Machine Learning Project 3

jjjj222

1 Overview

This k-Nearest Neighbor (KNN) program is implemented in Ruby. It uses information gain and fisher score, for discrete and continuous attributes respectively, to perform feature weighting. It also supports NT-Growth algorithm to reduce the number of training examples, and Principle Component Analysis (PCA) for feature transformation. The datasets used for testing are Wine [1], Breast Cancer [2], Statlog (Vehicle Silhouettes) [3], Pima Indians Diabetes [4], and Glass Identification [5].

2 Setup

2.1 Installation

There is no installation requirement since this program is developed in Ruby, However, please make sure the Ruby executable is in the path:

```
/usr/bin/ruby
```

2.2 Execution

Type the command below under the top directory:

```
./run_knn.rb <data_file> <attribute_file> <setup_file>
```

The formats of "`<data_file>`", "`<attribute_file>`", and "`<setup_file>`" are described below.

2.2.1 Data File

The data file is a comma-separated file with one example per line, and the target class should be placed in the last column. Missing data should be specified as "?". Note that all the white space (e.g. space, tab) will be removed before parsing.

```
Sunny,      85,     High,   Weak,   No
Rain,       57,     High,   Weak,   Yes
Rain,       64,     Normal, ?,      Yes
Rain,       ?,      Normal, Weak,   No
Sunny,      65,     Normal, Strong, Yes
Overcast,   33,     ?,      Strong, Yes
...
```

2.2.2 Attribute File

In attribute file, every line indicates an attirbute in the data, and there should be exact 2 columns in a line: the first column is the name of the attribute, and the second column should specify that the attribute is either continuous (c) or discrete (d). Note that the total number of lines in the attribute file should be the same as the total number of columns in data file.

```
Outlook,        d
Temperature,    c
Humidity,       d
Wind,           d
PlayTennis,     d
```

2.2.3 Setup File

The setup file is a JSON file with key-value pairs, specifiying the parameters of this KNN program. Those not specified or set as "default" will be set to default values as follows: (Note that if the "k" is set to "-1", it will be set to the total number of training examples internally.)

```
{
    "k"                         : 1,
    "distance-weighting"        : false,
    "feature-weighting"         : false,
    "PCA"                       : false,
    "PCA-energy-ratio"          : 0.9,
    "NTGrowth"                  : false,
    "NTGrowth-drop-ratio"       : 0.75
}
```

2.3 Project Structure and Evnironment

The development environment is as follows:

- `OS: OSX El Capitan 10.11.3`
- `Ruby: ruby 2.3.0p0 (2015-12-25 revision 53290)[x86_64-darwin15]`

The project structure is as follows:

- `run_knn.rb` : excutable
- `report.pdf` : report
- `makefile`: for test only
- `src/` : source codes
- `data/` : datasets for testing
- `test/` : results

3 Implementation

3.1 Pre-processing

The data, attribute, and setup file are parsed and pre-processed in "`class Tester`". All possible values for each discrete attribute will be extracted, and the mean and standard deviation of each continuous attribute will be calculated. Examples will be shuffled and then be partitioned into 10 parts for the 10-fold cross-validation. In each iteration, 1 part of the sub-examples will be kept for testing, and the rest will be passed down to "`class KNN`" for k-NN classifier.

3.2 Learning

There is no learning in k-NN: every training examples are simply stored in an array; however, continuous attributes will be normalized to make sure that each attribute has equal importance while calculating the distance.

3.2.1 NT-growth

Users can turn on NT-Growth by specifying `"NTGrowth" : true` in setup file. By using NT-Growth algorithm, training examples will be preprocessed to reduce the total number and noises. The algorithm is as follows: create a new empty set and add training examples to it one by one; however, if the current set can correctly use 1-NN to classify the to-be-added example, then drop the example instead. In the meantime, update the accuracy of the example which is chosen to classify to-be-added example. By the end, those with low accuracy will be dropped out of the set as well, resulting a smaller new set of training examples with hopefully fewer noise. The threshold to drop the noise can be specified in setup file with `"NTGrowth-drop-ratio"` option.

