MPCS 51087
Problem Set 3
Machine Learning for Image Classification
Milestone 2

Winter 2024

due: Sunday March 3 @6pm

1 Model Updates

For the second milestone we will make some updates to our basic model to make it more robust. Implement the following changes in your milestone 1 code. We will refer to this model as Model 2. Even though we are using a single hidden layer, your code must be general and accept any number of layers and neurons per layer.

- Change the model from 2 hidden layers to a single hidden layer with $n = 800$.
- Change the activation function in the hidden layer from Sigmoid to ReLu: $R(z) = max(0, z)$
- Use the Kaiming initialization scheme: $w = \mathcal{N}(0, 2/n)$ – i.e. a normal distribution centered at zero with variance $2/n$.
- For the output layer, replace the sigmodid activation function with the *softmax activation*:

  $$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

  where $K = 10$ here. Instead of the MSE loss, use the categorical cross-entropy loss function, where for a single sample:

  $$CE = -\sum_{c=1}^{K} y_c \log(p_c)$$

  where $p_c = \sigma(\mathbf{z})_i$ output layer values.

Note, the backpropagation formulas will be different from those derived in section 2.3 for MSE loss with sigmoid activations. Derive (or lookup) the gradient calculations for this network architecture. See footnote for a good discussion of why cross-entropy is preferable to MSE loss for this classification problem.[1]

2 Implementation Updates

Once the new functionality is in place, include the following in your CPU version:

- The ability to process any number of input samples concurrently, so that the algorithm is composed mostly of matrix products rather than matrix vector operations, as discussed in class.

[1] `https://bit.ly/3wLUjvb`

- If you haven't already, use a BLAS implementation (OpenBLAS or MKL) to carry out the matrix-matrix products.
- If you haven't already, hand-code your own matrix-matrix routines (this is for timing comparisons). Your code should be able to use either your hand coded or BLAS linear algebra routines.

3 Baseline GPU Version

Once these minor updates to your CPU version are in place and tested, develop a baseline GPU implementation. At this stage the GPU implementation has no specific performance requirements other than correctness and reasonable (not optimal) implementation choices. We will refine the GPU version in the final milestone.

4 Performance Benchmarks

Using a value of $\alpha = .01$, 5 epochs, and batch sizes of 256 and 512, show the time to solution, grind rate, and accuracies for the CPU version without BLAS, the CPU version with BLAS, and the baseline GPU version.