**Recurrent Neural Networks**

Source: IBM Cloud Education. (2020, September 14). *Recurrent Neural Networks*. IBM Cloud Learn Hub. Retrieved August 8, 2022, from https://www.ibm.com/cloud/learn/recurrent-neural-networks

* Recurrent neural network (RNN): type of artificial neural network that uses sequential or time series data
  + Uses: ordinal and temporal problems
  + Common uses: translation, NLP, speech recognition, and image captioning
  + Has built-in “memory” that uses prior inputs to influence the current i/o (i.e., makes no assumption that i/o are independent of each other)

Shape, circle

Description automatically generated

* + Works great when scenario must occur a very specific order (e.g., language)
* Traits
  + Parameters are shared across each layer of the network (unlike feedforward networks)
  + Leverages Backpropagation through time (BPTT) algorithm to determine gradients of backpropagation as specific to sequence data
    - Model trains itself by calculating errors from its output layers to its input layers
    - Sums the errors at each timestep (unlike feedforward networks)
  + Commonly has issues with exploding/vanishing gradients
    - Vanishing becomes when weight parameters are so insignificant that they are practically zero (i.e., no longer learning)
    - Exploding occurs when gradient is too large, and the model is destabilized
      * Best handled by reducing the number of hidden layers in the neural network
* Types of RNNs
  + One-to-one (what we will use)
  + One-to-many
  + Many-to-one
  + Many-to-many
* Activation functions
  + Sigmoid:
    - Inflection point around ½
  + Tanh:
    - Inflection point around 0
  + ReLU:
* RNN architectures
  + Bidirectional RNN (BRNN)
    - Pulls from both previous and future inputs to make predictions about its current state
  + Long short-term memory (LSMT)
    - Addresses vanishing gradient problem
    - If the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state
      * Essentially the farther the information that useful is from the current state, the less likely the model is going to be able to retain that info long enough to use it correctly
      * LSTM address this issue by having a cell of three gates--input, output, and forget gates--to control the flow of information
  + Gated recurrent units (GRUs)
    - Works similarly to LSTMs in addressing short-term memory problems
    - Uses hidden states instead of a cell with reset and update gates
      * Both gates are used to control how much and which information to retain

**Choosing an Activation Function**

Source: Brownlee, J. (2021, January 21). *How to choose an activation function for deep learning*. Machine Learning Mastery. Retrieved August 8, 2022, from https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

* Activation function: the weighted sum of inputs transformed into an output from a node or nodes in a layer of network
  + AKA transfer function, squashing function
  + Used within or after internal processing of each node of network
  + Activation function will dependent of layer: input, hidden, or output layers
  + Activation functions are typically differentiable in order to be able to be used with backpropagation
* Types
  + Rectified linear activation function (ReLU)

**Chart, line chart

Description automatically generated**

* + - Less susceptible to vanish gradients
    - Can have issues with saturated or dead units
    - Best practice to use normalization between 0 and 1 and ‘he normal’ weigh initialization
  + Sigmoid

Chart

Description automatically generated

* + - AKA logistic function (same as used in logistic regression classification algorithm)
    - Ranges values between 0 and 1
    - Also benefits from normalization range between 0 and 1 and “Xavier Normal” weight initialization
  + Tanh

Chart, line chart

Description automatically generated

* + - Use normalization between -1 and 1 and “Xavier Normal” weight initialization
* How to choose
  + Keep all layers with the same activation function
  + Start with sigmoid and work from there

**Diagram

Description automatically generated**

* Output layer activation functions
  + Options: linear, logistic, and softmax
    - Linear
      * Linear activation function does not change the weight sum of input any way
      * Not great for classification
    - Sigmoid
      * Same as hidden
    - Softmax
      * Outputs a vector of values that sum to 1.0 and can be interpreted as probabilities of class membership
  + Choice
    - Binary: one node, sigmoid
    - Multiclass: one node per class, softmax
    - Multilabel: one node per class, sigmoid
    - Regression: linear

