IFT 6390

Kaggle Competition 1 Extreme Weather

Report

Team name on kaggle: HX

Member: Xinyi HE 20202160

Introduction:

This competition is about detection of extreme weather events from atmospherical data. The dataset used is from ClimateNet and contains 19 features, 16 of which are atmospheric variables, as well as latitude, longitude and time. There are three labels for the classification, Standard background conditions, Tropical cyclone and Atmospheric river.

In this kaggle, logistic regression had to be used. In addition, SVM, Naive Bayes and Random Forests were used for additional attempts. From the results, the logistic regression model is about 0.73. Among the other three models, random forest has the best result of about 0.765.

Feature Design:

After loading the dataset using pd.read_csv, first use data.isnull().sum(axis=0) to see if there are missing values in each feature, 0 means there are no missing values. After deleting the index column 'S.No', the data is checked and found that there are only 25,763 unduplicated data out of 47,760 data. Therefore, drop_duplicates() is used to deduplicate.

Regarding feature selection, the seaborn was used to explore the relationship between features and labels, and it was found that several of the features had almost the same distribution, but in total 19 features were associated with labels. For the time feature, in order to make better use of it, it is divided into three parts, year, month, and day. Since the year in the dataset is from 1996 to 2010, and the test set is from 2011 to 2013, the year reference is not very meaningful, so it is discarded and the month and day information is kept.

After identifying the 20 desired features, feature scaling is performed on these features, and by standardizing them, different features can be made to have the same scale. In this part, normalization and standardization were tried, and after comparing the results, it was found that the accuracy and results of standardization were significantly better than the normalized ones. Because of the influence of extreme values in the feature data, standardization can better handle these data.

Finally, add a column of bias to this dataframe and initialize it to 1. Thus, the final processed dataframe contains a total of 21 columns with latitude and longitude, 16 atmospheric features, and the month and day converted from time, and a column of bias.

For labels, since it is a multiclassification problem, one-hot form is used to re-present the labels. Using csr_matrix and todense() from the scipy.sparse library, the labels of 1 column are divided into three columns of one-hot form.

The processed features and labels are shown below:

	lat	lon	TMQ	U850	V850	UBOT	VBOT	QREFHT	PS	PSL	T200	T500	PRECT	TS	TREFHT	Z1000	Z200	ZBOT	month	day	BIAS
0	−0.908686	−0.655686	−1.446629	−0.572219	1.037355	−0.713554	1.695781	−1.001604	1.394119	1.392846	1.962688	−0.686451	−0.168176	−0.662828	−0.849857	1.434832	−0.396766	−0.922959	0.136821	−1.272931	1
1	1.216774	0.193733	0.984026	1.298239	2.045043	1.076641	2.149791	1.041804	−1.377296	−1.383923	1.432447	1.332695	−0.193809	1.115359	1.439154	−1.493233	1.273084	1.402851	0.964361	−1.055355	1
2	1.216774	0.171380	−1.787944	−0.298059	−1.361854	−0.243334	−1.104079	−1.697052	0.591978	0.589156	−0.437588	−1.299803	−0.193809	0.432000	−0.840439	0.569268	−1.383491	−1.076930	−1.932029	1.664350	1
3	0.764985	−0.394900	1.395829	0.080610	−0.318709	0.940007	−0.334766	1.110756	−1.338133	−1.344684	0.632722	0.852439	0.183366	0.885178	0.783078	−1.463771	0.655926	0.844398	0.550591	−1.055355	1
4	−0.898418	−0.678039	−0.885710	−1.014248	−1.162562	−1.228055	−0.851582	−0.656311	1.808368	1.807895	−1.123181	−0.200704	−0.169139	−0.611811	−0.409376	1.916569	−0.204134	−0.446988	0.550591	−1.055355	1

Figure 1: final data_features.head()

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

Figure 2: data_label after one-hot

Algorithms:

Logistic regression

- Softmax: Logistic regression can be used for classification problems that logically separate categories. For Multinomial Logistic Regression, softmax is used to solve the K classification problem. The softmax function is obtained by training K-1 independent binary logistic regression models with the sum of the conditional probabilities of each category being 1:

$$P(y^{(i)} = k|x^{(i)}; \theta) = \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}$$

