Project 2

Neal Zhao

12/17/2021

CS 135 Machine Learning

1. BoW

The BoW will be as large as possible to represent the information of words in the document. The order does not matter because the feature order does not influence model result. Very rare words shall be include. Some commenter might use rare vocabularies (such as some expression in ancient English or Latin) to express their altitude, and these information could be key to the altitude in the comment. On the other hand, super common expression should be removed. If a word shows up in most of the comment, that would indicate that this word does not show altitude. However, our document does not have words with such high frequencies.

The count of occurrences of a word should be used because multiple existence of a word should enhance the strength of a comment. However, the *tf-idf* stat should not be used because common words such as "good" and "bad" has a very strong indicating power and also have high frequencies in the data set, and thus should not be devalue by *idf*.

At first, I think that bi-grams should be used because that will handle `not` as it can combine `not` with the following word, and thus the new combo would be treated separately. However, the model result voted against this idea.

The pipeline that prepare the data:

- lower the case of all characters
- extend all contractions. (i.e. can't -> cannot)
- remove all punctuations and numbers
- remove non-English characters
- remove extra spaces
- vectorize the document by count

2. Logistic Regression

1

This model is a Logistic Regression classifier, a linear classifier that models the possible outcomes using a logistic function.

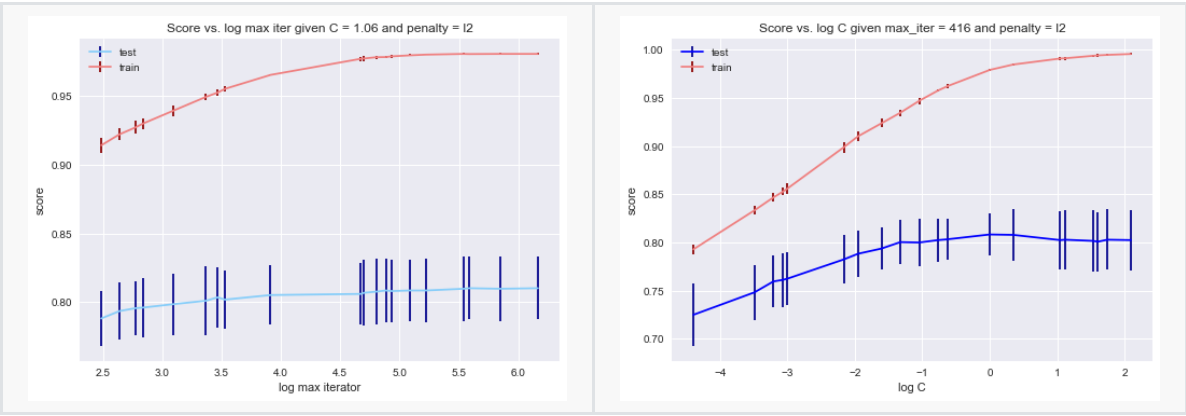
At first, to reduce the hyperparameter searching range, the model's regularization strength `C`, penalty type `penalty`, iteration limit, and `solver` are tuned. All types of `penalty` and `solver` are in the parameter grid. A randomized search with cross validation is performed on the model. The result shows that the `l2` penalty norm and `saga` solver performs not bad. These two hyperparameters will be fixed. These rough tuning also offers a hint to the range of `C` and `max_iter`, thus a closer search can be performed.

2

And then, the `RandomizedSearchCV` on logistic regression is performed with `solver = 'saga'`, `'penalty = 'l2'`, with 100 samples. The parameter grid tries `C` for 40 values logarithmically distributed between 0.1 and 10, and also tries `max_iter` for 50 values logarithmically distributed between $10^{1.7}$ and $10^{2.7}$. The value range is picked based on observations from the first rough search. Higher `C` value meaning lower regularization strength, and `max_iter` can stop the algorithm before convergence to reduce overfitting. This model also perform 5-fold cross validation and use their mean scores as the metrics for model selection. This searching finds the best model at `C = 1.0608` and `max_iter = 416` with a mean_test_score of 0.8100.

