Deep Learning
Winter Holiday 2022
LinkedIn

1/1/23

- Open jupyter notebooks using command prompt

Open folder, conda activate deeplearning, jupyter notebooks

- Deep learning: neural networks w/ 3 or more layers
  - Imitates human decision making and information processing
  - Starts w/ random initialization and works toward right values via trial and error
- Linear Regression
  - Dependent & independent variables, slope, intercept
  - Y = ax+b
- Logistic Regression
  - binary model
  - Relation btw 2 or more variables
  - Output 0/1
  - Y, x, slope, intercept, f - activation function
  - Y = f(ax+b)d
  - F used to convert the continuous variable coming out of ax + b into a boolean value
- Perceptron: unit for learning in artificial neural network
  - Represents an algo for supervised learning for binary classification
  - Like the cell in human brain // a cell in the neural network

    - Based on logistic regression
    - Replace slope w/ weight, w and intercept w/ bias, b
    - Weights and biases are the parameters for a neural network
- Artificial neural networks (ANN) : networks of perceptrons
    - Perceptrons called nodes
    - Nodes organised as layers
    - Each node w/ its own weight. Biases and fs
    - Each node connected to all nodes in next layer (w/ exceptions)
    - Working of ANN:
        - Inputs (independent var) sent from input layer
        - Passed onto nodes in hidden layer
        - Each node computes output based on its weights, biases and f
        - Node output passed onto next layer
- Training an ANN
    - Model is represented by parameters and hyperparameters
    - training : determining optimal " to max accuracy
    - Inputs, weight and biases may be n-array
    - Process:
        - Use training data and create network architecture with intuition
        - Start w/ random values for wights and biases
        - Compute error in output
        - Adjust weights and biases to reduce errors
        - Also fine tune hyperparameters by adjusting layers, nose counts and others
        - Until error is an acceptable value
- The input layer
    - Vectors: ordered list of values
        - Used as inputs
        - A tuple

        - Usually defines as NUmPy array
        - Represents feature variables
    - Sample: an instance of a real world example (data set made of features (c) and samples (r)) and features individual attributes of a sample
    - Input preprocessing: features need to be converted to numeric representations
    -

Input Type	Preprocessing Needed
Numeric	Centering and scaling
Categorical	Integer encoding, one-hot encoding
Text	TF-IDF, embeddings
Image	Pixels – RGB representation
Speech	Time series of numbers

    - E.g raw data -> centered and scaled -> transposed (switch x and y in table) [optional]
- Hidden layers
    - Typically 2n nodes
    - Neural network's architecture is defined by no of layers and nodes
    - Fully connected
    - Each node "learns" smt abt the feature-target relationship and this knowledge is persisted in its weights and biases
    - Inc nodes and layers Inc accuracy (not always true)
    - Architecture decided via experimentation
- Weights and biases
    - Represent trainable parameters in ANN
    - At a layer level, weights and bias are handled as RAs

-
Layer	Inputs	Nodes	Weights	Biases
HL 1	3	4	12	4
HL 2	4	5	20	5
HL 3	5	3	15	3
Output	3	2	6	2
Total			**53**	**14**
  - 3 inputs, 4 nodes, therefore each node has 3 weight values = 12 weights overall. One bias value per node, so 4 biases
  - Weights and biases are maintained as a matrix as well as inputs & outputs
- Activation func
  - Determines if a node will propagate info onto the next layer
  - Therefore, filters noise and normalizes output
  - Converts output to non linear value bc of matrix multiplication
  - Critical in learning patterns in the model
  - **Popular Activation Functions**

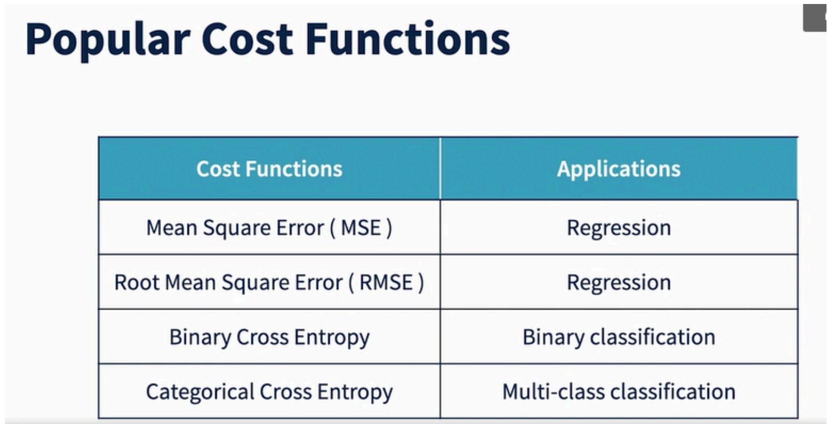
Activation Function	Output
Sigmoid	0 to 1
Tanh	-1 to +1
Rectified Linear Unit (ReLU)	0 if x < 0; x otherwise
Softmax	Vector of probabilities, with sum=1

