Hidden Layers: Takeaways

Syntax

- Training a classification neural network:

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(1,), activation='logistic')
```

- Training a regression neural network:

```
from sklearn.neural_network import MLPRegressor
mlp = MLPClassifier(hidden_layer_sizes=(1,), activation='relu')
```

- Specifying hidden layers as a tuple object:

```
mlp = MLPClassifier(hidden_layer_sizes=(1,), activation='relu')
mlp = MLPClassifier(hidden_layer_sizes=(10,10), activation='relu')
```

- Specifying the activation function:

```
mlp = MLPClassifier(hidden_layer_sizes=(1,), activation='relu')
mlp = MLPClassifier(hidden_layer_sizes=(10,10), activation='logistic')
mlp = MLPClassifier(hidden_layer_sizes=(10,10), activation='tanh')
```

- Generating data with nonlinearity:

```
from sklearn.datasets import make_moons
data = make_moons()
```

Concepts

- The intermediate layers are known as **hidden layers**, because they aren't directly represented in the input data or the output predictions. Instead, we can think of each hidden layer as intermediate features that are learned during the training process. This is actually very similar to how decision trees are structured. The branches and splits represent some intermediate features that are useful for making predictions and are analagous to the hidden layers in a neural network.
- Each of these hidden layers has its own set of weights and biases, which are discovered during the training process. In decision tree models, the intermediate features in the model represented something more concrete we can understand (feature ranges).
- The number of hidden layers and number of neurons in each hidden layer are hyperparameters that act as knobs for the model behavior.

Resources

- Sklearn Hyperparameter Optimization
- Neural Network Hyperparameter Optimization
- Deep Learning on Wikipedia