Last time:

- Statistical Learning
- Core Optimization Problem:

$$\theta^* = \arg\min_{\theta \in \Theta} L(\theta, D)$$

- Optimization to approximate θ^*:

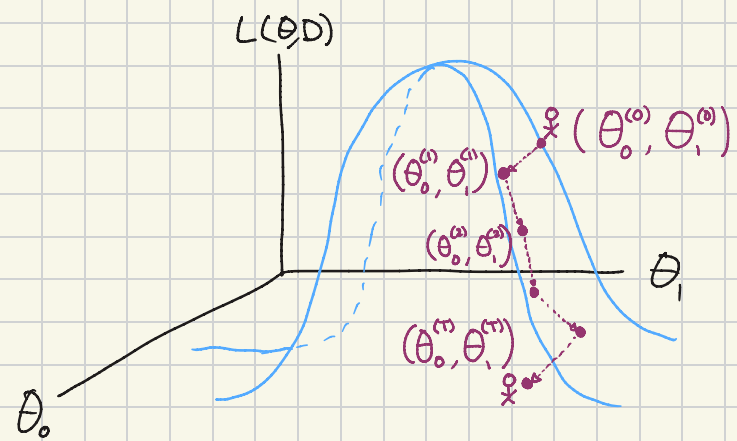
$$\hat{\theta} = \text{opt}_{\theta \in \Theta} L(\theta, D)$$

  - Gradient Descent

Today:

- Gradient Descent (GD) (cont.)
- Potential Optimization Issues
- Improvements to standard GD

Gradient Descent



Update Rule: $\theta^{(t+1)} = \theta^{(t)} - \alpha_t [\nabla_\theta L(\theta, D)]\Big|_{\theta = \theta^{(t)}}$

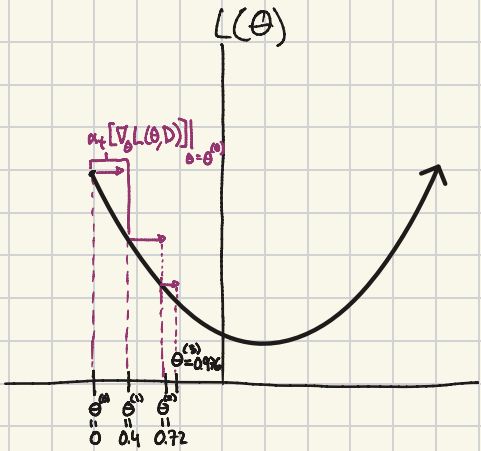
Made with Goodnotes

A small example:

$L(\theta) = (\theta - 2)^2 + 1$

$\nabla_\theta L(\theta) = 2(\theta - 2) = 2\theta - 4$

Consider $\alpha = 0.1$, $\theta^{(0)} = 0$



Step 1: $[\nabla_\theta L(\theta)]\big|_{\theta=\theta^{(0)}} = 2(0) - 4 = -4$

Update: $\theta^{(1)} = 0 - 0.1(-4) = 0.4$

Step 2: $[\nabla_\theta L(\theta)]\big|_{\theta=\theta^{(1)}} = 2(0.4) - 4 = -3.2$

Update: $\theta^{(2)} = 0.4 - 0.1(-3.2) = 0.72$

Step 3: $[\nabla_\theta L(\theta)]\big|_{\theta=\theta^{(2)}} = 2(0.72) - 4 = -2.56$

Update: $\theta^{(3)} = 0.72 - 0.1(-2.56) = 0.976$

Potential Issues

(1) Get stuck at local critical points (mins, maxs, saddle points)
(2) No guarantee of convergence to any fixed point
(3) Slow convergence (notice above that $\alpha_t [\nabla_\theta L(\theta, D)]\big|_{\theta^{(t)}} \to 0$ as $t \to \infty$)
(4) $\nabla_\theta L(\theta, D)\big|_{\theta=\theta^{(t)}}$ may be expensive to compute

