Machine Learning for Applied Physics and High Energy Physics

Lecture 5 : Gradient descent and its generalisations – II

In SGD, with and without momentum, we still have to specify a "schedule" for tuning the learning rate η_t as a function of time. We have seen that the learning rate is limited by the steepest direction which can change with t, that is with the current position in the parameter space. Ideally, the algorithm would keep track of the curvature in the parameter space and take large steps in shallow, flat directions, and small steps in steep, narrow directions. Second-order methods accomplish this task by calculating or approximating the Hessian and normalising the learning rate by the curvature. However, this is very computationally expensive for models with a huge number of parameters (as is the case in ML). We would like to be able to adaptively change the step size to match the landscape without paying the price of calculating or approximating the Hessian.

5.1. Methods that use the second moment of the gradient

The aforementioned task can be accomplished by a set of methods that track not only the gradient, but also its second moment. These methods include

ADAGRAD	RMSPROP
ADADELTA	ADAM

RMSPROP (Root Mean Square Propagation)

In addition to keeping a running average of the first moment of the gradient, one also takes into account the second moment of the gradient, which we denote as

$$\vec{s}_t = \mathbb{E}\left[\vec{g}_t^{\,2}\right]$$

The update rule for RMSPROP is given by

$$\vec{g}_t = \nabla_\theta E(\vec{\theta})$$

$$\vec{s}_t = \beta\, \vec{s}_{t-1} + (1-\beta)\, \vec{g}_t^{\,2}$$

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t \frac{\vec{g}_t}{\sqrt{\vec{s}_t + \varepsilon}}$$

(Multiplication, division and square root of vectors are understood to be element-wise operations)

where: β controls the averaging time of the second moment. Typical values are $\beta = 0,9$; η_t is a learning rate typically chosen to be 10^{-3}, and $\varepsilon \sim 10^{-8}$ is a small regulator to prevent the ratio diverge. The learning rate is reduced in directions where the gradient is consistently large. This greatly speeds up the convergence by allowing us to use a larger learning rate for flat directions.

ADAM (Adaptive Minimiser)

One keeps a running average of both the first and second moment of the gradient and use this information to adaptively change the learning rate for different parameters. In addition to keeping a running average of the first and second moments of the gradient, which we denote as

- $\vec{m}_t = \mathbb{E}[\vec{g}_t]$
- $\vec{s}_t = \mathbb{E}[\vec{g}_t^{\,2}]$

ADAM performs an additional bias correction to account for the fact that we are estimating the first two moments of the gradient using a running average (denoted with a hat below). The update rule for ADAM is given by

$$\vec{g}_t = \nabla_\theta E(\vec{\theta})$$

$$\vec{m}_t = \beta_1 \vec{m}_{t-1} + (1-\beta_1)\vec{g}_t$$

$$\vec{s}_t = \beta_2 \vec{s}_{t-1} + (1-\beta_2)\vec{g}_t^{\,2}$$

$$\hat{\vec{m}}_t = \frac{\vec{m}_t}{1-(\beta_1)^t}$$

$$\hat{\vec{s}}_t = \frac{\vec{s}_t}{1-(\beta_2)^t}$$

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{s}_t + \varepsilon}}$$

where β_1 and β_2 set the "memory lifetime" of the first and second moment and are typically taken to be 0,9 and 0,99 respectively. Like in RMSPROP, the effective step size of a parameter depends on the magnitude of its gradient squared. To understand this better, let us rewrite the next move in terms of the variance $\vec{\sigma}_t^{\,2} = \hat{s}_t - (\hat{m}_t)^2$. Consider a single parameter ϑ_t. The update rule for this parameter is given by 4

$$\Delta \vartheta_{t+1} = - \eta_t \frac{\hat{m}_t}{\sqrt{\sigma_t^2 + \hat{m}_t^2} + \varepsilon}$$

We now examine different limiting cases of this expression

- Gradient estimates are all very consistent, therefore the variance is small. In this limit $(\hat{m}_t^2 \gg \sigma_t^2)$

$$\Delta \vartheta_{t+1} = - \eta_t \quad (\text{assuming } \hat{m}_t^2 \gg \varepsilon)$$

This is equivalent to cutting off large persistent gradients at 1 and limiting the maximum step size in steep directions.

- Gradient estimates are all fluctuating very much between gradient descent steps. In this limit $(\hat{m}_t^2 \ll \sigma_t^2)$

$$\Delta \vartheta_{t+1} = - \eta_t \frac{\hat{m}_t}{\sigma_t}$$

The learning rate is adapted so that it is proportional to the signal-to-noise ratio.

In other words, the standard deviation serves as a natural adaptive scale to decide whether a gradient is large or small. Thus, ADAM has the beneficial effects of

1) Adapting our step size so that we cut off large gradient directions (and hence prevent oscillations and divergences)
2) Measuring gradients in terms of a natural length scale, the standard deviation σ_t.

5.2 Comparison of various methods

To better understand the five methods that we have illustrated (GD, GD with momentum, NAG, ADAM, RMSprop) let us visualize their performance. To this purpose, we use a parameter space modelled by Beale's function

$$f(x,y) = (1{,}5 - x + xy)^2 + (2{,}25 - x + xy^2)^2 + (2{,}625 - x + xy^3)^2$$

This function has a global minimum at $(x,y) = (3;0{,}5)$.
See the Jupyter notebook to visualise the results.

5.3 Gradient descent in practice: practical tips

We conclude this lecture by compiling some practical tips for getting the best performance from gradient-descent based algorithms.

1. Randomise the data when making mini-batches. Otherwise the gradient descent method can fit spurious correlations resulting from the order in which the data is presented

2. Transform your inputs. Learning becomes difficult when you have an admixture of flat and steep directions. Tip: standardise the data by subtracting the mean and normalising the variance of input variables.

3. Monitor the out-of sample performance. It is important to always split the data into training and validation sets. Early stopping consists in stopping minimisation when the value of the loss function deteriorates on the validation set.

4. Adaptive optimisation methods do not always have good

generalisation. Simple procedures, such as properly tuned SGD, may work equally well as - if not better than - ADAM or ADAGRAD.