

Data Mining and Machine Learning

Course Work 2

Issa Haddad

H00278537

Nov 7, 2017

Code, Snapshots, and Data visualization trees are all available in the link mentioned below. The file is structured in a way that is easy to follow. Code folder source classes are divided in such a way to incorporate a code for every item in this coursework. You can open the code project file in eclipse to view code for entire coursework once. Results folder includes screenshots, output results, and tree visualization in DOT format. Results folder is also divided in such a way to incorporate a file for each item in this coursework. Please view example below on how to access code and results for item or question “x”.

**Note**: I used programming (java in eclipse) to run all of the questions except for question 12.

* Coursework\_2/DM&ML\_CW2\_CODE/src/Qx
* Coursework\_2/Results/Qx\_results/Screenshots
* Coursework\_2/Results/Qx\_results/Trees

Link:

**Data conversion, data randomization, reducing the size and dealing with computational constraints**

Using Java with Eclipse and added Weka Archives. I read contents of the csv file line by line, I then proceed to store all class attribute values in one array, and all the non-class attribute values in a 2-D array. Using Weka libraries, I build my Attribute-Relation File Format ARFF data with the correct attributes. I also make sure that I place the class attribute in the last to facilitate procedures in the future. I then proceed to format and add my instances one by one to an object called “Instances”. After creating my ARFF data, I save it into an ARFF file using “ArffSaver”. I was using an Array List, which is dynamic, to store all the attributes from the csv file. The file is too large and initially took forever to run while crashing multiple times. I had to then create an arrays of fixed sizes (35887 and 2304) to store all the attributes. This sped the process and I managed to execute the program without crashing. After creating my ARFF data. I used the code “data.randomize( new Random());” to shuffle all the instances before saving it on another ARFF file. I used Eclipse to run the code, and had no trouble with heap size.

**Classification: Performance of Naïve Bayes Algorithm on the given data set**

Accuracy or True positive Rate or Recall per class 🡪

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Class 0 | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Overall accuracy |
| **TP Rate** | 0.047 | 0.208 | 0.055 | 0.154 | 0.398 | 0.598 | 0.151 | 21.6318% |

**Deeper analysis of the data:**

1. Running Naïve Bays on 50% of the randomized instances

Accuracy or True positive Rate or Recall per class 🡪

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Class 0 | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Overall accuracy |
| TP Rate | 0.066 | 0.268 | 0.058 | 0.132 | 0.399 | 0.587 | 0.153 | 21.3955% |

The accuracies obtained with 50% of the randomized instances are almost equal or identical to the accuracies obtained with the full non-randomized instances.

1. First 10 fields in order of absolute correlation value for each emotion

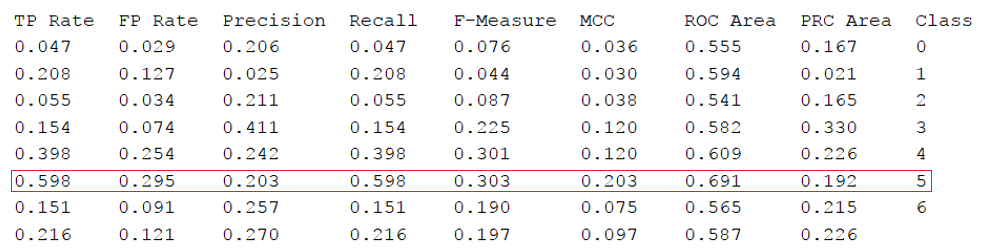
**Note**: attribute names start from 1 and not from 0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Angry | Disgust | Fear | Happy | Neutral | Sad | Surprise |
| Attribute 1 | Pixel 1362 | Pixel 24 | Pixel 818 | Pixel 1897 | Pixel 11 | Pixel 551 | Pixel 1408 |
| Attribute 2 | Pixel 1314 | Pixel 25 | Pixel 770 | Pixel 1896 | Pixel 12 | Pixel 599 | Pixel 1456 |
| Attribute 3 | Pixel 1363 | Pixel 30 | Pixel 866 | Pixel 1849 | Pixel 53 | Pixel 600 | Pixel 1409 |
| Attribute 4 | Pixel 1409 | Pixel 26 | Pixel 819 | Pixel 1945 | Pixel 197 | Pixel 648 | Pixel 840 |
| Attribute 5 | Pixel 1315 | Pixel 27 | Pixel 867 | Pixel 1944 | Pixel 89 | Pixel 552 | Pixel 744 |
| Attribute 6 | Pixel 1410 | Pixel 29 | Pixel 911 | Pixel 1850 | Pixel 59 | Pixel 647 | Pixel 745 |
| Attribute 7 | Pixel 1361 | Pixel 31 | Pixel 771 | Pixel 1898 | Pixel 150 | Pixel 601 | Pixel 792 |
| Attribute 8 | Pixel 1423 | Pixel 19 | Pixel 863 | Pixel 1848 | Pixel 245 | Pixel 649 | Pixel 793 |
| Attribute 9 | Pixel 1375 | Pixel 78 | Pixel 914 | Pixel 1993 | Pixel 137 | Pixel 553 | Pixel 1361 |
| Attribute 10 | Pixel 1457 | Pixel 79 | Pixel 623 | Pixel 1992 | Pixel 14 | Pixel 550 | Pixel 1455 |

