8.3 Recurrent Neural Networks

Lecture based on "Dive into Deep Learning" **http://D2L.AI** (Zhang et al., 2020)

Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

- In n-gram models the conditional probability of word x_{t+1} at position $t+1$ only depends on the $n-1$ previous words.

$$\begin{aligned} p(x_{t+1}|x_t, \dots x_1) &\approx p(x_{t+1}|x_t, \dots x_{t-n}) \\ &= p(x_{t+1}|\ f(x_t, \dots x_{t-n})\ ) \end{aligned}$$

  - To check the effect of words earlier than x_{t-n} on x_{t+1}, we need to increase n.
  - We need to store $|V|^n$ (exponentially many) numbers for a vocabulary V.

- Instead we will use a **latent variable model**

$$\begin{aligned} p(x_{t+1}|x_t, \dots x_1) &\approx p(x_{t+1}|x_t, h_t) \\ &= p(x_{t+1}|\ \underbrace{h_{t+1}}_{f(x_t, h_t)}\ ) \end{aligned}$$

  - For a *sufficiently powerful function* f this is not an approximation. h_{t+1} could simply store all the data it observed so far.

MLP for sequences

- $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ is the mini-batch input for step t
- Hidden layer $\mathbf{H} \in \mathbb{R}^{n \times h}$ with h hidden units

$$\mathbf{H} = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{b}_h).$$

  - weight parameter $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$
  - bias parameter $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$

- The output layer is given by

$$\mathbf{O} = \mathbf{H}\mathbf{W}_{hq} + \mathbf{b}_q.$$

  - $\mathbf{O} \in \mathbb{R}^{n \times q}$ is the output variable
  - $\mathbf{W}_{hq} \in \mathbb{R}^{h \times q}$ is the weight parameter
  - $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ is the bias parameter of the output layer

- We can use softmax$(\mathbf{O})$ to predict output probabilities

Recurrent Neural Networks (RNN) with Hidden States

- $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ be the mini-batch input for step t
- $\mathbf{H}_t \in \mathbb{R}^{n \times h}$ be the hidden variable for step t modeled by an **RNN**
- $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$ be the hidden variable for step $t-1$

$$\mathbf{H_t} = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h).$$

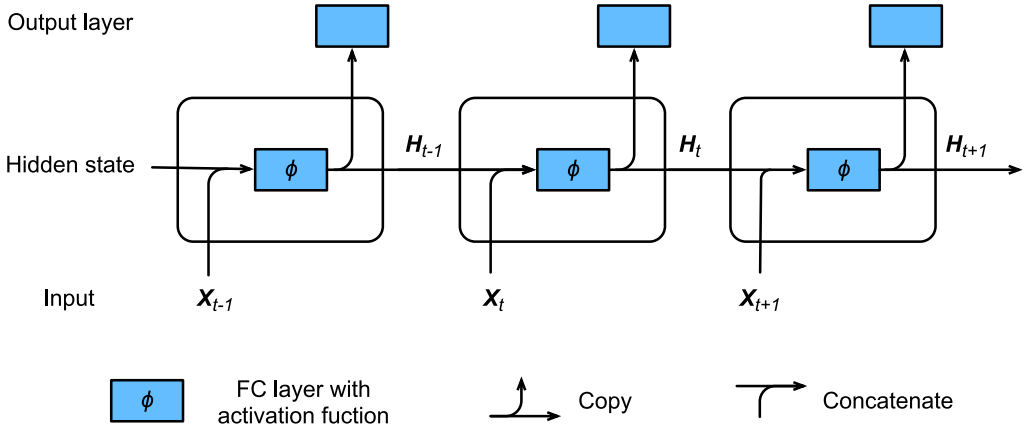
- weight parameter $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$
- bias parameter $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$
- weight parameter $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$, to relate the previous time step in the current time step

- The output layer is given by

$$\mathbf{O} = \mathbf{H}\mathbf{W}_{hq} + \mathbf{b}_q.$$

- $\mathbf{O} \in \mathbb{R}^{n \times q}$ is the output variable
- $\mathbf{W}_{hq} \in \mathbb{R}^{h \times q}$ is the weight parameter
- $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ is the bias parameter of the output layer

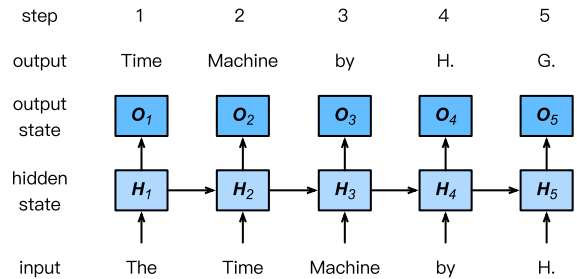
- We can use softmax$(\mathbf{O})$ to predict output probabilities


Output layer
Hidden state
Input
ϕ
H_{t-1}
H_t
H_{t+1}
X_{t-1}
X_t
X_{t+1}
ϕ
FC layer with activation fuction
Copy
Concatenate

In a **language model**, we use the next word x_{t+1} as a label for the prediction at time step t

$$p(x_{t+1}|x_t, \dots x_1) \approx p(x_{t+1}|x_t, h_t)$$
$$= p(x_{t+1}| \underbrace{h_{t+1}}_{f(x_t, h_t)})$$

- **input** sequence: `The Time Machine by H.`
- **label** sequence: `Time Machine by H. G.`



One-hot encoding vectors map each word to a different unit vector:

- Let the number of different tokens in the vocabulary be N (the `len(vocab)`)
- Each token has a one-to-one correspondence with a single index value from 0 to $N-1$.
- For each token with index i, we create a vector $\mathbf{e}_i^\top = \mathbf{0}_N$ and set the element at position i to 1.

Example

`The monkey in the bar`

- **encoding**: 1-hot encoded row of $\mathbf{X_t}$
  (e.g. mapping the token `the` to $\mathbf{e}_2^\top = (0, 0, 1, 0, \ldots)$
- **embedding**: i-th row of $\mathbf{W}$) corresponding to token i.

$$\mathbf{e}_2^\top \mathbf{W} = (0, 0, 1, 0, \ldots) \begin{bmatrix} \mathbf{w}_0^\top \\ \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \vdots \end{bmatrix} = \mathbf{w}_2^\top$$

Summary

- A network that uses recurrent computation is called a recurrent neural network (RNN).
- The hidden state of the RNN can capture historical information of the sequence up to the current time step.
- The number of RNN model parameters does not grow as the number of time steps increases.
- We can create language models using RNNs.
- One-hot encodings separate the **encoding** of a token from its **embedding**