Stochastic Gradient Methods for Neural Networks

Chih-Jen Lin
National Taiwan University

Last updated: May 25, 2020

Outline

Outline

NN Optimization Problem I

- Recall that the NN optimization problem is

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$$

where

$$f(\boldsymbol{\theta}) = \frac{1}{2C}\boldsymbol{\theta}^T\boldsymbol{\theta} + \frac{1}{l}\sum\nolimits_{i=1}^{l} \xi(\boldsymbol{z}^{L+1,i}(\boldsymbol{\theta}); \boldsymbol{y}^i, Z^{1,i})$$

- Let's simplify the loss part a bit

$$f(\boldsymbol{\theta}) = \frac{1}{2C}\boldsymbol{\theta}^T\boldsymbol{\theta} + \frac{1}{l}\sum\nolimits_{i=1}^{l} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})$$

- The issue now is how to do the minimization

Gradient Descent I

- This is one of the most used optimization method
- First-order approximation

$$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta}$$

- Solve

$$\begin{aligned} \min_{\Delta\boldsymbol{\theta}} \quad & \nabla f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta} \\ \text{subject to} \quad & \|\Delta\boldsymbol{\theta}\| = 1 \end{aligned} \tag{1}$$

- If no constraint, the above sub-problem goes to $-\infty$

Gradient Descent II

- The solution of (1) is

$$\Delta\boldsymbol{\theta} = -\frac{\nabla f(\boldsymbol{\theta})}{\|\nabla f(\boldsymbol{\theta})\|}$$

- This is called steepest descent method
- In general all we need is a descent direction

$$\nabla f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta} < 0$$

Gradient Descent III

- From

$$f(\boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}) = f(\boldsymbol{\theta}) + \alpha\nabla f(\boldsymbol{\theta})^T\Delta\boldsymbol{\theta} + \frac{1}{2}\alpha^2\Delta\boldsymbol{\theta}^T\nabla^2 f(\boldsymbol{\theta})\Delta\boldsymbol{\theta} + \cdots,$$

if

$$\nabla f(\boldsymbol{\theta})^T\Delta\boldsymbol{\theta} < 0,$$

then with a small enough α,

$$f(\boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}) < f(\boldsymbol{\theta})$$

Line Search I

- Because we only consider an approximation

$$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta}$$

  we may not have the strict decrease of the function value
- That is,

$$f(\boldsymbol{\theta}) < f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta})$$

  may occur
- In optimization we then need a step selection procedure

Line Search II

- Exact line search

$$\min_{\alpha} f(\boldsymbol{\theta} + \alpha \Delta \boldsymbol{\theta})$$

  This is a one-dimensional optimization problem
- In practice, people use backtracking line search
- We check

$$\alpha = 1, \beta, \beta^2, \ldots$$

  with $\beta \in (0, 1)$ until

$$f(\boldsymbol{\theta} + \alpha \Delta \boldsymbol{\theta}) < f(\boldsymbol{\theta}) + \nu \nabla f(\boldsymbol{\theta})^T (\alpha \Delta \boldsymbol{\theta})$$

Line Search III

- Here
$$\nu \in (0, \frac{1}{2})$$
- The convergence is well established.
- For example, under some conditions, Theorem 3.2 of Nocedal and Wright (1999) has that
$$\lim_{k \to \infty} \nabla f(\boldsymbol{\theta}^k) = 0,$$
where k is the iteration index
- This means we can reach a stationary point of a non-convex problem

Practical Use of Gradient Descent I

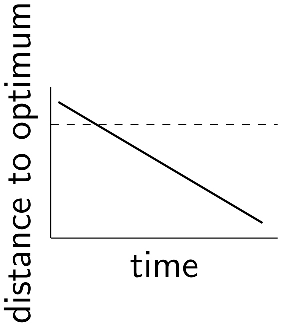
- The standard back-tracking line search is simple and useful
- However, the convergence is slow for difficult problems
- Thus in many optimization applications, methods of using second-order information (e.g., quasi Newton or Newton) are preferred

$$f(\boldsymbol{\theta}+\Delta\boldsymbol{\theta}) \approx f(\boldsymbol{\theta})+\nabla f(\boldsymbol{\theta})^T\Delta\boldsymbol{\theta}+\frac{1}{2}\Delta\boldsymbol{\theta}^T\nabla^2 f(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$$

