Admin

Final project proposals due today

Start working on the projects!
- Log hours that you work

Mentor hours this week?

---

**A Single Neuron/Perceptron**

Each input contributes: $x_i \cdot w_i$

$\text{in} = \sum w_i x_i$

$\sum g(\text{in})$ threshold function

**Activation functions**

- **hard threshold:**
  
  $g(\text{in}) = \begin{cases} 
  1 & \text{if } \text{in} \geq T \\
  0 & \text{otherwise} 
  \end{cases}$

- **sigmoid**
  
  $g(x) = \frac{1}{1 + e^{-x}}$

- **tanh x**
Many other activation functions

- Rectified Linear Unit
- Softmax (for probabilities)

Neural network

Neural network inputs: some inputs are provided/entered

Each perceptron computes and calculates an answer.
Neural networks inputs those answers become inputs for the next level.

Recurrent neural networks inputs hidden layer(s) output.

$x_t =$ input
$h_t =$ hidden layer output
$y_t =$ output

Figure 9.1 from Jurafsky and Martin.
Recurrent neural networks

$x_i = \text{input}$

$h_{t-1} = \text{hidden layer output from previous input}$

$h_t = \text{hidden layer output}$

$y_t = \text{output}$

Figure 9.2 from Jurafsky and Martin

Say you want the output of $x_1, x_2, x_3, ...$
Recurrent neural networks

Figure 9.2 from Jurafsky and Martin
Still just a single neural network

\[ x_t = \text{input} \]
\[ h_{t-1} = \text{hidden layer output from previous input} \]
\[ h_t = \text{hidden layer output} \]
\[ y_t = \text{output} \]

RNN language models

U, W and V are the weight matrices

How can we use RNNs as language models \( p(w_1, w_2, \ldots, w_n) \)?
How do we input a word into a NN?

"One-hot" encoding

For a vocabulary of \( V \) words, have \( V \) input nodes

All inputs are 0 except the one corresponding to the word

apple

\[ x_t \]
RNN language model

Figure 9.6 from Jurafsky and Martin

Softmax = turn into probabilities

25

26

Training RNN LM

Figure 9.6 from Jurafsky and Martin
**Generation with RNN LM**

- Sampled Word
- Softmax
- RNN
- Embedding
- Input Word

Figure 9.9 from Jurafsky and Martin

**Stacked RNNs**

- Multiple hidden layers
- Still just a single network run over a sequence
- Allows for better generalization, but can take longer to train and more data!

Figure 9.10 from Jurafsky and Martin

**Challenges with RNN LMs**

- What context is incorporated for predicting $w_i$?
Challenges with RNN LMs

Just like with an n-gram LM, only use previous history.

What are we missing if we’re predicting $p(w_1, w_2, \ldots, w_n)$?

Bidirectional RNN

Figure 9.11 from Jurafsky and Martin

Normal forward RNN

Bidirectional RNN

Normal forward RNN

Backward RNN, starting from the last word

Figure 9.11 from Jurafsky and Martin
Bidirectional RNN

Prediction uses collected information from the words before (left) and words after (right)

Figure 9.11 from Jurafsky and Martin

Challenges with RNN LMs

Can we use them for translation (and related tasks)?

Any challenges?

Challenges with RNN LMs

Can we use them for translation (and related tasks)?

Any challenges?

Challenges with RNN LMs

Translation isn’t word-to-word

Worse for other tasks like summarization
Encoder-decoder models

Idea:
- Process the input sentence (e.g., sentence to be translated) with a network
- Represent the sentence as some function of the hidden states (encoding)
- Use this context to generate the output

Encoder-decoder models: simple version

The context is the final hidden state of the encoder and is provided as input to the first step of the decoder

Encoder-decoder models: improved

The context is some combination of all of the hidden states of the encoder

How is this better?

Encoder-decoder models: improved

Each step of decoding has access to the original, full encoding/context
Encoder-decoder models: improved

Even with this model, different encoding steps may care about different parts of the context.

Attention

Context is dependent on where we are in decoding step and the relationship between encoder and decoder hidden states.

Simple version attention is static, but can learn attention mechanism (i.e., relationship between encoder and decoder hidden states).
Key RNN challenge: computation is sequential
- This prevents parallelization
- Harder to model contextual dependencies

How is this setup different from the RNN?

Another model

Do not rely on the hidden states for context information
Parallel: computation can all happen at once

Self-attention

Self-attention:
- Input is some context (for LMs, the previous words)
- Learn what parts of the context are important based
**Self-attention**

![Self-attention Diagram](image1.png)

Figure 10.1 from Jurafsky and Martin

**Transformer block**

![Transformer block Diagram](image2.png)

Figure 10.4 from Jurafsky and Martin

**Transformer network**

![Transformer network Diagram](image3.png)

Transformer network vs. RNN

![Transformer network vs. RNN Diagram](image4.png)
GPT

- Generative: outputs things
- Pre-trained: previously trained on a large corpus
- Transformer: uses the transformer network

Pre-trained language models

- Pre-trained language models are general purpose and are trained on a very large corpus
- They can be used as-is to:
  - Ask $p(w_1 \mid w_2 \ldots w_n)$
  - Generate text given some seed, $p(w_i \mid w_1 w_2 \ldots w_{i-1})$
- They can also be "fine-tuned" for particular tasks: take the current weights and update them based on a specific application

ChatGPT

ChatGPT is an artificial intelligence (AI) chatbot developed by OpenAI and released in November 2022. It is built on top of OpenAI’s GPT-3.5 and GPT-4 families of large language models (LLMs) and has been fine-tuned (an approach to transfer learning) using both supervised and reinforcement learning techniques.

ChatGPT

The fine-tuning process leveraged both supervised learning as well as reinforcement learning in a process called reinforcement learning from human feedback (RLHF).[7][8] Both approaches use human trainers to improve the model’s performance. In the case of supervised learning, the model was provided with conversations in which the trainers played both sides: the user and the AI assistant. In the reinforcement learning step, human trainers first ranked responses that the model had created in a previous conversation.[9] These rankings were used to create "reward models" that were used to fine-tune the model further by using several iterations of Proximal Policy Optimization (PPO).[7][9]