Stochastic Gradient Methods for Deep Learning

Chih-Jen Lin

National Taiwan Univ.

MBZUAI

June 2023

Outline

Optimization and Stochastic Gradient I

- Optimization is an area rich of methods and theory
- In standard textbooks we see methods like gradient descent, Newton, etc. with their convergence analysis
- Further, convex optimization is an important focus
- On the other hand, in deep learning, which is an important topic of machine learning, a special type of optimization methods (stochastic gradient) is highly popular
- The optimization problem is non-convex

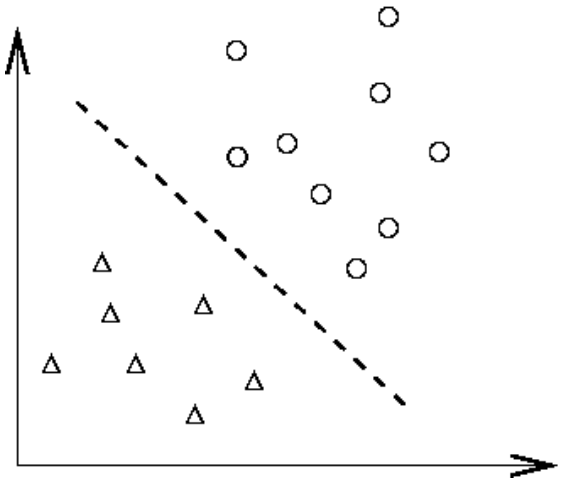
Optimization and Stochastic Gradient II

- Interestingly the stochastic gradient method is less used in areas other than machine learning.
- In this talk, we discuss basic concepts and explain why it is widely used in deep learning
- We also share some stories about its practical use.

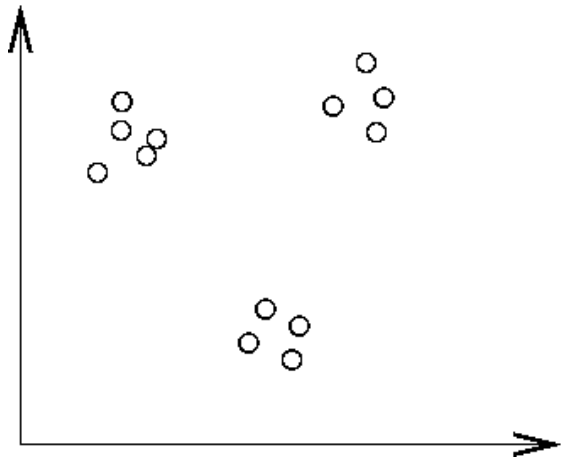
Outline

What is Machine Learning?

- Extract knowledge from data



Classification



Clustering

- We focus on classification. From data with labels, we build a model for prediction

Minimizing Training Errors

- A classification method often starts with minimizing the training errors

$$\min_{\text{model}} \quad \text{(training errors)}$$

- That is, all or most training data with labels should be correctly classified by our model
- A model can be a decision tree, a neural network, or other types
- This is called empirical risk minimization

Empirical Risk Minimization I

- Training data $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}, \boldsymbol{x}_i \in R^n, i = 1, \ldots, l$, $\boldsymbol{y}_i \in \{0,1\}^{\#\text{labels}}$
- l: # of data, n: # of features

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}), \quad f(\boldsymbol{\theta}) = \frac{1}{2C}\boldsymbol{\theta}^T\boldsymbol{\theta} + \frac{1}{l}\sum_{i=1}^{l} \xi(\boldsymbol{\theta}; \boldsymbol{y}_i, \boldsymbol{x}_i)$$

- $\boldsymbol{\theta}$: variables of the optimization problem; also model weights of neural networks
- $\xi(\boldsymbol{\theta}; y, \boldsymbol{x})$: loss function, a way to measure training errors $\Rightarrow$ 2nd term is the average training loss

Empirical Risk Minimization II

- $\boldsymbol{\theta}^T\boldsymbol{\theta}/2$: regularization term (to avoid overfitting)
- C: regularization parameter (chosen by users)

