# Neural Networks: weight change momentum and weight decay

Momentum $\alpha$ is used to diminish the fluctuations in weight changes over consecutive iterations:

$$\Delta\omega_i(t+1) = \omega_i - \eta\frac{\partial E}{\partial w_i} + \alpha\Delta\omega_i(t),$$

where $E(\mathbf{w})$ is the error function, $\mathbf{w}$ - the vector of weights, $\eta$ - learning rate.

Weight decay $\lambda$ penalizes the weight changes:

$$\Delta\omega_i(t+1) = \omega_i - \eta\frac{\partial E}{\partial w_i} - \lambda\eta\omega_i$$

**The question is if it makes sense to combine both tricks during the back-propagation and what effect it would have?**

$$\Delta\omega_i(t+1) = \omega_i - \eta\frac{\partial E}{\partial w_i} + \alpha\Delta\omega_i(t) - \lambda\eta\omega_i$$

neural-networks regularization gradient-descent

asked Sep 16 '13 at 1:56
Oleg Shirokikh
224 2 12

## 1 Answer

Yes, it's very common to use both tricks. They solve different problems and can work well together.

One way to think about it is that weight decay changes *the function that's being optimized*, while momentum changes *the path you take to the optimum*.

Weight decay, by shrinking your coefficients toward zero, ensures that you find a local optimum with small-magnitude parameters. This is usually crucial for avoiding overfitting (although other kinds of constraints on the weights can work too). As a side benefit, it can also make the model easier to optimize, by making the objective function more convex.

Once you have an objective function, you have to decide how to move around on it. Steepest descent on the gradient is the simplest approach, but you're right that fluctuations can be a big problem. Adding momentum helps solve that problem. If you're working with batch updates (which is usually a bad idea with neural networks) Newton-type steps are another option. The new "hot" approaches are based on Nesterov's accelerated gradient and so-called "Hessian-Free" optimization.

But *regardless* of which of these update rules you use (momentum, Newton, etc.), you're still working with the same objective function, which is determined by your error function (e.g. squared error) and other constraints (e.g. weight decay). The main question when deciding which of these to use is how quickly you'll get to a good set of weights.

edited Sep 15 '14 at 19:20 answered Sep 16 '13 at 15:16
David J. Harris
4,476 12 29

Thanks for the complete clear answer, David – Oleg Shirokikh Sep 16 '13 at 16:04