2. Training practicalities

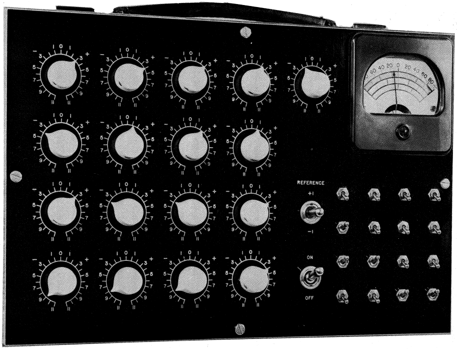
2.2 Optimizers

Manel Martínez-Ramón
Meenu Ajith
Aswathy Rajendra Kurup

Gradient descent

The basic optimization technique is the gradient descent.

- Stochastic gradient descent (SDG): the gradient is estimated with one sample at a time.
- Gradient descent (GD): the gradient is estimated with a batch of data.



- The Adaline (figure) was the first machine to be trained with SDG. The device consisted of a neuron with a sign activation. The training used the LMS algorithm.
- Modified 1st order algorithms have been proposed that improve the SG.

Momentum optimization

- Assume (without loss of generality) that a set of weights are arranged as a vector $\mathbf{w}$ simulating the position of a particle in a space, with initial velocity $\mathbf{v}_0 = 0$.
- With a given acceleration $\mathbf{a}(t)$ applied to the particle, the velocity will be

$$d\mathbf{v} = \mathbf{a}(t)dt$$

- Now, if the medium has viscosity η, there will be a deceleration proportional to the velocity, as

$$d\mathbf{v}(t) = -\eta\mathbf{v}(t) + \mathbf{a}(t)dt$$

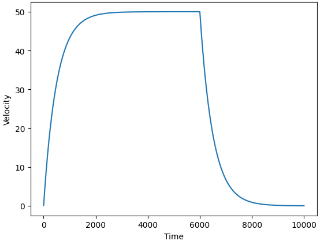
Momentum optimization

By discretizing the differential equations we obtain

$$\Delta \mathbf{v}_{k+1} = -\eta \mathbf{v}_k + \mathbf{a}_k \Delta t$$

$$\mathbf{v}_{k+1} = \gamma \mathbf{v}_k + \mathbf{a}_k$$

which is obtained by assuming $\Delta t = 1$ and $\gamma = 1 - \eta$



Example: A particle is applied a force $F = mg$ where $g = 0.1$ in a viscous fluid. The particle has a viscosity coefficient $\gamma = 0.002$. At $t = 6000$, the force is suppressed.

Momentum optimization

This can be applied to optimization, by assuming that the particle is on the cost function surface and that the acceleration is proportional to the gradient (the slope of the surface). The velocity of the particle is

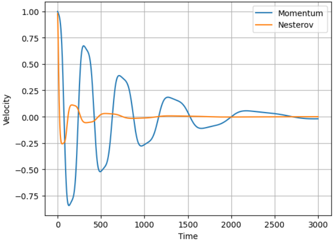
$$\mathbf{v}_k = \gamma \mathbf{v}_{k-1} - \mu \nabla_{\mathbf{w}} J(\mathbf{w}_k)$$

and its position is the integral of the velocity

$$\mathbf{w}_{k+1} = \mathbf{w}_0 + \sum_{k'=0}^{k} \mathbf{v}_{k'} = \mathbf{w}_k + \mathbf{v}_k$$

Nesterov Accelerated gradient

When the gradient is zero, the Momentum optimization will still have a velocity. The approach introduced by Yuri Nesterov (1983) computes the gradient one step ahead to modify the velocity.



Velocity of the particle around a minimum of the cost function using the Momentum and the Nesterov optimizers.

Assuming a present position $\mathbf{w}_k$ and velocity $\mathbf{v}_k$, the next position is computed as

$$\tilde{\mathbf{w}}_{k+1} = \mathbf{w}_k + \mathbf{v}_k$$

from which the velocity is modified

$$\mathbf{v}_k = \gamma \mathbf{v}_{k-1} - \mu \nabla_{\mathbf{w}} J(\tilde{\mathbf{w}}_k)$$

and then the position is updated

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_k$$

Root Mean Square Propagation (RMSProp)

The RMSProp adapts the learning rate of each of the weights by an average of the squared norm of the corresponding component of the weight through a decaying window:

$$\mathbf{g}_k = \beta \mathbf{g}_{k-1} + (1-\beta)\, \nabla_{\mathbf{w}} J(\mathbf{w}_k) \odot \nabla_{\mathbf{w}} J(\mathbf{w}_k)$$

Each weight is updated as

$$w_{i,k+1} = w_{i,k} - \frac{\mu}{\sqrt{g_{i,k}} + \varepsilon} \left[\nabla_{\mathbf{w}} J(\mathbf{w}_k)\right]_i$$

Adaptive Momentum Estimation (Adam)

It is a combination of the Momentum and the RMSProp optimizers. First, a velocity is computed as

$$\mathbf{v}_k = \beta_1 \mathbf{v}_{k-1} + (1-\beta_1)\nabla_{\mathbf{w}} J(\mathbf{w}_k)$$

and then an average of the square norm of each component of the gradient

$$\mathbf{g}_k = \beta_2 \mathbf{g}_{k-1} + (1-\beta_2)\,\nabla_{\mathbf{w}} J(\mathbf{w}_k) \odot \nabla_{\mathbf{w}} J(\mathbf{w}_k)$$

Adaptive Momentum Estimation (Adam)

Then, these magnitudes are unbiased

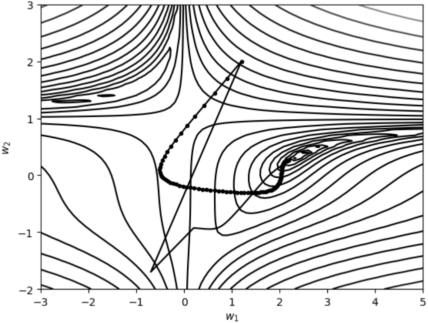
$$\tilde{\mathbf{v}}_k = \frac{\mathbf{v}_k}{1-\beta_1^k}$$
$$\tilde{\mathbf{g}}_k = \frac{\mathbf{g}_k}{1-\beta_2^k}$$

This is justified by seeing that the facts β_1 and β_2 produce a bias in the magnitudes.

Adaptive Momentum Estimation (Adam)

Finally, each element of the weight vector $\mathbf{w}_k$ is updated as

$$w_{i,k+1} = w_{i,k} - \mu \frac{\hat{v}_{i,k}}{\sqrt{\hat{g}_{i,k}} + \varepsilon}$$



The authors of the algorithm suggest in (Kingma and J. Ba 2014) to set the parameters at values $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. The robustness of this choice is shown in a variety of experiments. However, in some circumstances, these values may need to be cross-validated.