Machine Learning Project Report

Rodrigo Castiel
Center for Informatics
Federal University of Pernambuco
Recife, Brazil
rcrs2@cin.ufpe.br

***Abstract*—In the first part of this report, we compare k-means to KCM-F-GH, its hard clustering variation in feature space proposed by Cunha et al [?]. Both methods are evaluated on the segmentation dataset available in [?] by computing the adjusted rand index to analyze the similarity between the predicted cluster solutions and the ground-truth labels. In the second part, we perform a comparative analysis between three classifiers: a maximum likelihood gaussian estimator, a k-nearest neighbors classifier and a hybrid-model committee classifier. Those classifiers are also tested on the segmentation dataset. We run cross-validation multiple times to estimate the accuracy and the error margin of each classifier. At the end, we perform Friedman test to determine the best algorithm. Our results show that KCM-F-GH outperforms the standard k-means in most cases, with the tradeoff of being considerably more time-consuming. In the second part, our experiments show that both KNN and the hybrid classifier are equally accurate in the dataset, and better than the maximum-likelihood estimator.**

***Index Terms*—clustering, supervised learning, classification, pattern recognition**

I. Introduction

This research report contains two main sections. The first one briefly describes how k-means and KCM-F-GH were implemented, and how they performed in the experiments. In the second one, we point some implementation strategies for the maximum likelihood gaussian estimator (MLE), the k-nearest neighbors classifier (KNN) and the hybrid-model committee classifier. Then, we compare their accuracy in a series of experiments.

The *Python* code, the development history and the dataset are all available on Rodrigo Castiel's personal github (click here). Basic repository structure:

- *part_1.py*: main script for running and comparing the clustering algorithms.
- *part_2.py*: main script for running and comparing the supervised classifiers.
- *classifiers*: contains a list of classes, each implementing a classifier or a clustering algorithm.
- *core*: contains *data_loader.py*, a utility module for managing the segmentation data.
- *data*: training and test datasets.

Additionally, we use *Numpy* and *Scikit*, *Python* libraries for linear algebra, statistics and learning utilities.

II. Part I - Clustering

A. Implementation

The test code start point is located in *part_1.py*. The script arguments, to be passed in from the terminal, are a list of views which will be tested. For example the list *RGB* and *SHAPE* tells the program to run both k-means and KCM-F-GH on the RGB and the shape view, separately. For each view, the script then builds each classifier and calls the method fit on it. The parameter num_times controls how many times KCM-F-GH will be executed before the best fit run is taken. The implementation of k-means and KCM-F-GH are located in *k_means_clustering.py* and *kcm_f_gh_clustering.py*, respectively. They are both thoroughly documented.

Note. During the initial tests of KCM-F-GH, we noticed that the update of the hyper-width parameters is not robust to constant-valued features. That is, if at a given moment a specific feature becomes constant within each cluster, the denominator of equation (24) becomes 0 [**?**], which breaks the execution. It means that before actually running this algorithm in feature space, we must remove redudant dimensions in the dataset. For this reason, we removed the features "REGION-PIXEL-COUNT" and "SHORT-LINE-DENSITY-2" from the shape view.