3

To better present the relations between model performance and parameters, additional model searching is performed with only one parameter varying. The result is plotted. Both mean testing and training scores are plotted, with a bar indicate the size of one standard deviation of the scores of 5-fold cross validation. To make the graph more readable, the x-value's natural log is taken so that the points would uniformly distributed horizontally.



4

Although higher `max_iter` would enlarge the overfitting (as the training score increases), the score of cross validation does not hurt. In contrast, the score is low when `C` is too high (low regularization strength, which results in overfitting, high training score) or too low (high regularization strength, which results in underfitting, low training score). Note that when the model overfits, the CV testing score does not drop a lot; therefore, this model does not suffer

severe overfitting. In conclusion, a moderate `C` should be chosen, and `max_iter` can be set high (with consideration of the running time).

3 MLP

1

This model is a MLP (multi-layer-perceptron) model. This model has one or more non-linear layers between the input and output layers. MLP classifier can learn non-linear models.

I first read an article https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw which points out that one hidden layer can deal with most of the problems. Thus, I can focus on single hidden layer.

A rough `GridSearchCV` with `MLP` is performed to help reduce the searching range of hyperparameters. The next model search is performed with hyperparameters' range that is centered around the best model find in the first `GSCV`.

2

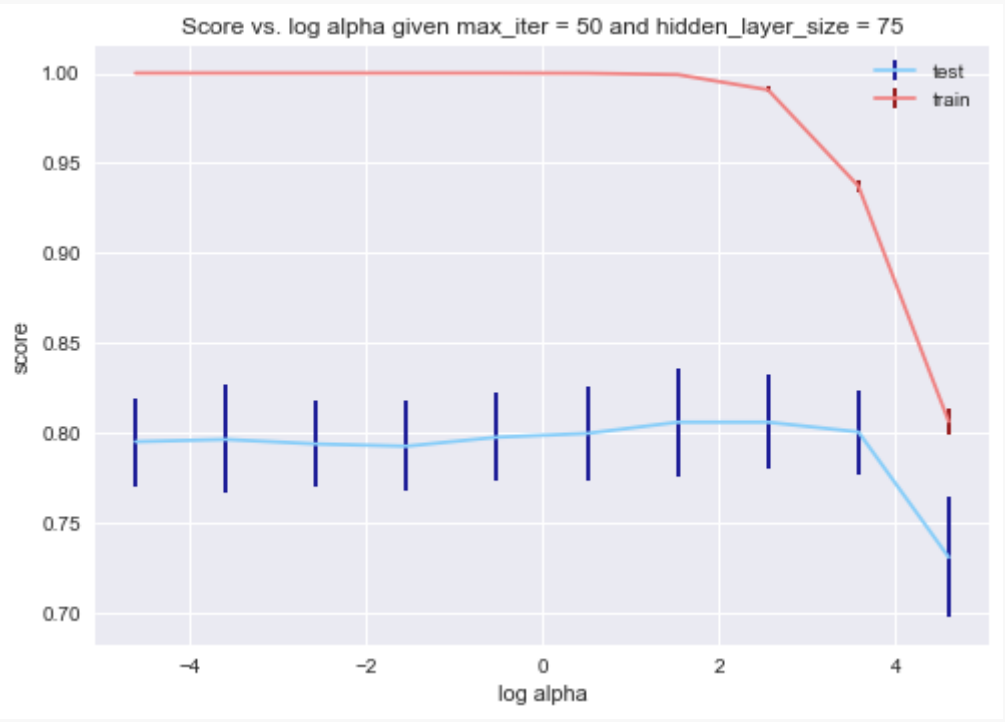
Then, I use the `GridSearchCV` to tune the following hyperparameters:

