Project Part 1
Pattern Recognition
ECE 759

Kudiyar Orazymbetov (`korazym@ncsu.edu`)
Nico Casale (`ncasale@ncsu.edu`)

March 7, 2018

Contents *(Note that the entries are links.)*

List of Figures

Listings

1 Introduction

2 Feature Selection

2.1 Decision Tree Feature Generation

2.1.1 MNIST

In working with the decision trees, we utilized the SVD of each image in in the training set.

3 Algorithm Implementations

3.1 Decision Tree Algorithm

A binary decision tree is a hierarchical structure that takes input data at its root and propagates it to one of many leaves. Each *leaf* of the tree represents a class designation. To reach a leaf, the features of the data are utilized at *nodes* to make a binary decision: proceed down the left or right *branch* of the tree? This structure needs to be generated before it can be used with test data. To generate a decision tree that appropriately classifies our test data according to the features we generated, we employ a recursive function. The function signature is

$$\texttt{tree} = \texttt{trainDecisionTree}(\texttt{set})$$

Where `set` is the training set, which is a MATLAB structure that contains the raw data (unused), class labels, and generated features. `tree` is the returned structure that can be used during testing. It is essentially a nested structure that contains two types of elements: nodes and leaves. At each node of the tree, an attribute and threshold are specified. If a test sample's feature at that particular attribute is less than the threshold, the sample is passed down the left branch of the node. Similarly, if the sample's feature is greater than the threshold, it goes through the right branch. This is repeated until we reach a leaf node, which specifies a class membership.

The decision tree algorithm has a few major steps, and proceeds by evaluating a metric called *information gain* at various configurations. For now, suffice it to say that information gain is a scalar that represents the improvement in prediction as we narrow down the set (by growing the tree) to find appropriate leaves.

1. Check stopping conditions:
   - If there are no more features to split on, return a leaf with the class mode of the set.
   - The set is smaller than `minLeaf`, which is a tuning parameter that is meant to reduce overfitting of the training data. If this condition is met, return a leaf with the class mode of the set.
   - If all samples in the set belong to the same class, return a leaf with the class.
   - If no feature yields an improvement to the information gain (discussed below), then return a leaf with the class mode of the set. Note that this condition is only evaluated after step 2.
2. Iterate over each feature. Sort the set along the current feature. We utilize a threshold that splits the set between adjacent feature values. Because the information gain across thresholds is convex on the whole, we use a line search that approximates the highest information gain for each threshold.

   Let `attributeBest` and `indBest` be the feature and index that yield the highest information gain. Since the set is sorted, we can simply split the set at the index given by `indBest` for the recursion.

3. Recur over the subsets given by `indBest` to find the next attribute that yields the highest information gain. Note that we exclude the attribute we chose in this execution of `trainDecisionTree(.)`.

This can be expressed in psuedocode as

Algorithm 3.1: Train Decision Tree

Data: *set* of training samples with class labels ($set2$) and features ($set1$).
Result: *tree*, a structure containing nodes and leaves.

```
begin
    $V \longleftarrow U$
    $S \longleftarrow \emptyset$
    for $x \in X$ do
        $NbSuccInS(x) \longleftarrow 0$
        $NbPredInMin(x) \longleftarrow 0$
        $NbPredNotInMin(x) \longleftarrow |ImPred(x)|$
    for $x \in X$ do
        if $NbPredInMin(x) = 0$ and $NbPredNotInMin(x) = 0$ then
            $AppendToMin(x)$
    while $S \neq \emptyset$ do
        remove $x$ from the list of $T$ of maximal index
        while $|S \cap ImSucc(x)| \neq |S|$ do
            for $y \in S - ImSucc(x)$ do
                { remove from $V$ all the arcs $zy$ : }
                for $z \in ImPred(y) \cap Min$ do
                    remove the arc $zy$ from $V$
                    $NbSuccInS(z) \longleftarrow NbSuccInS(z) - 1$
                    move $z$ in $T$ to the list preceding its present list
                    {i.e. If $z \in T[k]$, move $z$ from $T[k]$ to $T[k-1]$}
                $NbPredInMin(y) \longleftarrow 0$
                $NbPredNotInMin(y) \longleftarrow 0$
                $S \longleftarrow S - \{y\}$
                $AppendToMin(y)$
        $RemoveFromMin(x)$
```

4 Code Listings

Below are the primary scripts that solve the project. Please see the `code` folder for supporting functions.