Tools & Models for Data Science

Deep Neural Networks (3): RNN

Chris Jermaine & Risa Myers

Rice University

Objectives

- Learn about another neural network architecture
- Learn about the types of problems which are well-suited to it

Data

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

Data Representation

- How have we represented the text data?
  - Bag of words

? What are the limitations of this representation?

Issues with Data Representation

- How have we represented the text data?
  - Bag of words
- What are the limitations of this representation?
  - Lose the order/context
  - Fixed dictionary

Consider our Simple Feed-Forward Networks

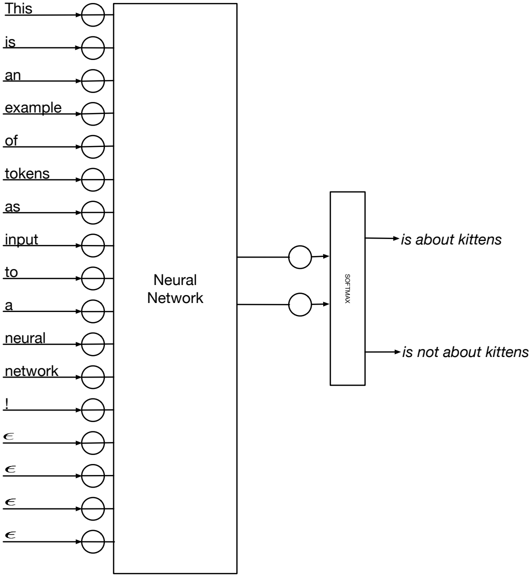
- They don’t easily handle sequences
- What if I want to classify text docs?
  - And I don’t want to do the bag-of-words thing
  - After all: bag-of-words loses word order
  - ? What are my options?

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?
    - Sequence of tokens

Standard Idea

- Use FF network with enough input units (e.g. 20,000 tokens or 20,000 characters)
  - To handle any document in training
  - 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?

? What other issues are there?

Other issues

- We are spending effort learning a network that is too big
- Likely not much training data for right-most inputs
- We can't handle bigger lengths (training or test)
- We have to use all the inputs

Token position issue

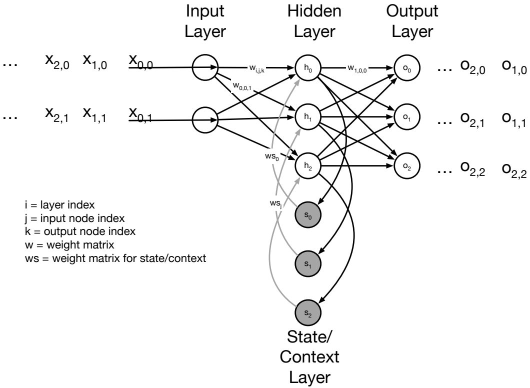
- Position i treated as different from position j
  - Not always the case!
  - If we see “kitten” at pos 34 or pos 1034, it is perhaps the same
  - We want to recognize pattern “kitten” regardless of position
  - Or consider swapping the order of 2 paragraphs on Wikipedia. No one would notice!
    -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

FFs on Fixed Length Sequences of Tokens

- Don't work well
- Alternatives?
  - Recurrent Neural Networks (RNNs)
- How do these help?
  - Add extra nodes that preserve context (order, state)
  - Incorporated via extra connections that cyclically link layers

What does an RNN look like?

- 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 hidden layer neurons output to the states
- There ARE weights from the states to the neuron inputs
- There is a 1-1 correspondence between the saved state and each hidden node



