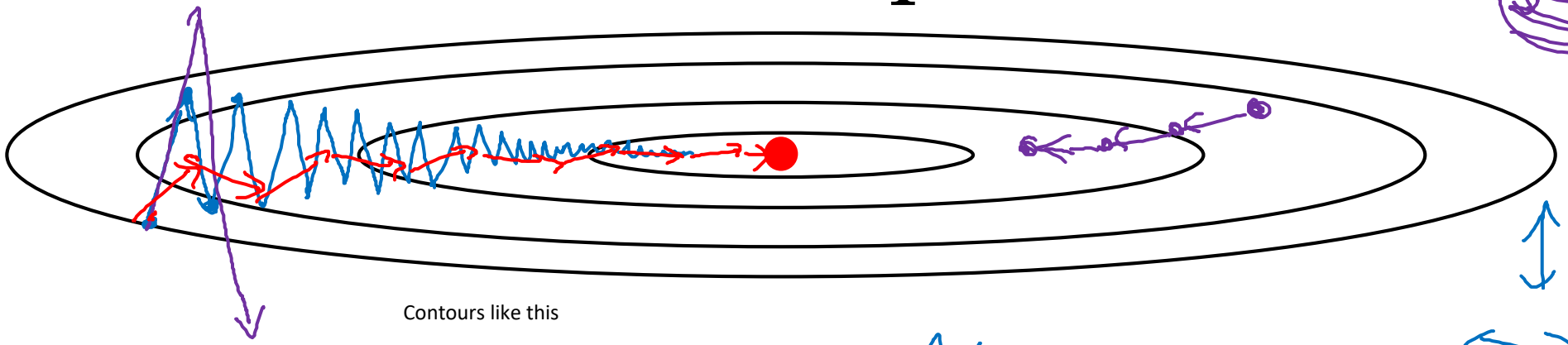
deeplearning.ai

Optimization Algorithms

Gradient descent with momentum

In one sentence, the basic idea is to compute an exponentially weighted average of your gradients, and then use that gradient to update your weights instead.

Gradient descent example



Contours like this

slower learning

faster learning.

And so the need to prevent the oscillations from getting too big forces you to use a learning rate that's not itself too large.

Momentum:

On iteration t:

Compute dW, db on current mini-batch.

$$V_{dW} = \beta V_{dW} + (1-\beta) dW$$

$$V_{db} = \beta V_{db} + (1-\beta) db$$

"$V_\theta = \beta V_{\theta-1} + (1-\beta)\theta_t$"

friction ↑ velocity ↑ acceleration

$$W := W - \alpha V_{dW}, \quad b := b - \alpha V_{db}$$

Andrew Ng

Implementation details

$V_{dW} = 0$, $V_{db} = 0$

On iteration t:

Compute dW, db on the current mini-batch

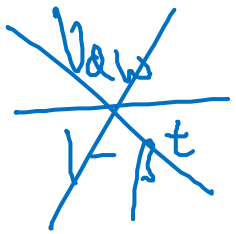
$$v_{dW} = \beta v_{dW} + (1-\beta) dW$$

$$v_{db} = \beta v_{db} + (1-\beta) db$$

$V_{dW} = \beta V_{dW} + dW$

it just affects what's the best value of the learning rate alpha.

$$W = W - \alpha v_{dW}, \quad b = b - \alpha v_{db}$$



Hyperparameters: α, β

$\beta = 0.9$

average over last ≈ 10 gradients