CO395 Machine Learning
CBC Assignment #4A
Case Based Reasoning

Group 11
Baisheng Song(bs2111) Jiayun Ding(jd1611)
Yiming Lin(yl8411) Yan Liu(ybl11)

November 30, 2014

1 Implementation Details

`n_fold_validation(examples, targets)`: This function performs the n(default 10) fold cross-validation. It splits the given arguments, examples and targets, into a train set and a test set for each fold. Then it passes the train set to `CBRinit` which initialises a CBR (Case Based Reasoning) system. Then we use `testCBR` which returns a vector of predictions for the test set. For each fold the confusion matrix is computed in `calculate_confusion_matrix`

`CBRinit(x,y)`: This function initialises and returns a CBR system. It takes examples and targets as inputs and uses `make_case` to map them to each other accordingly. It also stores the typicality of the mapping, which indicates how frequent it has appeared throughout the train set. If there are multiple equivalent mappings, it will increase the typicality of the CBR case instead of storing it twice.

`make_case(x,y)`: It takes one example and one target and returns a CBR case. Each case is a struct consisting of {AU, Class, Typicality}. The target gets mapped to the example in the case. Each cases typicality is initialised to 1.

`testCBR(CBR,x2)`: This function produces a vector of the corresponding prediction of targets from the unseen examples. For each example it will predict the target by the best suited appromate case from the CBR system. After that it stores the new case to the CBR system.

`retrieve(cbr, newcase)`: It scans through the CBR system and returns an array with the 5 closed cases to the newcase and an array with their respective distances. All CBR cases that have the same distance as the last array element will also be appended to the array.

`reuse(cases,newcase,cbr)`: Reuse performs a voting of the (k=5) closest cases and return the newcase an emotion class with the highest vote. This voting is performed by multiplying the typicality and the inversion of similaritiy between the closest cases and the newcase. Then, it is devided by the total number of typicality in the cbr for each class of emotion.

`retain(cbr, newcase)`: This function saves a new case to the existing cbr system. If the same case is already inside the system, the typicality of it will be increased by one, else the new case will be appended.

2 Evaluation

The results presented are for the modified Distance-weighted 5-NN algorithm with Manhattan distance on the clean dataset. For noisy dataset, we used 8-NN with Manhatte We chose these distances and values for k because they performed best during our tests.

2.1 Clean Dataset

2.1.1 Confusion Matrix

This is the sum of ten confusion matrices for each fold.

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**105**	11	4	1	10	1
	2	9	**174**	1	6	7	1
	3	4	3	**96**	1	4	11
	4	0	5	1	**208**	0	2
	5	7	17	1	3	**102**	2
	6	0	0	8	3	1	**195**

Table 1: Confusion Matrix for the *Clean* Dataset

2.1.2 Average Recall, Precision and F1 Measures

		Recall	Precision	F_1
Actual class	1	79.55%	84.00%	81.71%
	2	87.88%	82.86%	85.29%
	3	80.67%	86.49%	83.48%
	4	96.30%	93.69%	94.98%
	5	77.27%	82.26%	79.69%
	6	94.20%	91.98%	93.08%

Table 2: Average Recall, Precision and F_1 Measure for the *Clean* Dataset

2.1.3 Average Classification Rate

$$CR = \frac{880}{1004} = 87.6\%$$

Figure 1: Classification Rate for the *Clean* Dataset using Confusion Matrix

2.2 Noisy Dataset

2.2.1 Confusion Matrix

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**48**	13	13	3	10	1
	2	8	**161**	8	4	4	2
	3	23	13	**113**	14	6	18
	4	10	7	8	**175**	2	7
	5	32	10	8	2	**54**	4
	6	5	2	10	4	12	**187**

Table 3: Confusion Matrix for the *Noisy* Dataset

2.2.2 Average Recall, Precision and F1 Measures

		Recall	Precision	F_1
Actual class	1	54.55%	38.10%	44.86%
	2	86.10%	78.16%	81.93%
	3	60.43%	70.63%	65.13%
	4	83.73%	86.63%	85.16%
	5	49.09%	61.63%	54.55%
	6	85.00%	85.39%	82.64%

Table 4: Average Recall, Precision and F_1 Measure for the *Noisy* Dataset

2.2.3 Average Classification Rate

$$CR = \frac{735}{1001} = 73.4\%$$

Figure 2: Classification Rate for the *Noisy* Dataset using Confusion Matrix

2.3 Discussion of Results Part VIII

We had a high classification rate on the clean dataset using Case Based Reasoning, which is 87.6%; a bit higher than the cr using Artificial Neural Networks and way higher than the cr using Decision Trees learning. This result indicates that the Case Based Reasoning is the best learning algorithm for application to emotion recognition based on provided dataset. The Case Based Reasoning can calculate a different, local approximation to the tagrget function for each encountered new example which reduces the complexity of the problem domain. Since Case Based Reasoning which is a lazy learning algorithm is very suitable for complex problem domains, and emotion recognition does have a complex domain, we believe that it is the reason why Case Based Reasoning performed so well in this particular emotion recognition task.

