CS224n Assignment 3: Dependency Parsing

David Lee

October 30, 2019

1 Machine Learning & Neural Networks

1.1 Adam Optimizer

1.1.1 Momentum: Briefly explain how using m stops the updates from varying as much and why this low variance may be helpful to learning, overall.

The momentum (exponentially weighted moving average) makes the current gradient not just dependent on its mini-batch gradient (like SGD did). In some case, if the learning rate is too large we will miss the optimized solution (diverge), but if the learning is too small then it will converge very slow.

This using momentum imply the benefit of:

- reduce oscillation: make the learning rate more stable (not change too fast)
- faster convergence: we can set a larger initial learning rate and it will adjust by itself

1.1.2 Adaptive Learning Rates: Since Adam divides the update by $\sqrt{\mathbf{v}}$, which of the model parameters will get larger updates? Why might this help with learning?

The model parameters which receive small or infrequent updates will get larger updates.

Consider the opposite situation, the model parameters which recive larger updates will have their effective learning rate reduced. Thus we can regard adaptive learning rates as normalization of the parameter update step by element wise.

1.2 Dropout

1.2.1 What must γ equal in terms of p_{drop}? Briefly justify your answer.

The purpose of γ is said to make the expected value of $\mathbf{h}_{\text{drop}}$ is still $\mathbf{h}$

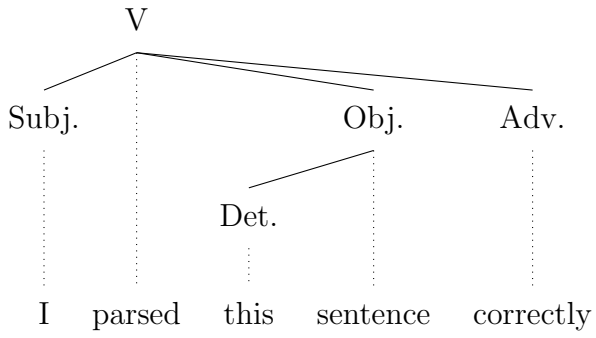
Thus γ must be $\frac{1}{(1-p_{\text{drop}})}$ because the expected value of $\mathbf{d}$ is $(1 - p_{\text{drop}})$

1.2.2 Why should we apply dropout during training but not during evaluation?

The regularization technique is aimed to prevent overfitting. Since overfitting only happen during training.

2 Neural Transition-Based Dependency Parsing

2.1 Parse the sentence "I parsed this sentence correctly" by given dependencies.



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]	I $\leftarrow$ parsed	LEFT-ARC
[ROOT, parsed, this]	[sentence, correctly]		SHIFT
[ROOT, parsed, this, sentence]	[correctly]		SHIFT
[ROOT, parsed, sentence]	[correctly]	this $\leftarrow$ sentence	LEFT-ARC
[ROOT, parsed]	[correctly]	parsed $\rightarrow$ sentence	RIGHT-ARC
[ROOT, parsed, correctly]	[]		SHIFT
[ROOT, parsed]	[]	parsed $\rightarrow$ correctly	RIGHT-ARC
[ROOT]	[]	ROOT $\rightarrow$ parsed	RIGHT-ARC

Table 1: Parsing the sentence "I parsed this sentence correctly" with optimal steps (assume we have trained a classifier).

2.2 A sentence containing n words will be parsed in how many steps (in terms of n)? Briefly explain why.

A sentence will be parsed in $2n$ times. Because we will eventually move all words from the *Buffer* to *Stack* (the SHIFT step) and then remove all words from the *Stack* (the ARC steps).