- Cost function: The predictions are not always correct, so in order to train the logistic model (e.g., via an optimization algorithm such as gradient descent), we need to define a cost function J that we want to minimize:

$$J(\theta) = -\left[\sum_{i=1}^{m}\sum_{k=1}^{K} 1\left\{y^{(i)} = k\right\} \log \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}\right]$$

- Gradient: We cannot solve for the minimum of J(θ) analytically, and thus as usual we'll resort to an iterative optimization algorithm. Taking derivatives, one can show that the gradient is:

$$\nabla_{\theta^{(k)}} J(\theta) = -\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = k\} - P(y^{(i)} = k|x^{(i)}; \theta)\right)\right]$$

The above three functions, are the main functions of Multinomial Logistic Regression. Based on these three functions, we iteratively update the weight matrix until we reach a specified number of epochs or reach the desired cost threshold.

Other method:

For Random forest, SVM, Bayes, the methods in sklearn can be used directly, so the algorithm is not detailed here.

Methodology:

Since other libraries are not allowed in logistic regression, the data is not split into training and validation sets in logistic regression. In other methods, the data is split into 70% training set and 30% validation set for validation accuracy using train_test_split in sklearn.model_selection.

In logistic regression, on the regularization side, the L2 is used to calculate the cost, adding the a priori knowledge that "the model parameters obey a zero-mean-normal distribution" to the model. The L2 norm is smoother (not sparse) than the L1 norm, but also ensures that there are more dimensions close to 0 (but not equal to 0, so it is relatively smooth) in the solution, reducing the complexity of the model.

The optimization algorithm chosen here is gradient descent. In solving for the minimum of the cost function, the gradient descent method can be used to solve step by step iteratively to obtain the minimized cost function and model parameter values.

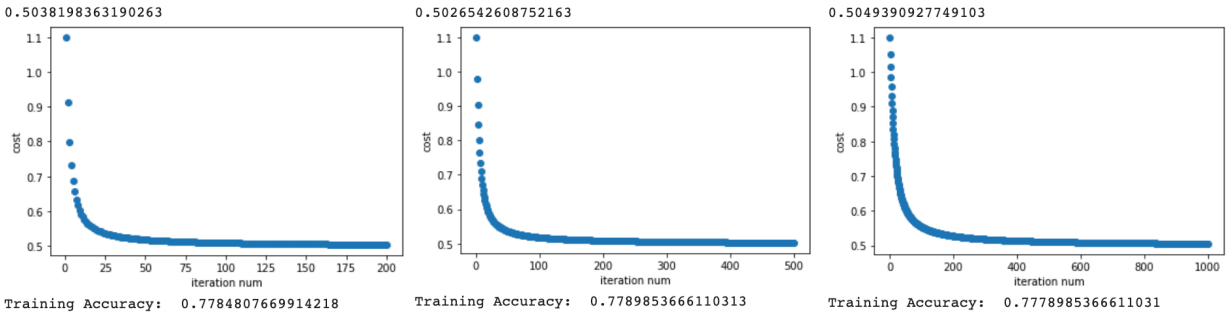
To make gradient descent work, the learning rate must be chosen smartly. The learning rate α determines how quickly the parameters are updated. If the learning rate is too large, the optimal value may be "exceeded". Similarly, if it is too small, it will take too many iterations to converge to the optimal value. Therefore, in order to get the best learning rate, first set the parameter of regularization to 0, compare the training accuracy of different learning rates when iterating to stability, and select the learning rate that gets the highest accuracy. The optimal learning rate is then kept constant and the regularization parameter that achieves the highest accuracy is selected. And set the maximum iteration times as the minimum number of iterations that can be stabilized.

Results:

Logistic Regression:

When lambda = 0

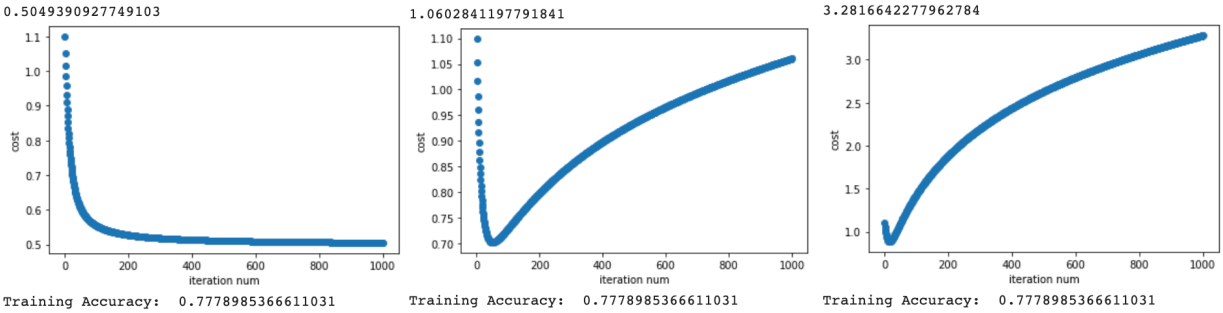
Learning Rate = 0.6 Learning Rate = 0.3. Learning Rate = 0.1



