



Week 3

태그

[cs224n-2023-lecture06-fancy-rnn.pdf](#)

[cs224n-2023-lecture05-rnnlm.pdf](#)

Homework 3

1

a-i: The usage of \mathbf{m} provides low variance in updating θ by preserving the previous momentum value for a portion of β_1 . This usage of momentum helps quick convergence by reducing the occurrence of oscillations, and also reduces the influence of learning rate α .

a-ii: Because Adam divides the update by \mathbf{v} , parameters with small \mathbf{v} 's will have larger updates. This division by \mathbf{v} is an important point on Adam because it creates an adaptive learning rate. This adaptive learning rate helps the model parameters with different gradients converge with optimal speed.

b-i:

$$h_{\text{drop}} = r d \cdot h, \quad E_{p_{\text{drop}}} [h_{\text{drop}}]_i = h_i$$

$$d = \underbrace{[0, \dots, 1, \dots, 0, \dots]}_{d_h}, \quad \begin{bmatrix} 0 & p_{\text{drop}} \\ 1 & 1-p_{\text{drop}} \end{bmatrix}$$

$$\rightarrow E_{p_{\text{drop}}} [r d \cdot h] = h_i \Rightarrow r E_{p_{\text{drop}}} [d \cdot h] = r E_{p_{\text{drop}}} [d] \cdot h = r(1-p_{\text{drop}})h$$

$$\rightarrow r(1-p_{\text{drop}})h = h$$

$$r = \frac{1}{1-p_{\text{drop}}}$$

b-ii: Dropout technique is very helpful to prevent overfitting during training. It helps the model to gain additional generalization power. But it should not be applied during evaluation because all the information in the evaluation process should not be lost. Random dropout will also lead to inconsistent results, which is not desirable in evaluation process.

2

(a)

Stack	Buffer	New Dependency	Transition
[ROOT]	[I, attended, lectures, in, the, NLP, class]		Initial Configuration
[ROOT, I]	[attended, lectures, in, the, NLP, class]		SHIFT
[ROOT, I, attended]	[lectures, in, the, NLP, class]		SHIFT
[ROOT, attended]	[lectures, in, the, NLP, class]	attended → I	LEFT-ARC
[ROOT, attended, lectures]	[in, the, NLP, class]		SHIFT

[ROOT, attended]	[in, the, NLP, class]	attended → lectures	RIGHT-ARC
[ROOT, attended, in]	[the, NLP, class]		SHIFT
[ROOT, attended, in, the]	[NLP, class]		SHIFT
[ROOT, attended, in, the, NLP]	[class]		SHIFT
[ROOT, attended, in, the, NLP, class]	[]		SHIFT
[ROOT, attended, in, the, class]	[]	class → NLP	LEFT-ARC
[ROOT, attended, in, class]	[]	class → the	LEFT-ARC
[ROOT, attended, class]	[]	class → in	LEFT-ARC
[ROOT, attended]	[]	attended → class	RIGHT-ARC
[ROOT]	[]	ROOT → attended	RIGHT-ARC

(b)

The steps would be $2n$ because all n words goes through the stages of SHIFT and ARC (left or right).

Coding Result

```
Epoch 10 out of 10
100% | 1848/1848 [01:41<00:00, 18.23it/s]
Average Train Loss: 0.0665105958549846
Evaluating on dev set
1445850it [00:00, 23011582.71it/s]
- dev UAS: 88.69
New best dev UAS! Saving model.
```

```
D:\대학원\스터디\24_Summer_NLP\3\student\run.py:157: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  parser.model.load_state_dict(torch.load(output_path))
Final evaluation on test set
2919736it [00:00, 25236128.53it/s]
- test UAS: 89.09
Done!
```