* Basics­­­­­­­­­
* Mini-batch SGD converge faster than SGD.­­­­­
* ***GRADIENT DESCENT:***­
  + Momentum: v[layer] = gamma \* v[layer] + alpha \* grad[layer]
* Model[layer] += v[layer]
  + Nesterov Momentum Instead of calculating gradient of the current position, it calculates the gradient at the approximated new position.

model\_ahead **=** {k: v **+** gamma **\*** velocity[k] **for** k, v **in** model**.**items()}

grad **=** get\_minibatch\_grad(model\_ahead, X\_mini, y\_mini)

**for** layer **in** grad:

velocity[layer] **=** gamma **\*** velocity[layer] **+** alpha **\*** grad[layer]

model[layer] **+=** velocity[layer]

* + RMSprop: cache[layer] = (1 – B)cache[layer] + B \* grad[layer]^2
    - Model[layer] = model[layer] – alpha \* grad[layer] / sqrt(cache[layer] + e).

Adam: Combine momentum and RMSProp

M[k] **=** beta1 **\*** M[k] **+** (1. **-** beta1) **\*** grad[k] (This is velocity).

R[k] **=** beta2 **\*** R[k] **+** (1. **-** beta2) **\*** grad[k]**\*\***2 ( This is cache).

m\_k\_hat **=** M[k] **/** (1. **-** beta1**\*\***(t))

r\_k\_hat **=** R[k] **/** (1. **-** beta2**\*\***(t))

model[k] **+=** alpha **\*** m\_k\_hat **/** (np**.**sqrt(r\_k\_hat) **+** eps)

NOTE : Adam also has a bias correction mechanism, it’s calculated in m\_k\_hat and r\_k\_hat. It’s useful to make the convergence faster, at several first iterations.

Large learning rate => Adaptive learning wins, else momentum wins.

**INITIALIZATION**

Xavier Initialization

**DROPOUT**

Prevent overfitting

Do not use in validation and testing.

**BATCH NORMALIZATION**

Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). To explain covariance shift, let’s have a deep network on cat detection. We train our data on only black cats’ images. So, if we now try to apply this network to data with colored cats, it is obvious; we’re not going to do well. The training set and the prediction set are both cats’ images but they differ a little bit. In other words, if an algorithm learned some X to Y mapping, and if the distribution of X changes, then we might need to retrain the learning algorithm by trying to align the distribution of X with the distribution of Y. ( Deeplearning.ai: Why Does Batch Norm Work? ([C2W3L06](https://www.youtube.com/watch?v=nUUqwaxLnWs" \t "https://towardsdatascience.com/_blank)))

Also, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.

Can use higher learning rate to train + also act as a kind of regularization