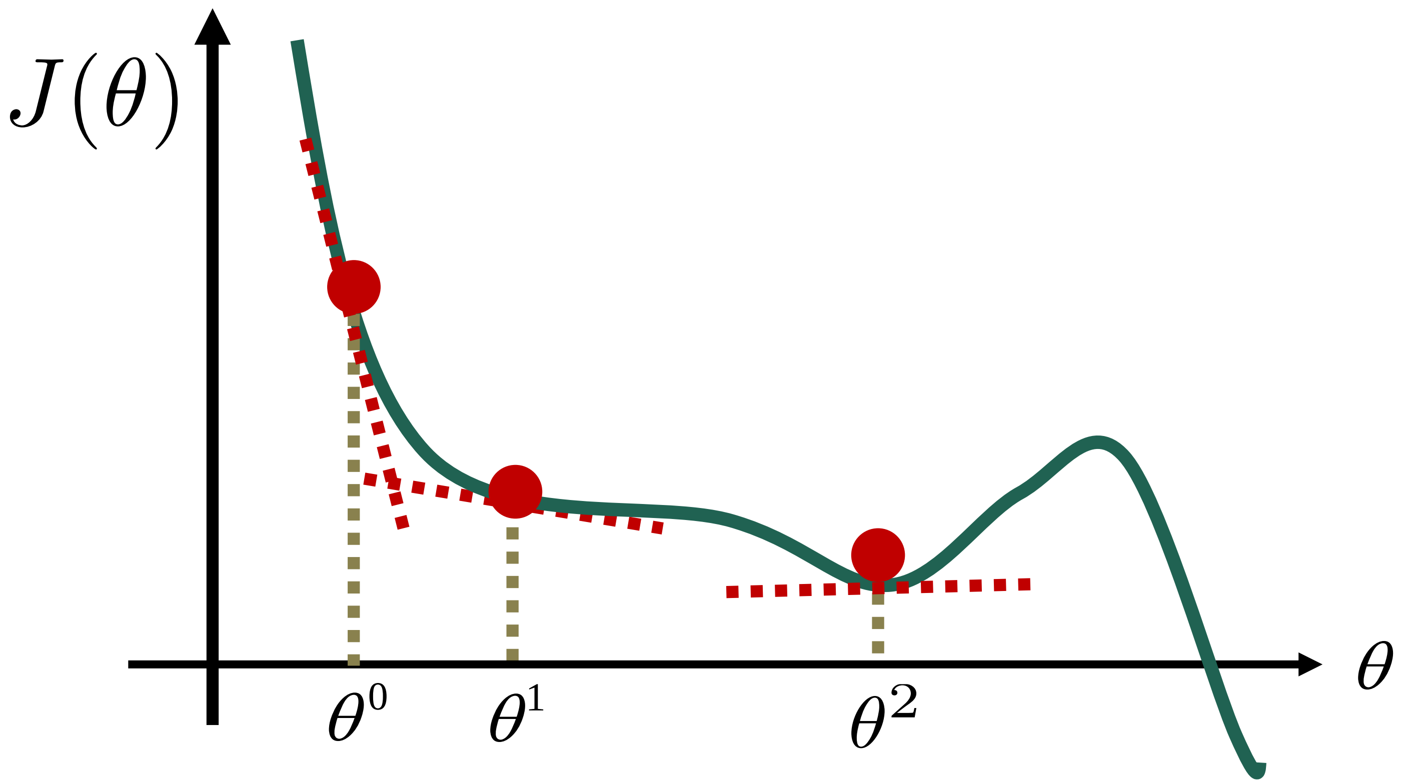
- Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples.
- This way reduces the variance of the parameter updates, which can lead to more stable convergence.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Local Minimum