Elman Network

- Has a set of context nodes (the "state layer") ("s" 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 value of states in context/state layer to zeros
for i = 1 to t
  read input x_i
  update hidden layer using x_i along with state 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
```

RNN applications

Data

1. Video data
2. Voice
3. Text
4. Other time sequence
    - Stock values
    - Temperatures

Input

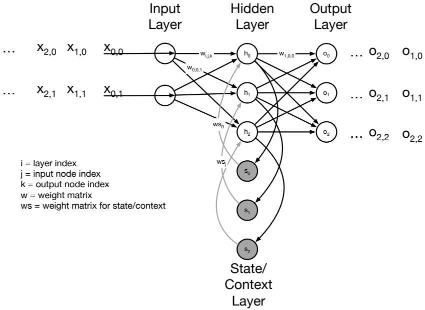
1. Sequence of images
2. Sequence of signal values
3. Sequence of characters/tokens
4. Sequence of [multidimensional] numeric values

Output

- Also a sequence
- Generated at each time tick
- Examples
    1. Video captions
    2. Translation
    3. Parts of speach
    4. Classification

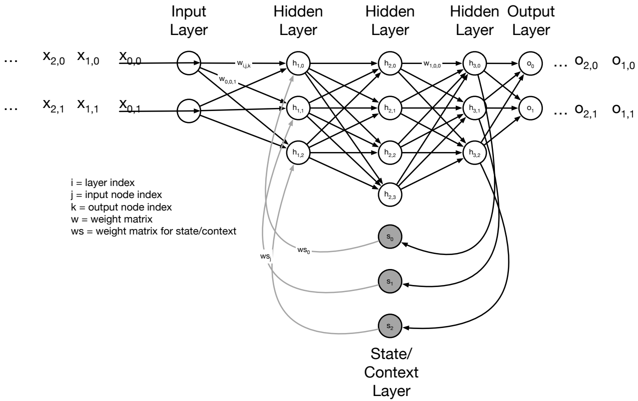
Elman Network in Practice

- 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



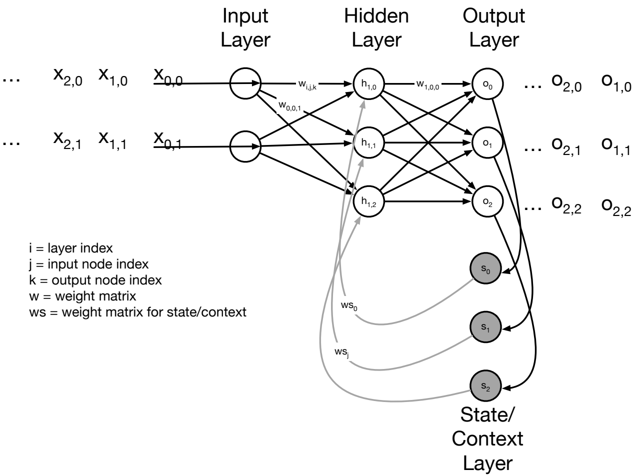
Elman Network

- 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

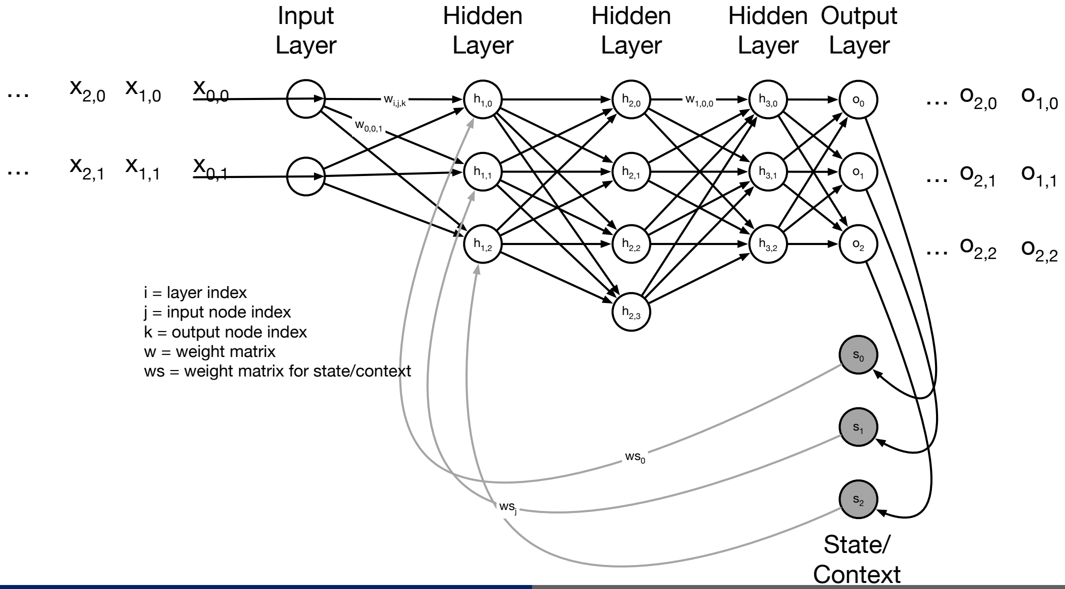


Jordan Network

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



Deep Jordan Network

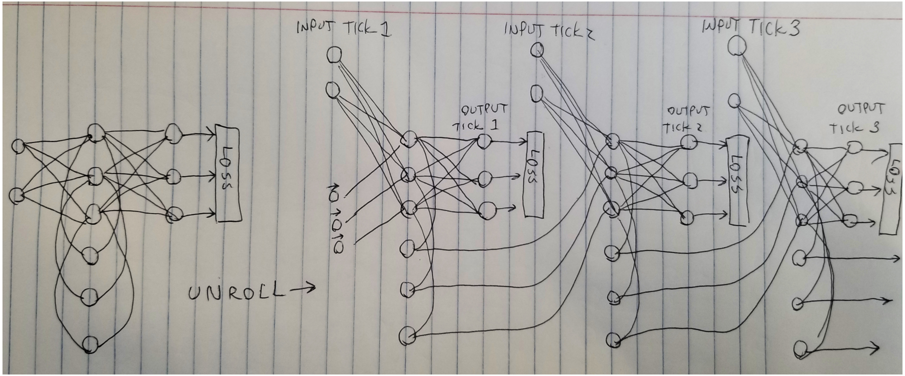


i = layer index
j = input node index
k = output node index
w = weight matrix
ws = weight matrix for state/context

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 much 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
  - Starting or ending data ends of being prioritized (depending on order data is fed in)
  - Mitigation
    - Batch normalization - normalize inputs to each layer

Problem With RNNs

- The "vanishing gradient" problem
