1 Machine Learning & Neural Networks

Solution to Problem (a):

i. The vector $\boldsymbol{m}$ is kind of a "velocity". It always change into to be a quite similar vector as before, because the hyperparameter β_1 is close to 1. Such a feature of "momentum" method alleviates wavering update of model parameters, thus the learning process becomes stable.

ii. Note that $\boldsymbol{v}$ is a moving average of (element-wise) square of the gradient. If the loss function did not change that much so far in terms of some parameters, then the entries of $\boldsymbol{v}$ associated to such parameters would be smaller. As a result, by dividing the update by $\boldsymbol{v}$, such parameters will get larger updates. In other words, the given trick allows the parameters whose effective learning rate is relatively poor to have more possibility to get modified. This helps with learning in a sense of element-wise equalizing impact.

■

Solution to Problem (b):

i. The value of γ should be equal to $1/(1-p_{\text{drop}})$, because of the last equality of the following calculations:

$$\begin{aligned}\mathbb{E}_{p_{\text{drop}}}[\boldsymbol{h}_{\text{drop}}]_i &= \mathbb{E}_{p_{\text{drop}}}[\gamma d_i h_i] \\ &= \gamma h_i\{p_{\text{drop}} \cdot 0 + (1-p_{\text{drop}}) \cdot 1\} \\ &= \gamma h_i(1-p_{\text{drop}}) = h_i.\end{aligned}$$

ii. According to the paper, applying Dropout during training 'prevents units from co-adapting too much' and thus resolves overfitting problem. In test time, however, Dropout must not be used, because the model should take advantage of all parameters it has learned.

■

2 Neural Transition-Based Dependency Parsing

Solution to Problem (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`

Solution to Problem (b):

We need exactly n reduction transitions (either `LEFT-ARC` or `RIGHT-ARC`): each of words is a dependent of a dependency; also, we need exactly n shift transitions, or `SHIFT`: each of words in the buffer is moved to the stack once and only once. Therefore, we need $2n$ steps to parse a sentence with n words. ■

Solution to Problem (c)~(d): (Pure Coding) ■

Solution to Problem (e): (Coding)

i. **Best UAS on dev set**: 88.24

ii. **UAS on test set**: 88.64

■

Solution to Problem (f):

i.
- **Error Type**: Verb Phrase Attachment Error
- **Incorrect dependency**: wedding → fearing
- **Correct dependency**: heading → fearing

ii.
- **Error Type**: Coordination Attachment Error
- **Incorrect dependency**: makes → rescue
- **Correct dependency**: rush → rescue

iii.
- **Error Type**: Prepositional Phrase Attachment Error
- **Incorrect dependency**: named $\rightarrow$ Midland
- **Correct dependency**: guy $\rightarrow$ Midland

iv.
- **Error Type**: Modifier Attachment Error
- **Incorrect dependency**: elements $\rightarrow$ most
- **Correct dependency**: crucial $\rightarrow$ most

■