---

Issue (2): Consider $\alpha = 1.0$ in our small example above

Step 1: $[\nabla_\theta L(\theta)]\big|_{\theta=\theta^{(0)}} = 2(0) - 4 = -4$

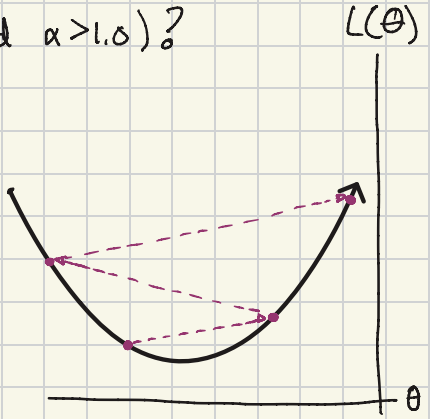
Update: $\theta^{(1)} = 0 - 1.0(-4) = 4$

Step 2: $[\nabla_\theta L(\theta)]\big|_{\theta=\theta^{(1)}} = 2(4) - 4 = 4$

Update: $\theta^{(2)} = 4 - 1.0(4) = 0 = \theta^{(0)}$  stuck in a loop

Made with Goodnotes

What about the case $\alpha = 2.0$ (α indeed $\alpha > 1.0$)?



Step 1: $[\nabla_\theta L(\theta)]|_{\theta=\theta^{(0)}} = 2(0) - 4 = -4$

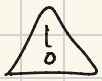
Update: $\theta^{(1)} = 0 - 2.0(-4) = 8$

Step 2: $[\nabla_\theta L(\theta)]|_{\theta=\theta^{(1)}} = 2(8) - 4 = 12$

Update: $\theta^{(2)} = 8 - 2.0(12) = -16$

Step 3: $[\nabla_\theta L(\theta)]|_{\theta=\theta^{(2)}} = 2(-16) - 4 = -36$

Update: $\theta^{(3)} = -16 - 2.0(-36) = 56$



optimization path moving away from minimum

We will introduce modifications to GD which use adaptive learning rates to reduce encountering these scenarios

First however we will address Issue (4) by introducing another GD modification - Stochastic Gradient Descent.

Stochastic Gradient Descent (SGD)

Idea: To avoid evaluating the expensive "whole data" gradient $[\nabla_\theta L(\theta, D)]|_{\theta=\theta^{(t)}}$, instead estimate this quantity with a cheaper estimator calculated using only a subset of D.

Def: A batch (B_t^m) is a subset of the training dataset containing m observations.

$$D = \{(x_i, y_i) : i = 1, \dots, n\}$$

$$B_t^m = \{(x_i, y_i) : i = 1, \dots, m\} \subset D$$

Made with Goodnotes

Def: An epoch is a collection of d B_t^m batches which together partition D. In other words:

$$\bigcup_{j=1}^{d} B_j^m = D \quad \text{and} \quad B_j^m \cap B_k^m = \emptyset \quad \forall j,k$$

- Generally the B_t^m which make up an epoch are obtained by randomly sampling D (w/o replacement)

The SGD update rule:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha_t \left[\nabla_\theta L(\theta, B_t^m)\right]_{\theta=\theta^{(t)}}$$

where $\nabla_\theta L(\theta, B_t^m)$ is the gradient wrt θ of the loss L evaluated on the m observations in B_t^m

Should think of $\nabla_\theta L(\theta, B_t^m)$ as an estimate of $\nabla_\theta L(\theta, D)$

Note: The SGD update rule is applied as follows. First, a new epoch is drawn/constructed. (for example by randomly shuffling D and then assigning the 1st m observations of this re-ordering to $B_{t_1}^m$, the second m to $B_{t_2}^m$ and so on)
Next, the update rule is applied to each batch in the epoch. Then a new epoch is constructed an the process is repeated.

SGD + Momentum

issue (1) [also helps (3)]

Intuition: To avoid getting trapped in shallow local minimums, accumulate "momentum" in directions of persistent travel that require persistent changes to direction to overcome.

Made with Goodnotes

Update Rule

$$\theta^{(t+1)} = \theta^{(t)} - \underbrace{v^{(t)}}_{\text{velocity}}$$

$$v^{(t)} = \underbrace{\gamma v^{(t-1)}}_{\text{gradient "memory"}} + \alpha_t \left[\nabla_\theta L(\theta, B_t^m)\right]\Big|_{\theta=\theta^{(t)}} \quad \text{for } t > 0$$

$$v^{(0)} = 0$$

momentum: $\gamma \geq 0$, rule-of-thumb $\gamma = 0.9$

This update rule produces a "smoothed" optimization path as compared to SGD by encouraging updates similar to previous ones

Made with Goodnotes