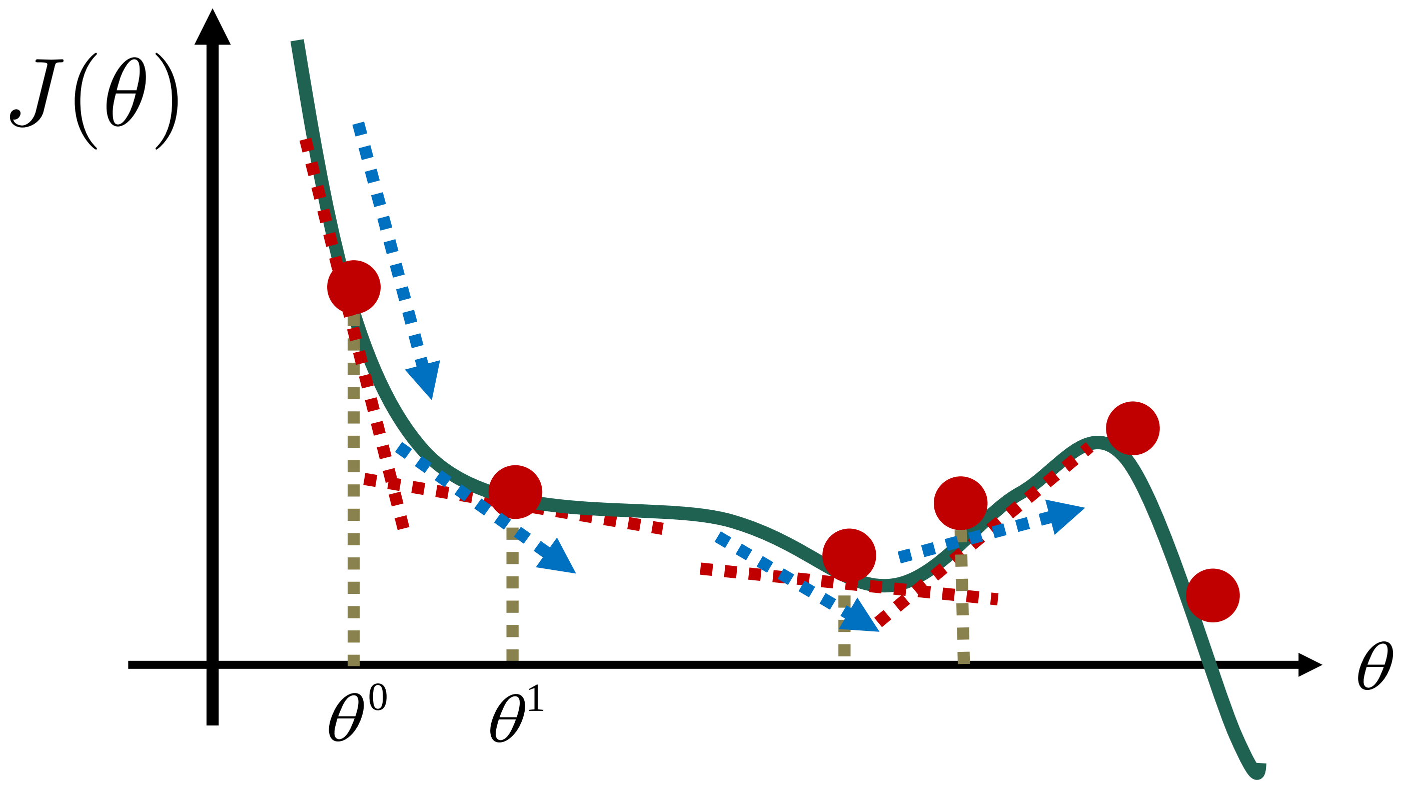
Local Minimum



Challenges

- Choosing a proper learning rate can be difficult.
- Learning rate schedules try to adjust the learning rate during training by e.g. annealing, i.e. reducing the learning rate according to a pre-defined schedule or when the change in objective between epochs falls below a threshold.
- These schedules and thresholds, however, have to be defined in advance and are thus unable to adapt to a dataset's characteristics

