Neural Network Optimizaton 2

CS 519 Deep Learning, Winter 2016

Fuxin Li

Tricks of the trade

- As you might have seen, it’s a bit difficult to make NN work well!
- Many tricks needed
- Some are optimization-related
  - Momentum
  - RMSprop
  - Adagrad
  - Adam, etc.
- Some are regularization-related
  - Dropout
  - Batch normalization
  - Particular regularizing network structures, etc.

Nesterov Momentum vs. Momentum

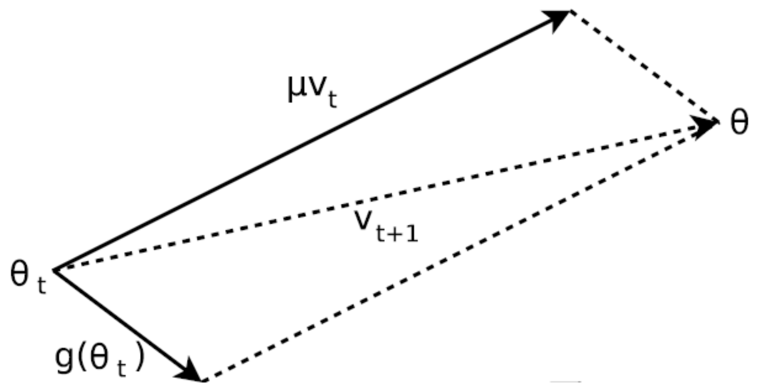
- Perform the "inertia" step first before taking gradient
- Better theoretical guarantees in convex optimization

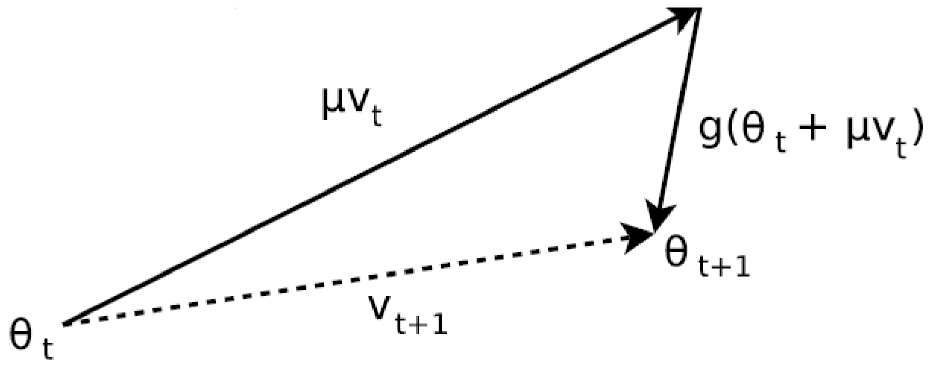
Normal momentum

$$\boldsymbol{D}_0 = 0$$
$$\boldsymbol{D}_{t+1} = \mu \boldsymbol{D}_t - \alpha \nabla f(\boldsymbol{W}_t)$$
$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \boldsymbol{D}_{t+1}$$

Nesterov momentum

$$\boldsymbol{D}_0 = 0$$
$$\boldsymbol{D}_{t+1} = \mu \boldsymbol{D}_t - \alpha \nabla f(\boldsymbol{W}_t + \mu \boldsymbol{D}_t)$$
$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \boldsymbol{D}_{t+1}$$





Sutskever et al. On the importance of momentum and initialization in deep learning. ICML 2013.

Nesterov Momentum vs. Momentum

- N: Nesterov
- M: Momentum
- Number: μ_{max}
- Schedule:

$$\mu_t = \min(1 - 2^{-1-\log_2(\lfloor t/250 \rfloor + 1)}, \mu_{\max})$$

task	$0_{(SGD)}$	0.9N	0.99N	0.995N	0.999N
Curves	0.48	0.16	0.096	0.091	**0.074**
Mnist	2.1	1.0	**0.73**	0.75	0.80
Faces	36.4	14.2	8.5	7.8	**7.7**
		0.9M	0.99M	0.995M	0.999M
Curves		0.15	0.10	0.10	0.10
Mnist		1.0	0.77	0.84	0.90
Faces		15.3	8.7	8.3	9.3

Sutskever et al. On the importance of momentum and initialization in deep learning. ICML 2013.

