Time Series Analysis with Recurrent Neural Networks (RNNs), and Roughly How Learning a Deep Net Works

George Chen
It’s Gauss’s birthday

One of the original “Al” researchers
Time series analysis with Recurrent Neural Networks (RNNs)
RNNs

What we’ve seen so far are “feedforward” NNs
RNNs

What we’ve seen so far are “feedforward” NNs

What if we had a video?
RNNs

Feedforward NN’s: treat each video frame separately

Time 0

Time 1

Time 2
RNNs

Feedforward NN’s: treat each video frame separately

RNN’s: feed output at previous time step as input to RNN layer at current time step

In *keras*, different RNN options: *SimpleRNN*, *LSTM*, *GRU*

Recommendation: don’t use *SimpleRNN*
RNNs

Feedforward NN’s: treat each video frame separately

RNN’s: feed output at previous time step as input to RNN layer at current time step

In *keras*, different RNN options: *SimpleRNN*, *LSTM*, *GRU*

Recommendation: don’t use *SimpleRNN*
Under the Hood

current_state = 0

for input in input_sequence:
    output = f(input, current_state)
    current_state = output

Different functions f correspond to different RNNs
Example: SimpleRNN

memory stored in current_state variable!

```python
current_state = 0
for input in input_sequence:
    output = activation(np.dot(W, input) + np.dot(U, current_state) + b)
    current_state = output
```

Activation function could, for instance, be ReLU

Parameters: weight matrices W & U, and bias vector b

Key idea: it’s like a dense layer in a for loop with some memory!
RNNs

Feedforward NN’s: treat each video frame separately

RNN’s:
feed output at previous time step as input to RNN layer at current time step

RNN layer

readily chains together with other neural net layers

like a dense layer that has memory

Time series

In keras, different RNN options:
SimpleRNN, LSTM, GRU

Recommendation:
don’t use SimpleRNN
RNNs

Feedforward NN’s: treat each video frame separately

RNN’s: feed output at previous time step as input to RNN layer at current time step

In *keras*, different RNN options: *SimpleRNN*, *LSTM*, *GRU*

Recommendation: don’t use *SimpleRNN*
RNNs

Feedforward NN’s: treat each video frame separately

RNN’s: feed output at previous time step as input to RNN layer at current time step

readily chains together with other neural net layers

like a dense layer that has memory

In keras, different RNN options: SimpleRNN, LSTM, GRU

Recommendation: don’t use SimpleRNN
RNNs

Example: Given text (e.g., movie review, Tweet), figure out whether it has positive or negative sentiment (binary classification).

Common first step for text: turn words into vector representations that are semantically meaningful.

In `keras`, use the `Embedding` layer.

Classification with > 2 classes: dense layer, softmax activation.

Classification with 2 classes: dense layer with 1 neuron, sigmoid activation.
RNNs

Demo
RNNs

- Neatly handles time series in which there is some sort of global structure, so memory helps
- If time series doesn’t have global structure, RNN performance might not be much better than 1D CNN
- An RNN layer by itself doesn’t take advantage of image/text structure!
  - For images: combine with convolution layer(s)
  - For text: combine with embedding layer
A Little Bit More Detail

Simple RNN: has trouble remembering things from long ago...

Figure 6.13 from Francois Chollet's book *Deep Learning with Python*
A Little Bit More Detail

Introduce a “carry” state for tracking longer term memory
A Little Bit More Detail

LSTM: figure out how to update “carry” state

Figure 6.15 from Francois Chollet’s book *Deep Learning with Python*
Learning a Deep Net
Gradient Descent

Suppose the neural network has a single real number parameter $w$

The skier wants to get to the lowest point

The skier should move rightward (positive direction)

The derivative $\frac{\Delta L}{\Delta w}$ at the skier’s position is negative

In general: the skier should move in opposite direction of derivative

In higher dimensions, this is called gradient descent
(derivative in higher dimensions: gradient)
Gradient Descent

Suppose the neural network has a single real number parameter $w$. 

![Graph showing gradient descent](image)
Gradient Descent

Suppose the neural network has a single real number parameter $w$
Gradient Descent

Suppose the neural network has a single real number parameter $w$. 

![Graph showing gradient descent](image)
Gradient Descent

Suppose the neural network has a single real number parameter $w$

In general: not obvious what error landscape looks like! → we wouldn’t know there’s a better solution beyond the hill

Popular optimizers (e.g., RMSprop, ADAM, AdaGrad, AdaDelta) are variants of gradient descent

In practice: local minimum often good enough
Gradient Descent

2D example

$L(w)$
Remark: In practice, deep nets often have $> \textit{millions}$ of parameters, so very high-dimensional gradient descent
Handwritten Digit Recognition

Training label: 6

28x28 image

A neural net is a function composition!

All parameters: \( \theta \)

Gradient: \( \frac{\partial}{\partial \theta} \frac{1}{n} \sum_{i=1}^{n} L(f_2(f_1(x_i)), y_i) \)

Automatic differentiation is crucial in learning deep nets!

Careful derivative chain rule calculation: back-propagation
Gradient Descent

We have to compute lots of gradients to help the skier know where to go!

Computing gradients using all the training data seems really expensive!

average loss

compute gradient and move skier
Stochastic Gradient Descent (SGD)

SGD: compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)
Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD): compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)
Stochastic Gradient Descent (SGD)

SGD: compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)
Stochastic Gradient Descent (SGD)

Training example 1 → Neural net → loss 1
Training example 2 → Neural net → loss 2
Training example 3 → Neural net → loss 3
Training example 4 → Neural net → loss 4
Training example 5 → Neural net → loss 5
... → Neural net → loss n

compute gradient and move skier

SGD: compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)
Stochastic Gradient Descent (SGD)

SGD: compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)
Stochastic Gradient Descent (SGD)

SGD: compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)
Stochastic Gradient Descent (SGD)

SGD: compute gradient using only 1 training example at a time (can think of this gradient as a noisy approximation of the “full” gradient)

An epoch refers to 1 full pass through all the training data
Mini-Batch Gradient Descent

Training example 1

Neural net

loss 1

Training example 2

Neural net

loss 2

Training example 3

Neural net

loss 3

Training example 4

Neural net

loss 4

Training example 5

Neural net

loss 5

…

Training example n

Neural net

loss n

average loss

compute gradient and move skier
Mini-Batch Gradient Descent

Batch size: how many training examples we consider at a time (in this example: 2)

Batch size: how many training examples we consider at a time (in this example: 2)