3.2.2 Principle Component Analysis

Users can turn on Principle Component Analysis (PCA) by specifying `"PCA" : true` in setup file. With PCA, covariance matrix along with its eigenvectors will be calculated. And the smallest set of eigenvectors with sum of their eigenvalues exceed a given ratio (`"PCA-energy-ratio"` in setup file) will be chosen and be used for transformation. Note that to use PCA, every attribute in the dataset should be continous.

3.3 Feature Weighting

For discrete attributes, information gains of each attributes are used as weighting; for continuous attributes, the fisher score are used as weightings. The cases with mixed feature types are not supported. The weightings will be stored for calculating distance during classification. Note that feature weighting will be performed before NT-growth if both options are turned on.

3.4 Classification

The test examples will first be normalized as well, and then the distances between the current test example and all the training examples will be calculated and sorted. Based on the paremters, the k cloest examples will be picked and used to decide the final class by either majority (`"distance-weighting : false"`) or distance-weighted (`"distance-weighting : false"`.)

3.4.1 Missing Value

For continuous attributes, the missing value will be set to the mean of the attribute. For discrete attributes, the missing value will be treated as an additional class in the distance calculation.

4 Experimental Result

The datasets used for testing are Wine [1], Breast Cancer [2], Statlog (Vehicle Silhouettes) [3], Pima Indians Diabetes [4], and Glass Identification [5]. All the detail information is in files "`./test/test_knn/test_knn_<setup>/<CASE>/<CASE>.out`". (The details of ID3 and majority classifier are in"`./test/test_id3/<CASE>/<CASE>.out`", of ANN are in "`./test/test_ann/test_ann_<setup>/<CASE>/<CASE>.out`")

4.1 Comparison between KNN, ANN and ID3

The Nearest-Neighbor algirhtm, though simple, performs surprisingly well. It out-performs ID3 and ANN in `wine`, `vehicle`, and `glass`, and has almost the same accuracies in the other cases. Even though the KNN we chose below is with $k = 5$ and distance-weighting, there is not many differences if we simply use $k = 1$ without any other options.

name	classifier	info	accuracy	SE	95% CI
wine	KNN	5,dw	96.0	1.9	(91.8, 100.2)
	ANN	1618.6	84.9	2.6	(79.0, 90.7)
	ID3	4.5	84.2	3.3	(76.9, 91.6)
	majority	N/A	40.1	4.9	(29.1, 51.1)
breast	KNN	5,dw	96.7	0.8	(94.9, 98.5)
	ANN	37.1	97.4	0.7	(95.9, 98.8)
	ID3	5.8	91.7	0.8	(89.9, 93.6)
	majority	N/A	62.7	1.9	(58.5, 67.0)
vehicle	KNN	5,dw	72.4	1.9	(68.0, 76.7)
	ANN	2411.5	57.3	1.5	(54.0, 60.7)
	ID3	81.6	68.3	1.3	(65.3, 71.3)
	majority	N/A	21.5	1.2	(18.9, 24.1)
diabete	KNN	5,dw	72.3	1.9	(68.1, 76.4)
	ANN	72.0	76.2	1.7	(72.3, 80.0)
	ID3	47.5	71.1	1.3	(68.1, 74.1)
	majority	N/A	65.1	1.4	(62.0, 68.2)
glass	KNN	5,dw	69.2	3.6	(61.2, 77.1)
	ANN	156.5	52.3	5.4	(40.2, 64.4)
	ID3	26.5	62.6	3.1	(55.8, 69.4)
	majority	N/A	35.5	2.4	(30.2, 40.7)

Table 1: Results of KNN, ANN, ID3, and majority classifier

name	k	option	accuracy	SE	95% CI
wine	1		95.5	1.6	(91.9, 99.1)
	3		94.9	1.8	(91.0, 98.9)
	5		95.5	1.4	(92.3, 98.6)
	7		96.6	1.2	(93.9, 99.4)
breast	1		94.7	0.9	(92.7, 96.8)
	3		96.3	0.7	(94.7, 97.9)
	5		96.5	0.9	(94.4, 98.6)
	7		96.5	0.9	(94.4, 98.6)
vehicle	1		70.9	2.2	(65.9, 75.9)
	3		71.6	1.9	(67.5, 75.8)
	5		73.2	1.8	(69.1, 77.3)
	7		73.1	1.5	(69.8, 76.3)
diabete	1		69.9	2.1	(65.1, 74.7)
	3		73.3	2.0	(68.8, 77.8)
	5		74.0	1.7	(70.1, 77.8)
	7		73.8	1.8	(69.8, 77.8)
glass	1		69.2	4.2	(59.8, 78.6)
	3		71.0	3.6	(62.9, 79.1)
	5		68.2	4.0	(59.4, 77.0)
	7		64.5	4.1	(55.3, 73.7)

