Batching & Optimizers + Assignments 6, 7

(Neural Networks Implementation and Application Tutorial)

Vilém Zouhar, Noon Pokaratsiri Goldstein

4th, 5th January 2022

Overview

- Assignment 6
- Batching
- Optimization algorithms
- Assignment 7

Assignment 6

- What was the hardest part?

Batching

Gradient descent 🧐

- What is it?
- What do we base the gradient estimates on?

What's the difference? 🤔

- Batch, stochastic, mini-batch gradient descent

What's the batch size for dataset size d? 🤔

- Batch gradient descent
  - d
- Stochastic gradient descent
  - 1
- Mini-batch gradient descent
  - $1 < b < d$

Batching

	batch size	estimate	train time	computation
(Full) Batch gradient descent	d	accurate	high	high
Mini-Batch gradient descent	$1 < b < d$	less accurate	faster	faster
Stochastic gradient descent	1	least accurate	fastest	fasterst

Notes

- All modes are (special cases of) mini-batch
- When people say “BGD” they usually mean “MBGD”

Note on common memory saving technique

- Accumulate gradient for $< b$ samples and update only after reaching mini-batch
- This way GPU/RAM only has $< b$ samples at the time
- Output same as for minibatch
- No extra memory cost, higher computation time

Batching

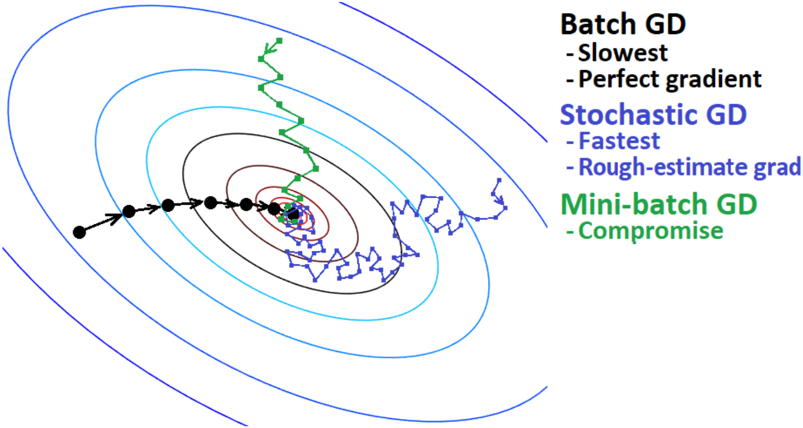

Figure 1: Convergence of SGD, MBGD, BGD [1]

Optimization Algorithms - Momentum

SGD/MBGD/BGD (known as SGD)

- Approximate gradient based on some set
- $\theta \leftarrow \theta - g$

SGD with momentum

- Approximate gradient based on some set
- Update direction in which the gradient is moving (+exponential decay)
- $v \leftarrow \alpha v - \epsilon g$
- $\theta \leftarrow \theta + v$

Remaining issue

- Some dimensions are very high frequency (need small learning rate)
- Some dimensions are uneventful (need large learning rate)

Optimization Algorithms - Adaptive learning rates

AdaGrad

- Keep sum of squared gradients ("speed"): $r \leftarrow r + g \odot g$
- Scale individual dimensions: $\theta \leftarrow \theta - \frac{\epsilon}{\delta + \sqrt{r}} \odot g$

RMSProp

- Same as AdaGrad but decay history
- Keep sum of squared gradients ("speed"): $r \leftarrow \rho r + (1 - \rho) g \odot g$
- Scale individual dimensions: $\theta \leftarrow \theta - \frac{\epsilon}{\delta + \sqrt{r}} \odot g$

Adam

- Combine momentum and adaptive learning rates
- $s \leftarrow \rho_1 s + (1 - \rho_1) g, r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$
- Bias correction (matters only at the beginning): $\hat{s} = s/(1 - \rho_1^t), \hat{r} = r/(1 - \rho_2^t)$
- Update: $\theta \leftarrow \theta - \frac{\epsilon}{\delta + \sqrt{\hat{r}}} \hat{s}$

Optimization Algorithms

Notes

- Look up Nesterov look-ahead momentum update 2
- Adam is the go-to algorithm nowadays
- Be able to write down pseudocode for the optimizers (in more detail than these slides)

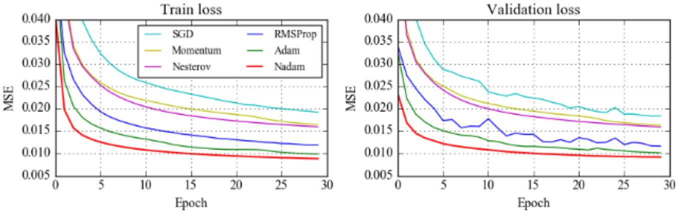

Figure 2: Comparison of optimizers [3]

See nice animated gradient optimizer visualizations.

Assignment 7

Any questions?

Resources

1. dragonnotes.org/DeepLearning/Optimization (very good notes!)
2. mlfromscratch.com/optimizers-explained/#/
3. towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c
4. gradient visualization