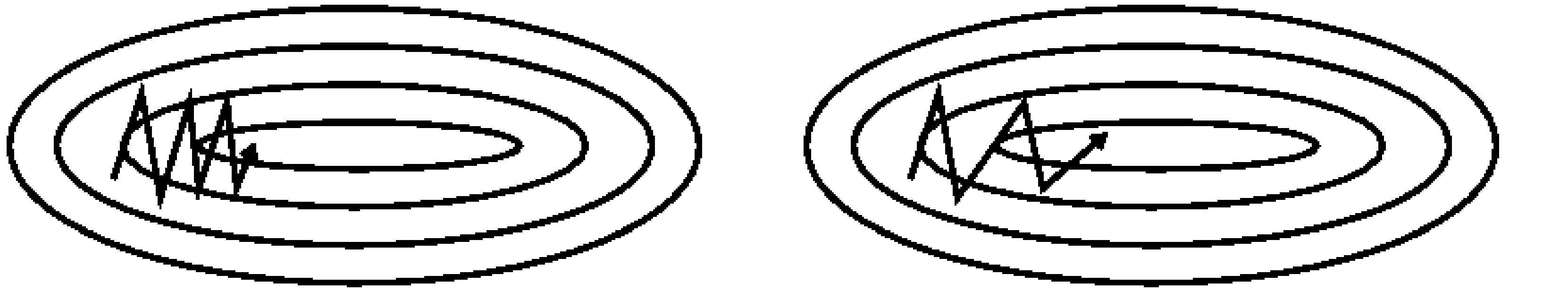
1.a Adam Optimizer

i. Momentum

Standard SGD have problem navigating around ravines. For example, in a 2-D setting, the gradient oscillates on Y direction but keep increasing to the X direction. In the following left image, on Y direction the gradient keeps oscillating from a large positive value to a large negative value, but on X direction it decreases it's magnitude keeps moving slowly to the local minima.

Without momentum	**With momentum**



By applying momentum, gradient changes to the same direction will keep increasing, which achieves faster convergence to the local minima; gradient changes in different directions will have it's magnitude decreased, which achieves reduced update oscillation.

ii. Adaptive learning rate

Since the update step is divided by the gradient magnitude in each direction, directions with higher gradient magnitude will have a smaller update rate; directions with lower gradient magnitude will receive higher update rate.

As learning is getting closer to the optima, changes in each direction will get smaller and smaller, hence the convergence will become much slower. Adaptive learning rate will scale up the update steps as the learning reaches to the optimal.

1.b Dropout

i. Value of γ

The derivation of γ:
Simplify $\mathbb{E}_{P_{drop}}[h_{drop}]_i$:

$$\begin{aligned}\mathbb{E}_{P_{drop}}[h_{drop}]_i &= \mathbb{E}_{P_{drop}}[\gamma d_i h_i] \\ &= \gamma h_i \mathbb{E}_{P_{drop}}[d_i] \\ &= \gamma h_i (1 - P_{drop})\end{aligned}$$

The last equation holds by applying the defionition of dropouts: d_i is 1 with probability of $1 - P_{drop}$.

The following equation holds:

$$\begin{aligned}\gamma h_i (1 - P_{drop}) &= h_i \\ \Rightarrow \gamma &= \frac{1}{1 - P_{drop}}\end{aligned}$$

So the chosen γ should be $\gamma = \frac{1}{1-P_{drop}}$.

ii. Why not dropout during evaluation

In deep learning, Dropout is a way to prevent over-fitting. There is no such problem in evaluation so the need for Dropout simply does not exist. Instead, all the activations are used but are reduced by a factor P_{drop} to account for the missing activations during training.

2.a Sequence of transitions

Stack	Buffer	New Dependency	Transition
[ROOT]	[I, parsed, this, sentence, correctly]		Init
[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 Number of steps

Total steps of the parsing is 2n:
Shift n times plus n times Left or Right Arc will be needed to make the buffer and stack empty.

2.e Best UAS achieved on the dev set and the UAS it achieves on the test set

- Best UAS on dev set: 88.73
- Test UAS: 88.80

2.f Dependency parse errors

1. - Error type: Verb Phrase Attachment Error
   - Incorrect dependency: wedding → fearing
   - Correct dependency: disembarked → fearing

1. - Error type: Coordination Attachment Error
   - Incorrect dependency: makes → rescue
   - Correct dependency: rush → rescue

1. - Error type: Prepositional Phrase Attachment Error
   - Incorrect dependency: named → Midland
   - Correct dependency: guy → Midland

1. - Error type: Modifier Attachment Error
   - Incorrect dependency: elements → most
   - Correct dependency: crucial → most

```
In [ ]:
```