Sequence Modeling
Sequence Model

- Models that can handle sequences as input, as output, or both;
- Examples of sequence:
  - Language
  - Speech
  - DNA
  - Protein
  - ...
- Usually we use \( x = [x_1, x_2, \ldots, x_n] \) to denote a sequence.
Language Modeling

- **Goal**: model a probability distribution over pieces of text:
  - $P(w_1, w_2, \ldots, w_t)$
- Chain Rule: $P(x, y) = P(x)P(y|x)$

\[
P(w_1, w_2, \ldots, w_T) = P(w_1)P(w_2, \ldots w_T | w_1) \\
= P(w_1)P(w_2 | w_1)P(w_3, \ldots, w_T | w_1, w_2) \\
= \prod_{t=1}^{T} P(w_t | w_1, w_2, \ldots, w_{t-1})
\]
Language Modeling

\[ P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P_{\theta}(w_t | w_1, w_2, \ldots, w_{t-1}) \]

Can we estimate these instead?

\[ P_{\theta}(w_t | w_1, w_2, \ldots, w_{t-1}) = \frac{\#(w_1, w_2, \ldots, w_t)}{\#(w_1, w_2, \ldots, w_{t-1})} \]

Next word

Context

Number of times this sequence appears in the corpus
Markov Assumption

\[ P(w_t | w_1, w_2, \ldots, w_{t-1}) \approx P(w_t | w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}) \]

Probability of a word only depends on the last \( n-1 \) words
Feedforward Neural Network

- Perceptron: $y = w^T x + b$
Feedforward Neural Network

\[ h_1 = g(W_1 f(x) + b_1) \]
\[ h_2 = g(W_2 h_1 + b_2) \]
\[ \ldots \]
\[ \Psi_{NN} = W_n h_{n-1} + b_n \]

This is a *n*-layer feedforward network (also called a multi-layer perceptron)
Activation Function

- Non-linear functions applied element-wise

ReLU is often the default choice
Feedforward Neural Language Model

\[
f(x) = [L(w_{t-n+1}); \ldots; L(w_t)]
\]

\[
h = g(W_1 f(x) + b_1)
\]

\[
s = W_2 h + b_2
\]

\[
p = \text{softmax}(s)
\]

Bengio et al., 2003
Loss Function

- **Idea**: minimize negative log-likelihood
  \(-\log(P(w_t|w_{t-n+1}, \cdots, w_{t-1}))\)
- Normalize the output of the last layer with Softmax:

\[
\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}
\]
Gradient Descent

Algorithm 5 Generalized gradient descent. The function BATCHER partitions the training set into $B$ batches such that each instance appears in exactly one batch. In gradient descent, $B = 1$; in stochastic gradient descent, $B = N$; in minibatch stochastic gradient descent, $1 < B < N$.

1: procedure GRADIENT-DESCENT($x^{(1:N)}, y^{(1:N)}, L, \eta^{(1...∞)}, \text{BATCHER}, T_{\text{max}}$)
2: \hspace{1cm} $\theta \leftarrow 0$
3: \hspace{1cm} $t \leftarrow 0$
4: \hspace{1cm} repeat
5: \hspace{1.5cm} $(b^{(1)}, b^{(2)}, \ldots, b^{(B)}) \leftarrow \text{BATCHER}(N)$
6: \hspace{1.5cm} for $n \in \{1, 2, \ldots, B\}$ do
7: \hspace{2cm} $t \leftarrow t + 1$
8: \hspace{2cm} $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta^{(t)} \nabla_{\theta} L(\theta^{(t-1)}, x^{(b^{(1)}), b^{(2)}, \ldots}, y^{(b^{(1)}), b^{(2)}, \ldots})$
9: \hspace{2cm} if Converged($\theta^{(1,2,\ldots,t)}$) then
10: \hspace{2.5cm} return $\theta^{(t)}$
11: \hspace{2cm} until $t \geq T_{\text{max}}$
12: return $\theta^{(t)}$
Backpropagation

Chain rule (of calculus): \( f(x) = u(v(x)) \Rightarrow \frac{\partial f}{\partial x} = \frac{\partial u}{\partial v} \frac{\partial v}{\partial x} \)

\[
\begin{align*}
  h &= g(W_1 f(x_i) + b_1) \\
  \Psi_{NN} &= W_2 h + b_2 \\
  p &= \text{softmax}(\Psi_{NN}) \\
  \ell_i &= -\log p(y^{(i)} | x^{(i)} ; \theta) = -\log p_{y^{(i)}} \\
  \nabla_{b_2} \ell_i &= \frac{\partial \ell_i}{\partial p} \frac{\partial p}{\partial \Psi_{NN}} \frac{\partial \Psi_{NN}}{\partial b_2} \\
  \nabla_{W_2} \ell_i &= \frac{\partial \ell_i}{\partial p} \frac{\partial p}{\partial \Psi_{NN}} \frac{\partial \Psi_{NN}}{\partial W_2} \\
  \nabla_{b_1} \ell_i &= \frac{\partial \ell_i}{\partial p} \frac{\partial p}{\partial \Psi_{NN}} \frac{\partial \Psi_{NN}}{\partial h} \frac{\partial h}{\partial b_1} \\
  \nabla_{W_1} \ell_i &= \frac{\partial \ell_i}{\partial p} \frac{\partial p}{\partial \Psi_{NN}} \frac{\partial \Psi_{NN}}{\partial h} \frac{\partial h}{\partial W_1}
\end{align*}
\]
Feedforward Neural Language Model

- Note that there is a fixed input size n-1
  - Why? Because MLP can only handle fixed size input
- Can we model sequences of any length?
  - Recurrent neural network
  - Transformer
Recurrent Neural Network

\[ h_t = g(Wx_t + Uh_{t-1} + b) \]

The number of parameters is independent of input length!

In principle, information can propagate long distances.

Computation is slow since it needs to be done sequentially.
RNN Language Model

- log \( y_a \)  
- log \( y_{\text{hole}} \)  
- log \( y_{\text{in}} \)  
- log \( y_{\text{the}} \)  
- log \( y_{\text{ground}} \)  

Input Embeddings

 soften over Vocabulary

RNN Layer(s)

Next word
Problems with RNN

- The entire word history is represented by one (or two) vectors
- We cannot parallelize computation across different words
- **Solution:** Transformer Networks
  - [https://drive.google.com/file/d/1x-Xr150w6ddOskWw2bvlZVIq_YjqAEM/view](https://drive.google.com/file/d/1x-Xr150w6ddOskWw2bvlZVIq_YjqAEM/view)
References

Duke CS 572, Introduction to NLP, Bhuwan Dhingra,
https://sites.duke.edu/compsci_572_01_f22/schedule/