Non-convex cost function optimization

Adagrad, RMSProp, Adam

Alex-Antoine Fortin

American Family Insurance

March 31, 2017

Optimizer

What role is playing your optimizer?

- Computing the cost of your intermediate model
- Compute $\nabla_{\theta} Cost$
- Tuning the learning rate
- Computing the parameter update to your model
- and more ...

Heuristic

delta-bar-delta (Jacobs, 1988)
Let

$$\nabla_{\boldsymbol{\theta}} Cost = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$$

then, update:

$$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta} + \epsilon \nabla_{\boldsymbol{\theta}} Cost$$

Intuition:
if $sign\left(\nabla Cost; \boldsymbol{\theta}\right)_i = sign\left(\nabla Cost; \boldsymbol{\theta}^*\right)_i$ then ϵ should be bigger
if $sign\left(\nabla Cost; \boldsymbol{\theta}\right)_i <> sign\left(\nabla Cost; \boldsymbol{\theta}^*\right)_i$ then ϵ should be smaller

AdaGrad (Duchi et al., 2011)

Idea

Parameters are scaled down proportionnaly to the square root of their historical value

Net effect

More progress in the gently sloped directions of the parameter space

AdaGrad: Algorithm

Algorithm 1 The AdaGrad algorithm

Require: Global learning rate ϵ

Require: Initial parameter $\boldsymbol{\theta}$

Require: Small constant δ for numerical stability

Initialize gradient accumulation variable $\mathbf{r} = \mathbf{0}$

while stopping criterion not met **do**

Get minibatch: $\left\{\left(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\right), \ldots, \left(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\right)\right\}$

Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$

Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$

Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

end while

RMSProp (Hinton, 2012)

Idea

AdaGrad is really good for convex optimization. In non-convex optimization, the learning trajectory may pass through many structures and eventually arrive at a locally convex region.

RMSProp use an EMA to discard updates from the distant past so that it can converge rapidly once it found a locally convex region.

Thus, RMSProp introduces a parameter ρ that controls the lenght scale of the MA.

RMSProp: Algorithm

Algorithm 2 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ

Require: Initial parameter $\boldsymbol{\theta}$

Require: Small constant δ for numerical stability

Initialize gradient accumulation variable $\mathbf{r} = \mathbf{0}$

while stopping criterion not met **do**

Get minibatch: $\left\{\left(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\right), \ldots, \left(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\right)\right\}$

Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$

Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho)\, \boldsymbol{g} \odot \boldsymbol{g}$

Compute update: $\boldsymbol{\Delta\theta} \leftarrow -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\Delta\theta}$

end while

Adam (Kingma and Ba, 2014)

Adam stands for *Adaptative moments*.

Idea

RMSProp has the capacity to tune ϵ for each θ_i individually as well as to forget distant past. However, the squared gradient accumulator, $\boldsymbol{r}$, initialized to $\mathbf{0}$ introduces a bias toward $\mathbf{0}$ in its calculation.

Adam introduces a bias correction term to both the calculated gradient and its second-moment, $\boldsymbol{r}$. It also adds momentum to the estimation of the gradient.

Adam: Algorithm

Algorithm 3 The Adam algorithm

Require: Global learning rate ϵ, decay rate ρ_1, ρ_2,param $\boldsymbol{\theta}$, const δ

Initialize accumulation variables $\mathbf{s} = \mathbf{0}, \mathbf{r} = \mathbf{0}$ and $t = 0$

while stopping criterion not met **do**

Get minibatch: $\left\{\left(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\right), \ldots, \left(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\right)\right\}$

Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

$t \leftarrow t + 1$

Update biased first moment: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\, \boldsymbol{g}$

Update biased second moment: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\, \boldsymbol{g} \odot \boldsymbol{g}$

Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$

Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$

Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\hat{\boldsymbol{r}}}} \odot \hat{\boldsymbol{s}}$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

end while