B. Experiments and Results

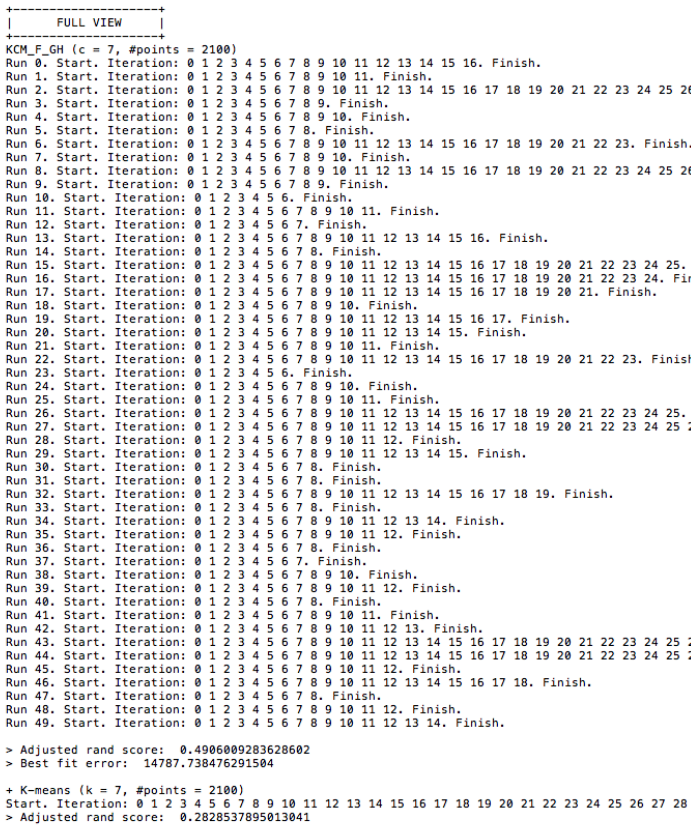
In the first experiments, we noticed that KCM-F-GH takes an average of approximately one hour to converge on the test dataset containing 2100 points. Since our personal computer's hardware is not powerful, executing KCM-F-GH 100 times per view is impossible in practice. Instead, we run it only 10 times per view.

III. Part II - Supervised Learning

A. Implementation

The test code start point is located in *part_2.py*. In the first moment, we perform a grid search in the training dataset to find the best hyper-parameters for the classifiers. Then, we run cross-validation multiple times in the training dataset to estimate the accuracy (and its standard deviation) for each classifier. After that, we run the classifiers on the test dataset to evaluate the overall accuracy. Last, we perform the Friedman test to compare all pairs of classifiers. The program writes the output log into *part_2_log.txt*.

Each classifier contains two main methods: *fit* and *predict*.

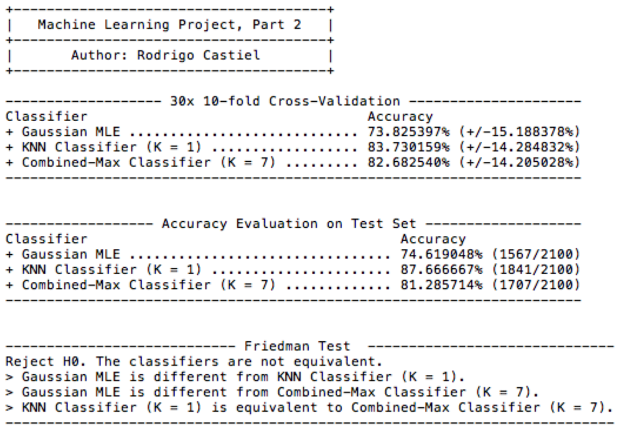


```
+--------------------+
|      FULL VIEW     |
+--------------------+
KCM_F_GH (c = 7, #points = 2100)
Run 0. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16. Finish.
Run 1. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11. Finish.
Run 2. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 2
Run 3. Start. Iteration: 0 1 2 3 4 5 6 7 8 9. Finish.
Run 4. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10. Finish.
Run 5. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 6. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23. Finish.
Run 7. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10. Finish.
Run 8. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 2
Run 9. Start. Iteration: 0 1 2 3 4 5 6 7 8 9. Finish.
Run 10. Start. Iteration: 0 1 2 3 4 5 6. Finish.
Run 11. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11. Finish.
Run 12. Start. Iteration: 0 1 2 3 4 5 6 7. Finish.
Run 13. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16. Finish.
Run 14. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 15. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25.
Run 16. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24. Fi
Run 17. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21. Finish.
Run 18. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10. Finish.
Run 19. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17. Finish.
Run 20. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15. Finish.
Run 21. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11. Finish.
Run 22. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23. Finis
Run 23. Start. Iteration: 0 1 2 3 4 5 6. Finish.
Run 24. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10. Finish.
Run 25. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11. Finish.
Run 26. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25.
Run 27. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Run 28. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12. Finish.
Run 29. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15. Finish.
Run 30. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 31. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 32. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19. Finish.
Run 33. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 34. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14. Finish.
Run 35. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12. Finish.
Run 36. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 37. Start. Iteration: 0 1 2 3 4 5 6 7. Finish.
Run 38. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10. Finish.
Run 39. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12. Finish.
Run 40. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 41. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11. Finish.
Run 42. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13. Finish.
Run 43. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Run 44. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Run 45. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12. Finish.
Run 46. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18. Finish.
Run 47. Start. Iteration: 0 1 2 3 4 5 6 7 8. Finish.
Run 48. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12. Finish.
Run 49. Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14. Finish.

> Adjusted rand score:  0.4906009283628602
> Best fit error:  14787.738476291504

+ K-means (k = 7, #points = 2100)
Start. Iteration: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
> Adjusted rand score:  0.2828537895013041
```

