Practical Considerations for Gradient Descent

The following are some practical issues concerning gradient descent.

Feature Scaling

Make sure that features have similar scales (range of values). One way to assure this is to normalize the training date so that each feature has a range of 1.

Simple technique: Divide by the Range of sample values.
For a training set $\{\vec{X}_m\}$ of M training samples with D values.

Range: $r_D = Max(x_d) - Min(x_d)$

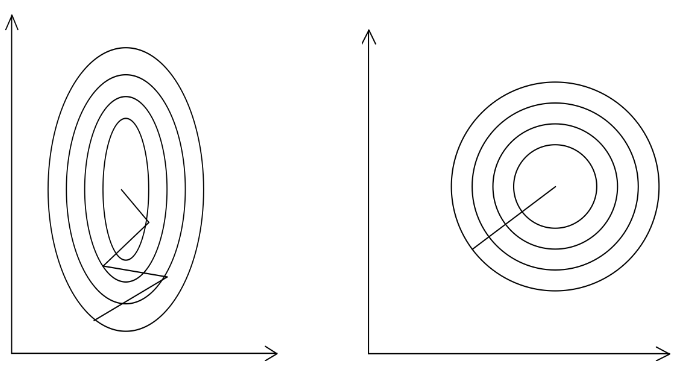
Then

$$\forall_{m=1}^{M} : x_{dm} := \frac{x_{dm}}{r_d}$$

Even better would be to scale with the mean and standard deviation of the each feature in the training data

$$\mu_d = E\{x_{dm}\} \qquad \sigma^2 = E\{(x_{dm} - \mu_d)^2\}$$

$$\forall_{m=1}^{M} : x_{dm} := \frac{(x_{dm} - \mu_d)}{\sigma_d}$$



Verifying Gradient Descent

The value of the loss function should always decrease:
Verify that $L(\vec{w}^{(i)}) - L(\vec{w}^{(i-1)}) < 0$.

if $L(\vec{w}^{(i)}) - L(\vec{w}^{(i-1)}) > 0$ then decrease the learning rate "α"

Gradient Descent vs Direct Solution

Form M training samples composed of D features:

<u>Direct Solution</u>:
Advantages:

1) No need to choose a learning rate (α)
2) No need to iterate - Predictable computational cost.

Inconvenient: Need to compute $(\vec{X}^T\vec{X})^{-1}$ which has a computational cost $O(M^3)$

<u>Gradient Descent</u>:
Advantages: Works well for Large
Inconvenient:

1) Need to choose a learning rate (α)
2) Can require many iterations to converge, each iteration costs O(M).
   (number of iterations not known in advance.)