Having run the simulation on the noisy dataset, we observed an expected decrease in the performance for all emotions. The classification rate was 73.4% which was better than the

results for DT and ANN. But the F1 for emotion 1 suggests that the system was still struggling to detect similarity between instances of this emotion. Also the F1 for emotion 3 and 5 were also low, but for 2, 4 and 6 was high. This maybe because there were more samples for these emotions in the dataset.

CBR performed very good compared to the DT and the ANN. We had the highest classification rates for both clean and noisy datasets.

3 Questions

3.1 Two or more best matches

We avoid such a situation from how the program selects cases. Basicly, the program does the calculation as descriped by the equation: score(cases.class) $= \sum \frac{\frac{1}{distance(cases,newcase)} * typicality(cases)}{total\ number\ of\ examples}$ for each emotion. Then select the class with the highest score.

We barely have any two cases with the same score, since the total number of examples for each class is barely the same. But if it is the case, then in our code, we reduce the k number to k-1, and keep reducing until each case has a unique score. If two cases still have the same x number with k=1, we use the random choice. But that is not likely the case.

3.2 Adding an existing case

In both the CBR initialisation and the retain function we scan through the CBR system and see if there already exists such a case with the same AU and class assignment. Instead of adding a new case to the CBR, we increase the typicality of the existing case.

3.3 Comparison of similarity measures

We used three similarity measures, which are Manhattan distance, Euclidean distance, Chebyshev distance and Jaccard distance.

Manhattan distance is the sum of distances in all dimensions. Advantage is it takes all dimensions into account. Also, it is quick to calculate. But it is unfair in the case of actual distance in geometry. But our best performance is based on Manhattan distance, maybe because the examples are only binary.

Euclidean distance is the square root of the sum of the squares of distances. It is more like the actual geometry distance between two points. For examples, in Manhattan distance, distance between (0,0) and (1,1) and distance between (0,0) and (0,2) are both 2. This is unfair, but as to Euclidean distance (1,1) will have a distance smaller than (0,2) which is reasonable. But since for this particular domain, all dimensions are whether 0 or 1, so Euclidean distance is not much different from Manhattan distance.

Chebyshev distance is poor on this case. Since the max of the distances is always 1 or 0 (exact much).

Jaccard distance is defined as the size of the intersection divided by the size of the union of the sample sets. Advantage of it is proportionately characterized the similarity to difference of the sample. Disadvantage is it runs the slowest and uses the most memory. We were hopeful

this would prove to be a good similarity measure, as it is intended to be used when finding the binary differences or similarities between two objects. It did prove to give a good accuracy however Manhattan similarity gave a better fit, so we opted with that in the final submission.

3.4 Initialisation of CBR

First, we we initialised an array of struct{Class, AU, Typicality}. For each AU and class combination we scan through the existing CBR array. If the AU and class combination are already registered in the CBR array, we increment the typicality of that field, else we append the them as a new struct to the CBR system.

3.5 Classes of learning algorithms

CBR uses the lazy learning algorithm. Compared to the eager learning method which is used in decision trees and neural networks, lazy learning does not try to construct a target function based on the training examples. It just stores the data inside the CBR system. Instead of following the set of rules to obtain the solution like in the case of decision trees and neural networks, CBR reapplies and adapts previous successful solutions to the new problems.

3.6 Clean vs. noisy dataset

The overall performance on the clean dataset is much better than the performance on the noisy dataset. There are some consistencies between the clean and noisy dataset. For example, emotion 4 performs the best among all six emotions on both clean and noisy dataset, but surprisely emotion 1 does not perfrom the worst on the clean dataset but appears to be the worst on the noisy dataset, it may because there are more error data in the noisy dataset for emotion 1. The highest recall, precision, f1 we get from the clean dataset are 96.30%, 93.69%, 94.98%,all in emotion 4. For the noisy dataset, the hightest recall appears in emotion 2(86.10%), the highest precision and f1 both appear in emotion 4(86.63% and 85.16% respectively). They are all abvious indicators of conclusions gained in the first place.