**Attempt a better Bayesian Classification**

**Note**: If you look back to the table above we can see duplicate attributes highlighted in red found for pixel 1409, thus we only ended up with 34 total attributes when gathering the top 5 for each emotion**.** Another duplicate attributes highlighted in blue was found for pixel 1361, thus we only ended up with 68 total attributes when gathering the top 10 for each emotion.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Top Attributes | Class 0 | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Overall Accuracy |
| TP Rate | Top 2 | 0.073 | 0.000 | 0.030 | 0.372 | 0.335 | 0.622 | 0.158 | **26.1097%** |
| Top 5: | 0.077 | 0.002 | 0.050 | 0.296 | 0.365 | 0.645 | 0.174 | **25.5747%** |
| Top 10: | 0.068 | 0.099 | 0.050 | 0.231 | 0.378 | 0.662 | 0.190 | **24.6552%** |

****

Detailed accuracy by class from part 4.

**Make Conclusions:**

Selecting top correlating attributes actually yields a higher overall accuracy in classifying our instances correctly. However, because of the complexity of this data set and because it is not very straightforward to determine an emotion from a couple of pixels only, the accuracy only increased by 5% max.

The emotion that is most difficult to recognize is the disgust emotion or class 1 emotion. This can be further validated from the recall rates obtained for that particular emotion throughout our experiment. We can notice that it’s the lowest. This is due to two reasons. First because it has significantly less records than all the other emotions. Second because the disgust emotion can be very interrelated with the anger emotion. They both can be expressed by more or less the same face expression.

The top 2 fields for class 5 “Surprise” were the most reliable. The top 2 because they yield higher accuracy than even the top 5 or top 10 attributes from each emotion. Class 5 or surprise, because as seen from the graph above, Naïve Bayes gives it the highest TP rate and precision. The purpose of items 4-6 was to experiment on removing redundant or not important attributes from our dataset. This has many advantages including yields a better classification accuracy. If the dataset on Items 4-6 were not randomized, they would still yield the same results. This is due to the fact that we dealt with attributes and not instances

**Beyond Naïve Bayes: complex Bayesian Network Architectures**

**Note:** Although I used Java programming to run this code, I couldn’t obtain tree using the code, so I had to resort to Weka GUI to produces and visualize the tree.

Before attempting this item. I ran a prep code to minimize the number of attributes in the randomized version of fer2017. I chose and created an ARFF file with the top 100 attributes for the Fer2017 File and I did the same for each emotion file.

In this Experiment I ran the K2 Search while gradually increasing the number of parents from 1 to 3. I also ran both the Hill-Climbing search and he Tabu search. I performed Bayes Net on question 1 through 6 again and recorded the following values:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | K2 | | | Hill Climbing | Tabu Search |
| No. of parents | **1** | **2** | **3** | **1** | **1** |
| Accuracy on Fer2017 | 27.8346% | 37.2391% | 49.3493% | 27.8346% | 27.9488% |
| Accuracy on 50% of Fer2017 | 26.2888% | 37.1844% | 49.2393% | 26.2888% | 26.5842% |
| Accuracy on top 2 Attributes | 26.6949% | 30.727% | 32.555% | 26.6949% | 26.3912% |
| Accuracy on top 5 Attributes | 26.8872% | 33.299% | 37.2224% | 26.8872% | 26.9095% |
| Accuracy on top 10 Attributes | 26.55% | 34.3188% | 42.1044% | 26.55% | 26.8537% |

K2 is an algorithm that uses greedy search and looks for optimal tree structure by adding parents to nodes in the structure. It only adds a parent if that particular arc from the node increase the fitness score of the tree structure. An overall significant improvement in accuracy was obtained for questions 4-6 over the Naïve Bayes, what is even more interesting is the fact the accuracies kept increasing as the number of parents increase, and as the number of attributes increase. Dependencies between attributes were very much mixed. It was difficult to infer any kind of behavior from it, except for the fact the fact the top attributes calculated in question 5 seemed to appear more as a parent node rather than a child node.

