Move 37
Neural Networks Study Guide

Feed Forward Neural Networks

multilayer perceptron: historical name for a feed forward neural network, the fundamental algorithm for deep learning. It was modeled after the brain, from a 1950s level of understanding. We commonly represent the network as a graph of interconnected nodes.
Each layer (hidden layer) performs a set of multidimensional transformations of the data.
Each node (hidden unit) is comprised of a set of weights, which are adjusted through training to learn the mapping between sets of input and output.

backpropagation algorithm: allows for iterative improvements of weights, updating from last layer to first based on a loss (error) from the output.

learning rate: controls how extreme our adjustments are to the weights at each backward pass.

activation function: needed between each layer in order for our network to learn nonlinearity.

overfitting: a common problem in which our model is unable to generalize to new data.

Normalization: a technique used to mitigate overfitting.

Universal Approximation Theorem: a feedforward network with at least one hidden layer, a squashing (activation) function, and a fixed number of neurons, can approximate any continuous function to a desired level of precision.

Recurrent Neural Network (1987)

Generalization of Backpropagation to Recurrent and Higher Order Neural Networks

RNN: Neural network specialized for sequential data; uses a tanh activation function
Hidden Vector: used to represent the past state of the input; produced by combining the hidden state and the input at each timestep. Inputs accumulate into the hidden state over time.

$$h_t = fw(h_{t-1},\ x_t)$$

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$

(vanilla example)

Styles of RNN architecture:

- one to many: fixed size input; variable sized output.
  Example: Image Captioning
- many to one: variable sized input, fixed size output.
  Example: Sentiment Analysis
- many to many: variable sized input, variable sized output.
  Example: Machine Translation
- many to many (sync): variable sized input, variable sized output.
  Example: Video Captioning

backpropagation through time: can lead to vanishing and exploding gradients. One simple hack for vanishing/exploding gradients is gradient clipping

Long Short-Term Memory (1997)

Long Short-Term Memory

- reduces the vanishing/exploding gradients encountered by RNNs
- maintains a hidden state as well as a cell state

$c_t = f \odot c_{t-1} + i \odot g$

$h_t = o \odot tanh(c_t)$

- combines input and hidden state to make 4 gates:
  i: input, whether to write to a cell
  f: forget, whether to erase a cell
  o: output, how much to reveal a cell to the next layers
  g: gate, how much to write to a cell

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

(note: some sources call the g-gate $\tilde{C}$ for candidate)

for i, f, and o we use the sigmoid activation function; for g we use tanh instead

Gated Recurrent Unit (2014)

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

- reduces the vanishing/exploding gradients
- simplifies the LSTM by reducing 4 gates down to just 2

$z_t = \sigma(W_z x_t + U_z h_{t-1})$

$r_t = \sigma(W_r x_t + U_r h_{t-1})$

- Uses an update gate *z* to modulate between the current activation and the candidate activation

$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t$

- Uses a reset gate *r* to allow forgetting the previous state

$\tilde{h}_t = tanh(W_{\tilde{h}} x_t + U_{\tilde{h}}(r_t \odot h_{t-1}))$

Recent Models:

Hierarchical Multiscale Recurrent Neural Networks (2017)

LARNN: Linear Attention Recurrent Neural Network (2018)

On Extended Long Short-term Memory and Dependent Bidirectional Recurrent Neural Network (2018)

RNN explainability:

Visualizing and Understanding Recurrent Networks (2015)

LISA: Explaining Recurrent Neural Network Judgments via Layer-wIse Semantic Accumulation and Example to Pattern Transformation (2018)

Also see:

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling (2014)

Understanding LSTM Networks (2015)

An Empirical Exploration of Recurrent Network Architectures (2015)

Deep Learning: Feedforward Neural Network (2017)

Stanford CS231n Lecture 10 (2017) ← Very informative talk, be sure to check this out

Understanding Gated Recurrent Unit (GRU) Deep Neural Network