Gradient Descent in Optimization

- We take the negative gradient direction and a suitable step size such that

$$f(\boldsymbol{\theta} - \eta \nabla f(\boldsymbol{\theta})) = f(\boldsymbol{\theta}) - \eta \nabla f(\boldsymbol{\theta})^T \nabla f(\boldsymbol{\theta}) + \cdots$$
$$< f(\boldsymbol{\theta})$$

Estimation of the Gradient I

- Recall the function is

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

- The gradient is

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

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

Estimation of the Gradient II

- Going over all data is time consuming
- If data are from the same distribution

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

then we may just use a subset S (often called a batch)

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

Stochastic Gradient Algorithm I

1: Given an initial learning rate η.
2: **while do**
3: Choose $S \subset \{1, \ldots, l\}$.
4: Calculate

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

5: May adjust the learning rate η (i.e., step size)

Stochastic Gradient Algorithm II

- Deciding a suitable learning rate is difficult
- Too small learning rate: very slow convergence
- Too large learning rate: the procedure may diverge
- This is very different from standard gradient descent methods in optimization, where we check new and existing function values to select the step size

Momentum I

- Because we use a subset of data to get an approximate gradient, the resulting directions may be noisy
- We can consider a moving average of sub-gradients
- A new vector $\boldsymbol{v}$ and a parameter $\alpha \in [0, 1)$ are introduced

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

Momentum II

- However, the rule in (1) may be biased toward the initial value
- Thus we need bias correction, which will be discussed later
- So far the learning rate η is the same for every component of the sub-gradient

AdaGrad I

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

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

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

AdaGrad II

ϵ and δ are given constants

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

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

  $\Rightarrow$ faster decrease of the learning rate
- Though details not shown, a conceptual explanation is that infrequent features correspond to small $\boldsymbol{g}$ components and need larger rates to learn

AdaGrad III

- The above analysis is for convex problems (e.g., logistic regression)
- 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

- This heuristic[1] thinks that AdaGrad's learning rate may be too small before reaching a locally convex region
- Thus they propose "exponentially weighted moving average" for summing $\boldsymbol{g} \odot \boldsymbol{g}$

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:

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

[1] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Adam (Adaptive Moments) I

- The update rule (Kingma and Ba, 2015)

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

Adam (Adaptive Moments) II

- t is the current iteration index
- Roughly speaking, Adam is the combination of
  - Momentum
  - RMSprop
- Adam is now a popular stochastic gradient method

Bias Correction in Adam I

- The two steps in Adam

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

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

  are called "bias correction"

- Due to the moving average, the vector is biased toward the initial value

Bias Correction in Adam II

- For the direction $\boldsymbol{s}$ used to update $\boldsymbol{\theta}$, we hope that its expectation is similar to the expected gradient

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

Weight Decay I

- Recall in our earlier description, the simple stochastic gradient update is

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

- In this calculation

$$\frac{\boldsymbol{\theta}}{C}$$

comes from the regularization term in $f(\boldsymbol{\theta})$

- The use of regularization follows from standard machine learning settings

Weight Decay II

- However, in the area of neural networks, this term may come from a setting called weight decay (Hanson and Pratt, 1988)

$$\boldsymbol{\theta} \leftarrow (1-\lambda)\boldsymbol{\theta} - \eta\left(\frac{1}{|S|}\nabla_{\boldsymbol{\theta}} \sum_{i:i\in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}_i, \boldsymbol{x}_i)\right)$$

  where λ is the rate of weight decay
- In fact, Hanson and Pratt (1988) did not give good reasons for decaying the weight of $\boldsymbol{\theta}$

Weight Decay III

- Clearly, if

$$\lambda = \frac{1}{C}$$

  then weight decay is the same as regularization
- However, as pointed out in Loshchilov and Hutter (2019), the equivalence does not hold if adaptive learning rate is used

