Optimization for Deep Learning

Sebastian Ruder
PhD Candidate, INSIGHT Research Centre, NUIG
Research Scientist, AYLIEN
@seb_ruder

Advanced Topics in Computational Intelligence
Dublin Institute of Technology

24.11.17

Agenda

Introduction

- Gradient descent is a way to minimize an objective function $J(\theta)$
  - $\theta \in \mathbb{R}^d$: model parameters
  - η: learning rate
  - $\nabla_\theta J(\theta)$: gradient of the objective function with regard to the parameters
- Updates parameters **in opposite direction** of gradient.
- Update equation: $\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$

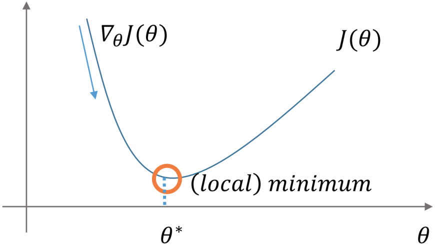


Figure: Optimization with gradient descent

Gradient descent variants

1. Batch gradient descent
2. Stochastic gradient descent
3. Mini-batch gradient descent

Difference: Amount of data used per update

Batch gradient descent vs. SGD fluctuation

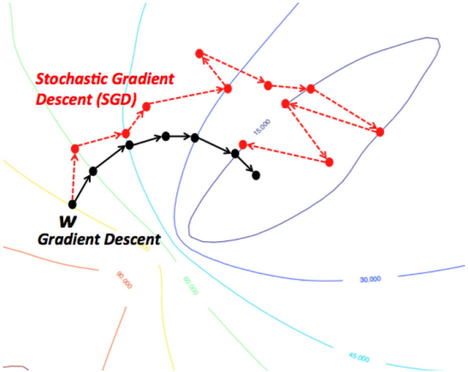


Figure: Batch gradient descent vs. SGD fluctuation (Source: wikidocs.net)

- SGD shows same convergence behaviour as batch gradient descent if learning rate is **slowly decreased (annealed)** over time.

Method	Accuracy	Update Speed	Memory Usage	Online Learning
Batch gradient descent	Good	Slow	High	No
Stochastic gradient descent	Good (with annealing)	High	Low	Yes
Mini-batch gradient descent	Good	Medium	Medium	Yes

Table: Comparison of trade-offs of gradient descent variants

Challenges

- Choosing a **learning rate**.
- Defining an **annealing schedule**.
- Updating features to **different extent**.
- **Avoiding suboptimal minima**.

Gradient descent optimization algorithms

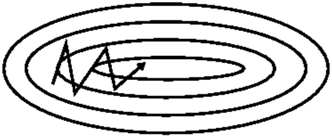
1. Momentum
2. Nesterov accelerated gradient
3. Adagrad
4. Adadelta
5. RMSprop
6. Adam
7. Adam extensions

Momentum

- SGD has trouble navigating **ravines**.
- Momentum [Qian, 1999] helps SGD **accelerate**.
- Adds a fraction γ of the update vector of the past step v_{t-1} to current update vector v_t. Momentum term γ is usually set to 0.9.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{1}$$

(a) SGD without momentum



(b) SGD with momentum

Figure: Source: Genevieve B. Orr

RMSprop

- Developed independently from Adadelta around the same time by Geoff Hinton.
- Also divides learning rate by a **running average of squared gradients**.
- RMSprop update:

$$
\begin{aligned}
E[g^2]_t &= \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t
\end{aligned}
\tag{12}
$$

  - γ: decay parameter; typically set to 0.9
  - η: learning rate; a good default value is 0.001

Adam

- Adaptive Moment Estimation (Adam) [Kingma and Ba, 2015] also stores **running average of past squared gradients** v_t like Adadelta and RMSprop.
- Like Momentum, stores **running average of past gradients** m_t.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{13}$$

  - m_t: first moment (mean) of gradients
  - v_t: second moment (uncentered variance) of gradients
  - β_1, β_2: decay rates

- m_t and v_t are initialized as 0-vectors. For this reason, they are biased towards 0.
- Compute bias-corrected first and second moment estimates:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{14}$$

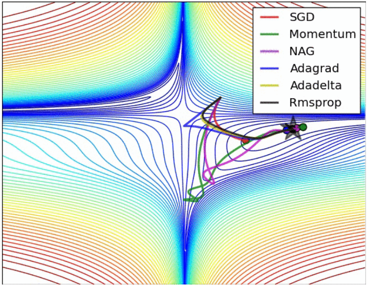
- Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \tag{15}$$

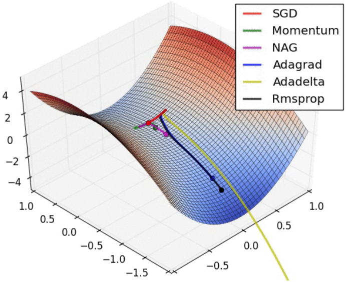
Adam extensions

1. AdaMax [Kingma and Ba, 2015]
    - Adam with ℓ_∞ norm
2. Nadam [Dozat, 2016]
    - Adam with Nesterov accelerated gradient

Visualization of algorithms



(a) SGD optimization on loss surface contours



(b) SGD optimization on saddle point

Figure: Source and full animations: Alec Radford

Which optimizer to choose?

- Adaptive learning rate methods (Adagrad, Adadelta, RMSprop, Adam) are **particularly useful for sparse features**.
- Adagrad, Adadelta, RMSprop, and Adam work well in similar circumstances.
- [Kingma and Ba, 2015] show that bias-correction helps Adam **slightly outperform RMSprop**.

Outlook

1. Tuned SGD vs. Adam
2. SGD with restarts
3. Learning to optimize
4. Understanding generalization in Deep Learning
5. Case studies

Tuned SGD vs. Adam

- Many recent papers use **SGD with learning rate annealing**.
- SGD with tuned learning rate and momentum is **competitive with Adam** [Zhang et al., 2017b].
- Adam **converges faster**, but **underperforms SGD** on some tasks, e.g. Machine Translation [Wu et al., 2016].
- Adam with **2 restarts and SGD-style annealing** converges faster and outperforms SGD [Denkowski and Neubig, 2017].
- **Increasing the batch size** may have the same effect as decaying the learning rate [Smith et al., 2017].