F20DL - Data Mining CW1 Report

F20DL_2018-2019
Data Mining and Machine Learning
Taught by Diana Bental and Ekaterina Komendantskaya

Mobeen Aftab - H00220767
Jonathan Mendoza - H00251229
Owen Welch - H00235788

The task in this coursework is to work with a large ‘emotion recognition’ dataset, created by the research group of Pierre-Luc Carrier and Aaron Courville. The data set (of 35887 examples) consists of 48x48 pixel grayscale images of faces and the overarching goal is to categorize each face based on the emotion shown in the facial expression in one of seven categories and then perform various data mining activities on that data such as attempting to recognise new data once trained off this training set.

Tools and libraries

The scripts written to accomplish this coursework is mostly written in Python using pandas and scikit-learn libraries. We initially accomplished all of the tasks using Weka GUI on a reduced subset and used the results to compare accuracy and performance. Source code used for this course work can be found on our private Github repository [7]. Other supplementary documentation is in our Google Drive [1].

Tasks 1-3: Data conversion, randomisation, and computational constraints

Step 1: Convert all csv files into the format suitable for pandas and scikit-learn

We wrote a script `convert.py` to convert the given datasets to a .csv that pandas can read using its `read_csv()` function. The function converts it into a Dataframe; from which, the class attribute and other attributes can be extracted.

Step 2: Produce versions of these files that have the instances in a randomised order

Sci-kit learn's `train_test_split()` function with the argument `shuffle=true` along with a seed value was used to produce versions of the dataset with instances in a random order.

Step 3: Reducing the size, dealing with computational constraints

When using our command-line Python scripts, pandas was able to handle the whole dataset.

We split our data up by having 75% of the instances for training, and 25% for testing.

On the times we used the Weka GUI, we used the filters:

- weka.filters.supervised.instance.Resample
- weka.filters.unsupervised.instance.Randomize

Task 4 - Classification performance of the Nearest Neighbour algorithm

We ran the Nearest Neighbour algorithm on the main dataset as well as the ones for specific emotions. The results are as follows:

Dataset	Accuracy
fer2018	0.34407044137316095
fer2018angry	0.8586714222024074
fer2018disgust	0.9829469460543915
fer2018fear	0.8390548372715114
fer2018happy	0.7248105216228266
fer2018neutral	0.7658270173874275
fer2018sad	0.8124164065983058
fer2018surprise	0.9005795809184128

Figure 1 - Performance of 3-Nearest-Neighbours algorithm

Effect of testing on training set vs. having a separate testing set

The model produces higher accuracies when evaluating on itself; however when using a separate training set, the classifier performs considerably worse. This can be due to overfitting of the model to the data. When unseen inputs are given, the accuracy is worse.

Computation Time

Running the nearest neighbour algorithm with all the attributes on such a huge dataset takes a long time. In our tests, the program typically took greater than 10 minutes to train and test combined.

Task 5 - Deeper Analysis: search for important attributes by correlation of each class

To find the top attributes correlated to the class attribute for each emotion dataset, we first did some background research into data correlation specifically reading into statistics to measure correlation for data mining applications (Rayward-Smith, 2007) and then used Sci-kit learn's `SelectKBest()` function with 2 univariate selection algorithms. `f_classif` (ANOVA f-values [2]) and `chi2` (Chi-squared test [3])

We recorded the results in tables that can be found here [4]. We have also done this task using the Weka GUI 'Select Attributes' feature using Correlation as the attribute evaluator. Results can be found here [5].

Nearest neighbour using only top correlated attributes

While gathering the top attributes for each specific emotion dataset (*fer2018EMOTION.csv*), we were curious to see what effects it would have if we ran Nearest Neighbour on the individual emotion datasets. The results can be found here [6].

Task 6 - Classification using the selected attributes from 5

Using only the top attributes gathered from each specific emotion dataset from step 5, we had new versions of the main dataset (fer2018.csv) with 14, 35 and 70 non-class attributes. Running the K Nearest Neighbour algorithm on the new data yielded the following accuracies:

Top 2	Top 5	Top10
0.2692822113241195	0.2656041016495765	0.28923316986179226

The accuracy decreased compared to running the K Nearest Neighbour algorithm using all the attributes; however, there were significant performance gains in terms of computing speed. K Nearest Neighbour took a very long time (>10min) to run with 48x48+1 attributes. This is a potential trade off that can be made; a slightly lower accuracy to the speed of executing the algorithm.

Task 7 - Conclusions

Which emotions are harder to recognise?

