Neural Networks

Packages for ML/Neural Networks

We use tensorflow because it has a good implementation for neural networks.

The caret package can do neural networks, but limited to one layer.

Neural Network Parameters

Many neural network parameters

- Nodes in Hidden layers
- Learning rate
- Dropout rate
- Batch size
- Epochs

Tune using cross-validation!

Avoiding Overfitting

Dropout rate: some proportion of nodes aren't used within a layer in a given pass through.

- Used to avoid overfitting
- Can think of it as (very roughly) analogous to random predictors in RF

Learning rate: How much the weights change with each update.

- Same as learning rate in Gradient Boosting/XGBoost.

Activation Functions

Activation functions **add non-linearity to neural networks**

- If they were all linear, you could calculate the linear form for the prediction using the weights.

Different activation functions add different types of non-linearity.

Typically, activation function is determined based on **type of neural network** (and type of outcome, for output activation function).

- Typically use ReLU for CNN, Sigmoid/Tanh for RNN.
- For numerical outcome, linear output activation function. Sigmoid/Logistic for Binary.

Convolutional vs Recurrent Neural Networks

Convolutional Neural Networks (CNN): Feed-forward neural network that only goes in one direction

Recurrent Neural Network (RNN): Uses outputs from previous time steps as inputs for current time step.

Normalizing/Scaling

In general: Normalizing or scaling features may or may not matter depending on the model being used.

- KNN needs it, while elastic net does not.

Recommendation is generally to scale because it doesn’t hurt even if it’s not necessary.

Computational Considerations

Neural networks are **very** computationally demanding.

Recurrent neural networks are more demanding due to needing to save information from previous inputs.

This is why large pre-built models exist!

Note: Tensorflow + Keras provides a way of using GPUs.