Introduction to Deep Learning

19. Recurrent Neural Networks

STAT 157, Spring 2019, UC Berkeley

Alex Smola and Mu Li

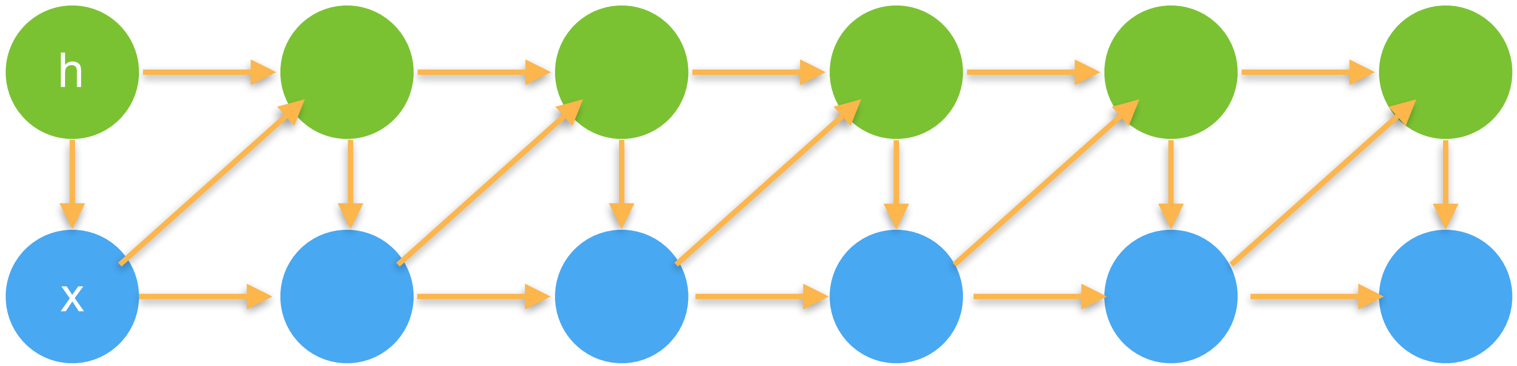
courses.d2l.ai/berkeley-stat-157

Recurrent Neural Networks

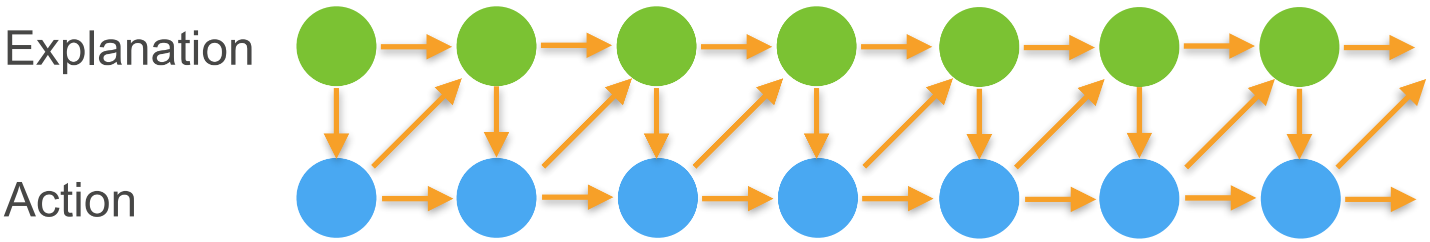
Latent Variable Autoregressive Models

Latent state summarizes all the relevant information about the past. So we get $h_t = f(x_1, \ldots x_{t-1}) = f(h_{t-1}, x_{t-1})$

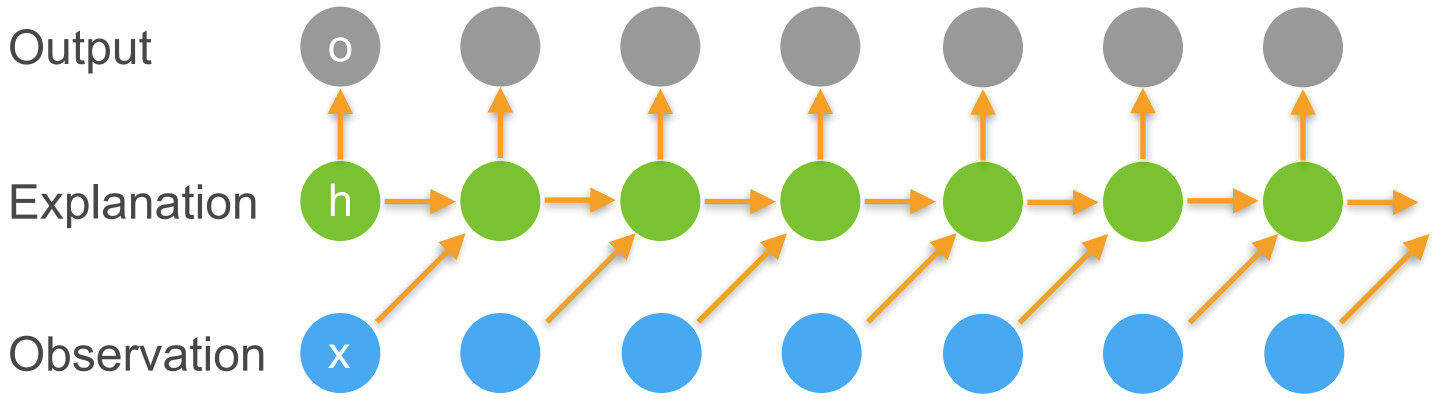
$p(h_t \mid h_{t-1}, x_{t-1})$ and $p(x_t \mid h_t, x_{t-1})$