Figure 1 shows the performance of 3-Nearest-Neighbours on all the datasets. The most easily identified emotion was disgust, and the least easy to identify was happiness. This could mean that more data is needed to train the model for detecting happiness.

What was the purpose of Tasks 5 and 6?

The purpose of tasks 5 and 6 was to see the effect of focusing on the highest correlated values. Having a smaller set of attributes could eliminate noise and improve the classification. Furthermore, having such a dataset with a large number of instances and attributes, having a model which only takes into account fewer attributes would lead to a reduced amount of time needed to classify new instances.

In the context of this course work, our results show that running k Nearest Neighbour on the main *fer2018* dataset using only the top attributes from each emotion actually reduced the accuracy by a little bit compared to using all 48x48 attributes. However, significant gains in terms of computing performance were observed. It previously took over 10 minutes to classify 8,000 new instances with all the attributes; with less attributes, this took half the time.

What would happen if the data sets you used in Tasks 4, 5 and 6 were not randomised?

KNN uses a memory based approach such that the classifier will adapt as the training data is learned. As KNN is an non parametric lazy algorithm it makes no assumptions of the underlying data distribution. It keeps any generalization of the training data to a minimum or none which KNN keeps all (or almost all) of the training data during the testing phase. If the training set is not randomised KNN will overfit the training set and perform poorly on unseen test set because KNN prediction calculates the similarity between each training instance in input sample.

What would happen if there is cross-correlation between the non-class attributes?

As discussed in Lecture 9, features that correlate with the target class may correlate with each other. They may actually have the same underlying cause; thus could be treated as a single feature. In this coursework, results from step 5 [4] show that, for instance on the *fer2018happy* dataset, the top 10 attributes are pixels very close to each other. There may be other interesting features that could improve the classification if the cross-correlated attributes were considered as one attribute.

Levels of contribution:

Mobeen Aftab - 33.33 %
Jonathan Mendoza - 33.33 %
Owen Welch - 33.33 %

Links:

[1] Google Drive:
https://drive.google.com/open?id=1SyGKWrumyfmoDq8m4GqKGBNMXvbOoe87
[2]: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html
[3]: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html
[4]:
https://docs.google.com/document/d/1dtFd3YHwtBQBruZv0xWUEqlevTmrVUc9KYSxY_D9URk/edit?usp=sharing
[5]:
https://docs.google.com/document/d/1Uz2eHrzbFr7OjZ4FceUkFSU7efgg0bEpFVnV7YgEbnc/edit?usp=sharing
[6]:
https://docs.google.com/document/d/1CO-IqTJ6VyAR74VP2JEsRx1b1qri_C9mzjIrUvOJ4Eo/edit?usp=sharing
[7]: https://github.com/MobeenAftab/F20DL

References

Mwagha, S., Muthoni, M. and Ochieg, P. (2014). *Comparison of Nearest Neighbor (ibk), Regression by Discretization and Isotonic Regression Classification Algorithms for Precipitation Classes Prediction*. [online] Ir.cut.ac.za. Available at: http://ir.cut.ac.za/bitstream/handle/11462/723/Comparison%20of%20Nearest%20Neighbor%20%28ibk%29%2C%20Regression%20by%20Discretization%20and%20Isotonic%20Regression%20Classification%20Algorithms%20for%20Precipitation%20Classes%20Prediction.pdf?sequence=1&isAllowed=y [Accessed 13 Oct. 2018].(Mwagha, Muthoni and Ochieg, 2014)

Scikit-learn.org. (2018). *sklearn.feature_selection.f_classif — scikit-learn 0.20.0 documentation*. [online] Available at: http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif [Accessed 13 Oct. 2018]. (Scikit-learn.org, 2018)

Williams, G. (2009). *Rattle: A Data Mining GUI for R*. [online] Journal.r-project.org. Available at: https://journal.r-project.org/archive/2009/RJ-2009-016/RJ-2009-016.pdf [Accessed 15 Oct. 2009].(Williams, 2018)

Weka.sourceforge.net. (2018). *weka.filters.unsupervised.instance*. [online] Available at: http://weka.sourceforge.net/doc.stable/weka/filters/unsupervised/instance/package-summary.html [Accessed 15 Oct. 2018].

Rayward-Smith, V. (2007). Statistics to measure correlation for data mining applications. *Computational Statistics & Data Analysis*, [online] 51(8), pp.3968-3982. Available at: https://www.sciencedirect.com/science/article/pii/S0167947306001897 [Accessed 15 Oct. 2018].