ASSIGNMENT - 3

SIMERJOT KAUR (sk3391)

(1) (a) (i) Momentum means that fraction of previous update is added to the current update, so the repeated updates in a particular direction compound. This means that it prevents the updates from changing too much, hence building up momentum and moving faster and faster towards local optimum. This reduces / dampens the oscillations around local optimum.

This helps with learning because this causes the gradient to be a rolling average over the updates multiple minibatches and hence gets closer to the true gradient over the full dataset.

(1)(a)(ii) Due to the division by $\sqrt{v}$, the parameters with smaller gradients will receive larger updates.

This helps in learning because parameters with low gradients can get stuck in a plateau region, hence by recieving larger updates such parameters can move off the plateau region.

(1)(b)(i) $\forall$ $E_{p_{drop}}[h_{drop}]_i = h_i$

$$E_{p_{drop}}[h_{drop}]_i = E[\gamma d_i h_i]$$

Given, $E[d_i] = p_{drop}\cdot(0) + (1-p_{drop})\cdot 1$

$$= (1-p_{drop})$$

$$\Rightarrow E[\gamma d_i h_i] = (E[d_i])\,\gamma h_i$$

$$= (1-p_{drop})\,\gamma h_i$$

and $E_{p_{drop}}[h_{drop}]_i = h_i$

$$\Rightarrow (1-p_{drop})\,\gamma h_i = h_i$$

$$\Rightarrow \boxed{\gamma = \frac{1}{1-p_{drop}}}$$

(1)(b)(ii) Dropout is needed to avoid overfitting. A fully connected neural n/w layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training, which curbs the individual power of each neuron leading to overfitting. Hence during training, randomly selected neurons are ignored.

Dropout is not used during evaluation because we want our neural n/w to be fixed and ~~hence~~ dont want our output to be random. If we implement dropout at test time, then it would add noise to our predictions which we dont want.

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

(2)(b) Since for each word there are two steps involved:
SHIFT from buffer onto stack and REDUCE via left-arc / right-arc.

$\Rightarrow$ for 1 word there are 2 steps involved

$\Rightarrow$ n words will require 2n steps.

(2) (f)

(i) ERROR TYPE : Verb Phrase Attachment Error

INCORRECT DEPENDENCY : wedding → fearing

CORRECT DEPENDENCY : heading → fearing

(ii) ERROR TYPE : Coordination Attachment Error

INCORRECT DEPENDENCY : mates → rescue

CORRECT DEPENDENCY : out → rescue

(iii) ERROR TYPE : Prepositional Phase Attachment Error

INCORRECT DEPENDENCY : named → Midland

CORRECT DEPENDENCY : guy → Midland

(iv) ERROR TYPE : Modifier Attachment Error

INCORRECT DEPENDENCY : elements → most

CORRECT DEPENDENCY : crucial → most