Objectives

- Optimizers
- Batch Gradient Descent
- Momentum
- RMSProp

---

1 Optimizers

1. Batch Gradient Descent
   In the past couple of classes we have been focusing primarily on Batch Gradient Descent and the math behind it.
   Issues
   (a). Computationally expensive
   Batch gradient descent computes the gradient at every single data point, which ensures points don't get passed over, but also requires a lot of needless computations
   (b). Notational issues
   $$loss = l(w, x)$$
   $$L(w, x) = \frac{1}{n}\sum l(w, x)$$
   $$\nabla_w L(w, x) = \nabla(\frac{1}{n}\sum l(w, x)) = \frac{1}{n}\sum \nabla l(w, x)$$
   The derivative of a sum should be equal to the sum of the derivatives
   (c). Slow convergence
   The steps get progressively smaller
   (d). Gets stuck on local minimums

   Related Types of Gradient Descent

   (a). Stochastic Gradient Descent (SGD)
   This type of gradient descent selects a random x every time
   $$l(w, x_r)$$

(b). Minibatch Gradient Descent
A combination of SGD and Batch Gradient Descent. It uses a sample of x values and selects a random value from that sample for computation.
The general rule for the batch size is 32 or less.

$$batchsize \leqslant 32$$

2. Momentum
A type of gradient descent that takes into consideration previous gradients in order to prevent getting stuck at local minimums.

$$w_{t+1} = w_t - \alpha \nabla_w L(w_t) - previous\ gradients$$

Features

(a). Faster convergence than Batch Gradient Descent

(b). Doesn't get stuck at local minimums

(c). The amount of past gradients used are controlled with exponentially decaying weighted averages

$$v_{t+1} = \beta v_t + (1 - \beta) \nabla_w L(w_t)$$

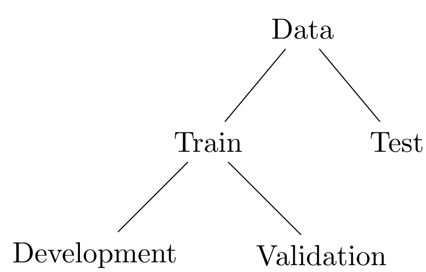
$$w_{t+1} = w_t - \alpha v_{t+1}$$

Hyperparameters

$$\alpha = learning\ rate$$

$$\beta = weights$$

(d). Hyperparameters are tuned with cross validation



(1)

The best practice has $\beta = 0.9$, but it just has to be less than 1

(e). Update function

$$w_{t+1} = \sum_{i=0}^{t} \beta^i (1 - \beta) \nabla L_{t-i}$$

Issues

(a). Can overshoot and miss the global minimum

(b). Still can get stuck at some local minimums

Unrolling the recurrence relation

$$v_{t+1} = \beta v_t + (1-\beta)\nabla L_t$$

$$v_{t+1} = \beta(\beta v_{t-1} + (1-\beta)\nabla L_{t-1}) + (1-\beta)\nabla L_t$$
$$v_{t+1} = \beta^2 v_{t-1} + \beta(1-\beta)\nabla L_{t-1} + (1-\beta)\nabla L_t$$
$$v_{t+1} = \beta^2(\beta v_{t-2} + (1-\beta)\nabla L_{t-2}) + \beta(1-\beta)\nabla L_{t-1} + (1-\beta)\nabla L_t$$
$$v_{t+1} = \beta^3 v_{t-2} + \beta^2(1-\beta)\nabla L_{t-2} + \beta(1-\beta)\nabla L_{t-1} + (1-\beta)\nabla L_t$$

3. RMSProp
   A type of gradient descent that is adaptable depending on the size of the gradient

$$\nabla L(w) = \begin{bmatrix} \delta L/\delta w_1 \\ \delta L \delta w_2 \\ \vdots \\ \delta L/\delta w_n \end{bmatrix}$$

Features

(a). Weighs gradients differently in different directions
   i. Large gradient → Small step
   ii. Small gradient → Large step

(b). Learning rates are different for each component

(c). Element wise operations

$$\rightarrow crossproduct \rightarrow vector$$

$$uxv \rightarrow dotproduct \rightarrow scalar$$

$$\rightarrow Hadamard\ product \rightarrow vector$$

4. AdaGrad

5. NAG (Nestrov Accelerated Gradient)

6. Adam
   Created in 2015, Adam is generally considered to be the best type of gradient descent. It takes the best features of both Momentum and RMSProp and combines them into a single type of gradient descent.