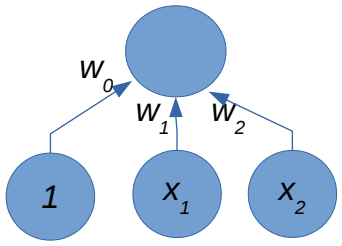
CS445 Gradient Descent Exercises

Name:

With help from:

1. Consider the following, very simple, "neural network":



where the activation of the output unit is just the dot product between the input and the weight vector: $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$

Assume that the current weights are $w_0 = 0, w_1 = 1, w_2 = .5$ and we have the following set of examples $D.train$:

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ -2 & 5 \\ 0 & 1 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} 1 \\ 6 \\ 1 \end{bmatrix}$$

Where each row in $\mathbf{X}$ represents a data point and each entry in $\mathbf{y}$ represents the corresponding target value.

a. What output will this network produce for $\mathbf{x} = \begin{bmatrix} 2 & 3 \end{bmatrix}^T$?

b. Calculate

$$L(\mathbf{w}) = \frac{1}{2} \sum_{(\mathbf{x}_i, y_i) \in D} (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

for this data set.

c. Can you find a set of weights that would result in less error?

2. Perform one round of **batch** gradient descent updates using the data set in question 1) and $\eta = .01$. Recall that the learning rule for our sum-squared error term is:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \sum_{(\mathbf{x}_i, y_i) \in D} (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i$$

Recalculate the the error with the new weights to confirm it went down.

3. Repeat the previous question using one round of **stochastic gradient descent**. Process the points in the order they appear in the data set. Is the final loss lower or higher in this case?

4. Calculate the partial derivatives of $f(x, y) = \sqrt{x^2 + y^2}$. Show your work.

5. Recall the least-squares error function for linear regression:

$$L(\boldsymbol{w}) = \frac{1}{2}(y - \boldsymbol{w}^T \boldsymbol{x})^2$$

(This is the error associated with a single training sample with input $\boldsymbol{x}$ and target value y.)

This objective function encodes a belief that bigger errors are *much* worse than smaller errors: in particular, that the penalty for making a mistake should grow with the square of the magnitude of the mistake. That seems reasonable[1], but it isn't the only possible error function. One problem with using a squared error function is that outliers can have a big impact on the result.

An alternative that is more robust to outliers is the absolute error (or L1 error):

$$\begin{aligned} L(\boldsymbol{w}) =& |y - \boldsymbol{w}^T \boldsymbol{x}| \\ & |y - (w_0 x_0 + ... + w_i x_i + ... + w_n x_n)| \end{aligned}$$

Your goal in this exercise is to develop a gradient-descent learning rule for this new objective function. (It will be helpful to know that $\frac{d}{dx}|u| = \frac{u}{|u|} \times \frac{d}{dx}u$)

Your final rule should have the form:

$$w_i \leftarrow w_i - \eta???$$

[1] In fact, there are good statistical reasons for using this error function, particularly if the noise in the data is normally distributed. This blog post outlines the reasoning: `http://aoliver.org/why-mse`