Deep Learning and Regularization

Technically, a deep Neural Network has 2 or more hidden layers (often, many more). Deep Learning involves Machine Learning with deep Neural Networks. However, the term Deep Learning is often used to broadly describe a subset of Machine Learning approaches that use deep Neural Networks to uncover otherwise-unobservable relationships in the data, often as an alternative to manual feature engineering. Deep Learning approaches are common in Supervised, Unsupervised, and Semi-supervised Machine Learning.

These are some common ways to prevent overfitting and regularize neural networks:

- Regularization penalty in cost function - This option explicitly adds a penalty to the loss function

$$J = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 + \lambda \sum_{j=1}^{m} W_i^2$$

- Dropout - This is a mechanism in which at each training iteration (batch) we randomly remove a subset of neurons. This prevents a neural network from relying too much on individual pathways, making it more robust. At test time the weight of the neuron is rescaled to reflect the percentage of the time it was active.
- Early stopping - This is another heuristic approach to regularization that refers to choosing some rules to determine if the training should stop.

Example:

Check the validation log-loss every 10 epochs.

If it is higher than it was last time, stop and use the previous model.

- Optimizers - This approaches are based on the idea of tweaking and improving the weights using other methods instead of gradient descent.

Details of Neural Networks

Training Neural Networks is sensitive to how to compute the derivative of each weight and how to reach convergence. Important concepts that are involved at this step:

Batching methods, which includes techniques like full-batch, mini-batch, and stochastic gradient descent, get the derivative for a set of points

Data shuffling, which aids convergence by making sure data is presented in a different order every epoch.

Keras

Keras is a high-level library that can run on either TensorFlow or Theano. It simplifies the syntax, and allows multiple backend tools, though it is most commonly used with TensorFlow.

This is a common approach to train a deep learning model using Keras:

1. Compile the model, specifying your loss function, metrics, and optimizer.

Fit the model on your training data (specifying batch size, number of epochs).

2.

Predict on new data.

3.

Evaluate your results.

4.

Below is the syntax to create a sequential model in Keras.

First, import the Sequential function and initialize your model object:

```
                    from keras.models import Sequential

                    model = Sequential()
```

Then add layers to the model one by one:

```
          from keras.layers import Dense, Activation

 # Import libraries, model elements

from keras.models import Sequential

from keras.layers import Dense, Activation

model = Sequential()

# For the first layer, specify the input dimension

model.add(Dense(units=4, input_dim=3))

# Specify activation function
```

```
model.add(Activation('sigmoid'))

# For subsequent layers, the input dimension is presumed from the previous layer
model.add(Dense(units=4))

model.add(Activation('sigmoid'))
```