Tools & Models for Data Science

Deep Neural Networks (3): RNN

Chris Jermaine & Risa Myers

Rice University

Data

? What kinds of input have we considered so far (in the course)?

Data

- What kinds of input have we considered so far (in the course)?
  - Rx counts and costs
  - Genome sequences
  - PubMed abstracts
  - Clinical trial descriptions
  - Books (Sherlock Holmes, War and Peace, Shakespeare)
  - Wikipedia newsgroups

? How have we represented the text data?

Data Representation

- What kinds of input have we considered so far (in the course)?
  - Rx counts and costs
  - Genome sequences
  - PubMed abstracts
  - Clinical trial descriptions
  - Books (Sherlock Holmes, War and Peace, Shakespeare)
  - Wikipedia newsgroups
- How have we represented the text data?
  - Bag of words

? What are the limitations of this representation?

Issues with Data Representation

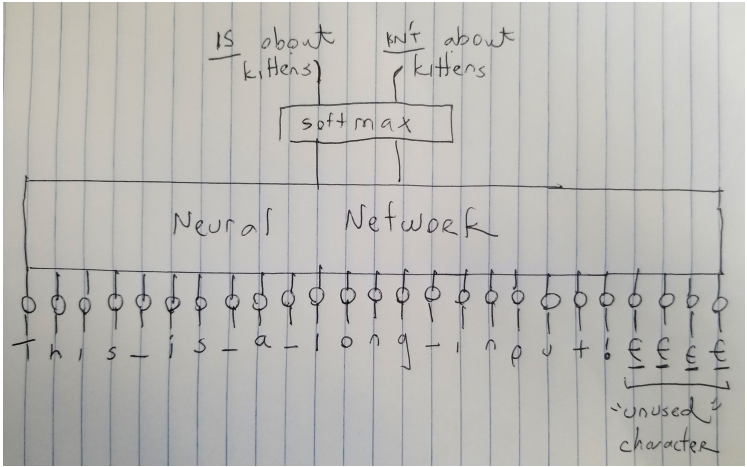
- What kinds of input have we considered so far (in the course)?
  - Rx counts and costs
  - Genome sequences
  - PubMed abstracts
  - Clinical trial descriptions
  - Books (Sherlock Holmes, War and Peace, Shakespeare)
  - Wikipedia newsgroups
- How have we represented the text data?
  - Bag of words
- What are the limitations of this representation?
  - Lose the order/context

Consider our Simple Feed-Forward Networks

- They don't easily handle sequences
- What if I want to classify text docs?
  - And I don't want to do the bag-of-words thing
  - After all: bag-of-words loses word order
  - ? What are my options?

Standard Idea

- Use FF network with enough input units (e.g. 20,000 words)
  - To handle any document in training/testing
  - Pad unused tokens with a special character



Issues with Standard idea

- High model complexity
  - Max n input tokens
  - Size m first hidden layer
  - Means $n \times m$ weights to learn

$$10^4 \times 10^5 = 10^9 = 1\text{B weights}$$

? What if max tokens is 100K, average is 1000?

Issues

- High model complexity
  - Max n input tokens
  - Size m first hidden layer
  - Means $n \times m$ weights to learn
  - What if max tokens is 100K, average is 1000?
    - We are spending effort learning a network that is too big
    - Not much training data for right-most inputs
  - ? What other issues are there?

Issues

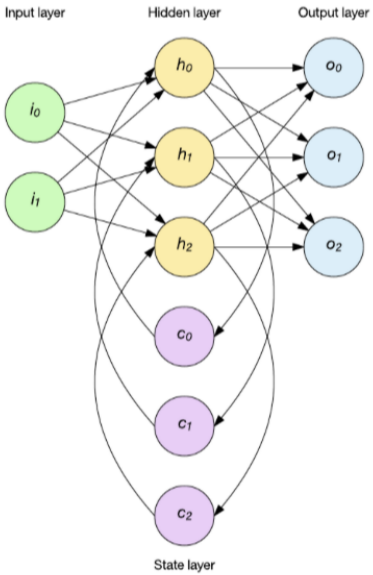
- High model complexity
  - Max n input tokens
  - Size m first hidden layer
  - Means $n \times m$ weights to learn
  - What if max tokens is 100K, average is 1000?
    - We are spending effort learning a network that is too big
    - Not much training data for right-most inputs
  - ? What other issues are there?
    - We can't handle bigger lengths
    - We have to use all the inputs
    - Words positions are considered equivalent

Words Position Issues

- Position i treated as different from position j
  - Not always the case!
  - If we see “kitten” at pos 34 or pos 1034, it is perhaps the same
  - We want to recognize pattern “kitten” regardless of position
  - Or consider swapping the order of 2 paragraphs on Wikipedia. No one would notice!

