Shape classification using supervised classifiers

Julen Etxaniz and Ibon Urbina

Abstract

In this document we report our proposal for the application of supervised classification methods to the shape classification problem. The datasets we have selected are plane and car shapes. The classifiers that we have selected for the classification tasks were: Decision Trees, Linear discriminant analysis and Logistic regression. We have implemented the classification process primarily using the scikit-learn and pandas library. We have learned the classifiers using the train data and computed the accuracy in the test data. We have performed several additional tasks: scaling, feature selection, feature engineering and pipeline optimization. On the one hand, we have concluded that the best option for plane data is Random Forest feature selection and LDA classifier. On the other hand, the best option for car data is LDA feature engineering and LDA classifier.

Contents

1 Description of the problem

The task we have to solve is the classification of the shape datasets, introduced in [1] [2] [3] and available in `http://biomecis.uta.edu/shape_data.htm`.

There are 4 shape datasets available. We have selected the plane and car datasets. In each dataset, shapes are represented as Cartesian coordinates of each point on the perimeter; all stored as MATLAB data files named as $Class\%d\_Sample\%d.mat$.

1.1 Plane dataset

Shape database was created by taking 640X480 resolution pictures of replica models of airplanes from top. The plane dataset has the following characteristics:

- $890 * 2 + 1 = 1781$ features. The minimum x and y values and perimeter length.
- 7 classes: (a) Mirage, (b) Eurofighter, (c) F-14 wings closed, (d) F-14 wings opened, (e) Harrier, (f) F-22, (g) F-15. See Figure 1.
- 210 instances.
- The data is perfectly balanced. There are 30 instances in each class.

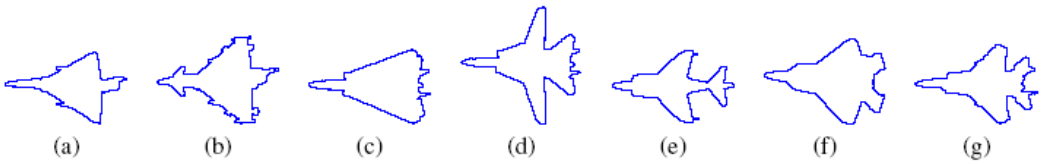

Figure 1: 7 plane classes: (a) Mirage, (b) Eurofighter, (c) F-14 wings closed, (d) F-14 wings opened, (e) Harrier, (f) F-22, (g) F-15.

1.2 Car dataset

Approach discussed in [4] was applied to 320X240 resolution videos to extract these shapes. As object extraction approach does not deal with shadows, the shapes are distorted in the bottom half. Samples show larger within-class variation, because vehicles of different makes and models vary. The car dataset has the following characteristics:

- $272 * 2 + 1 = 545$ features. The minimum x and y values and perimeter length.
- 4 classes: (a) Sedan, (b) Pickup, (c) Minivan, (d) SUV. See Figure 2.
- 120 instances.
- The data is perfectly balanced. There are 30 instances in each class.

2 Description of our approach

We organized the implementation of the project according to the tasks:

1. Importing the libraries
2. Reading the datasets
3. Preprocessing the datasets
4. Scaling the data
5. Dividing train and test data
6. Classification
7. Validation
8. Feature Selection
9. Feature Extraction
10. Pipeline Optimization

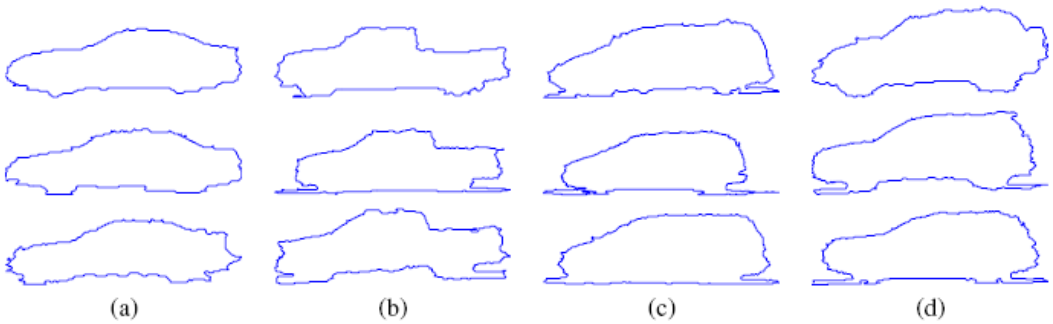

Figure 2: 4 car classes: (a) Sedan, (b) Pickup, (c) Minivan, (d) SUV.