Table 2: KNN with different k values

4.2 Comparison between different options

4.2.1 Distance Weighting

In these 5 cases, the distance weighting doesn't seem to help much. The accuracies of KNN with distance weighting for $k = 5$ does not have significant change. Maybe there isn't many noises in these 5 cases, or maybe it's just because the distance weighting is not that useful as I thought.

4.2.2 Feature Weighting

The feature weighting provides a little increase in accuracy, except the `vehicle` case. I think the features in `vehicle` are special, because the accuracy of it decreased for every option. Overall I believe the feature weighting will help if fine-tuning the algorithms and parameters carefully.

4.2.3 Principle Component Analysis (PCA)

PCA drastically improves the accuracies in `diabete` and `glass`, yet also decreases the accuracy in `vehicle` a lot. It uses 5 eigenvectors out of 18 in the `vehicle` case while the energy ratio is 0.9. I tried the PCA with energy ration with 0.99 and it increases the accuracy of `vehicle` to 69.9%; however, it decreases the accuray of `diabete` to 69.9%. It seems that the performance of feature selections vary among cases. In sum, apart from `vehicle`, it provides a very good improvement overall.

4.2.4 NT-Growth

NT-Growth reduces the stored examples as well as runtime, but it decreases the accuracies in every case. Actually I'm not pretty sure whether is a correct result. Maybe it's due to some unknown bugs.

name	k	option	accuracy	SE	95% CI
wine	5		95.5	1.4	(92.3, 98.6)
	5	dw	96.0	1.9	(91.8, 100.2)
	5	fw	97.2	1.3	(94.3, 100.0)
	1	pca	97.2	1.9	(92.9, 101.5)
	1	nt	88.2	1.8	(84.2, 92.1)
breast	5		96.5	0.9	(94.4, 98.6)
	5	dw	96.7	0.8	(94.9, 98.5)
	5	fw	96.5	1.0	(94.3, 98.7)
	1	pca	96.5	0.7	(94.8, 98.1)
	1	nt	94.6	1.5	(91.2, 97.9)
vehicle	5		73.2	1.8	(69.1, 77.3)
	5	dw	72.4	1.9	(68.0, 76.7)
	5	fw	70.0	1.5	(66.7, 73.2)
	1	pca	56.6	1.7	(52.9, 60.4)
	1	nt	64.7	1.6	(61.0, 68.3)
diabete	5		74.0	1.7	(70.1, 77.8)
	5	dw	72.3	1.9	(68.1, 76.4)
	5	fw	74.7	1.8	(70.7, 78.8)
	1	pca	94.3	0.8	(92.4, 96.2)
	1	nt	74.3	1.9	(70.2, 78.5)
glass	5		68.2	4.0	(59.4, 77.0)
	5	dw	69.2	3.6	(61.2, 77.1)
	5	fw	72.4	2.9	(65.9, 78.9)
	1	pca	95.8	1.6	(92.2, 99.4)
	1	nt	59.5	4.9	(48.5, 70.5)

Table 3: KNN with different options

References

[1] Wine Data Set
`http://archive.ics.uci.edu/ml/datasets/Wine`

[2] Breast Cancer Wisconsin (Diagnostic) Data Set
`http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`

[3] Statlog (Vehicle Silhouettes) Data Set
`http://archive.ics.uci.edu/ml/datasets/Statlog+%28Vehicle+Silhouettes%29`

[4] Pima Indians Diabetes Data Set
`http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes`

[5] Glass Identification Data Set
`http://archive.ics.uci.edu/ml/datasets/Glass+Identification`