So Idea Is To Allow Recursive Links

- Results in an RNN
- Classic RNN is Elman Network
- The output from the hidden layer is pushed to the output AND copied to the state layer
- Input from context neurons is fed in with the input data
- There are NO weights from the neurons output to the states
- There ARE weights from the states to the neuron inputs



Elman Network

- Have a set of context nodes (the "state layer") (“c” neurons in previous slide)
- They read value of the hidden layer
  - Non-trainable (e.g. values are not transformed)
  - Value simply remembered for one time tick
- To process t ticks of data:

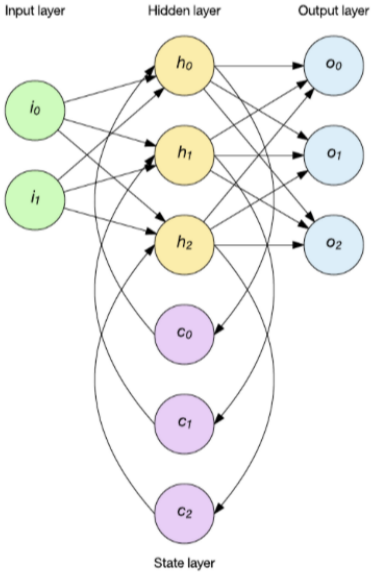
```
init state layer to zeros
for i = 1 to t
  read input x_i
  x_i along with state layer used to update hidden layer
  if (i == t) // for sequence-to-sequence, omit "if clause"
    hidden layer used to produce output
  hidden layer copied to state layer
end for
```

Sequence-to-Sequence?

- Input is a sequence
- Output is a sequence
- Output is generated at each time tick
- Lengths do not have to be identical
- Examples
  - Machine translation
  - Creating captions for video
  - Questions and answers

Elman Network in Practice

- Concatenate x_i and the state values
- Perform the matrix multiplication to get the inputs to the next layer
- At the top, use only the hidden layer to product the output

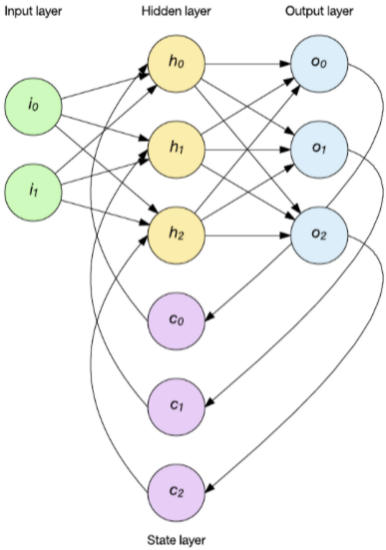


Elman Network

- We can have many hidden layers
- That is, a “deep net”
- In this case
  - Last hidden layer output copied to state
  - State used as input to first hidden layer...
  - In next time tick

Jordan Network

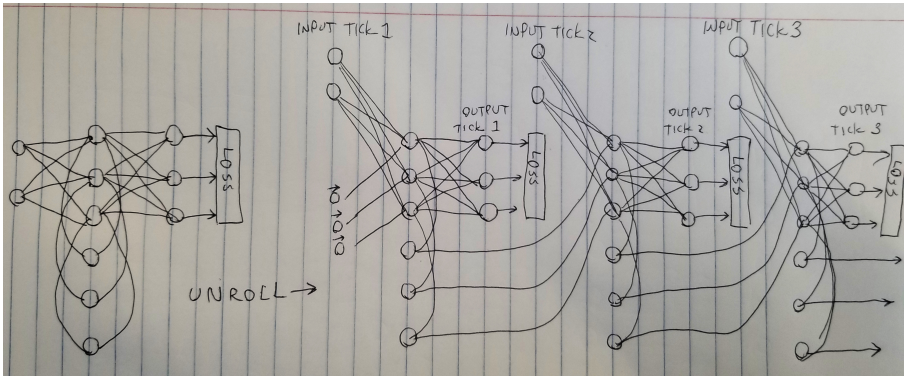
- Similar, but copy output values, not hidden values
- Can be used for sequence-to-sequence
- Must be producing output at each tick



Training

- Classic algorithm is back-propagation through time
- That is, view RNN as compact representation for a complex graph
- Unroll the complex graph
- And then use back-propagation on that
  - Key difference from classic back-propagation:
  - Weights are constrained to repeat

Example: Unrolling an Elman Network



- Example of unrolling a network for three time ticks
- Note distance backpropped error from last time tick must travel
  - Goes through output neurons at time tick 3
  - Through hidden neurons time tick 3
  - Through hidden neurons time tick 2
  - Through hidden neurons time tick 1

Training Difficulty: Vanishing Gradient

- Errors fall off exponentially as backpropped thru layers
  - Problem is that derivative of loss wrt activation function often $<< 1$
  - Repeatedly multiplying causes backpropped errors to tend to zero
  - Happens with deep, feed-forward nets, too
  - But unrolled RNNs are often especially deep

Training Difficulty: Vanishing Gradient

- Errors fall off exponentially as backpropped thru layers
  - Problem is that derivative of loss wrt activation function often $<< 1$
  - Repeatedly multiplying causes backpropped errors to tend to zero
  - Happens with deep, feed-forward nets, too
  - But unrolled RNNs are often especially deep
- Means that in a deep net...
  - ...backprop does not affect weights in first (leftmost) few layers

Training Difficulty: Vanishing Gradient

- Especially a problem if there is just one output at end of unrolled RNN
- Like in a pure classification task
  - Means that you will learn to classify
  - ...using only the last few time-tick's worth of data
  - Because early data can't interact with backpropped error

Other Issues with RNNs

- Very expensive to train
- Still expensive to use

LSTMs

- Long Short Term Memory networks
  - Special RNN designed to deal with vanishing gradient problem
  - In LSTM, long term memory is not pushed through activation functions
  - So we don't have vanishing gradients
  - Will discuss next time!

Questions?