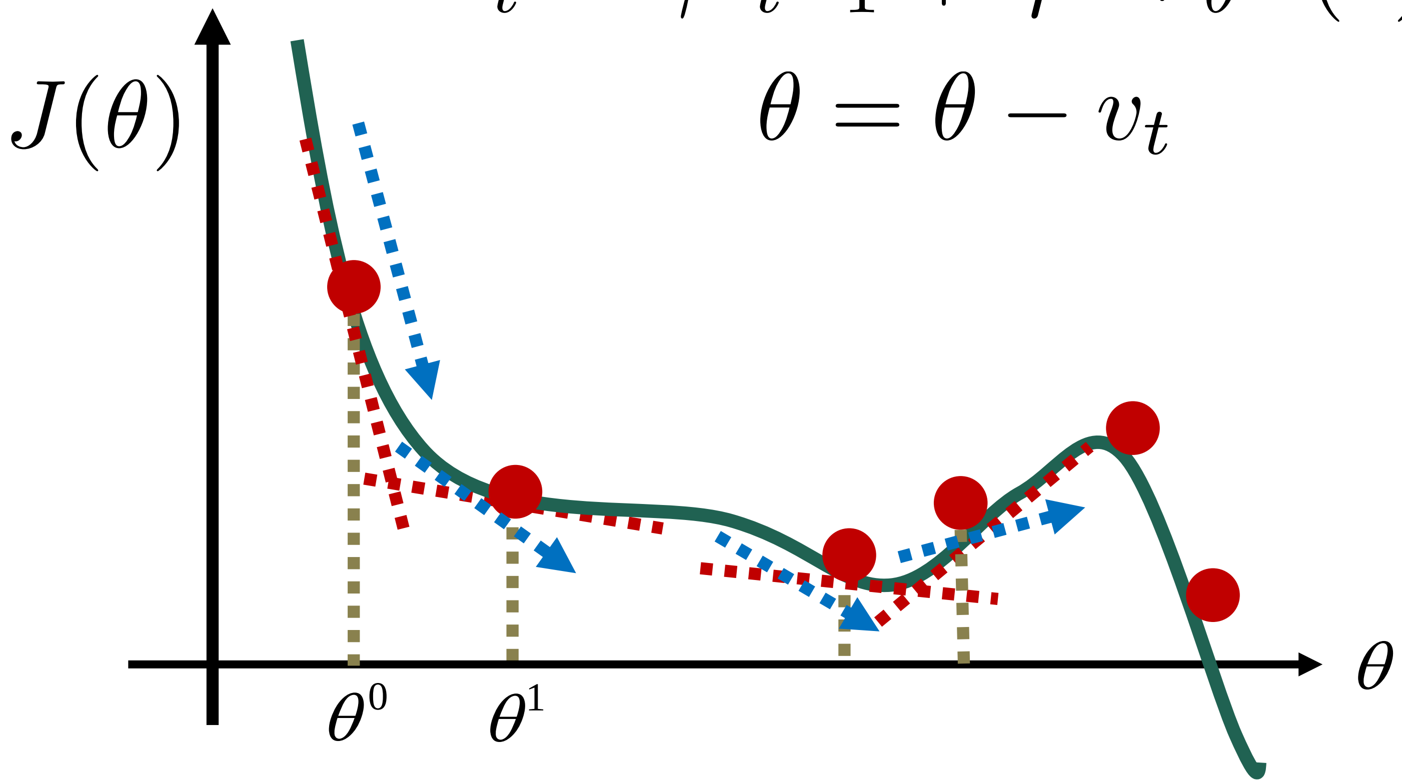
Momentum



Momentum

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$



Momentum

- Mini-batch accumulates the gradient of the past steps to determine the direction to go.
- faster convergence and reduced oscillation.

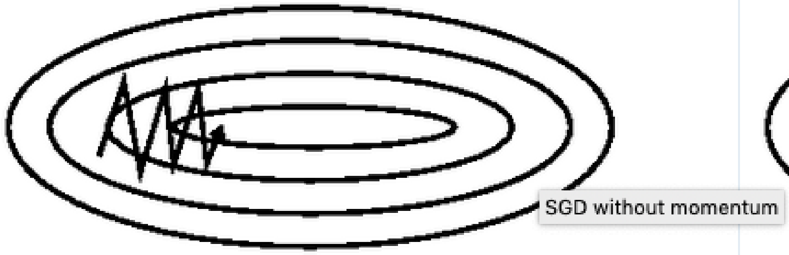


Image 2: SGD without momentum

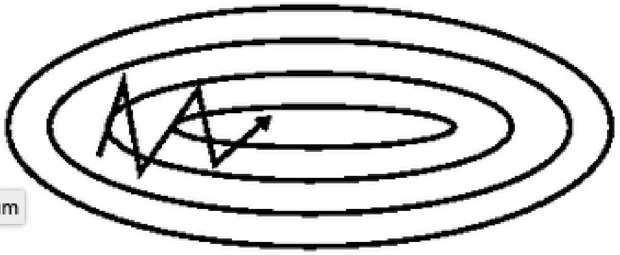


Image 3: SGD with momentum

Nesterov Accelerated Gradient

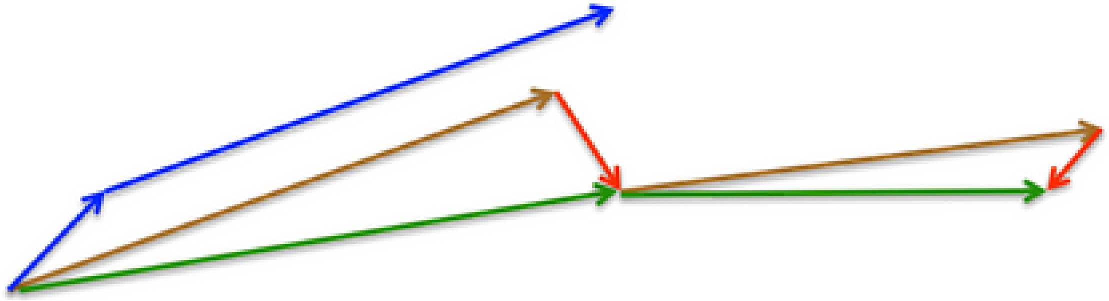
- However, a ball that rolls down a hill, blindly following the slope, is highly unsatisfactory. We'd like to have a smarter ball, a ball that has a notion of where it is going so that it knows to slow down before the hill slopes up again.
- Look ahead by calculating the gradient not w.r.t. to our current parameters θ but w.r.t. the approximate future position of our parameters:

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma v_{t-1})$$

Nesterov Accelerated Gradient

- This anticipatory update prevents us from going too fast and results in increased responsiveness.