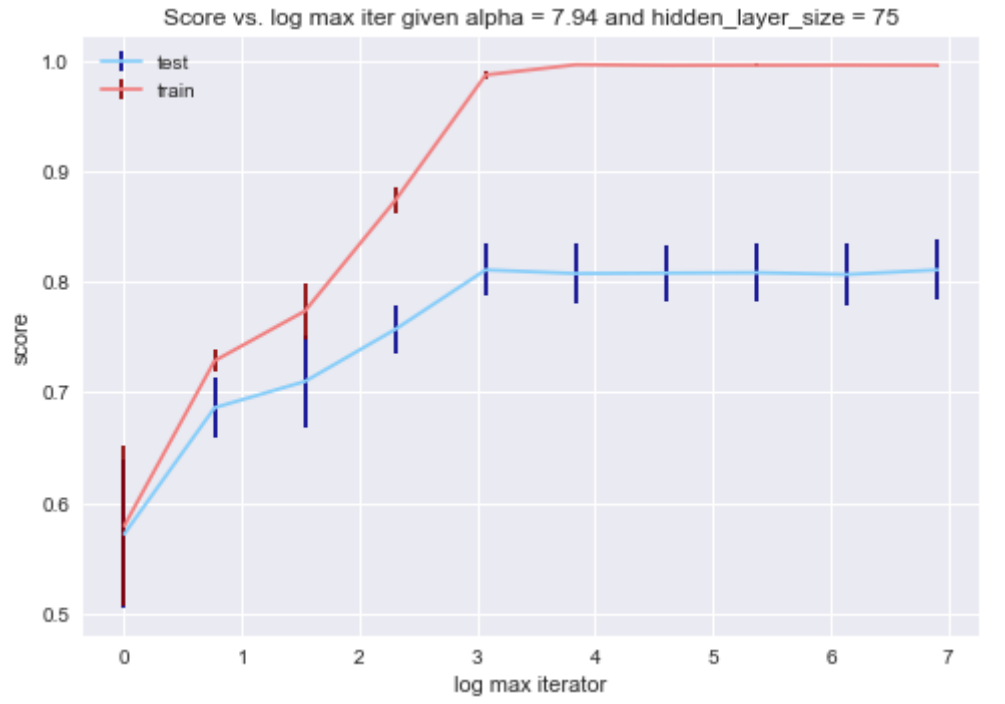
- hidden layer size: 50, 75, 100, 125, 150
- alpha, the inverse of regularization strength: 1.26, 3.16, 7.94 (`np.logspace(0.1, 0.9, 3)`)
- maximum number of iterations: 10, 30, 50

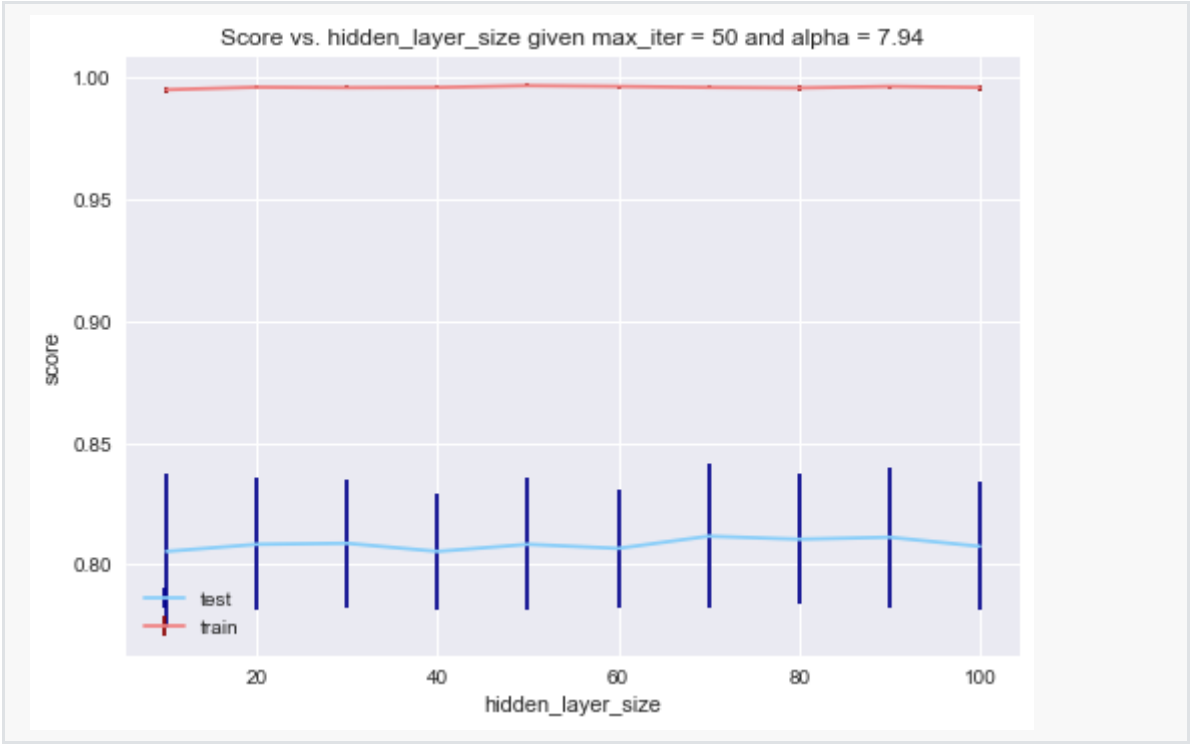
This model also perform 5-fold cross validation and use their mean scores as the metrics for model selection. The model search find the best model at `alpha = 7.94, hidden_layer_sizes = (75,), max_iter = 50` with mean_test_scores 0.8117.

