

Distance to containing variety: The twonorm dataset

An experimental demonstration, using the UCI **twonorm** dataset, that the failures of the independent stream evaluator for binary classifiers can be used to find nearly independent trios on unlabeled data.

We read some common code for the three datasets

```
In[*]:= nbDir = NotebookDirectory[];  
SetDirectory[nbDir];  
NotebookEvaluate[FileNameJoin[{nbDir, "CommonCode.nb"}]]
```

Getting the data and taking a look at the available features

```
In[*]:= {tsvHeader, benchmarkData} = ImportPennMLBenchmarksDataset["twonorm"];  
  
In[*]:= tsvHeader  
Out[*]=  
{A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11,  
 A12, A13, A14, A15, A16, A17, A18, A19, A20, target}  
  
In[*]:= twonormFeatures  
Out[*]=  
{{1, Numerical}, {2, Numerical}, {3, Numerical}, {4, Numerical}, {5, Numerical},  
 {6, Numerical}, {7, Numerical}, {8, Numerical}, {9, Numerical}, {10, Numerical},  
 {11, Numerical}, {12, Numerical}, {13, Numerical}, {14, Numerical}, {15, Numerical},  
 {16, Numerical}, {17, Numerical}, {18, Numerical}, {19, Numerical}, {20, Numerical}}  
  
In[*]:= Length /@ benchmarkData  
Out[*]=  
<| 1 → 3697, 0 → 3703 |>
```

Experiments

```

In[*]:= manyFeaturePartitions = Table[RandomSample[twonormFeatures] //
    Partition[#, UpTo@3] & //
    Take[#, 3] &, {1000}] //
    Map[Sort, #, {2}] & // Sort /@# & // DeleteDuplicates //
    RandomSample[#, 300] & // Partition[#, 100] & //
    Take[#, 3] &;

For[i = 1, i < 4, i++,
  nTrain = 200;
  globalTrainSize = 600;
  nClassifiers = 3;
  classifierTypes = Table[{"LogisticRegression", "L1Regularization" → 0.1,
    "L2Regularization" → 0.5, "OptimizationMethod" → "Newton"}, {nClassifiers}];
  (* We separate a held-out test set that is never seen in any of the training
    runs below *)
  aTrainIndices = RandomSample[Range@Length@benchmarkData[0], globalTrainSize];
  bTrainIndices = RandomSample[Range@Length@benchmarkData[1], globalTrainSize];
  globalTrainTestSplit =
    MakeTrainTestSplit[benchmarkData, aTrainIndices, bTrainIndices];
  globalTrainData = SelectTrainDataFromSplit[globalTrainTestSplit];
  globalH0Data = SelectTestDataFromSplit[globalTrainTestSplit];
  nTest = 2000;
  runs = Table[
    {classifiersFeatures,
      Table[
        (* Select random, disjoint partitions of the training data,
          grouped by classifier *)
        trainIndices = Transpose@{
          (* The disjoint partition of a subset of the alpha training data *)
          ClassifiersSampleIndices[globalTrainData[0], nTrain],
          (* The disjoint partition of a subset of the beta training data *)
          ClassifiersSampleIndices[globalTrainData[1], nTrain]};
        (* Prepare the training data for the current feature partitions *)
        trainClassifiersData = Table[Map[#[[First@Transpose@features]] &,
          globalTrainData, {2}], {features, classifiersFeatures}];
        classifiers = TrainClassifiersDisjoint[
          trainClassifiersData, classifierTypes, trainIndices,
          Map[(Last@Transpose@#) &, classifiersFeatures], "TrainingSpeed"];

        (* Now evaluate *)
        evalIndices = {

```

```

(* The disjoint partition of a subset of the alpha training data *)
SampleIndices[globalH0Data[0], nTest],
(* The disjoint partition of a subset of the beta training data *)
SampleIndices[globalH0Data[1], nTest]};
testClassifiersData = Table[Map[#[[First@Transpose@features]] &,
  SelectFromData[globalH0Data, First@evalIndices, Last@evalIndices],
  {2}], {features, classifiersFeatures}];
vcbl = LabelCounts[classifiers, testClassifiersData];
Which[
  Not@HasAllDecisionFrequenciesQ@vcbl,
  Nothing,
  Not@IndependentEvaluationIdealIsNonEmptyQ@vcbl,
  Nothing,
  sols = AlgebraicallyEvaluateClassifiers[vcbl];
  Not@HasOnlyRealNumbersQ@sols,
  Nothing,
  Not@InsideUnitCubeQ@sols,
  Nothing,
  True,
  vcbl],
{10}]],
{classifiersFeatures, manyFeaturePartitions[[i]]}];
runs >> "./twonorm-ni-" ~~ ToString[i] ~~ ".m"
]

```