SGD, Momentum, Nesterov's Accelerated Gradient (NAG)

Algorithm 1 Gradient Descent

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \mathbf{g}_t$$

Algorithm 2 Classical Momentum

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + \mathbf{g}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \mathbf{m}_t$$

Algorithm 3 Nesterov's accelerated gradient

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1} - \eta \mu \mathbf{m}_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + \mathbf{g}_t$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \mathbf{m}_t$$

Adagrad

Algorithm 4 AdaGrad

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{n}_t \leftarrow \mathbf{n}_{t-1} + \mathbf{g}_t^2$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\mathbf{g}_t}{\sqrt{\mathbf{n}_t + \varepsilon}}$$

- Adagrad divides η of every step by the L_2 norm of all previous gradients
- The monotonic learning rate usually proves too aggressive and stops learning too early
- Assume the gradient dx and parameter vector x

```
cache += dx**2
x += -learning_rate * dx / (np.sqrt(cache) + eps)
```

RMSprop

Algorithm 5 RMSProp

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\mathbf{g}_t}{\sqrt{\mathbf{n}_t + \varepsilon}}$$

- The RMSProp update adjusts the Adagrad method in a very simple way in an attempt to reduce its aggressive, monotonically decreasing learning rate
- Assume the gradient dx and parameter vector x

```
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

Adam

Algorithm 6 Adam

$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1})$$
$$\mathbf{m}_t \leftarrow \mu \mathbf{m}_{t-1} + (1-\mu)\mathbf{g}_t$$
$$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\mu^t}$$
$$\mathbf{n}_t \leftarrow \nu \mathbf{n}_{t-1} + (1-\nu)\mathbf{g}_t^2$$
$$\hat{\mathbf{n}}_t \leftarrow \frac{\mathbf{n}_t}{1-\nu^t}$$
$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{n}}_t}+\varepsilon}$$

- Adam is a recently proposed update that looks a bit like RMSProp with momentum
- Assume the gradient dx and parameter vector x

```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

Adadelta

- Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate
  - Adagrad + Second order (Hessian Approximation → similar to momentum)

$$\Delta x = \frac{\frac{\partial f}{\partial x}}{\frac{\partial^2 f}{\partial x^2}} \Rightarrow \frac{1}{\frac{\partial^2 f}{\partial x^2}} = \frac{\Delta x}{\frac{\partial f}{\partial x}}$$

$$\Delta x_t = -\frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t$$

$$\text{RMS}[g]_t = \sqrt{E[g^2]_t + \epsilon}$$

Algorithm 1 Computing ADADELTA update at time t

Require: Decay rate ρ, Constant ϵ

Require: Initial parameter x_1

1: Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta x^2]_0 = 0$

2: **for** $t = 1 : T$ **do** %% Loop over # of updates

3: Compute Gradient: g_t

4: Accumulate Gradient: $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$

5: Compute Update: $\Delta x_t = -\frac{\text{RMS}[\Delta x]_{t-1}}{\text{RMS}[g]_t} g_t$

6: Accumulate Updates: $E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho) \Delta x_t^2$

7: Apply Update: $x_{t+1} = x_t + \Delta x_t$

8: **end for**