2.1 Importing the libraries

We import all the necessary libraries. Firstly, we used scipy.io.loadmat to read the dataset. Next, we used *pandas* and numpy for preprocessing the dataset, and scikit-learn for the classification tasks. Moreover, we used matplotlib for viasualization. Finally, we used TPOT for pipeline optimization.

2.2 Reading the datasets

The main problem of both plane and car datasets is that each sample is a MATLAB file, which makes it to be only visible in MATLAB. To solve this, we use a helper function from the scipy.io package called loadmat which makes the data readable in python.

2.3 Preprocessing

We used the pandas library to represent the datasets as dataFrames. This structure makes the data flexible, easy to manipulate and good looking.

The main problem of both plane and car datasets is that each sample is a MATLAB file, which makes it to be only visible in MATLAB. To solve this, we use a helper function from the scipy.io package called loadmat which makes the data readable in python.

First, we collected the data in python lists and then transferred them to pandas DataFrame. When loading data with loadmat function it includes some information such as header, version and globals that is not useful for our classification; thus, we dropped them from the dataFrame.

As we mention before, the only data available in each shape sample are the points that conform the perimeter. It is not mentioned to what class corresponds that data. This has to be taken from the file name which is a string concatenation of: Class + class number + Sample + sample number + .mat.

Finally, to finish with the dataFrame representation, 1) we added perimeter length feature and 2) decided to change the way the points on the perimeter were described. Regarding to the first one, we thought that normalizing each sample array to the minimum perimeter length found in each dataset would make a more consistent and organized data: in the plane dataset the minimum perimeter corresponds to a sample formed by 890 and in the car dataset, corresponds to 272 points. Of course, doing this, the original length of the perimeter is lost; so, before resizing it, we stored the perimeter in a feature called Perimeter length.

On the other hand, the feature responsible for structuring the points on the perimeter is a single list containing one list for each sample that once again, contains a list of two numbers to represent a concrete point. The first number corresponds to the axis x and the second one to the axis y. Taking into account that in principle features must be unique values (either number or strings) and not lists, our structure is not a valid input for the classifiers when fitting them. In this situation, we made the decision of making one feature per axis value. So, if one sample is defined with this unique feature: [[x1, x2], [x2, y2], ..., [xn, yn]]. Now, is going to be defined by n*2 features: [x1, y1, x2, y2, ..., xn, yn].

2.4 Scaling the data

We scaled the data using $StandardScaler$. This scaler standardizes features by removing the mean and scaling to unit variance. Scaling the data before classification is recommended for some classifiers because their performance is better.

Parameters:

- copy=True
- with_mean=True
- with_std=True

2.5 Dividing train and test data

The data was split into two different sets of the same length: train and test; even numbers for the first one and odd ones for the second. Again both sets are divided in two: features (perimeter length, axes x and axes y) and targets(the corresponding class of each sample).

Train and data sets are essential for the classifier. The first one is used in the learning process and the second one in the measure of its performance.

2.6 Classification

We used three different classifiers. The parameters that are different from default are in **bold**.

1. Decision trees with parameters:

   - criterion='gini'
   - splitter='best'
   - max_depth=None
   - min_samples_split=2
   - min_samples_leaf=1
   - min_weight_fraction_leaf=0.0
   - max_features=None
   - random_state=None
   - max_leaf_nodes=None
   - min_impurity_decrease=0.0
   - min_impurity_split=None
   - class_weight=None
   - presort=False

2. Linear discriminant analysis with parameters:

   - solver='svd'
   - shrinkage=None
   - priors=None
   - n_components=None
   - store_covariance=False
   - tol=0.0001

3. Logistic regression with parameters:

   - penalty='l2'
   - dual=False
   - tol=0.0001
   - C=1.0
   - fit_intercept=True
   - intercept_scaling=1
   - class_weight=None
   - random_state=None
   - solver='liblinear'
   - **max_iter=2000**
   - multi_class='ovr'
   - verbose=0,
   - warm_start=False
   - n_jobs=1

2.7 Validation

To validate our results we computed the classifier accuracies in the test data for the three classifiers.