Weight Decay IV

- For example, in AdaGrad, the update rule is

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\epsilon}{\sqrt{\boldsymbol{r}} + \delta} \odot \left(\frac{1}{|S|} \nabla_{\boldsymbol{\theta}} \sum_{i:i \in S} \xi(\boldsymbol{\theta}; \boldsymbol{y}_i, \boldsymbol{x}_i)\right) - \frac{\epsilon}{\sqrt{\boldsymbol{r}} + \delta} \odot \frac{\boldsymbol{\theta}}{C}$$

  so the regularization term is scaled in a component-wise way

- Loshchilov and Hutter (2019) advocate to decouple the weight decay step and propose AdamW

AdamW I

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

AdamW II

- This is not equivalent to Adam because in Adam, $\boldsymbol{\theta}/C$ has been used in calculating $\boldsymbol{g}$ and then scaled after
- Why is the decoupled setting better? Some discussions are in Section 3 of Loshchilov and Hutter (2019), but more investigation is needed

Choosing Stochastic Gradient Algorithms

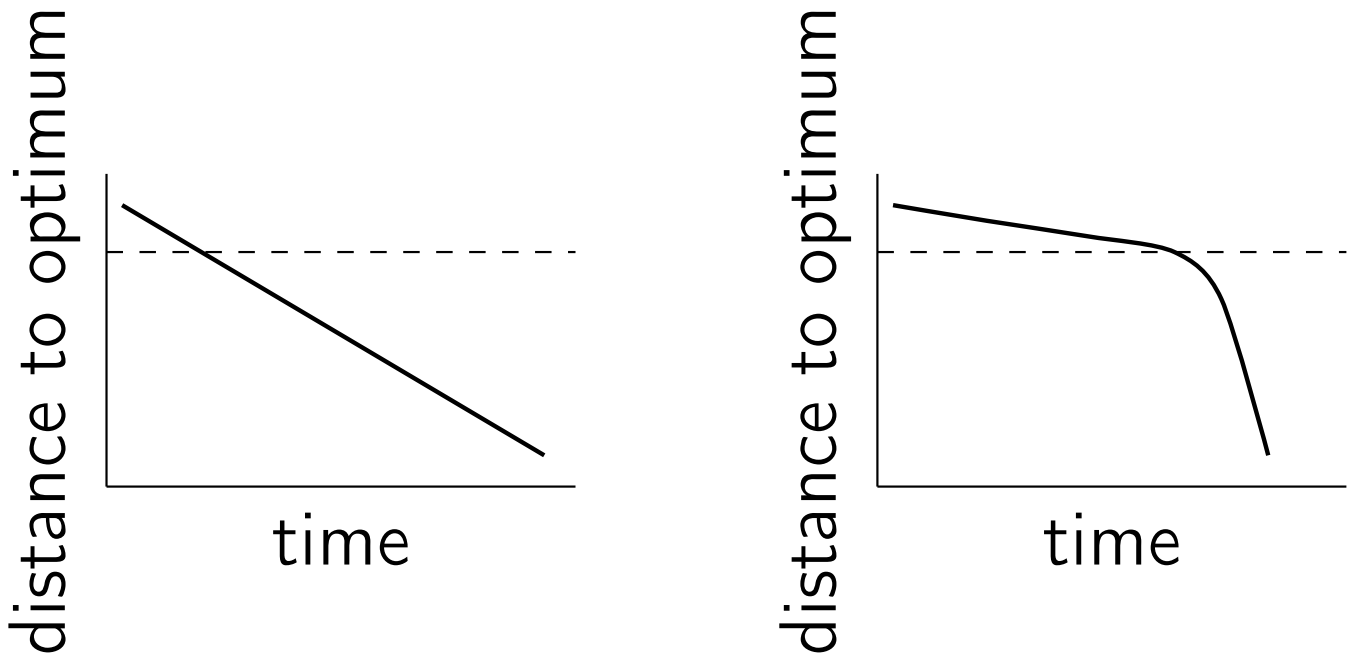
- From Goodfellow et al. (2016), "there is currently no consensus"
- Further, "the choice ... seemed to depend on the user's familiarity with the algorithm"