    Choice depends on problem and experimentation.
- Output layer
  - Activation func maybe diff than the hidden layer
    - E.g. Softmax activation used for classification problems
  - Output may need further post-processing to convert to business values
  - Output layer size depends on problem
    - 1 for binary classification
    - N for n-class classification

        - 1 for regression problems

TRAINING A NEURAL NETWORK

- Input Preprocessing
    - Input data is split into:
        - Training set: used to fit/dtermine the parameters
        - Validation set: used for model selction/ fine tuning
        - Test set: used to measure the final model performance
        - 80:10:10 usual split of data
    - Need to select values for layers and nodes in layers, activation function and hyperparameters
        - Initial selection based on intuition
        - Adjusted based on results
    - Weights and bias parameters need to be initialized
        - 0 initialization (not recommended)
        - Random initialization: values from a std normal distribution (mean =0, SD =1)
- Forward Propagation
    - ^y (y w/ cap) (y hat) : prediction; y: actual
    - Send each sample thru neural network and obtain value of ^y
    - Repeat for all samples and collect a set of ^y
    - Compare values of ^y to y to obtain error rates
- Error in prediction
    - Loss and cost function
    - Loss: measures prediction error for a single sample
    - Cost: measures error across a set of samples

        - 

Cost Functions	Applications
Mean Square Error (MSE)	Regression
Root Mean Square Error (RMSE)	Regression
Binary Cross Entropy	Binary classification
Categorical Cross Entropy	Multi-class classification

    - Measuring accuracy:
        - Send a set of samples through ANN and predict outcome
        - Estimate prediction error btw predicted outcome and expected outcome using a cost function
        - Use back propagation to adjust weights based on the error value
- Back Propagation
    - Opp. of forward propogation
    - Start from output layer
    - Compute delta value based on error found
    - Apply delta to adjust weights and biases in the layer
    - Derive new error value
    - Back propagate new error to previous layer and repeat
- Gradient descent
    - Process of repeating forward and backward propagations in order to reduce error and move closer to the desired model
    - Repeat the learning process
        - Forward propagation
        - Estimate error
        - Backward propagate
        - Adjust weights and biases
- Batches and Epoch help control the number of passes during the learning process

- Batches: set of samples sent through ANN in a *single* pass
    - batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters.
    - Training data set can be divided into one or more batches
    - Training data is sent to the ANN one batch at a time
    - Cost estimates and parameters updates one batch at a time
    - 2 types:
        - Batch gradient descent: batch size = training set size
        - Mini-batch gradient descent: batch size < traing set size
- Epoch: no. of times the entire training set is sent through the ANN
    - Epoch has one or more batches
    - Training process completes when all epoch is complete
    - Epoch sizes can be higher for better accuracy
- Traning set size = 1000, batch size - 128, epoch =50
    - Batches per epoch = ceil (1000/128) = 8 // ceiling value
    - The last batch will have fewer samples than 128
    - Total iterartions (passes) through ANN = 8*50 = 400
        - Weights and biases updated 400 times

- Validation and testing
    - Validation: after each epoch and corresponding parameter updates, model can be used to predict for the validation data set
        - accuracy/loss can be measured and investigated
    - Evaluation: data set used to evaluate
- ANN model
    - Parameters: weights and biases
    - Hyperparameters: no. of layers, nodes in each layer, f, cost func, batch size, epoch

DEEP LEARNING EXAMPLE 1 - THE IRIS CLASSIFICATION PROBLEM

```
#Use a Label encoder to convert String to numeric values
#for the target variable

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
iris_data['Species'] = label_encoder.fit_transform(
                        iris_data['Species'])
```

-