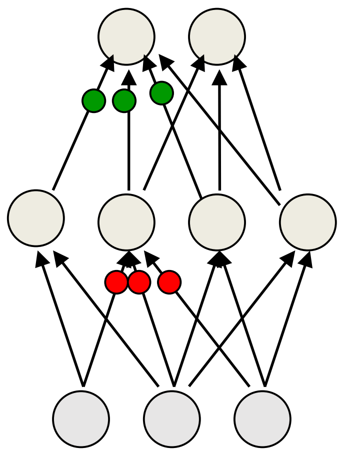
Trick #2: Adaptive learning rates

- Recall, learning rate should go to zero for convergence
- Is fixed learning rate a great idea?
- Simple learning rate decay:
  - Reduce the learning rate after every few epochs
  - Seems to work relatively well if all the weights are quite balanced

The intuition behind separate adaptive learning rates

- Learning rates can be set per weight or layer
  - Gradient **magnitudes** differ across layers
  - fan-in of a unit determines the size of the "overshoot" effects

  - Use a global learning rate times local gain



Gradients can get very small in the early layers of very deep nets.

The fan-in often varies widely between layers.

AdaGrad (Duchi 2011)

- If the gradient is large, stepsize should be small
- Use square-root of accumulated gradient norm to be step size

$$r_{Tk} = \sum_{i=1}^{T} \|G_{ik}\|^2 = r_{T-1,k} + \|G_{ik}\|^2$$

$$\alpha_k = \frac{\alpha_0}{\sqrt{r_{Tk}}}$$

Try:

$$\min_{\mathrm{w}_1,\mathrm{w}_2} (w_1 - 1)^2 + 100(w_2 - 1)^2$$

AdaGrad

Algorithm 8.4 The AdaGrad algorithm

Require: Global learning rate ϵ
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, perhaps 10^{-7}, for numerical stability
Initialize gradient accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$
Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. (Division and square root applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

RMSprop

- AdaGrad does not work well for non-convex problems
- Remembered too much history
- In stochastic non-convex optimization, history might be not very useful

$$r_{Tk} = \rho r_{T-1,k} + (1-\rho)\|G_{Tk}\|^2$$

e.g. $\rho = 0.9$

RMSprop

Algorithm 8.5 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, usually 10^{-6}, used to stabilize division by small numbers.
  Initialize accumulation variables $\boldsymbol{r} = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute parameter update: $\Delta \boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. $\quad (\frac{1}{\sqrt{\delta + \boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$
  end while

Combining RMSProp with Nesterov Momentum (Sutskever 2013)

- Use RMSprop on the gradient part of the momentum
- Somehow it doesn't seem to work well with normal momentum

$$\boldsymbol{D}_0 = 0$$
$$\boldsymbol{D}_{t+1} = \mu \boldsymbol{D}_t - \alpha \nabla f(\boldsymbol{W}_t + \mu \boldsymbol{D}_t)$$
$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \boldsymbol{D}_{t+1}$$

$$\boldsymbol{D}_{\boldsymbol{0}} = 0$$
$$\boldsymbol{D}_{t+1} = \mu \boldsymbol{D}_t - \frac{\alpha}{\sqrt{\boldsymbol{r}_t}} \odot \nabla f(\boldsymbol{W}_t + \mu \boldsymbol{D}_t)$$
$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \boldsymbol{D}_{t+1}$$

Adam

- Running average of momentum estimates and RMS estimates are both biased
- Take RMSprop estimate at stationary state:

$$r_{Tk} = \rho r_{T-1,k} + (1-\rho)\|G_{Tk}\|^2$$

$$= (1-\rho)\sum_{i}^{T} \rho^{T-i}\,\|G_{Tk}\|^2$$

$$\mathbb{E}(r_{Tk}) = \mathbb{E}(\|G_k\|^2)(1-\rho)\sum_{i}^{T} \rho^{T-i}$$

$$= \mathbb{E}(\|G_k\|^2)(1-\rho^T)$$

Adam

- Robust choice of step size
- Two moment estimates:
  - Bias-correcting momentum(T here is not transpose, is T-th power!), e.g. $\rho_1 = 0.99$

$$D_T = \frac{\rho_1}{1-\rho_1^T} D_{T-1} + \frac{1-\rho_1}{1-\rho_1^T} G_T$$

  - Bias-correcting RMS estimate (average gradient norm), e.g. $\rho_2 = 0.999$

$$r_{Tk} = \frac{\rho_2}{1-\rho_2^T} r_{T-1,k} + \frac{1-\rho_2}{1-\rho_2^T} \|G_{Tk}\|^2$$

  - Final update:

$$W_k = W_k - \epsilon \frac{D_{Tk}}{\sqrt{r_{Tk}}}$$

Polyak averaging

- $\widehat{W_T} = \alpha\widehat{W_{T-1}} + (1-\alpha)W_T$
- Sometimes used only at the end of optimization to create a "momentum"-like effect for the final model

When to use these things?

- Adam is usually good if you want to avoid tuning learning rate
- However, sometimes fixed learning rate + some manual decreases work just fine
- RMSprop + Nesterov momentum also works well
- Usually Adagrad is not used in deep learning
- Polyak averaging is a bit redundant with momentum, hence mostly used at the end (testing time)