Another possibility was to compute the cross-validation in the complete dataset but we used the split between train and test because we assumed that either training and test data were sufficient.

As an additional validation step we computed the confusion matrices for the three classifiers.

2.8 Feature Selection

We used feature selection to reduce the number of features. We had too many features, two features for each point in the contour perimeter, which makes a total number of 1781 features for the plane dataset: 2*890 (number of points in the perimeter) + 1 (the feature corresponding to the perimeter length) and 545 features for the car dataset: 2*272 (number of points in the perimeter) + 1 (the feature corresponding to the perimeter length). Remember that we have normalized each contour perimeter to the minimum length perimeter found in each dataset.

We used 2 different methods for feature selection: *SelectKBest* and *RandomForest*. Both being supervised methods as they use each feature list and corresponding labels.

The parameters of *SelectKBest*:

- score_func=f_classif
- score_func=f_classif
- score_func=f_classif
- score_func=f_classif
- score_func=f_classif
- **k=100**
- **k=200**
- **k=300**
- **k=400**
- **k=500**

The parameters of *RandomForest*:

- **n_estimators=1-60**
- criterion='gini'
- max_depth=None
- min_samples_split=2
- min_samples_leaf=1
- min_weight_fraction_leaf=0.0
- max_features='auto'
- max_leaf_nodes=None
- min_impurity_decrease=0.0
- min_impurity_split=None
- bootstrap=True
- oob_score=False
- n_jobs=None
- random_state=None
- verbose=0
- warm_start=False
- class_weight=None
- ccp_alpha=0.0
- max_samples=None

2.9 Feature Engineering

As mentioned above, our features length was a problem. Thus, we tried feature engineering for features dimensionality reduction. We extracted 3 feature so that we can plot them in 3 dimensions.

We used 2 different methods for feature engineering: *PCA* and *LDA*. The first one is an unsupervised technique as it tries to find the directions of maximal variances, while the second one attempts to find a features subspace that maximizes class separability. Logically, to separate one class from another needs to know the corresponding feature-class information, making it easier with supervised technique.

The parameters of PCA:

- **n_components=3**
- copy=True
- whiten=False
- svd_solver='auto'
- tol=0.0
- iterated_power='auto'
- random_state=None

The parameters of LDA:

- solver='svd'
- shrinkage=None
- priors=None
- **n_components=3**
- store_covariance=False
- tol=0.0001

2.10 Pipeline Optimization

We used TPOT to generate an optimal pipeline to compare its accuracy with ours. This way we can know how good our classifiers are. We can compare the accuracies obtained with TPOT with ours.

The parameters of the TPOT classifier:

- **generations=5**
- **population_size=10**
- offspring_size=None
- mutation_rate=0.9
- crossover_rate=0.1
- scoring='accuracy'
- cv=5
- subsample=1.0
- n_jobs=1
- max_time_mins=None
- max_eval_time_mins=5
- **random_state=16**
- config_dict=None
- template=None
- warm_start=False
- memory=None
- use_dask=False
- periodic_checkpoint_folder=None
- early_stop=None
- **verbosity=2**
- disable_update_check=False
- log_file=None

3 Implementation

All the project steps were implemented in Python. Firstly, we used scipy.io.loadmat to read the dataset. Next, we used *pandas* and numpy for preprocessing the dataset, and scikit-learn for the classification tasks. Moreover, we used matplotlib for viasualization. Finally, we used TPOT for pipeline optimization.

We illustrate how the implementation works with explanations in the Jupyter notebook `P59_Shape_Classification_ML_D2.ipynb`. In most of the cases, we defined the functions first and then used them. This avoids repeating the code in different steps or on the same step with different datasets. The project description, the notebook, the dataset and the written report are available at `https://github.com/juletx/shape-classification-ml`.