- During back-propagation
  - Update magnitude drops exponentially with distance from output
  - Recall

$$\frac{\partial L}{\partial w_{i,j,k}} = \frac{\partial L}{\partial y_{i,k}} \frac{\partial y_{i,k}}{\partial v_{i,k}} \frac{\partial v_{i,k}}{\partial w_{i,j,k}}$$

  - If activation function is logistic function σ then $\frac{\partial y_{i,k}}{\partial v_{i,k}} = \sigma(v_{i,k})(1 - \sigma(v_{i,k}))$
  - Means you have a multiplier that maxes out at 0.25
  - Max value is when $v_{i,k} = 0.5$
  - $\sigma(0.5) = 0.25$
  - As you back-propagate, you keep multiplying by this in DP... $.25 \times .25 \times .25 \ldots$ gradient gets tiny
- Result is that output at time tick t...
  - ...has little interaction with input at tick $t - 100$ during back-propagation
  - Practically speaking: means back-propagation has limited-duration memory

Other Issues with RNNs

- Very expensive to train
- Still expensive to use
- Alternatives?
  - Long Short Term Memory networks (LSTMs)

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
  - `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Tokens or characters?

- Depends on what you are trying to learn
- Tokens
  - Can stem words / tokens
  - Fewer parameters needed
  - Lower computational cost
- Characters
  - Smaller input dataset
  - Little to no preprocessing needed
  - Better for some languages (morphologically rich)
  - Grammar is reflected more in words than in position

Wrap up

? How can we use what we learned today?

- What RNNs are and how they work

? What do we know now that we didn't know before?

- How to handle sequential data in a neural network