Transfer Learning Experiment

November 28, 2014

1 The Idea

- `Target Dataset` is a **small and labeled** dataset from a certain domain
- `Source Dataset(s)` can be one or more **large and labeled** datasets from *related* domains
- If we would like to build a model for the `Target Dataset` using it alone as `Training` dataset, then the model will likely perform undesirably as the dataset is quite small!
- The idea is to make maximum use of the `Source Datasets` to augment the `Target Dataset` and classify data from its domain (this is Instance Transfer Learning)

2 The Data

- Two `Source` datasets TS3-Sapphire.arff and TS6-Sapphire.arff (This is for strain *Plasmodium vivax* for assays TS3 and TS6 respectively)
- The TL algorithm (TransferBoost) supports having multiple source datasets as it assigns weights to instances as well as source datasets
- I have done **two experiments** with two different `Target` Datasets:
  - Data from the Venus Channel for assay TS6 (This is for strain *Plasmodium falciparum*)
    - this dataset has 1435 instances and the number of `Active` instances is 27
    - details of this experiment are in **experiments6.pdf**
  - Data from the Sapphire Channel for assay TS7 (This is for strain *Leishmania major*)
    - this dataset has 1326 instances and the number of `Active` instances is 12
    - details of this experiment are in **experiments7.pdf**
- Iteratively, I split each of these two datasets to create `Target` and `Test` datasets
- I started with a very small `Target` dataset of all `Inactives` and doubled the size at each iteration
- As the size increased, I started to include Active instances (with the same proportion for *Plasmodium falciparum* data)
- I built models using the `Target` dataset, evaluated via `cross validation` and `using the Test dataset`

- Remember. the `Target` dataset is used as `Training` dataset to build TL (Transfer Learning .. which also uses the source datasets), Naive Bayes (NB), Decision Trees (J48), Support Vector Machine (SMO) and k-Nearest Neighbour (IBk) models
- Details of the various datasets are shown in the following tables:

Exp. No.	**Target Dataset (Training)**	**Test Dataset**
1	Size=3 (3 Inactive + 0 Active)	Size=1432 (1405 Inactive + 27 Active)
2	Size=6 (6 Inactive + 0 Active)	Size=1429 (1402 Inactive + 27 Active)
3	Size=12 (12 Inactive + 0 Active)	Size=1423 (1396 Inactive + 27 Active)
4	Size=24 (24 Inactive + 0 Active)	Size=1411 (1384 Inactive + 27 Active)
5	Size=49 (48 Inactive + 1 Active)	Size=1386 (1360 Inactive + 26 Active)
6	Size=98 (96 Inactive + 2 Active)	Size=1337 (1312 Inactive + 25 Active)
7	Size=196 (192 Inactive + 4 Active)	Size=1239 (1216 Inactive + 23 Active)
8	Size=392 (384 Inactive + 8 Active)	Size=1043 (1024 Inactive + 19 Active)
9	Size=784 (768 Inactive + 16 Active)	Size=651 (640 Inactive + 11 Active)

Table 1: Details of Datasets for Strain *Plasmodium falciparum*

Exp. No.	**Target Dataset (Training)**	**Test Dataset**
1	Size=3 (3 Inactive + 0 Active)	Size=1323 (1311 Inactive + 12 Active)
2	Size=6 (6 Inactive + 0 Active)	Size=1320 (1308 Inactive + 12 Active)
3	Size=12 (12 Inactive + 0 Active)	Size=1314 (1302 Inactive + 12 Active)
4	Size=25 (24 Inactive + 1 Active)	Size=1301 (1290 Inactive + 11 Active)
5	Size=50 (48 Inactive + 2 Active)	Size=1276 (1266 Inactive + 10 Active)
6	Size=100 (96 Inactive + 4 Active)	Size=1226 (1218 Inactive + 8 Active)
7	Size=200 (192 Inactive + 8 Active)	Size=1239 (1122 Inactive + 4 Active)

Table 2: Details of Datasets for Strain *Leishmania major*

2.1 Column Names in Results Tables:

I have abbreviated column names in the tables to save display space. In order from left to right, they're as follows:
corr = Percentage of Correct guesses,
inco = Percentage of Incorrect guesses,
auc = Area Under the Curve (for class Active),
k = Kappa statistic,
mae = Mean Abs Error,
rmse = Root Mean Squared Error,
rae = Relative Abs Error,
rrse = Root Relative Squared Error,
prec = Precision (for class Active),
rec = Recall (for class Active),
fM = F-Measure (for class Active),
eR = Error Rate

3 My Conclusion:

Transfer Learning outperforms ordinary algorithms when there is little data. As more data becomes available, some other algorithms perform equally well!