Hill Climbing is another greedy search algorithm. It starts with a random tree structure and proceeds to add arcs to the node that only increase the fitness score of the tree. It gave same accuracies as the K2, that’s because they both behave similarly.

Tabu search is like Hill Climbing, except that when it reaches the local maximum, it can still choose with a certain probability to add an arc that lowers the fitness. This probability decrease as the number of iterations increase. Tabu search also never rediscovers the same route it took, thus allowing it to explore more solutions that could possible yield a better accuracy than the local optimum it found initially. This explains the little improvement obtained over K2 or Hill Climbing. We can infer from the tree visualization that some adjacent pixels seemed to form some kind of dependencies, specially the top attributes obtained in question 5 for each emotion seemed to have more dependency than the rest.

**Clustering, K-means**

**Note:** I used the ARFF files created in the previous section to perform this clustering

Here I also used all the ARFF files from the previous item; with the top 100 attributes in each file (not each emotion). I ran K-means on Fer2017 file and on all Emotion files. Clustered Instance excluding and including class instance were the same on every Fer2017Emotion file, but only differed in the Fer2017 File that classifies all emotions.

Number of clusters for Fer2017 with all emotions = 7 because classes can be assigned from 0 to 6. Number of clusters for each emotion = 2 because classes can be either assigned 0 or 1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **All Emotions** | **Angry** | **Disgust** | **Fear** | **Happy** | **Neutral** | **Sad** | **Surprise** |
| **Accuracy** | 21.6429% | 57.2759% | 57.7674% | 55.9562% | 54.4375% | 52.1832% | 61.2004% | 51.6928% |

The results or accuracy obtained for the Fer2017 data on all emotions equals 21.6429% which is almost identical to the accuracy obtained in question 4 for Naïve Bays. However, performance decreased when compared to the supervised method of Bayes Nets. The closer the initial points were from each other the less likely it is for k-means to find the optimal solution.

**Beyond K-means, tools for computation of optimal number of clusters**

I have used the Farthest-First Clustering mechanism in this dataset. Farthest first selects its first point randomly and each consequent point is the farthest away from the list of points selected earlier, hence the name Farthest-First. I applied this form clustering with Weka’s default settings and gradually increasing the number of clusters from 2 to 8. I obtained the following results.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **All Emotions** | **Angry** | **Disgust** | **Fear** | **Happy** | **Neutral** | **Sad** | **Surprise** |
| **Accuracy 2 clusters** | 25.7893% | 68.9888% | 83.5874% | 62.4599% | 70.9449% | 58.4836% | 80.6114% | 81.0934% |
| **Accuracy 3 clusters** | 25.8255% | 66.9072% | 83.9635% | 61.8385% | 70.4489% | 47.9143% | 79.9315% | 79.3686% |
| **Accuracy: 4 clusters** | 25.749% | 66.8543% | 81.1185% | 57.0123% | 70.2761% | 47.2929% | 79.6751% | 78.9701% |
| **Accuracy: 5 clusters** | 25.8227% | 65.4471% | 80.1154% | 56.1819% | 70.2037% | 44.6067% | 79.2571% | 75.9662% |
| **Accuracy: 6 clusters** | 26.3382% | 64.2489% | 67.1023% | 55.5744% | 69.5377% | 43.336% | 79.2488% | 75.8074% |
| **Accuracy: 7 clusters** | 25.5575% | 63.7692% | 66.8292% | 53.217% | 68.9832% | 41.2461% | 79.0509% | 75.6736% |
| **Accuracy: 8 clusters** | 25.2125% | 62.8361% | 66.6453% | 50.9516% | 67.6485% | 39.2956% | 78.828% | 75.6096% |

Farthest-First algorithm has the ability to explore clusters on different ends of the spectrum. This is good to create clusters with points that have the most different set of features. Hence it potentially can find a more optimal solution that normal k-means. This can be proven by comparing the accuracy of k-means with the accuracy of Farthest-First algorithm obtained with 7 number of clusters. Farthest-First yields a better accuracy. Furthermore, 6 number of clusters obtained the highest accuracy in our experiment for the dataset that contains all the emotions. This is due to the fact that the disgust emotion has significantly less records than all the rest, thus, it is harder of unsupervised algorithms to place it on a different cluster on its own. Majority of 2 number of clusters obtained the highest accuracy for the data sets containing each emotion.