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma v_{t-1})$$



Adagrad

- It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features.
- For this reason, it is well-suited for dealing with sparse data.
- G is the accumulation of previous gradient values.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

RMSProp

- Instead of inefficiently storing all previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients.
- resolve Adagrad's radically diminishing learning rates
- **RNN?**

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \odot g_t$$

Adam

- **common choice**
- first moment + second moment (momentum + RMSprop)

$$m_t = \gamma_1 m_{t-1} + (1 - \gamma_1) g_t$$

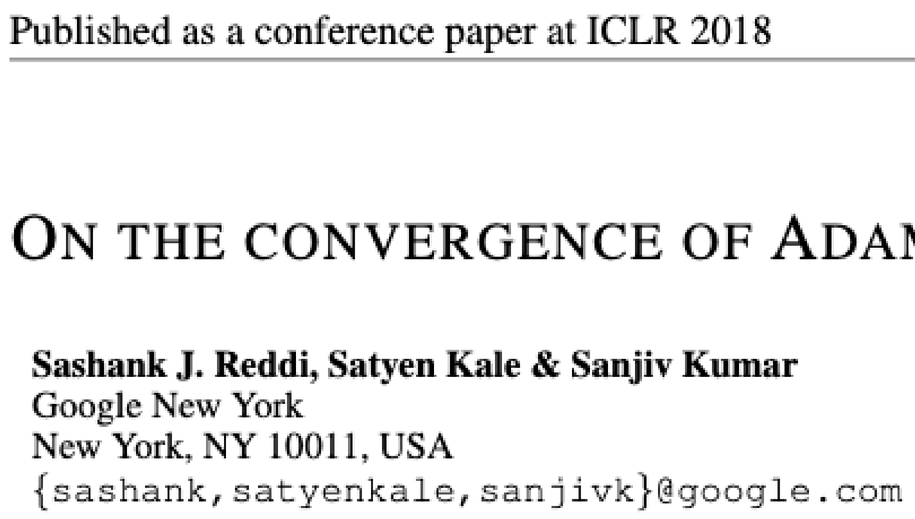
$$v_t = \gamma_2 v_{t-1} + (1 - \gamma_2) g_t^2$$

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

Which to use?

Adam is the best?

- Issue of non-convergence



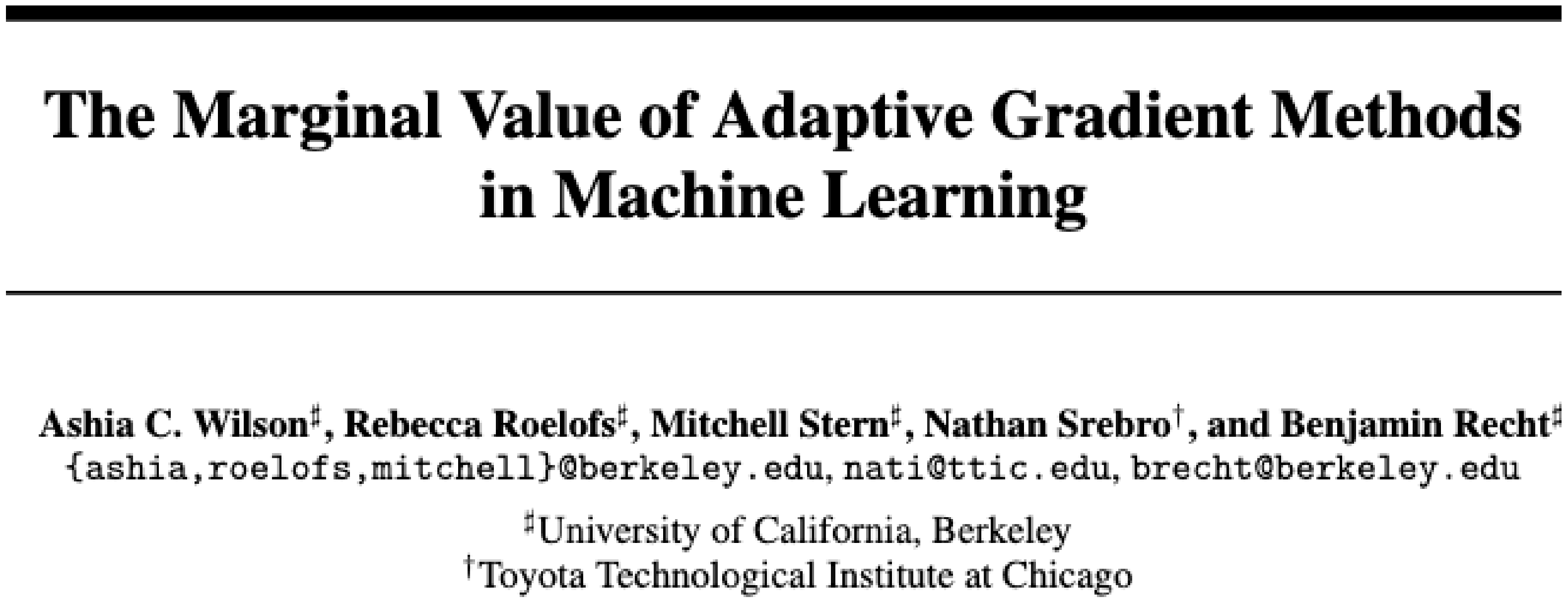
Published as a conference paper at ICLR 2018