- These methods have fast final convergence

Practical Use of Gradient Descent II

- An illustration (modified from Tsai et al. (2014))



Slow final convergence   Fast final convergence

Practical Use of Gradient Descent III

- But fast final convergence may not be needed in machine learning
- The reason is that an optimal solution $\boldsymbol{\theta}^*$ may not lead to the best model
- We will discuss such issues again later

Outline

Estimation of the Gradient I

- Recall the function is

$$f(\boldsymbol{\theta}) = \frac{1}{2C}\boldsymbol{\theta}^T\boldsymbol{\theta} + \frac{1}{l}\sum\nolimits_{i=1}^{l}\xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})$$

- The gradient is

$$\frac{\boldsymbol{\theta}}{C} + \frac{1}{l}\nabla_{\boldsymbol{\theta}}\sum_{i=1}^{l}\xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})$$

- Going over all data is time consuming

Estimation of the Gradient II

- What if we use a subset of data

$$E(\nabla_{\boldsymbol{\theta}} \xi(\boldsymbol{\theta}; \boldsymbol{y}, Z^1)) = \frac{1}{l} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{l} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})$$

- We may just use a subset S

$$\frac{\boldsymbol{\theta}}{C} + \frac{1}{|S|} \nabla_{\boldsymbol{\theta}} \sum_{i:i \in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})$$

Algorithm I

Given an initial learning rate η.

while **do**

  Choose $S \subset \{1, \ldots, l\}$.

  Calculate

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\left(\frac{\boldsymbol{\theta}}{C} + \frac{1}{|S|} \nabla_{\boldsymbol{\theta}} \sum_{i:i \in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})\right)$$

  May adjust the learning rate η

end while

- It's known that deciding a suitable learning rate is difficult

Algorithm II

- Too small learning rate: very slow convergence
- Too large learning rate: the procedure may diverge

Stochastic Gradient "Descent" I

- In comparison with gradient descent you see that we don't do line search
- Indeed we cannot. Without the full gradient, the sufficient decrease condition may never hold.

$$f(\boldsymbol{\theta} + \alpha\Delta\boldsymbol{\theta}) < f(\boldsymbol{\theta}) + \nu\nabla f(\boldsymbol{\theta})^T(\alpha\Delta\boldsymbol{\theta})$$

- Therefore, we don't have a "descent" algorithm here
- It's possible that

$$f(\boldsymbol{\theta}^{\text{next}}) > f(\boldsymbol{\theta})$$

- Though people frequently use "SGD," it's unclear if "D" is suitable in the name of this method

Momentum I

- This is a method to improve the convergence speed
- A new vector $\boldsymbol{v}$ and a parameter $\alpha \in [0, 1)$ are introduced

$$\begin{aligned}\boldsymbol{v} &\leftarrow \alpha \boldsymbol{v} - \eta\left(\frac{\boldsymbol{\theta}}{C} + \frac{1}{|S|}\nabla_{\boldsymbol{\theta}} \sum_{i:i\in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})\right) \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \boldsymbol{v}\end{aligned}$$

Momentum II

- Esssentially what we do is

$$
\begin{aligned}
\boldsymbol{\theta} \leftarrow\ & \boldsymbol{\theta} - \eta(\text{current sub-gradient}) \\
& -\alpha\eta(\text{prev. sub-gradient}) \\
& -\alpha^2\eta(\text{prev. prev. sub-gradient}) - \cdots
\end{aligned}
$$

- There are some reasons why doing so can improve the convergence speed, though details are not discussed here

Outline

AdaGrad I

- Scaling learning rates inversely proportional to the square root of sum of past gradient squares (Duchi et al., 2011)
- Update rule:

$$\begin{aligned}
\boldsymbol{g} &\leftarrow \frac{\boldsymbol{\theta}}{C} + \frac{1}{|S|}\nabla_{\boldsymbol{\theta}} \sum_{i:i\in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i}) \\
\boldsymbol{r} &\leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g} \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \frac{\epsilon}{\sqrt{\boldsymbol{r}} + \delta} \odot \boldsymbol{g}
\end{aligned}$$

