50 points total. This is a short assignment to practice math skills. Show your work. Make sure your results are readable — either as a scan of handwritten solutions, or typed up equations. (If typing it up, do not use computer code-like notation like “A[x,y]”; use proper mathematical notation like $A_{x,y}$.)

Follow the same collaboration and citation policies as before: http://people.cs.umass.edu/~brenocon/anlp2017/grading.html

1 Nonlinear gradients

We define the softmax function $\mathbb{R}^K \to \mathbb{R}^K$ as:

$$S(y) = \frac{e^y}{\sum_j e^{y_j}}$$

So for one output dimension $i$,

$$[S(y)]_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

The normalizer is shared across all output dimensions.

1.1 Question (5 points)

Derive the gradient of the log softmax indexed at output dimension $w$ in terms of one the inputs $i$, 

$$\frac{\partial \log[S(y)]_w}{\partial y_i}$$

1.2 Question (3 points)

Derive the gradient of the scalar tanh function $g(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$,

$$\frac{\partial g(x)}{\partial x} = \frac{\partial}{\partial x} \left( \frac{e^{2x} - 1}{e^{2x} + 1} \right)$$

and make sure to represent the final answer in terms of $g$. 
2 Softmax MLP

Consider a model with one hidden layer where next word \(w\) is generated from input vector \(x\) (omitting bias terms for simplicity), with elementwise tanh \(g\):

\[
h = g(Ax) \tag{1}
\]
\[
p = S(Bh) \tag{2}
\]
\[
w \sim \text{Categ}(p) \tag{3}
\]

Let \(D\) be the input dimensionality for \(x \in \mathbb{R}^D\) (for example, in the n-gram MLP it could be the concatenation of word embeddings of context words), \(K\) the hidden layer dimensionality for \(h \in \mathbb{R}^K\), and \(V\) the output vocabulary size for \(p \in \mathbb{R}^V\). We are using conventional column vector notation (as opposed to the convention in the Goldberg reading). For example, the entire model can be written \(P(w \mid x) = S(Bg(Ax))\).

2.1 Question (2 points)

What are the dimensionalities of \(A\), \(Ax\), \(B\), and \(Bh\)?

Gradients

To make a probabilistic prediction from \(x\), a forward pass simply computes \(h\) and \(p\) in turn. With these quantities, gradients can be calculated with backpropagation. We’ll use the convention that \(w\) is an index \((w \in \{1..V\})\) representing the word. (If you are good at multivariate calculus, feel free to use one-hot notation instead, but be clear what convention you’re following.) When you derive the following gradients, make sure they’re in a form that is practical to compute (as opposed to using sums over a combinatorial number of paths).

2.2 Question (20 points)

Derive the gradient for the parameters of the final softmax (multiclass logreg) layer

\[
\frac{\partial}{\partial B} \log p_w
\]

2.3 Question (20 points)

Derive the gradient for the hidden-layer parameters

\[
\frac{\partial}{\partial A} \log p_w
\]

Notes

A good approach is to first derive the gradient for a particular hidden-layer value \(\frac{\partial}{\partial h_k} \log p_w\), before further expanding with the chain rule to attack the final \(\frac{\partial}{\partial A_k,d} \log p_w\).

You will probably want to use the multiple-paths chain rule. Generally, for \(z = f(y_1, y_2, ..., y_n)\) where each \(y_i = g_i(x)\),

\[
\frac{\partial z}{\partial x} = \sum_i \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}
\]
See the illustrations in the first 9 slides of: https://cs224d.stanford.edu/lectures/CS224d-Lecture6.pdf
Also potentially helpful: https://colah.github.io/posts/2015-08-Backprop/