**One-Hot Layers**

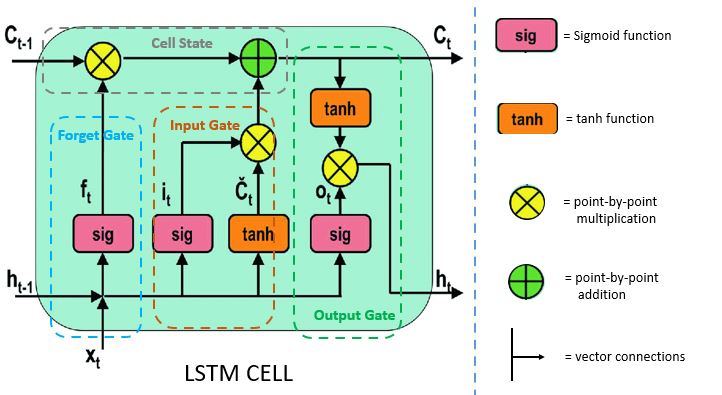
Source: Dalvi, F. (2018, April 7). *One-hot layer in Keras's sequential API*. Fahim Dalvi. Retrieved August 8, 2022, from https://fdalvi.github.io/blog/2018-04-07-keras-sequential-onehot/

* One-hot layer: take in values of previous layer and make the largest 1 and the rest 0
  + Best used after softmax layers, though could maybe improve performance between hidden layers

**LSTM Networks**

Source: GeeksForGeeks. (2021, June 25). *Understanding of LSTM networks*. GeeksForGeeks. Retrieved August 8, 2022, from https://www.geeksforgeeks.org/understanding-of-lstm-networks/

* Long Short-Term Memory
  + Advanced version of RNN designed to model chronological sequences and their long-range dependencies more precisely than RNN
  + Really targets exploding and vanishing gradients
  + Short-term memory: taking into consideration previous output (feedback) and storing its memory for a short period time
* Problems with traditional RNN
  + Failing to store information for longer periods of time (long-term dependencies)
  + No finer control over which part of context needs to be carried forward and how much the past needs to be forgotten
  + Exploding and vanishing gradients
    - LSTM specifically eliminates this with altering the training model
    - Backtracking: models specific method of minimizing loss by altering the weights that contribute to a given layer
    - Scaling effect: if some components from a previous layer are small, results obtained (gradient) will be even smaller
      * Can cause alterations of weights to become infinitely small, thus essentially not changing the weights at all >> vanishing gradient
      * Opposite occurs when increasingly large values from the previous layer are produced, making the gradient even bigger; eventually this spirals and dominates the target of DL model >> exploding gradient
  + Need for a finite number of states from beforehand are needed (such as HMM)
* Architecture
  + Differs from RNN in that hidden layer of LSTM is gated unit (cell)
  + LSTMs have three logistic sigmoid gates and one tanh layer
    - These gates limit the info passed through the cell, noting what info needs to be passed and what needs to be discarded
    - Output of gats range from 0 (reject all) and 1 (include all)
  + Each LSTM cell has three inputs and two outputs
    - Inputs
      * is the hidden state
      * is the cell state/memory
      * is the current datapoint or input
    - Outputs
      * is the previous cell state
      * is the previous hidden state
  + Conventional model



* + - Second sigmoid layer (input gate): takes in the input and previous hidden states to present new candidate states for memory
    - First sigmoid layer (forget gate): takes activates what part of the previous cell state to remember and is combined with the input gate
    - Third sigmoid layer (output gate): takes information from the input and information decided to remember to create an output
  + Model variants
    - Most target a way to reduce computational complexity
    - Gers and Schmidhuber peephole connections: allow gate layers to have knowledge about the cell state at every instance
    - Coupled input and forget gate that helped make decisions simultaneously
    - Gated recurrent unit (GRU): reduces the number of gates by using a combination of cell state and hidden state and also an update gate to merge forgotten and input gates

Diagram, schematic

Description automatically generated

* + Structural options
    - Multiple layers that feed into one another (B)
    - Recurrent projection layer (C)
* GRUs VS LSTMs
  + Have update (what knowledge is carried forward) and reset gates (determines how much knowledge needs to be forgotten (between two successive units))
  + Does not store cell state in any way; does not restrict amount of memory to which the next unit is exposed
    - LSTMs regulate the amount of info being included in the cell
* Drawbacks
  + Does not completely remove vanishing gradients
  + Require a lot of resources and time to get trained and ready for real world applications
  + Some models need memory for even longer time than LSTMs
  + Affected by random weight initialization
  + Prone to overfitting and difficult to apply dropout algorithm to curve the issue

**GRUs (and More LSTM)**

Source: Phi, M. (2020, June 28). *Illustrated guide to LSTM's and GRU's: A step by step explanation*. Medium. Retrieved August 8, 2022, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Diagram

Description automatically generated

* Reset gate decides what information to forget

Update gate determines what to remember