**Conclusions**

So far in this coursework, we have only used hard clustering. This means that any point in space will have to be classified to evaluated to exactly one class. Unlike soft clustering that involves a probability on a single point to belong to a certain class. Output of the experiments I conducted showed that Farthest-first clustering seemed more efficient in correctly classifying my set of instances than normal k-means. Overall K-means and Farthest-First performance as the number of records increases is far better than other clustering mechanism such as the hierarchical algorithm and has time complexity of o(n). The disadvantages of algorithms like K-means and Farthest-first is that they are both highly sensitive to initial points and often get stuck in the local optimal.

**Research Question**

Calculating correlation values for categorical data

In the fast evolving fields of data mining and machine learning, scientists often perform many experiments and collect data regarding various real world problems and scenarios. Experiments are conducted to usually find some level of pattern and induce some level of prediction to help solve that particular problem. Scientists often find useful to infer some kind of association between different attributes in our dataset. These associations help us in predicting certain behaviors with regards to a subset of our features. Furthermore, they can also be used to infer and predict certain missing values that are crucial to the correct classifications of our instances. By better understanding our data, we can make better conclusions regarding the problem that we are solving (Al-Maolegi and Arkok, 2014).

Perhaps the simplest of these associations is the linear regression. Linear regression is used to find relationship between two numeric values by fitting a line through data points. That means that you would always have a response variable and an explanatory variable, in other words an x value and a y value. These variables are both present in each attribute, and the scatterplot records one variable by manipulating the other. Another approach to measuring the level of association is the correlation coefficient. This method is better than the linear regression in most cases because the majority of the time, there exists no one variable in which you experimentally manipulate. An example of a variable manipulation is time, concentration, and ratio. The correlation coefficient simply measures how much the second variable usually changes when the first variable does (Bewick et al., 2003). The top three most used types of correlation are Pearson, Kendall, and Spearman. These types are only used to measure continuous variable or ordinal variables to the least, but what if we wanted to find the correlation between a numeric and a nominal variable or between two nominal variables?

For determining correlation between numeric and nominal variables, the Analysis of Variance, or ANOVA, along with a post hoc test called the Tukey HSD can be conducted. To simply put it, the ANOVA is the difference between the means of the different categories of the nominal variable. This test performs over the null hypothesis that all the nominal category means are equal. It then proceeds to determine whether or not any of those means where significantly different than the rest. If this happens to be the case, ANOVA generally accepts the alternative hypothesis that there exists at least two group means that are significantly different than the rest. However, ANOVA cannot tell you exactly which group is different. This is where Tukey HSD comes into play to find the exact group. It compares all possible pairs of means using the formula: where “MSw” is the mean square and “nh” is the number in the group (https://statistics.laerd.com, 2013).

For determining correlation between two nominal variables, the Chi-squared test of independence can be conducted. The Chi-squared test creates a contingency table, or a frequency distribution table, as well as uses the formula where “O” is the observed value and “E” is the expected variable in the “ith” position of the contingency table. The expected value is usually calculated as the total frequency divided by the number of variables in the contingency table. This test computes a statistic number that measures the difference between your observed counts and your expected counts. Thus, the smaller this statistic, the higher the relationship or correlation between your variables, and the larger this statistic, the lower the relationship or correlation between your variables (Ugoni and Walker, 1995).

Reference:

Al-Maolegi, M, and Arkok, B, (2014). AN IMPROVED APRIORI ALGORITHM FOR ASSOCIATION RULES. International Journal on Natural Language Computing (IJNLC) 3(1).

Bewick, V, Cheeck L, Ball, J, (2003). Statistics review 7: Correlation and regression. Cirt Care, 7(6), 451-459.

statistics.laerd.com, (2013). one-way-anova-statistical-guide. [online] Available at: URL [https://statistics.laerd.com/statistical-guides/one-way-anova-statistical-guide.php].

Ugoni, A. and Walker, B.F. (1995). THE CHI SQUARE TEST: An Introduction. COMSIG Rev, 4(3), 61–64.