ON THE CONVERGENCE OF ADAM AND BEYOND

Sashank J. Reddi, Satyen Kale & Sanjiv Kumar
Google New York
New York, NY 10011, USA
{sashank,satyenkale,sanjivk}@google.com

Missing Global-Optima

- The solutions found by adaptive methods generalize worse (often significantly worse) than SGD, even when these solutions have better training performance. These results suggest that practitioners should reconsider the use of adaptive methods to train neural networks



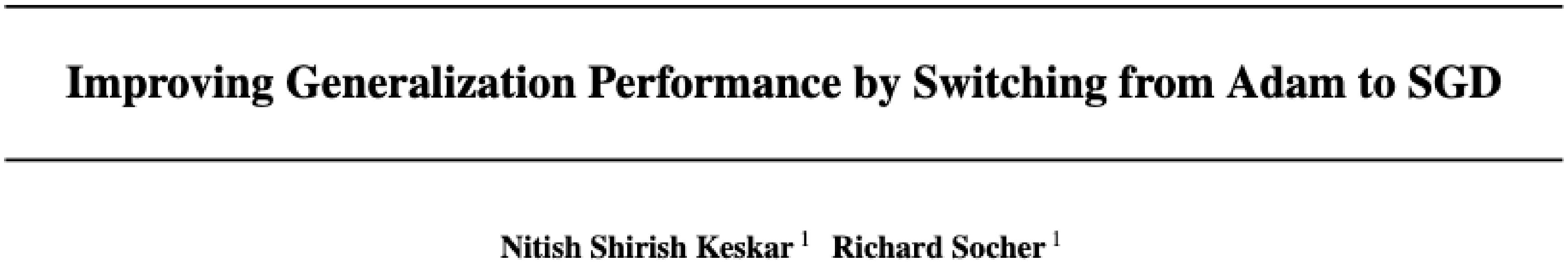
The Marginal Value of Adaptive Gradient Methods in Machine Learning

Ashia C. Wilson[♯], **Rebecca Roelofs**[♯], **Mitchell Stern**[♯], **Nathan Srebro**[†], **and Benjamin Recht**[♯]
{ashia,roelofs,mitchell}@berkeley.edu, nati@ttic.edu, brecht@berkeley.edu

[♯]University of California, Berkeley
[†]Toyota Technological Institute at Chicago

Adam + SGD

- prior period : Adam for fast convergence
- last period: SGD for gradually seeking the global optima



Improving Generalization Performance by Switching from Adam to SGD

Nitish Shirish Keskar [1] Richard Socher [1]

Back to the Data

算法固然美好，数据才是根本。

另一方面，Adam之流虽然说已经简化了调参，但是并没有一劳永逸地解决问题，默认参数虽然好，但也不是放之四海而皆准。因此，在充分理解数据的基础上，依然需要根据数据特性、算法特性进行充分的调参实验，找到自己炼丹的最优解。而这个时候，不论是Adam，还是SGD，于你都不重要了。

少年，好好炼丹吧。

references

- http://ruder.io/optimizing-gradient-descent/
- http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Lecture/Gradient%20Descent%20(v2).pdf
- http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Lecture/DNN%20tip.pdf