15-780 HW4

Due 2/21

A Two-layer Neural Network Language Model

For this problem, you should build on your implementation from Homework 3 and implement a Shakespeare language model with a *two-layer neural network*. Specifically, you should implement a two-layer neural network of the form

$$h_\theta(X) = \sigma(XW_1)W_2 \tag{1}$$

for $X \in \mathbb{R}^{m\times n}$, $W_1 \in \mathbb{R}^{n\times d}$ and $W_2 \in \mathbb{R}^{d\times k}$, and where σ denotes here the ReLU nonlinearity. For this setting, recall that the gradients we derived in class were given by

$$\nabla_{W_1}\ell_{\text{ce}}(h_\theta(X), Y) = X^T\left(\text{softmax}(h_\theta(X)) - I_Y)W_2^T \circ \sigma'(XW_1)\right) \tag{2}$$

$$\nabla_{W_2}\ell_{\text{ce}}(h_\theta(X), Y) = \sigma(XW_1)^T(\text{softmax}(h_\theta(X)) - I_Y). \tag{3}$$

Note that there should be no changes to the input embeddings. And as with HW3, you can use any of the `torch` functions, but not the `torch.nn` library.

Specifically, you should start with your Homework 3 code, and replace the `epoch_linear_language_model` with an alternative function `epoch_nn_language_model` that takes the same arguments except `W1` and `W2` as arguments, and runs one epoch of SGD to update these parameters in place.

You should initialize W_1 and W_2 with the values (we'll discuss later why these are good):

```
W1 = torch.randn(n,d) * np.sqrt(2/n)
W2 = torch.randn(d,k) * np.sqrt(2/d)
```

As before, plot the training loss over the epoch, evaluate the test loss, and generate some samples. Experiment with different numbers of hidden units, and different learning rates. Comment on the training versus test loss of the neural network model versus the linear language model.

Residual Connection

Recall that for a generic L-layer neural network with cross-entropy loss we derived the forward pass

$$Z_1 = X \tag{4}$$

$$Z_{i+1} = \sigma(Z_iW_i), i = 1, \ldots, L \tag{5}$$

for $Z_i \in \mathrm{R}^{m\times n_i}$, $W_i \in \mathbb{R}^{n_i\times n_{i+1}}$, and the corresponding backward pass

$$G_{L+1} = (\text{softmax}(Z) - I_Y) \tag{6}$$

$$G_i = (G_{i+1} \circ \sigma'(Z_iW_i))W_i^T \tag{7}$$

$$\nabla_{W_i}\ell(Z_{L+1}, Y) = Z_i^T(G_{i+1} \circ \sigma'(Z_iW_i)) \tag{8}$$

where $G_i \in \mathbb{R}^{m\times n_i}$ represents the gradient $\nabla_{Z_i}\ell(Z_{L+1}, Y)$.

For this question, derive the backpropagation equations from the following alternative forward pass

$$Z_{i+1} = \sigma(Z_iW_i) + Z_iU_i, \tag{9}$$

where $Z_i \in \mathrm{R}^{m \times n_i}$, $W_i \in \mathbb{R}^{n_i \times n_{i+1}}$, and $U_i \in \mathbb{R}^{n_i \times n_{i+1}}$. Derive the format and backward pass for the Z_i and G_i terms, and also derive gradients of the loss with respect to W_i and U_i.

[Note: to make the problem a bit easier, I decided to change this slightly from the form in class to include the U_i matrix. This makes the sizes for each hidden unit in the forward pass different, and lets you derive the precise update more easily via matching sizes.]