Assignment 2 writeup

Thomas Lu

1 Problem 1

(c) Placeholders are nodes in a TF computation graph whose values are populated at runtime. These nodes are often used to populate training data. Feed dictionaries, meanwhile, are what actually do the populating at runtime; they specify at runtime a mapping of placeholder nodes to actual values.

(e) When `train_op` is called, we compute during forward propagation (for a batch) predictions $\hat{y} = \text{softmax}(xW + b)$ and the loss $J = CE(y, \hat{y})$, where CE denotes cross-entropy loss. Then we compute during backpropagation the partial gradients $\partial J/\partial W$ and $\partial J/\partial b$ and add a negative multiple of these gradients to our variables W and b.

2 Problem 2

(a)

stack	buffer	new dependency	transition
[ROOT]	[I, parsed, this, sentence, correctly]		Initial Configuration
[ROOT, I]	[parsed, this, sentence, correctly]		SHIFT
[ROOT, I, parsed]	[this, sentence, correctly]		SHIFT
[ROOT, parsed]	[this, sentence, correctly]	parsed → I	LEFT-ARC
[ROOT, parsed, this]	[sentence, correctly]		SHIFT
[ROOT, parsed, this, sentence]	[correctly]		SHIFT
[ROOT, parsed, sentence]	[correctly]	sentence → this	LEFT-ARC
[ROOT, parsed]	[correctly]	parsed → sentence	RIGHT-ARC
[ROOT, parsed, correctly]	[]		SHIFT
[ROOT, parsed]	[]	parsed → correctly	RIGHT-ARC
[ROOT]	[]	ROOT → parsed	RIGHT-ARC

(b) A written sentence of n words will require $2n$ steps: each word requires one step to move it from the buffer to the stack, and one to move it from the stack to the dependency tree.

(f) We have

$$h_i = \mathbb{E}_{drop}[h_{drop}]_i = \gamma(1 - p_{drop})h_i,$$

so $\gamma = 1/(1 - p_{drop})$.

(g)

- (i) Momentum basically slows the rate at which we adjust our updates: instead of immediately updating using the current gradient, we instead continue mostly going in the direction of our previous momentum and only assign a partial influence to the current gradient on our next update. This can help us avoid large "bad" updates when we see a particularly anomalous batch of training data or when we hit a particularly steep gradient wall.

- (ii) Adam amplifies the movement of parameters with small gradient contributions and reduces that of parameters with large gradient contributions. This can help in cases where we might have large almost-flat regions and small steep walls in our gradient function.