- $\boldsymbol{r}$: sum of past gradient squares

AdaGrad II

ϵ and δ are given constants

- $\odot$: Hadamard product (element-wise product of two vectors/matrices)
- A large $\boldsymbol{g}$ component

  $\Rightarrow$ a larger $\boldsymbol{r}$ component

  $\Rightarrow$ fast decrease of the learning rate
- Conceptual explanation from Duchi et al. (2011):
  - frequently occurring features $\Rightarrow$ low learning rates
  - infrequent features $\Rightarrow$ high learning rates

AdaGrad III

"the intuition is that each time an infrequent feature is seen, the learner should take notice."

- But how is this explanation related to $\boldsymbol{g}$ components?
- Let's consider linear classification. Recall our optimization problem is

$$\frac{\boldsymbol{w}^T\boldsymbol{w}}{2} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}; y_i, \boldsymbol{x}_i)$$

AdaGrad IV

- For methods such as SVM or logistic regression, the loss function can be written as a function of $\boldsymbol{w}^T\boldsymbol{x}$

  $$\xi(\boldsymbol{w}; y, \boldsymbol{x}) = \hat{\epsilon}(\boldsymbol{w}^T\boldsymbol{x})$$

  Then the gradient is

  $$\boldsymbol{w} + C\sum_{i=1}^{l} \hat{\epsilon}'(\boldsymbol{w}^T\boldsymbol{x}_i)\boldsymbol{x}_i$$

- Thus the gradient is related to the density of features

AdaGrad V

- The above analysis is for linear classification
- But now we have a non-convex neural network!
- Empirically, people find that the sum of squared gradient since the beginning causes too fast decrease of the learning rate

RMSProp I

- The original reference seems to be the lecture slides at `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`
- Idea: they think AdaGrad's learning rate may be too small before reaching a locally convex region
- That is, OK to sum all past gradient squares in convex, but not non-convex
- Thus they do "exponentially weighted moving average"

RMSProp II

- Update rule

$$\begin{aligned} \boldsymbol{r} &\leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g} \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g} \end{aligned}$$

- AdaGrad:

$$\begin{aligned} \boldsymbol{r} &\leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g} \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \frac{\epsilon}{\sqrt{\boldsymbol{r}} + \delta} \odot \boldsymbol{g} \end{aligned}$$

RMSProp III

- Somehow the setting is a bit heuristic and the reason behind the change (from AdaGrad to RMSProp) is not really that strong

ADAM (Adaptive Moments) I

- The update rule (Kingma and Ba, 2015)

$$
\begin{aligned}
\boldsymbol{g} &\leftarrow \frac{\boldsymbol{\theta}}{C} + \frac{1}{|S|} \nabla_{\boldsymbol{\theta}} \sum_{i:i \in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i}) \\
\boldsymbol{s} &\leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1) \boldsymbol{g} \\
\boldsymbol{r} &\leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2) \boldsymbol{g} \odot \boldsymbol{g} \\
\hat{\boldsymbol{s}} &\leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t} \\
\hat{\boldsymbol{r}} &\leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t} \\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \frac{\epsilon}{\sqrt{\hat{\boldsymbol{r}}} + \delta} \odot \hat{\boldsymbol{s}}
\end{aligned}
$$

ADAM (Adaptive Moments) II

- t is the current iteration index
- Roughly speaking, ADAM is the combination of
  - Momentum
  - RMSprop
- From Goodfellow et al. (2016),

$$\frac{\epsilon}{\sqrt{\hat{\boldsymbol{r}}} + \delta} \odot \hat{\boldsymbol{s}}$$

(i.e., the use of momentum combined with rescaling) "does not have a clear theoretical motivation"

ADAM (Adaptive Moments) III

- The two steps

$$\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$$
$$\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t}$$

  are called "bias correction"
- Why "bias correction"?