4 Results

4.1 Not scaled data

4.1.1 Plane dataset

The accuracies produced by the Decision tree, LDA and Logistic regression classifiers were, respectively: 0.7714, 0.9523, and 0.9428. Therefore, the best classifier for not scaled plane data was LDA. Decision tree obtains a very low accuracy compared to the other two.

The results of the computation of the confusion matrices for Decision tree, LDA and Logistic regression classifiers are respectively shown in Tables 1, 2, and 3.

The analysis of the confusion matrices shows that the most difficult discrimination is for the first 2 classes, Mirage and Eurofighter. This occurs because they are mistaken with other planes. For example, the Eurofighter is commonly mistaken with F-15.

col_0 Class	1	2	3	4	5	6	7
1	7	2	2	1	0	1	2
2	2	8	0	0	1	0	4
3	1	1	10	0	0	2	1
4	0	1	0	14	0	0	0
5	0	0	0	0	14	1	0
6	0	0	0	0	0	15	0
7	2	0	0	0	0	0	13

Table 1: Confusion matrix produced by the Decision tree classifier on plane data

col_0 Class	1	2	3	4	5	6	7
1	13	0	0	0	0	2	0
2	0	12	0	0	0	0	3
3	0	0	15	0	0	0	0
4	0	0	0	15	0	0	0
5	0	0	0	0	15	0	0
6	0	0	0	0	0	15	0
7	0	0	0	0	0	0	15

Table 2: Confusion matrix produced by the LDA classifier on plane data

col_0 Class	1	2	3	4	5	6	7
1	13	0	1	0	0	1	0
2	1	11	0	0	0	0	3
3	0	0	15	0	0	0	0
4	0	0	0	15	0	0	0
5	0	0	0	0	15	0	0
6	0	0	0	0	0	15	0
7	0	0	0	0	0	0	15

Table 3: Confusion matrix produced by the Logistic regression classifier on plane data

col_0 Class	1	2	3	4
1	13	0	0	2
2	0	13	1	1
3	0	0	15	0
4	0	7	2	6

Table 4: Confusion matrix produced by the Decision tree classifier on car data

col_0 Class	1	2	3	4
1	13	0	0	2
2	0	15	0	0
3	0	1	13	1
4	0	0	2	13

Table 5: Confusion matrix produced by the LDA classifier on car data

4.1.2 Car dataset

The accuracies produced by the Decision tree, LDA and Logistic regression classifiers were, respectively: 0.7833, 0.9000, and 0.8666. Therefore, the best classifier for not scaled car data was LDA.

The results of the computation of the confusion matrices for Decision tree, LDA and Logistic regression classifiers are respectively shown in Tables 4, 5, and 6.

The analysis of the confusion matrices shows that there are more differences between classifiers. Decision tree for example has by far the most problems with class 4. The others classifiers make less mistakes but we can also see slight differences between classes.

4.2 Scaled data

The accuracies produced by the Decision tree, LDA and Logistic regression classifiers were, respectively: 0.7809, 0.9523, and 0.9238. Therefore, the best classifier for scaled plane data was LDA. We can see that the accuracy for Decision tree increased, LDA is the same and Logistic regression decreased.

The accuracies produced by the Decision tree, LDA and Logistic regression classifiers were, respectively: 0.8333, 0.9000, and 0.9166. Therefore, the best classifier for scaled car data was Logistic regression. We can see that the accuracy for Decision tree increased, LDA is the same and Logistic regression increased.

4.3 Conclusions regarding to scaling or not scaling the data

For our thoughts the results were kind of unexpected. We thought that scaling the data would be decisive regarding to improving the accuracy scores (things were not so decisive). However, best

col_0 Class	1	2	3	4
1	14	0	1	0
2	0	14	1	0
3	0	1	12	2
4	0	0	3	12

Table 6: Confusion matrix produced by the Logistic regression classifier on car data

results in the plane dataset were given by the LDA classifier and match exactly with the percentage obtain from the scaled and not scaled data: 0.9523. On the other hand, for the car dataset we obtained a better score with the scaled data; so here yes, our prognostic was succeed : 0.8666 vs 0.9166 in the Logistic regression classifier.

4.4 Feature Selection

4.4.1 SelectKBest

For this method, we decided to compute the accuracy evolution starting with 100 features and ending with 500. Each time, we summed up 100 features to the algorithm. This process was done either for scaled and not scaled data.

Plane dataset: accuracy scores regarding to the selected number of features **without scaling** them:

- 100 features:
  DT: 0.5619
  LDA: 0.2571
  LR: 0.6190
- 200 features
  DT: 0.6761
  LDA: 0.4952
  LR: 0.6285
- 300 features
  DT: 0.7333
  LDA: 0.7047
  LR: 0.7047
- 400 features
  DT: 0.7523
  LDA: 0.7714
  LR: 0.8190
- 500 features
  DT: 0.7142
  LDA: 0.7809
  LR: 0.8761

Plane dataset: accuracy scores regarding to the selected number of features **scaling** them:

- 100 features:
  DT: 0.5047
  LDA: 0.2571
  LR: 0.6190
- 200 features
  DT: 0.6571
  LDA: 0.4952
  LR: 0.6285
- 300 features
  DT: 0.7238
  LDA: 0.7047
  LR: 0.7047
- 400 features
  DT: 0.7619
  LDA: 0.7714
  LR: 0.8190
- 500 features
  DT: 0.7333
  LDA: 0.7809
  LR: 0.8761

Car dataset: accuracy scores regarding to the selected number of features **without scaling** them:

- 100 features:
  DT: 0.7666
  LDA: 0.4333
  LR: 0.6
- 200 features
  DT: 0.8333
  LDA: 0.6666
  LR: 0.7166
- 300 features
  DT: 0.8166
  LDA: 0.8333
  LR: 0.8833
- 400 features
  DT: 0.8666
  LDA: 0.8666
  LR: 0.85
- 500 features
  DT: 0.85
  LDA: 0.8833
  LR: 0.8666

Car dataset: accuracy scores regarding to the selected number of features **scaling** them:

- 100 features:
  DT: 0.7833
  LDA: 0.4333
  LR: 0.6
- 200 features
  DT: 0.8166
  LDA: 0.6666
  LR: 0.7166
- 300 features
  DT: 0.75
  LDA: 0.8333
  LR: 0.8833
- 400 features
  DT: 0.8333
  LDA: 0.8666
  LR: 0.85
- 500 features
  DT: 0.75
  LDA: 0.8833
  LR: 0.8666

4.5 Conclusions regarding to SelectKBest algorithm

The main conclusion we reach in this process is that not always more features implies better accuracy scores; it depends also, for example, in the classifier you are using. If we take a look in the accuracy score (plane datases + not scaled) and observe the scores obtained with 300 features and 500 features in the decision tree classifier, we see that with less features it obtains an accuracy of 0.7333 and with 500, 0.7142. But, for the logistic regression classifier, with 300 features obtain a 0.7047 and with 500, 0.8761.

4.5.1 RandomForest

RandomForest is a good algorithm for feature selection. As we see in the theorical classes, the algorithm to learn decision trees adopts a greedy divide-and-conquer strategy selecting always the most important attribute first. The most important attribute is the one that makes the most important difference to the classification; and this, is usually measured by a rank where the features make the maximal impurity decrease.

However, there is no way to know fully certain which is the number of trees that makes the best feature selection. The reason behind this is that RandomForest is an ensemble of decision trees; each one learned on a bootstrap sample. This means that each bootstrap sample is computed using a subset of features/variables that are selected randomly each time.

To avoid this and quantify the best performance for the number of trees used when selecting the features that make the best accuracy score, we measured ten different trials. Each trial, consisted on selecting from 1 to 60 the number of trees used and then, computing the accuracy scores. All this is constrained by a condition: we are able to to sum up one tree to the current tree numbers parameter if the current selected number of trees has obtained a better accuracy score than the previous numbers of trees obtained. In each trial, the obtained maximum accuracy score, the name of the classifier with which we obtain that score and the number of features used to obtain the maximum scores were taken.

Plane dataset, not scaled: results regarding to the RandomForest algorithm (making a mean of the 10 trials):

- More often appeared classifier: LDA
- Mean number of used trees: 8
- Mean number of used features: 81
- Mean accuracy score: 0.9637

Car dataset, not scaled: results regarding to the RandomForest algorithm (making a mean of the 10 trials):

- More often appeared classifier: LG
- Mean number of used trees: 6

- Mean number of used features: 44
- Mean accuracy score: 0.8833

For both datasets it was worth using RandomForest feature section process against SelectKBest algorithm, as it obtained better ratio regarding to selected/used features and accuracy scores.

4.6 Feature Engineering

4.6.1 PCA

The accuracies produced by the LDA classifier on PCA reduced plane data was 0.6190. The accuracy obtained is much lower to using all features. Therefore, PCA is not a good option for feature engineering or dimensionality reduction.

The accuracies produced by the LDA classifier on PCA reduced car data was 0.5166. The accuracy obtained is much lower to using all features. Therefore, PCA is not a good option for feature engineering or dimensionality reduction.

4.6.2 LDA

The accuracies produced by the LDA classifier on LDA reduced plane data were 0.9333 and 0.9523 for not scaled and scaled data, respectively. The accuracy obtained is similar to using all features. Therefore, PCA is a good option for feature engineering or dimensionality reduction.

The accuracies produced by the LDA classifier on LDA reduced car data was 0.9000 and 0.9000 for not scaled and scaled data, respectively. The accuracy obtained is similar to using all features. Therefore, PCA is a good option for feature engineering or dimensionality reduction.

4.7 Pipeline Optimization

We have realised that TPOT produces different results sometimes and we know that the produced pipeline is not always the optimal. But, we think that it is still interesting to have an idea of how good our solution is.

The best internal CV score of the pipeline obtained by TPOT on the **plane data** was 0.9714. The accuracy on test data was 0.9142. The pipeline we obtained was:

[('linearsvc', LinearSVC(C=25.0, dual=False, random_state=16, tol=1e-05))]

The accuracy obtained is a lot lower than our best option. So, we can say that our classifier is very good.

The best internal CV score of the pipeline obtained by TPOT on the **car data** was 0.8166. The accuracy on test data was 0.9166. The pipeline we obtained was:

[('gradientboostingclassifier', GradientBoostingClassifier (learning_rate=0.5, max_depth=4, max_features=0.4, min_samples_leaf=5, min_samples_split=8, random_state=16, subsample=0.5))]

The accuracy obtained is a bit higher than our best option. So, we can say that our classifier is quite good.

5 Conclusions

As you may see, we have made a huge experiment trying out different algorithms regarding to data preprocessing, classifiers, feature selection and feature engineering. For example, at first we obtained lower accuracies after feature selection and engineering. So, we had to try different options to improve the accuracy.

In general, results were quite good, but the accuracy for car data was significantly lower. We think that the reason for obtaining a lower accuracy in the car data is that the images are more similar and distorted. We also saw that even if the datasets are similar the best solution is not the same.

For our databases and used algorithms, this is the best solution combination we found:

For the **plane dataset**:

- Not scaled data
- Feature selection: Random Forest
  Number of trees: 8
  Number of features: 81
- Classifier: Linear Discriminant Analysis
- Mean accuracy score: 0.9637

For the **car dataset**:

- Not scaled data
- Feature engineering: LDA
- Classifier: Linear Discriminant Analysis
- Mean accuracy score: 0.9000

References

[1] N. Thakoor, J. Gao, and S. Jung. Hidden markov model-based weighted likelihood discriminant for 2-d shape classification. *IEEE Transactions on Image Processing*, 16(11):2707–2719, 2007.

[2] N. Thakoor, Sungyong Jung, and J. Gao. Hidden markov model based weighted likelihood discriminant for minimum error shape classification. In *2005 IEEE International Conference on Multimedia and Expo*, pages 342–345, 2005.

[3] N. Thakoor and J. Gao. Shape classifier based on generalized probabilistic descent method with hidden markov descriptor. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 495–502 Vol. 1, 2005.

[4] N. Thakoor and J. Gao. Automatic video object shape extraction and its classification with camera in motion. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–437, 2005.