Lab Report of Exercise 1

Yun Wang
yun.wang@uzh.ch

Part 1 SGDClassifier and Multinomial Naive Bayes

Code link

https://colab.research.google.com/drive/1TKE2H-f5AxdTWeHfCpDO4UXo5Es5mHXm?usp=sharing

Approach

1. **Clean the data**

Remove '@someone', '#..' and urls from the tweet contents.

2. **Encode the data**

Use the bag of words method to vectorize features in the training and testing sets by the sklearn function CountVectorizer().
Encode labels in training set with numbers by function lbl_enc().

3. **Define a hyperparameter space**

To shorten the training time, I limite it to a small combinations of parameters:

For SGDClassifier, the searched hyperparameter space:
param_grid = {'sgdc__loss':['hinge', 'log'], 'sgdc__penalty':['l2','l1'], 'sgdc__alpha':[0.0001, 0.1, 100], 'sgdc__early_stopping':[True,False]}

For MNB, the searched hyperparameter space is:
param_grid = {'nb__alpha': [0.001, 0.01, 0.1, 1, 10, 100]}

4. **Search Hyperparameters Combinations and Train models**

I use the training set to train SGDClassifier / MNB and search the hyperparameter spaces to get the optimal hyperparameters for each classifier by GridSearchCV.

As the labels in the training set are imbalanced, I didn't use accuracy as the scorer. I use macro f1 score as the scorer, by choosing the best f1 score to get the best models.

The result shows that the best model of SGDClassifier is SVM, the combination of hyperparameters is: {alpha: 0.0001, loss: 'hinge', penalty: 'l2', Early_stopping: False}

The best hyperparameter for MNB is : {alpha: 0.1}

I set refit=True, so after GridSearchCV finds the best estimator, it retrains the best estimator on the whole training set. Then, I get the best model which can be used to predict the labels on the test set.

5. Predict on Test set

I use the best models of SGDClassifier and MNB to predict the labels of the test set, then inversely transform the numeric labels back to string labels, and compare the predicted labels with the true test labels to get the accuracy, precision, recall and confusion matrix.

Result on Test Set

1. Accuracy

Accuracy of best SGDClassifier: **0.821**, and accuracy of best Multinomial Naive Bayes classifier: 0.731

2. Average Macro precision and recall

As the labels are imbalanced, macro precision and recall are important metrics to evaluate the classifier's performance. Average macro (precision, recall) of SGDClassifier is (0.377, 0.227) , average macro (precision, recall) of MNB is **(0.397, 0.264) .**

3. Precision and Recall for each label

From Precision and recall of each label (see data in colab file), we can see some languages have high precision and recall, such as 'ar', 'en','fr', the good performance because of large data of these languages in the training set; some languages have high precision but low recall, or zero precision and recall, such as 'ar_LATN', 'el', 'fa', this is because of the very little data of these languages in the training set (see table in colab file).

4. Compare SGDClassifier and Multinomial Naive Bayes

The best SGDClassifier model (SVM) gets a higher test accuracy , lower average macro precision and recall than Multinomial Naive Bayes. Why?

Naive Bayes assumes that the features are independent from each other. However, in this project, the words are not really independent from others, for the adjacent words are probably from the same language. The assumption of Naive Bayes is violated, so the performance of this model is not as good as

SGDClassifier. But when the data is very small, Multinomial Naive Bayes works better than SVM, as we can see, the precision and recall are higher than those of SVM.

Part 2 Multi-layer Perceptron

Colab Link

https://colab.research.google.com/drive/17hi3Happ-APkyTOHoLveutJnU5NpCCZV?usp=sharing

Approach

1. **Preprocess the data as in part I**

Cleaning and encoding the data is the same as Part 1. Whenever shuffle the data, the random state is also the same as in part 1.

2. **Train Models**

As the training of MLP is very slow, I cannot successfully complete the grid search cross validation. So I trained the model on the whole training set and set hyperparameters manually.

I set solvers='adam', early_stopping=True, shuffle=True, random_state is the same as in part 1. Then I mainly focus on trying different hidden_layer_sizes and activation functions.

When the activation function is RELU, the computing is faster, so I first set activation='relu', then try different hidden_layer_sizes. After I get better hidden_layer_sizes, I change the activation function to 'tanh' and try this new hyperparameter combination.

3. **Predict on Test Set**

The same as I did in part 1.

Result on Test Set

1. **Models and Results**

See table below:

Hidden layer sizes	Activation	Accuracy	(Precision, Recall)
(100,)	relu	0.837	(0.2759188541076257, 0.1993901101672538)
(1000,500,100)	relu	0.841	(0.23302884586979927, 0.19263543706367553)
(500,100,64,32,)	relu	0.831	(0.2045939139501389, 0.17473585877433812)
(1000,)	relu	0.843	(0.3409707076283162, 0.23082523112717837)
(2000,)	relu	0.839	(0.292643308729043, 0.2150874749570274)
(800,)	**relu**	**0.845**	**(0.34510497259717143, 0.22272265553409448)**
(800,)	**tanh**	**0.842**	**(0.3799551750905021, 0.24026011207184375)**

From the result, when the hidden_layer_sizes is (800,), the performance is good. This hidden layer sizes with ReLU as the activation function, the test accuracy is the highest (0.845). When using this hidden layer sizes with tanh as activation function, the accuracy is still good (0.842) , and the macro precision and recall are the best in all these MLP models, which is 0.380 and 0.240. So, I choose the last model in the table as the best MLP model.

2. Compare with Best Linear Models

Model	Accuracy	Macro Precision	Macro Recall
SGDClassifier	0.821	0.377	0.227
Multinomial NB	0.731	**0.397**	**0.264**
MLP	**0.842**	0.380	0.240

Compared with the best SGDClassifier I found in part 1, the accuracy, macro precision and recall of MLP are all larger. Compared with the Multinomial NB, MLP model has much higher accuracy, however the macro precision and recall are a bit smaller. According to the overall result, MLP beats both SGDClassifier and Multinomial NB, but it also requires much more computation resources.