3

To better present the relations between model performance and parameters, additional model searching is performed with only one parameter varying. The result is plotted. Both mean testing and training scores are plotted, with a bar indicate the size of one standard deviation of the scores of 5-fold cross validation. To make the graph more readable, the `alpha`'s and `max_iter`'s natural log is taken so that the points would uniformly distributed horizontally.


Score vs. log alpha given max_iter = 50 and hidden_layer_size = 75
test
train
score
log alpha
1.00
0.95
0.90
0.85
0.80
0.75
0.70
−4
−2
0
2
4


Score vs. log max iter given alpha = 7.94 and hidden_layer_size = 75
test
train
score
log max iterator
1.0
0.9
0.8
0.7
0.6
0.5
0
1
2
3
4
5
6
7



4

From the graphs we can see that the value of `alpha` has a very big impact on the model performance. When this regulation parameter is high, the model underfit and both training and testing score is very low. When the `alpha` is low, the model overfit, the training score is approaching perfect, but meanwhile the testing score does not drop by a lot, which indicates that the model does not suffer too much from overfitting.

At low value of `max_indicator`, both testing and training score is low, because the model barely just start the fitting process. When the value of `max_indicator` grows higher and higher, the model gets more and more closer to convergence, leaving the training score closer and closer to 1, and meanwhile the testing score also "converge", holding to a value and does not change.

In this range of layer size, it has little influence to the performing of the model.

In conclusion, a moderate `alpha` should be chosen, and `max_iter` can be set high (with consideration of the running time).

4 SVC

1

SVC is the support vector machine classifier that can efficiently classify data with high-dimensions.

The model is a Support Vector Classifier with probability output enabled. However, this probability output is not provided directly, but instead comes from a 5-fold CV.

2

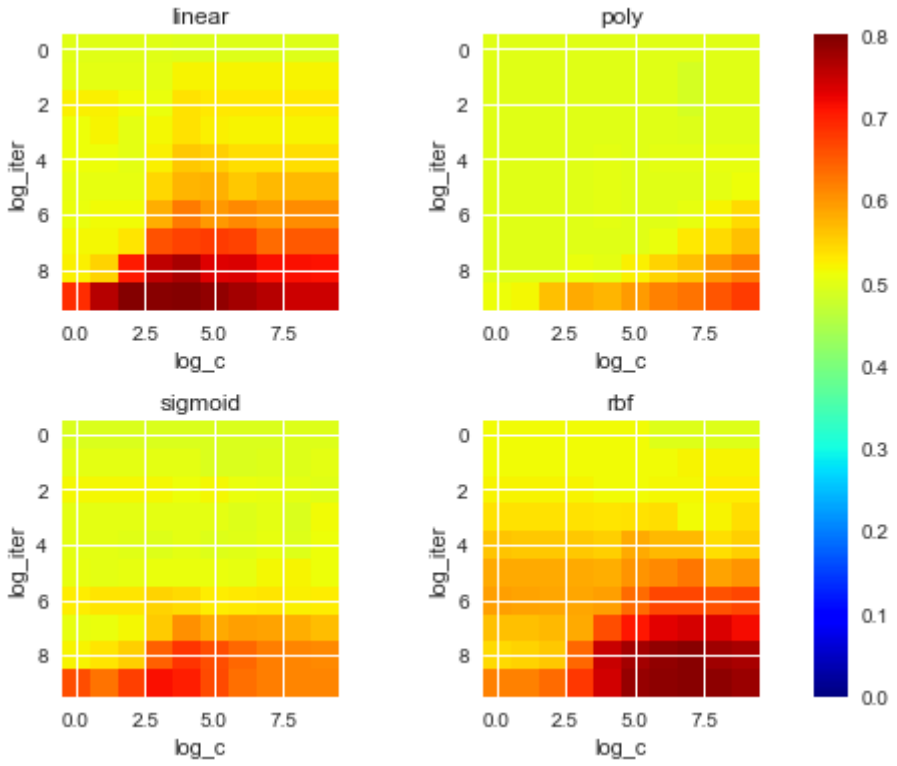
The model is tuned with following parameters and range:

- regularization parameter C: 10 numbers logarithmically distributed between 0.01 and 100.
- kernel: four types of kernel provided in `sklearn`: 'linear','poly','rbf', 'sigmoid'.
- max_iter: 10 numbers logarithmically distributed between 1 and 1000.

These sets of parameters are tested through `GridSearchSV`. This model also perform 5-fold cross validation and use their mean scores as the metrics for model selection. The search explore through 400 possible models and finds the best model at `C = 0.0774, kernel = 'linear', max_iter = 1000` with a mean_test_scores 0.7967

3

The mean-test-scores are plotted with four heatmaps, one for each type of kernel. The axis represents their relative size on a log scale.



4

We can observe that the better scores are distributed near the high end of `max_iter`, which indicate that the models perform better when closer to convergence. Also we can see that `C` has a crucial influence on the performance of the models: both overfitting and underfitting will worsen the performance of the models.

Note that for `poly`, the best model is near the right boundary of the searching grid. Thus there might be a potential maximum score exists when the `C` value is higher than the test range.

5

The MLP performs the best among the three models. This could because the data is not linear separable, which does not work well with logistic regression or support vector machine, but MLP can deal with non-linear situation. Also not that SVM is highly-influenced by the size of the features, but in this case it does not make sense to standardize the data because the different words should have different factor of influences to the model. Although MLP took a lot of time to compute (thus the hyperparameters are not tuned as much as other two models), it still works the best.

After the model is picked, the training data are split into a training (80%) and a testing (20%) set. The training and fitting are then performed on them, ending with the following confusion matrix:

true\predict	**1**	**0**
1	198	39
0	35	208

This matrix have roughly the same rate of type 1 and type 2 error, indicating that the model does not more often misclassify one class over another. The score is 0.8458

By looking at the misclassified examples, following features of error are found:

- Numbers
    - example: `All in all I give this one a resounding 9 out of 10.` (imbd)
    - reason: the data cleaning process erased all the numbers, but the sentiment is expressed through numbers in this example.
- The text does not correspond to sentiment value:
    - example: It was just not a fun experience. (yelp)
    - reason: somehow this evaluation is labeled positive.
- Not
    - reason: BoW model does not consider the positions of words. Thus the model works poorly when analyzing the meaning of 'not'.

Majority of errors comes from imbd texts. I think this is because imbd comments are longer and involves more complicate thoughts, and because BoW model only consider moods of each word and does not process any logic of sentences, the model works poor on these long sentences. On the other hand, the comments on yelp and amazon are in general shorter and contains more direct expression of mood, so their comments works better with this BoW model.

6.

The result on auto-grader (0.835) is a little bit worse than the score on my split-test set (0.846), and is better than the cross validation scores (0.812). One reason that may cause this is that CV process uses a smaller training set and thus CV score is low. The slightly lower score from the auto-grader can be explain by just probabilities. There could be also a reason that the auto-grader's input has a different distribution than the given training set, thus the scores would be different.