Recurrent Neural Networks (with hidden state)



Recurrent Neural Networks (with hidden state)



- Hidden State update

$$\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$$

- Observation update

$$\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$$

Code …

Implementing an RNN Language Model

Input Encoding

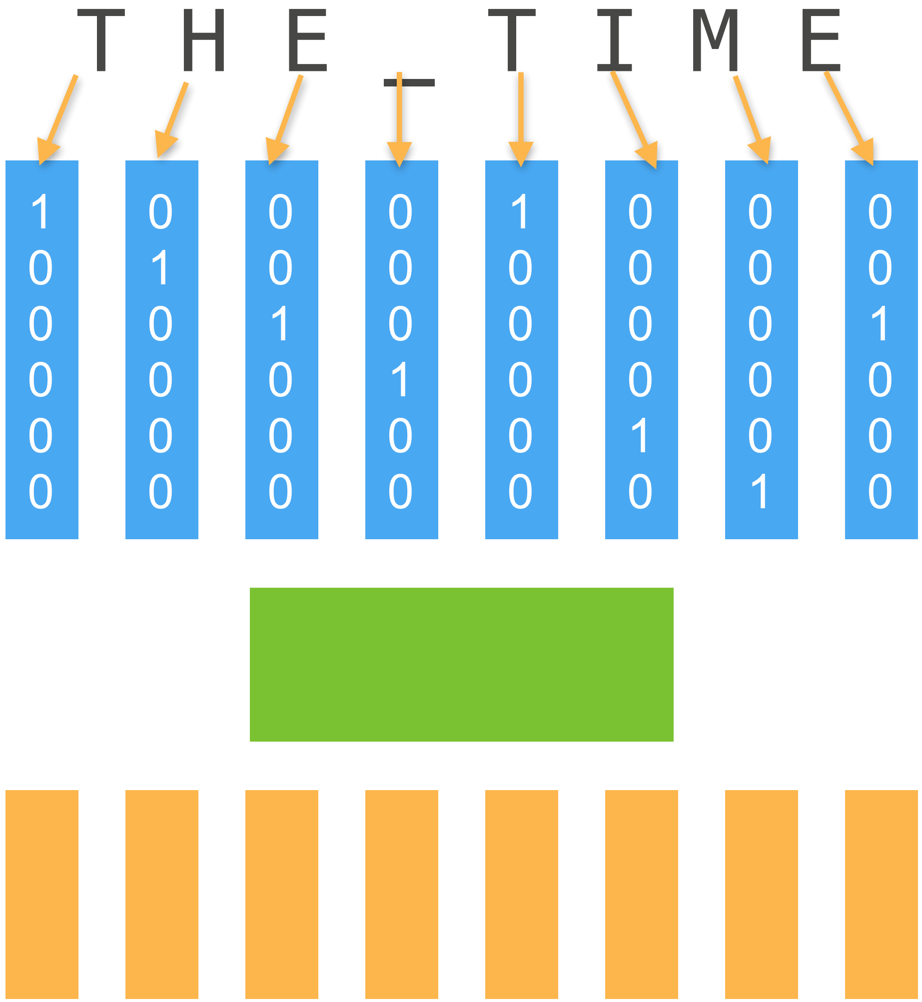
- Need to map input tokens to vectors
  - Pick granularity (words, characters, subwords)
  - Map to indicator vectors

```
nd.one_hot(nd.array([0, 2]), vocab_size)
```

```
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
<NDArray 2x43 @cpu(0)>
```

- Multiply by embedding matrix W

Input Encoding



Canonical Vectors $\mathbf{v}$

Embedding Matrix $\mathbf{W}$

Embedded Vectors $\mathbf{v}'$

RNN with hidden state mechanics

- Input
  vector sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$
- Hidden States
  vector sequence $\mathbf{h}_1, \ldots, \mathbf{h}_T$ where $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$
- Output
  vector sequence $\mathbf{o}_1, \ldots, \mathbf{o}_T$ where $\mathbf{o}_t = g(\mathbf{h}_t)$

Read sequence to generate hidden states, then start generating outputs. Often outputs (symbols) are used as input for next hidden state (and thus output).

Output Decoding

Output Vectors $\mathbf{o}$

Decoding Matrix $\mathbf{W}'$

$$p(y \mid \mathbf{o}) \propto \exp\left(\mathbf{v}_y^\top \mathbf{o}\right) = \exp(\mathbf{o}[y])$$

One-hot decoding

Gradients

- Long chain of dependencies for backprop
  - Need to keep a lot of intermediate values in memory
  - Butterfly effect style dependencies
  - Gradients can vanish or diverge (more on this later)
- Clipping to prevent divergence

$$\mathbf{g} \leftarrow \min\left(1, \frac{\theta}{\|\mathbf{g}\|}\right)\mathbf{g}$$

rescales to gradient of size at most θ

Perplexity

- Typically measure accuracy with log-likelihood
  - This makes outputs of different length incomparable (e.g. bad model on short output has higher likelihood than excellent model on very long output)
  - Normalize log-likelihood to sequence length

$$-\sum_{t=1}^{T} \log p(y_t \mid \text{model}) \quad \text{vs.} \quad \pi := -\frac{1}{T}\sum_{t=1}^{T} \log p(y_t \mid \text{model})$$

- Perplexity is exponentiated version $\exp(\pi)$ (effectively number of possible choices on average)

Code …