The approximate accuracies are all approximate, and slightly larger learning rates drop off quickly and do not require a large number of iterations. Smaller learning rates can achieve the same result after more iterations. Here we use a learning rate of 0.1 and 1000 iterations to confirm the appropriate regular parameters.

When Learning Rate = 0.1 iter = 1000,

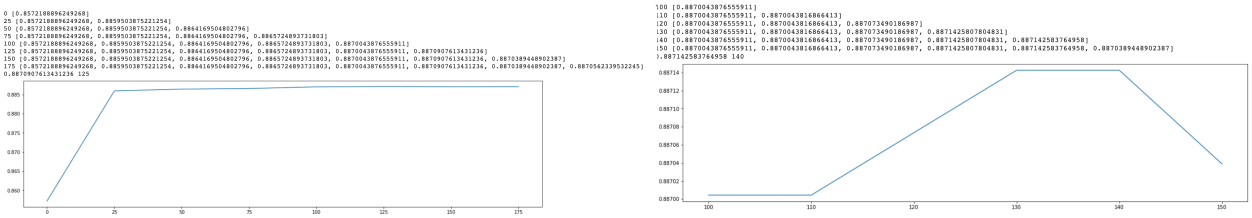
Lambda = 0 Lambda = 0.1 Lambda = 0.5



As the results show, the accuracy is highest when lambda is 0 and the cost is smaller.

Random Forest:

In order to find the optimal n_estimators, first find the most suitable region in a large area. And then in the most suitable area, more refined search



Finally get the right n_estimators = 140

Naive Bayes

```
BernoulliNB, correct prediction num: 3341, accuracy: 51.93%
GaussianNB, correct prediction num: 4014, accuracy: 51.93%
```

SVM

```
linear kernel: The accuracy is 0.7524906197438219
rbf kernel: The accuracy is 0.8179583387242851
```

Comparing the test accuracy of the four methods, we can see that Random Forest has the highest accuracy and Parsimonious Bayes has the lowest accuracy.

Discussion:

Comparing these four methods, it is easy to see that the advantages of logistic regression are simple and easy to understand, the interpretability of the model is very good. From the weights of the features can see the impact of different features on the final results. Small computational effort

and fast. The disadvantages are sensitive to extreme values, difficult to deal with data imbalance, the accuracy is not very high, because the form is very simple.

The advantage of Random Forest, which gives the best results this time, is that it can handle very high dimensional data and does not have to do feature selection. The random forest algorithm is ideal for handling datasets with unbalanced label classification. It can handle large datasets efficiently. These are the points that make it a good fit for this dataset.

For the other two models that do not work well, the naive Bayesian is ineffective due to its use of the assumption that features are independent, while features are clearly correlated in the dataset. And SVM is due to the difficulty of implementing this model for large training dataset and the difficulty of solving multiple classification problems with SVM

A direction for improvement is the learning rate, which can be dynamic. Because it is not really appropriate to keep the same learning rate. At the beginning, when the parameters are just starting to be learned, the parameters are far from the optimal solution, so it is necessary to keep a large learning rate to approach the optimal solution as soon as possible. However, after a period of learning, the parameters and the optimal solution are already close to each other, so if we keep the initial large learning rate, it is easy to cross the optimal point and oscillate back and forth around the optimal point. Therefore, setting the dynamic learning rate can find the optimal solution more quickly and accurately.

Statement of Contributions.

Since I am a student of IFT6390, this is an individual task and is done by me alone. I hereby state that all the work presented in this report is that of the author.

References:

https://zhuanlan.zhihu.com/p/113014922

https://zhuanlan.zhihu.com/p/81857985

http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/

https://www.cnblogs.com/ljygoodgoodstudydaydayup/p/10944431.html

Appendix (optional)