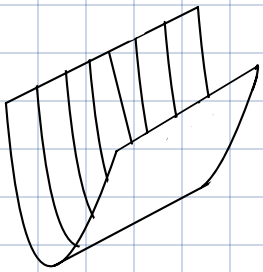
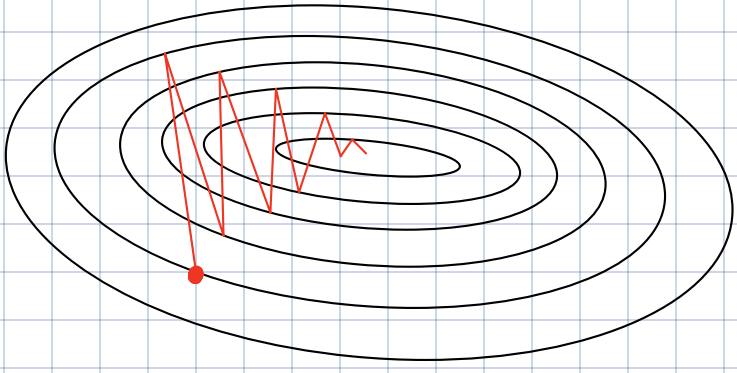
Fancier Optimizations:



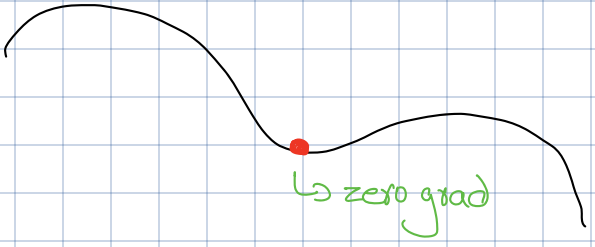
- Sensitive in vertical direction & not in horizontal

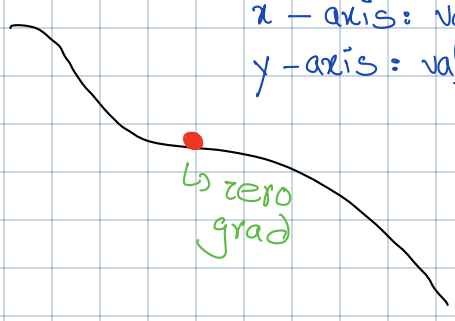
SGD:



↳ Direction of gradient doesn't align with direction towards minima

Local minima / Saddle points





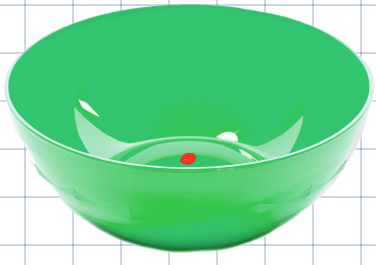
x – axis: value of a param
y – axis: value of loss

↳ SGD will get stuck

Saddle point:

A point where loss increases in one direction & decreases in another

Local minima:



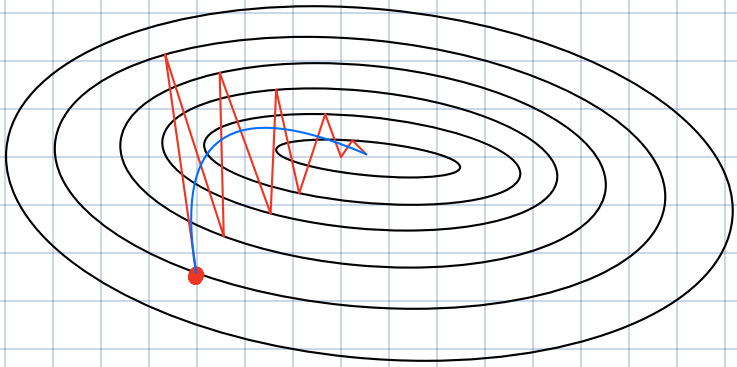
Anywhere else the loss increases

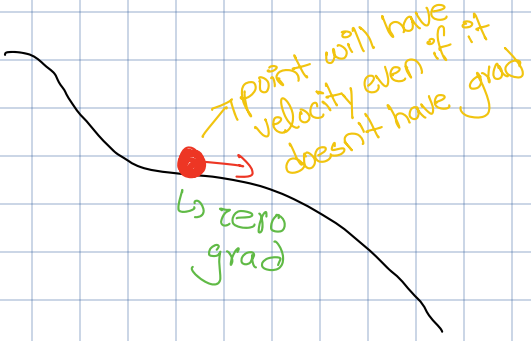
SGD + momentum:

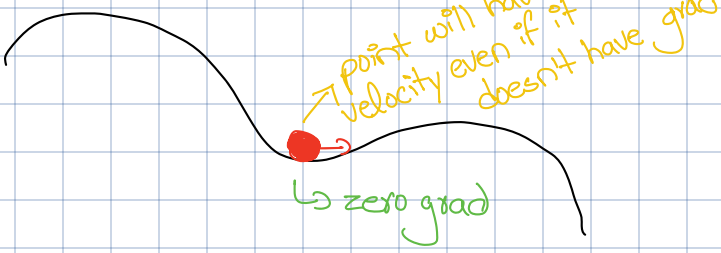
friction ~ 0.9 or 0.99

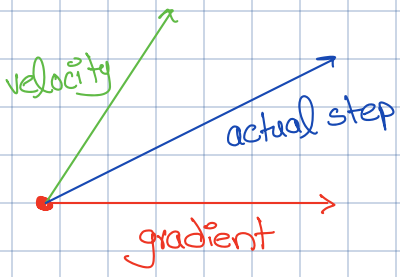
$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_t = x_t - \alpha v_{t+1}$$

