A4 - part 1: Deriving Backpropagation for Batch Gradient Descent

1) **Review of Stochastic Gradient Descent:**

In Stochastic Gradient Descent (SGD), we update the weights of a neural network iteratively for each training example. The update rule for the weights in each iteration is given by:

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} - \eta \frac{\partial L(y, \hat{y})}{\partial w_{ij}^{(k)}}$$

Where:
$w_{ij}^{(k+1)}$ represents the weight between neuron i in layer k and neuron j in layer k+1.
η is the learning rate.
$\frac{\partial L(y,\hat{y})}{\partial w_{ij}^{(k)}}$ is the gradient of the loss function L with respect to the weight $w_{ij}^{(k+1)}$.
y is the true target, and $\hat{y}$ is the predicted target.

2) **Understand Batch Gradient Descent:**

Batch Gradient Descent (BGD) differs from SGD in that it computes the gradient of the loss function with respect to the weights using the entire training dataset in each iteration. It calculates the average gradient over the entire dataset and then updates the weights. In BGD, you sum the gradients of the loss function over all training examples and then update the weights. The update rule for BGD is similar to SGD but with the average gradient:

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} - \eta \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L(y_i, \hat{y_i})}{\partial w_{ij}^{(k)}}$$

Where:

$N$is the number of training examples.

$\frac{\partial L(y_i,\hat{y_i})}{\partial w_{ij}^{(k)}}$ is the gradient of the loss for the i-nth training example.

3) **Derive Backpropagation for Batch Gradient Descent:**

To derive backpropagation equations for BGD, you need to compute the gradient of the loss function with respect to each weight for the entire dataset and then update the weights using the BGD update rule.

$$\frac{\partial L}{\partial w_{ij}^{(k)}} = -\frac{1}{N} \sum_{i=1}^{N} 2(y_i - \hat{y_i}) \cdot \frac{\partial \hat{y_i}}{\partial a_j^{(L)}} \cdot \frac{\partial a_j^{(L)}}{\partial w_{ij}^{(k)}}$$

$$\frac{\partial a_j^{(L)}}{\partial w_{ij}^{(k)}} = z_j^{(k-1)}$$

Gradient Calculation for Output Layer (MSE Loss):

$$\frac{\partial L}{\partial w_{ij}^{(L)}} = -\frac{1}{N}\sum_{i=1}^{N} 2(y_i - \hat{y}_i) \cdot \frac{\partial \hat{y}_i}{\partial w_{ij}^{(L)}}$$

Gradient Calculation for Hidden Layers (MSE Loss):

$$\frac{\partial L}{\partial w_{ij}^{(k)}} = -\frac{1}{N}\sum_{i=1}^{N} 2(y_i - \hat{y}_i) \cdot \frac{\partial \hat{y}_i}{\partial a_j^{(L)}} \cdot \frac{\partial a_j^{(L)}}{\partial w_{ij}^{(k)}}$$

$$\frac{\partial a_j^{(L)}}{\partial w_{ij}^{(k)}} = z_j^{(k-1)}$$

Further information comparing Stochastic and Batch gradient descent methods:

Gradient Descent is used to train Linear Regression models by adjusting parameters like feature weights and bias terms through iterative optimization. It ensures minimal cost function over training data and is guaranteed to find the optimal solution given enough time and an appropriate learning rate. In Linear Regression and Neural Networks, two primary variants of Gradient Descent are used: Batch Gradient Descent and Stochastic Gradient Descent (SGD).

Batch Gradient Descent processes the entire training set in each iteration, making it accurate and suitable for convex or smooth error landscapes. However, it's slow on large datasets and demands more computational resources. On the other hand, Stochastic Gradient Descent processes single training examples or small subsets, leading to faster convergence but less accuracy.

In a nutshell, Batch Gradient Descent is slower but more accurate, while Stochastic Gradient Descent is faster but less accurate. The choice depends on the specific problem, dataset, and computational resources available.