Connect-4 Decision Tree

July 11, 2019

```
In [1]: import time
        from sklearn.tree import export_graphviz
        from sklearn.externals.six import StringIO
        from sklearn.model_selection import train_test_split
        from IPython.display import Image
        import pydotplus

        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import confusion_matrix

        from sklearn.preprocessing import LabelEncoder
        from keras.utils import np_utils

        import numpy as np
```

```
/home/shashwati/anaconda3/envs/py35/lib/python3.5/site-packages/h5py/__init__.py:36: FutureWarn
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
In [2]: # fix random seed for reproducibility
        seed = 7
        np.random.seed(seed)
```

```
In [3]: # load connect 4 dataset
        dataset = np.genfromtxt("connect-4.csv", dtype='str', delimiter=",")
```

```
In [4]: # split into input (X) and output (Y) variables
        preX = dataset[:,0:42]
        preY = dataset[:,42]

        X = np.zeros(preX.shape)
        Y = np.zeros(preY.shape)
```

```
In [5]: # converting predictors to numbers
        for i, row in enumerate(preX):
            for j, col in enumerate(row):
                if col == 'x':
                    X[i,j] = 1.0
                if col == 'o':
                    X[i,j] = -1.0
                if col == 'b':
                    X[i,j] = 0.0

In [6]: # converting categorical classes into number
        encoder = LabelEncoder()
        # code: 0 - draw; 1 - loss; 2 -win
        encoded_Y = encoder.fit_transform(preY)

In [7]: # making 80% training and 20% test data set
        train, test, label_train, label_test = train_test_split(X, encoded_Y, test_size = 0.2)

In [8]: from sklearn.tree._tree import TREE_LEAF

        def prune_index(inner_tree, index, threshold):
            if inner_tree.value[index].min() < threshold:
                # turn node into a leaf by "unlinking" its children
                inner_tree.children_left[index] = TREE_LEAF
                inner_tree.children_right[index] = TREE_LEAF
            # if there are shildren, visit them as well
            if inner_tree.children_left[index] != TREE_LEAF:
                prune_index(inner_tree, inner_tree.children_left[index], threshold)
                prune_index(inner_tree, inner_tree.children_right[index], threshold)

In [9]: # making the decision tree of depth 12
        start = time.time()

        dt = DecisionTreeClassifier(max_depth = 12, min_samples_leaf = 100)
        dt.fit(train, label_train)
        prune_index(dt.tree_, 0, 5)
        end = time.time()

In [10]: # predicting for test data
         pred = dt.predict(test)

In [11]: # function to calculate the metrics
         def metrics(cm, cl, size):
             cm = np.array(cm)
             tp = cm[cl][cl]
             fp = sum(cm[x, cl] for x in range(3))-cm[cl][cl]
             fn = sum(cm[cl, x] for x in range(3))-cm[cl][cl]
             tn = size - tp - fp - fn
             precision = tp/(tp+fp)
             recall = tp/(tp+fn)
             fmeasure = 2*(precision*recall)/(precision + recall)
             accuracy = (tp + tn)/size
             return precision, recall, fmeasure, accuracy
```

```
In [12]: # getting the confusion matrix
         cm = confusion_matrix(label_test, pred)
         print("Confusion matrix: ")
         print(cm)
```

```
Confusion matrix:
[[  20  329  896]
 [  19 1896 1475]
 [  26  872 7979]]
```

```
In [13]: # metrics for class 0 (draw)
         precision0, recall0, f0, acc0 = metrics(cm, 0, len(test))
         print("                Precision Recall F-measure Accuracy")
         print("Class 0 (draw): ", round(precision0, 3), "  ", round(recall0, 3), \
               " ", round(f0, 3), "   ", round(acc0,3))
```

```
                Precision Recall F-measure Accuracy
Class 0 (draw):  0.308    0.016   0.031     0.906
```

```
In [14]: # metrics for class 1 (lose)
         precision1, recall1, f1, acc1 = metrics(cm, 1, len(test))
         print("                Precision Recall F-measure Accuracy")
         print("Class 1 (loss): ", round(precision1, 3), "  ", round(recall1, 3), \
               " ", round(f1, 3), "   ", round(acc1,3))
```

```
                Precision Recall F-measure Accuracy
Class 1 (loss):  0.612    0.559   0.585     0.801
```

```
In [15]: # metrics for class 2 (win)
         precision2, recall2, f2, acc2 = metrics(cm, 2, len(test))
         print("                Precision Recall F-measure Accuracy")
         print("Class 2 (win): ", round(precision2, 3), "  ", round(recall2, 3), \
               " ", round(f2, 3), "   ", round(acc2,3))
```

```
                Precision Recall F-measure Accuracy
Class 2 (win):  0.771    0.899   0.83     0.758
```

```
In [16]: # average metrics
         avg_p = (precision0 + precision1 + precision2)/3.0
         avg_r = (recall0 + recall1 + recall2) / 3.0
         avg_f = (f0 + f1 + f2) / 3.0
         avg_a = (acc0 + acc1 + acc2)/ 3.0
         print("        Precision Recall F-measure Accuracy")
         print("Average: ", round(avg_p, 3), "  ", round(avg_r, 3), \
               " ", round(avg_f, 3), "   ", round(avg_a,3))
```

```
        Precision Recall F-measure Accuracy
Average:  0.564    0.491   0.482     0.822
```

```
In [17]: # training time
         print("Training time: ", round(end - start, 5), " sec")
```

```
Training time:  0.23134  sec
```

```
In [18]: # accuracy score - number of instances correctly classified
         print("Accuracy score: ", round(accuracy_score(label_test, pred), 5))
```

```
Accuracy score:  0.73231
```