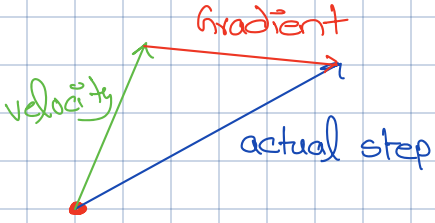








Nesterov Momentum:



↳ Step in the point where velocity will take you
↳ Calculate the grad at that point
↳ Come back to your original point & mix together those two

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t + \rho v_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

OR

Let $\bar{x}_t = x_t + \rho v_t$

$$v_{t+1} = \rho v_t - \alpha \nabla f(\bar{x}_t)$$

$$\bar{x}_{t+1} = \bar{x}_t - \rho v_t + (1+\rho) v_{t+1}$$

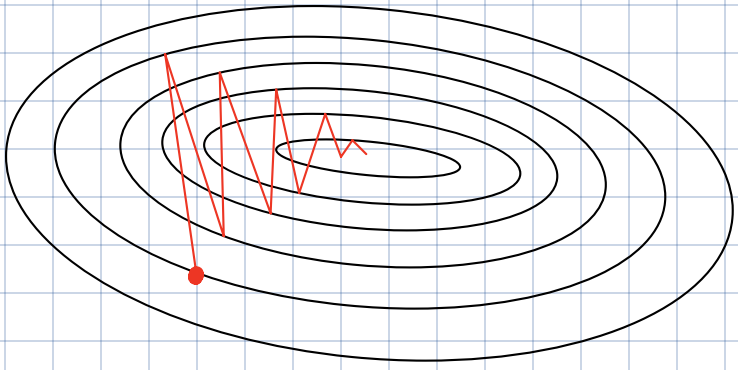
$$= \bar{x}_t + v_{t+1} + \rho (v_{t+1} - v_t)$$

Adagrad:

$$\text{grad\_sq} = \text{grad\_sq} + (\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\sqrt{grad\_sq} + ep}$$

↳ 1e-7



↳ In vertical direction, grads are high so we divide by the large gradient & then take small update step

↳ In horizontal direction, grads are small so we divide by the small gradient & then take a large update step

↳ One problem is that over time, grad_sq becomes so big that our update step becomes too small.

RMSprop:

0.9 or 0.99

$$grad\_sq = grad\_sq * decay\_rate + (1 - decay\_rate) * \nabla f(x_t)^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\sqrt{grad\_sq} + ep}$$

↳ 1e-7

Adam:

↳ combines SHO + momentum & RMSprop

$$v_{t+1} = beta1 * v_t + \nabla f(x_t)$$

$$grad\_sq = grad\_sq * beta2 + (1 - beta2) * \nabla f(x_t)^2$$

$$x_{t+1} = x_t - \alpha \frac{v_{t+1}}{\sqrt{grad\_sq} + ep}$$

↳ At $t=0$, v_t & grad-sq $=0$. So, grad-sq becomes ≈ 0 & we end up taking large steps at the beginning

So, we end up with:

$$v_{t+1} = beta1 * v_t + \nabla f(x_t)$$

$$grad\text{-}sq = grad\text{-}sq * beta2 + (1 - beta2) * \nabla f(x_t)^2$$

$$v_{t+1}\text{-}correction = \frac{v_{t+1}}{(1 - beta1^t)}$$

$$grad\text{-}sq\text{-}correction = \frac{grad\text{-}sq}{(1 - beta2^t)}$$

$$x_{t+1} = x_t - \alpha \frac{v_{t+1}\text{-}correction}{\sqrt{grad\text{-}sq\text{-}correction} + ep}$$

Learning rate:

↳ Decay α over time:

- Exponential decay:

$$\alpha = \alpha_0 e^{-kt}$$

- $1/t$ decay:

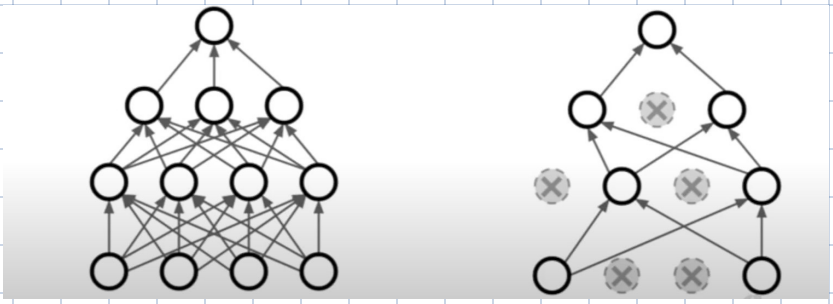
$$\alpha = \frac{\alpha_0}{(1 + kt)}$$

- Step decay:

↳ Decay α by $\frac{1}{2}$ every few epochs

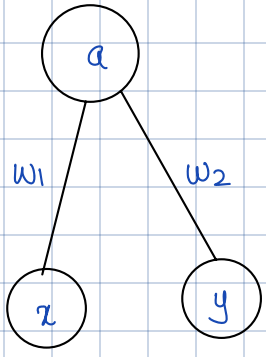
Dropout:

↳ During forward pass, set randomly some neurons to 0



No dropout dropout

At test time:



$$E[a] = w_1 x + w_2 y$$

$$E[a] = \frac{1}{4}(w_1 x + w_2 y) + \frac{1}{4}(w_1 x + 0y) + \frac{1}{4}(0x + 0y) + \frac{1}{4}(0x + w_2 y)$$

$$= \frac{1}{2}(w_1 x + w_2 y)$$

$$z = (wx + b) * p$$

DropConnect:

↳ Randomly zero out values of w

fractional Pooling:

↳ Randomize pooling Regions

Stochastic depth:

↳ Randomly drop layers

Assignment helper: