ML project report

Author

Matteo Posenato

1. Introduction

I choose the emojify- sentiment analysis of Text Natural Language Processing Challenge, in which the goal is classify written sentence and map it into an emoji. The difficulty is to process the data so that the sentences make sense, combining the various meanings of the words within them. This has been possible through word embedding. We have not enought big training set and we have a lot of feature: this produce overfit. This can be fixed by increasing the number of training examples, because the number of feature is already cleaned (stopwords, punctuation). The best results are produced by support vector machine method with an accuracy of 0.72 and better shapes of the curves. This is the correct and incorrect prediction of this method:

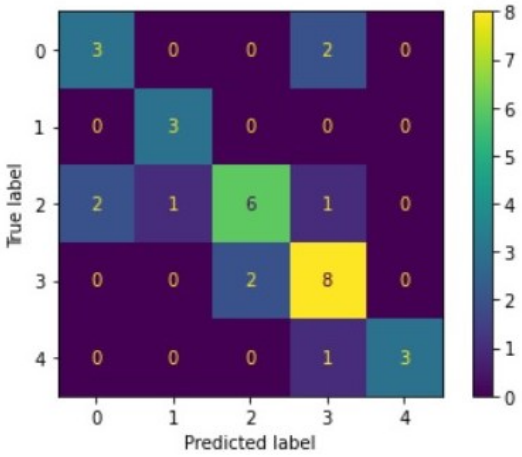


Figure 1. **Confusion matrix of SVM**

2. Dataset

The Dataset was provided by the Kaggle challenge, already cleaned and split in Train set and Validation set. Its structure is: x_train : Is composed by 100 sentences, the sentence is translated into an array of dimensions (250,) where each word is identified by 25 float numbers, the length is set by the longest sentence, in this case a sentence composed by 10 words. y_train: contain the classification of the training sentences. x_validation: Is composed by 32 sentences, each sentence is an array as the training set. This is the set we want to classify. y_validation: is the correct label of the validation sentences, we use this to evaluate performances of the models. An example of a part of the dataset is:

```
EXAMPLES:
####################
I want to go play -> ⚾
I want to have sushi for dinner -> 🍴
I am always working -> 😒
she is happy -> 😄
What is wrong with you -> 😞
I am proud of your achievements -> 😄
funny -> 😄
do you like pizza  -> 🍴
no one likes him -> 😞
he is the best player -> ⚾
```

Figure 2. **Example data in the dataset**

The single word as explain above is n-dimensional vectors where the distance between points is a correspondence respect to similarity between word semantics, similar words are closely, this representation is know as word embeddings, in the graph below we can see an example:

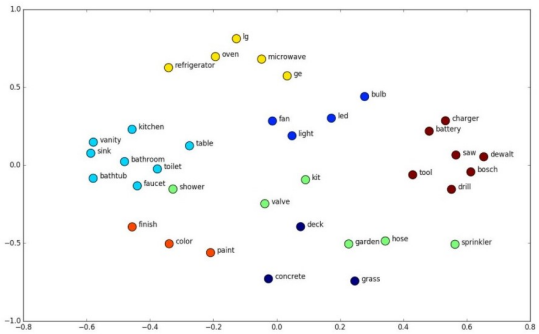


Figure 3. **Word Embeddings**

With a briefly exploratory data analysis is possible to see that the training set is not balanced between the five classes of emoji.

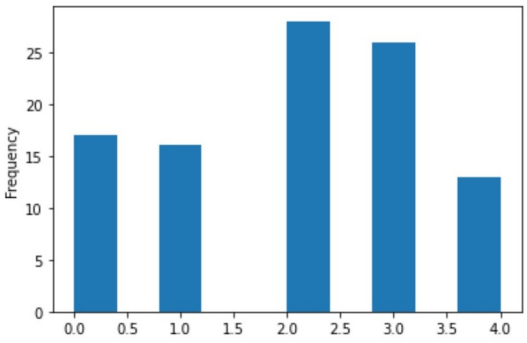


Figure 4. **Class frequency**

For this reason we'll focus on the precision-recall curve, ROC curve and Learning Curve for analyze the different methods.

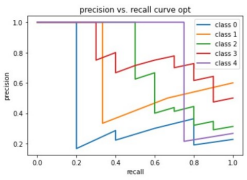
3. Methods

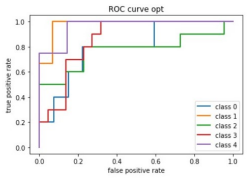
3.1. Logistic Regression

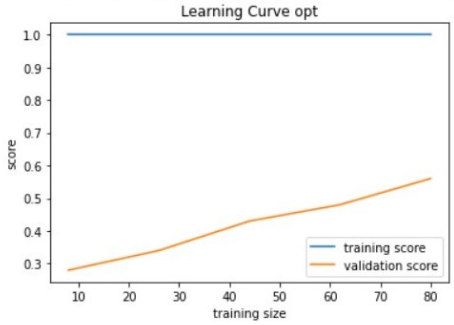
I start with the simplest model, excluding linear regression that is not a great idea applied to classification task, then i try with Logistic regression. I implement the model with all hyperparameters as default value and see the results:

	precision	recall	f1-score	support
0	0.60	0.43	0.50	7
1	0.67	0.67	0.67	3
2	0.60	0.67	0.63	9
3	0.60	0.67	0.63	9
4	0.75	0.75	0.75	4
accuracy			0.62	32
macro avg	0.64	0.64	0.64	32
weighted avg	0.62	0.62	0.62	32

As we see the accuracy is 0.62 and the fourth class is the best classified, but is the smallest one. Then I try to search the best combination of the hyperparameters. for do this I used the built-in sklearn GridSearchCV that retrieve the best hyperparameters for the model, after that I put all these parameters and predict another time the x_validation set but the performance remains the same. Then I decided to study the various curves as precision-recall, learning and ROC curve, below can see the graph





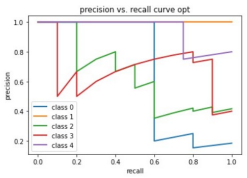


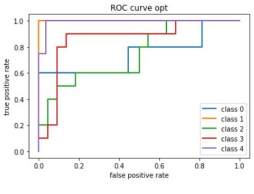
As we see the three curves show that the model didn't work well, the precision-recall curve show of big steps and fairly horizontal curves, learning curve show a large gap between train and validation set that is an high variance and thus overfit. This can be fixed by improve the number of the training example.

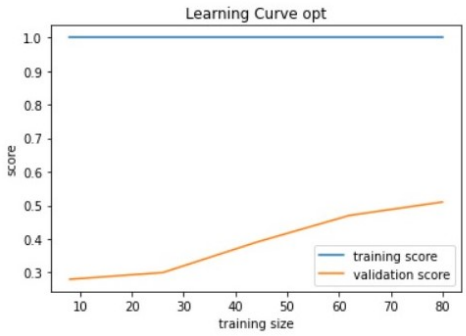
3.2. Support Vector Machine

For the second method I used the same approach of the logistic regression, I implement the method with default values, I try to optimize the hyperparameters and then I studied the results and the curves. I choose the support vector machine because is powerful method that can perform really good with non linear classification using different kernel parameter. Below is possible to see the result of the application:

	precision	recall	f1-score	support
0	0.60	0.60	0.60	5
1	1.00	0.75	0.86	4
2	0.60	0.75	0.67	8
3	0.80	0.67	0.73	12
4	0.75	1.00	0.86	3
accuracy			0.72	32
macro avg	0.75	0.75	0.74	32
weighted avg	0.74	0.72	0.72	32





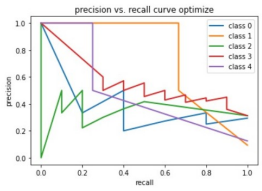


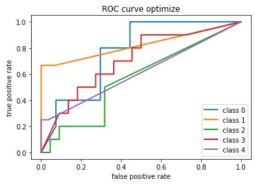
This report have better results than the logistic regression, we have an accuracy of 0.72 and the graph of precision-recall and Roc curve shows an improvement in this model but the learning curve have the same problematic of the logistic regression.

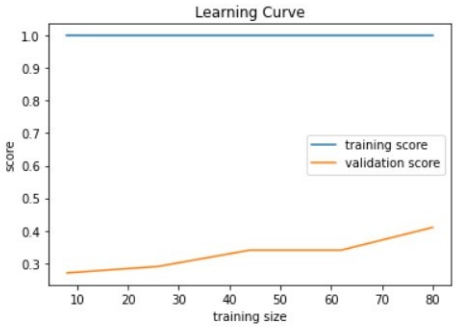
3.3. K-Nearest Neighbors

For the last method I chose a non parametric model based on word embeddings. I tried to see how it respond based on the distance between the sentences, as the previous model I used the same steps.

	precision	recall	f1-score	support
0	0.60	0.43	0.50	7
1	0.67	0.67	0.67	3
2	0.60	0.67	0.63	9
3	0.60	0.67	0.63	9
4	0.75	0.75	0.75	4
accuracy			0.62	32
macro avg	0.64	0.64	0.64	32
weighted avg	0.62	0.62	0.62	32







This method have the same accuracy of the logistic regression model (0.62), but the curves show a worsening in terms of operation of the method.

4. Experiments

For all the models I tried to scale the x_train set and perform another time all the analysis but the performance in terms of learning of the algorithm and the accuracy on prediction doesn't improve. For example with the best model (SVM) the performance drops down.

	precision	recall	f1-score	support
0	0.60	0.30	0.40	10
1	0.00	0.00	0.00	0
2	0.00	0.00	0.00	0
3	0.90	0.43	0.58	21
4	0.25	1.00	0.40	1
accuracy			0.41	32
macro avg	0.35	0.35	0.28	32
weighted avg	0.79	0.41	0.52	32

Another experiment that I did is on the LogisticRegression methods having regard to the low results, I try the LogisticRegressionCV method, that include the logistic regression classifier but with cross validation estimator, provided by sklearn package, but as the above experiments the results drops down as we can see in the figure below:

	precision	recall	f1-score	support
0	0.40	0.40	0.40	5
1	0.67	0.67	0.67	3
2	0.60	0.60	0.60	10
3	0.60	0.67	0.63	9
4	0.75	0.60	0.67	5
accuracy			0.59	32
macro avg	0.60	0.59	0.59	32
weighted avg	0.60	0.59	0.59	32

5. References

Scikit-Learn documentation