Why Stochastic Gradient Widely Used? I

- Stochastic gradient is known to converge slowly. However, in machine learning, fast final convergence may not be important
  - An optimal solution $\boldsymbol{\theta}^*$ may not lead to the best model
  - Further, we don't need a point close to $\boldsymbol{\theta}^*$. Suppose the decision value at $\boldsymbol{\theta}^*$ is $0.3 > 0$ and a positive label is predicted. Then an approximate decision value of 0.29 makes no difference. A not-so-accurate $\boldsymbol{\theta}$ may be good enough

Why Stochastic Gradient Widely Used? II

- Thus a method with slow final convergence may be efficient enough



Slow final convergence Fast final convergence

This illustration is modified from Tsai et al. (2014)

Why Stochastic Gradient Widely Used? III

- The special property of data classification is essential

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

- We can cheaply get a good approximation of the gradient
- Easy implementation. It's simpler than methods using, for example, second derivative
- Now gradient is calculated by automatic differentiation

Why Stochastic Gradient Widely Used? IV

- We draw a network and the gradient can be calculated
- Non-convexity plays a role
  - For convex, other methods may possess advantages to more efficiently 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 (as loss is minimized), but for a stationary point we are less sure

Why Stochastic Gradient Widely Used? V

- All these explain why SG is popular for deep learning

Outline

Decrease of Function Values I

- In a gradient descent method, we can choose a small enough η such that

$$f(\boldsymbol{\theta} - \eta \nabla f(\boldsymbol{\theta})) = f(\boldsymbol{\theta}) - \eta \nabla f(\boldsymbol{\theta})^T \nabla f(\boldsymbol{\theta}) + \cdots$$
$$< f(\boldsymbol{\theta})$$

- This relies on that

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

- However, for stochastic gradient, we no longer have this property.

Decrease of Function Values II

- Now assume the simplest situation of selecting one data instance at a time. The sub-gradient

$$\nabla_i f(\boldsymbol{\theta}) \equiv \nabla_{\boldsymbol{\theta}} \xi(\boldsymbol{\theta}; \boldsymbol{y}_i, \boldsymbol{x}_i)$$

  in general does not satisfy

$$-\eta \nabla_i f(\boldsymbol{\theta})^T \nabla f(\boldsymbol{\theta}) < 0$$

- What we can show is the decrease in expectation

Convergence in Expectation I

- Suppose we consider up to T iterations and run up to a random number of iteration τ
- And τ follows a probability distribution such as

$$P(\tau = t) = \frac{\eta_t}{\sum_{k=0}^{T-1} \eta_k},$$

where η_t is the learning rate at each iteration t

- Then we bound the expected squared-norm of the gradient

$$E_{\tau, \tilde{i}_0, \ldots, \tilde{i}_{\tau-1}}[\|\nabla f(\boldsymbol{\theta}_\tau)\|^2],$$

where $\tilde{i}_t$ is the index selected at iteration t

Convergence in Expectation II

- Under some assumptions, we can prove the following convergence rate

$$E[\|\nabla f(\boldsymbol{\theta}_\tau)\|^2] = O(\frac{\log T}{\sqrt{T}})$$

- Note that we consider the simplest situation of selecting one data instance at a time
- It is more complicated to establish the proof for commonly used stochastic gradient methods
- For some of them, the proofs are still lacking

Outline

A Story on the Use of Stochastic Gradient I

- Recently pre-trained language models (e.g., ChatGPT) draw lots of attention
- Behind them an optimization problem is solved by stochastic gradient methods
- BERT (Devlin et al., 2019) is a pioneer of such pre-trained models, cited by tens of thousands papers
- It applies Adam for the optimization
- But the authors forgot to do the bias correction step
- Later people realized this (e.g., Zhang et al., 2021) and pointed out the instability in some situations

A Story on the Use of Stochastic Gradient II

- For example, the process may under-fit the data and require more iterations
- However, BERT did achieve the best test performance on many applications when it was proposed
- This interesting story reflects the role of optimization in machine learning
- Optimization is an important tool for machine learning but we also need useful models and other things to achieve good final test performance

Conclusions

- Stochastic gradient methods are the dominant optimization technique for deep learning
- We have discussed commonly used stochastic gradient methods and explain why they are widely used for deep learning
- However, many issues from theoretical convergence to the practical use remain to be investigated