ADAM (Adaptive Moments) IV

- They hope that

$$E[\boldsymbol{s}_t] = E[\boldsymbol{g}_t]$$

and

$$E[\boldsymbol{r}_t] = E[\boldsymbol{g}_t \odot \boldsymbol{g}_t],$$

where t is the iteration index

ADAM (Adaptive Moments) V

- For $\boldsymbol{s}_t$, we have

$$
\begin{aligned}
\boldsymbol{s}_t &= \rho_1 \boldsymbol{s}_{t-1} + (1-\rho_1)\boldsymbol{g}_t \\
&= \rho_1(\rho_1 \boldsymbol{s}_{t-2} + (1-\rho_1)\boldsymbol{g}_{t-1}) + (1-\rho_1)\boldsymbol{g}_t \\
&= (1-\rho_1)\sum_{i=1}^{t} \rho_1^{t-i}\boldsymbol{g}_i
\end{aligned}
$$

We assume that $\boldsymbol{s}$ is initialized as 0

ADAM (Adaptive Moments) VI

- Then

$$
\begin{aligned}
E[\boldsymbol{s}_t] &= E[(1-\rho_1)\sum_{i=1}^{t}\rho_1^{t-i}\boldsymbol{g}_i] \\
&= E[\boldsymbol{g}_t](1-\rho_1)\sum_{i=1}^{t}\rho_1^{t-i}
\end{aligned}
$$

- Note that we assume

$$E[\boldsymbol{g}_i], \forall i \geq 1$$

are the same

ADAM (Adaptive Moments) VII

- Next,

$$
\begin{aligned}
&(1-\rho_1)\sum_{i=1}^{t}\rho_1^{t-i}\\
=&(1-\rho_1)(1+\cdots+\rho_1^{t-1})\\
=&1-\rho_1^t
\end{aligned}
$$

- Thus

$$E[\boldsymbol{s}_t] = E[\boldsymbol{g}_t](1-\rho_1^t)$$

and they do

$$\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t}$$

ADAM (Adaptive Moments) VIII

- The above derivation on bias correction partially follows from `https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimiz`
- The situation for $E[\boldsymbol{g}_t \odot \boldsymbol{g}_t]$ is similar
- How about ADAM's practical performance?
- From Goodfellow et al. (2016), "generally regarded as being fairly robust to the choice of hyperparmeters, though the learning rate may need to be changed from the default"

ADAM (Adaptive Moments) IX

- However, from the web page we referred to for deriving the bias correction, “The original paper ... showing huge performance gains in terms of speed of training. However, after a while people started noticing, that in some cases Adam actually finds worse solution than stochastic gradient”
- One example of showing the above is Wilson et al. (2017)
- We may do some experiments later

Outline

Choosing Stochastic Gradient Algorithms

- From Goodfellow et al. (2016), "there is currently no consensus"
- Further, "the choice ... seemed to depend on the user's familarity with the algorithm"
- This isn't very good. Can we have some systematic investigation?

Why Stochastic Gradient Widely Used? I

- The special property of data classification is essential

$$E(\nabla_{\boldsymbol{\theta}} \xi(\boldsymbol{\theta}; \boldsymbol{y}, Z^1)) = \frac{1}{\ell} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{\ell} \xi(\boldsymbol{\theta}; \boldsymbol{y}^i, Z^{1,i})$$

  Indeed stochastic gradient is less used outside machine learning
- Easy implementation. It's simpler than methods using, for example, second derivative
- Non-convexity plays a role

Why Stochastic Gradient Widely Used? II

  - For convex, other methods are efficient to find the global minimum
  - But for non-convex, efficiency to reach a stationary point is less useful
  - A global minimum usually gives a good model (loss minimized), but for a stationary point we are less sure
- All these explain why SG is popular for deep learning
- What are your opinions? Any other reasons you can think of

Issues of Stochastic Gradient I

- We have shown several variants
- Don't you think some settings are a bit ad hoc?
- There are reasons behind each change. But some are just heuristic
- Can we try a paradigm completely different?
- But before that we need some first-hand experiences and know implementation details.

References I

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, NY, 1999.

C.-H. Tsai, C.-Y. Lin, and C.-J. Lin. Incremental and decremental training for linear classification. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014. URL http://www.csie.ntu.edu.tw/~cjlin/papers/ws/inc-dec.pdf.

A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.