Fig. 1. Output log of Part I on full view (RGB + shape). We run KCM-F-GH $num\_times = 50$ with random initial representatives in each cluster. Then, we pick the best fitting KCM-F-GH and compute its adjusted rand index (see bottom). KCM-F-GH achieved 0.4906 while k-means achieved 0.2826.



```
+------------------------------------+
|   Machine Learning Project, Part 2  |
+------------------------------------+
|       Author: Rodrigo Castiel       |
+------------------------------------+

------------------- 30x 10-fold Cross-Validation ---------------------
Classifier                                 Accuracy
+ Gaussian MLE ........................... 73.825397% (+/-15.188378%)
+ KNN Classifier (K = 1) ................. 83.730159% (+/-14.284832%)
+ Combined-Max Classifier (K = 7) ......... 82.682540% (+/-14.205028%)
----------------------------------------------------------------------


------------------ Accuracy Evaluation on Test Set -------------------
Classifier                                     Accuracy
+ Gaussian MLE ............................... 74.619048% (1567/2100)
+ KNN Classifier (K = 1) ..................... 87.666667% (1841/2100)
+ Combined-Max Classifier (K = 7) ............ 81.285714% (1707/2100)
----------------------------------------------------------------------


---------------------------- Friedman Test  ------------------------------
Reject H0. The classifiers are not equivalent.
> Gaussian MLE is different from KNN Classifier (K = 1).
> Gaussian MLE is different from Combined-Max Classifier (K = 7).
> KNN Classifier (K = 1) is equivalent to Combined-Max Classifier (K = 7).
--------------------------------------------------------------------------
```

Fig. 2. Output log of Part II. The first section shows the average accuracy and its error margin measured by running 10-fold cross-validation 30 times on the training dataset. The second section shows the overall accuracy measured on the test dataset. The last section shows the Friedman test result (on the training dataset).

- *gaussian_mle.py*: defines class *GaussianMLE*. In its method *fit*, the training dataset is split up into subsets sharing the same label. Then, the gaussian distribution parameters (*i.e.*, the mean μ and the covariance matrix Σ) are estimated for each class. Once trained, the MLE classifier is also able to compute the a posteriori probabilities for a given point. This is needed for the hybrid classifier.
- *knn_classifier.py*: defines class *KNNClassifier*. It performs no computation in the method *fit*. Instead, it only stores the labeled points. In the method *predict*, we then compute the distances from each input point to all examples and use a heap to keep track of the nearest k points. The predicted label is computed by picking the class of the majority.
- *combinex_max_classifier.py*: defines class *CombinedMaxClassifier*, the hybrid classifier. It takes a list of views to be tested on the dataset, and the hyper-parameter k for its KNN classifiers. Internally, it performs a cartesian product between the list of classifiers (MLE and KNN) and all views. It forwards *fit* and *predict* to all its classifier-view pairs, and returns the combined answer.

B. *Experiments and Results*

Figure III-B was taken from the output log of Part II. Grid-search finds that the best hyper-parameters are $k = 1$ for KNN and $k = 7$ for the hybrid method. As we can see in the cross-validation table, Gaussian MLE has the lowest accuracy among all classifiers ($\approx 74\%$), while both hybrid and KNN achieve similar accuracies ($\approx 83\%$). On the complete test set containing 2100 points, KNN achieves the highest accuracy, followed by hybrid and then Gaussian MLE. In fact according to the Friedman test results, KNN and hyrid are statistically equivalent.

When it comes to the comparative computational costs, KNN is certainly cheaper than hybrid, since hybrid needs to run multiple classifiers internally before combining their answers. Gaussian MLE is indeed the cheapest method, because its prediction consists only of simple likelihood computations.

IV. Conclusion

blah blah blah blah blah blah.

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first . . ."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.