Mathematic of Deep Learning-Homework 2

Description of Algorithms

TSE M2 Data science for social science
Yun RU
Xu ran HUANG
2021.11

Catalogue

Introduction

In this sentiment prediction project, we implemented one clustering method to group the tweets, two machine learning methods and two deep learning methods for prediction. Using K-means clustering method, we obtained 6 clusters of the tweets. Then, the machine learning methods we used are Penalized Multinominal Logistic Regression and Random Forest, and for the deep learning part, we used Multilayer Perceptrons and LSTM with embedding without pre-training.

We made the comments in the python notebook to explain our project process in detail. In this complementary pdf file, we will explain the theoretical part of the algorithms in the project and how we applied them.

1. K-means algorithm

1.1 Introduction

The K-means method is the most popular one among those partitioning clustering methods. The first step of the method is to fix a number k for clusters and find some points randomly as representant of each cluster. Then, all the individuals are allocated to its closest group. The centers of each cluster will be computed, based on the new centers, we can reallocate the individuals to their closest group with the nearest center. We iterate the above procedure many times until the centers become stable.

1.2 Algorithm

The principle behind the K-means can be explained by the composition of total inertia I

Notation: X1, . . . , Xn denote the n individuals and $X^{(r)}$ *denotes the individuals in group r.*

I = Within-cluster Inertia + Between-cluster Inertia

$$I = \sum_{r=1}^{k} \sum_{i \in C_r} \frac{1}{n} d^2(X_i, \bar{X})$$

$$Within\ cluster\ Inertia = \sum_{r=1}^{k} \frac{1}{n} \sum_{i \in C_r} d^2(X_i, \bar{X}^{(r)})$$

$$Between\ cluster\ Inertia = \sum_{r=1}^{k} \frac{n_r}{n} d^2(\bar{X}^{(r)}, \bar{X})$$

where d represents the euclidian distance: $d(X_i, X_j) = \sqrt{\sum_{k=1}^{p}(X_i^k - X_j^k)^2}$

The K-means clustering tries to minimize the within-cluster inertia and maximize the between cluster inertia. This is what we do during the iteration process of K-means.

1.3 Advantage and drawback

The good point of K-means method is that it is very fast even for big data, however the number of clusters must be fix before the clustering. What's more, the k-means cluster depends on the represent individual that the computer chose at the very beginning, meaning that the clusters are not fixed, and they can change if you do the K-means several times.

1.4 Determination of the number of clusters

As we explained above, one of the bothered things for K-means is to choose the number of clusters. Nevertheless, there exists numbers of ways which help us to decide the number. In our project, we use the silhouette method and elbow method to figure out the number k.

Silhouette method measures how similar un individual is to its own cluster compared to other clusters. It varies from -1 to 1. The higher the score is, the better the individual match to its cluster. In the project, we decide our cluster number by looking at the silhouette score of each group and the thickness of each group in the graph.

Elbow method is a heuristic method for finding the clustering number. The intuition of this method is to find the cluster number where we cannot get a much better modeling by adding another cluster group. We can plot the model's performance against the number of groups. The optimum number is situated at the "elbow" of the performance curve. The performance of model can be evaluated by the percentage of variation explained. Generally, we use the ratio of between-group variance to the total variance. However, in our project, the performance is evaluated by the sum of squared distances of individuals to their closest cluster center. The lower the value is, the better the model is.

2. Penalized Multinominal Logistic Regression (Ridge)

2.1 Introduction

Multinomial logistic regression is a simple extension of binary logistic regression that allows for more than two categories of the outcome variable. In our project, we chose Ridge penalty to avoid the overfitting problem. Ridge regression consists in giving the variables with minor contribution

the coefficients close to zero.

2.2 Algorithm

The penalized multinominal logistic regression tries to realize optimization below:

$$\underset{\beta \in R^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} l(Y_i, h(\underline{x}_i{}^T \beta)) + pen(\beta)$$

$$Where\ h(t) = \frac{e^t}{e^t + 1}\ and\ \ pen(\beta) = \lambda ||\beta||_2^2$$

The first term of the optimization function is the loss function, and the second term is the penalty term. The logistic regression works quite rapid compared to the Ensemble method as the computer should only do one time of the regression model.

2.3 Parameters

For the parameters in the logistic regression, we should choose 'multinominal' for multi_class and 'l2' for penalty. And we have to choose a solver, we chose the Newton's method in the project which uses the quadratic approximation and is suitable for multinominal case. The parameter that we tune by GridSearchCV is the λ , which corresponds to the parameter C in the scikit learning. In fact, C is the inverse of λ. The smaller the C is, the bigger the λ is and the less overfitting the model is. We give a set of value for C and let the GridSearchCV give us the best value.

3. Random Forest

3.1 Introduction

Random Forest is one of the ensemble methods which constructs numbers of decision trees during the training step to improve the accuracy. Like bagging-of-trees, random forests predict via majority voting from decision trees trained on the bootstrap samples. The difference is that the split in each tree is allowed among predictors randomly selected out of the predictors. In fact, if one predictor is very strong in the bagging methods, most bagging trees would use this predictor in the top split and would generate highly correlated trees. However, the random forest can decorrelate the trees and improve the model.

3.2 Algorithm

Denote as $\emptyset s(x)$ the predicted class for predictor value x returned by the classification tree associated with sample $S = \{(X_i, Y_i), i = 1, \dots n\}$.

Random Forest considers predictions from B bootstrap samples

$$S^{*1} = ((X_1{}^{*1}, Y_1{}^{*1}), \dots, (X_n{}^{*1}, Y_n{}^{*1})) \rightarrow \emptyset s^{*1}(x)$$
$$\dots$$
$$S^{*b} = ((X_1{}^{*b}, Y_1{}^{*b}), \dots, (X_n{}^{*b}, Y_n{}^{*b})) \rightarrow \emptyset s^{*b}(x)$$
$$\dots$$
$$S^{*B} = ((X_1{}^{*B}, Y_1{}^{*B}), \dots, (X_n{}^{*B}, Y_n{}^{*B})) \rightarrow \emptyset s^{*B}(x)$$

then proceeds by majority voting:

$$\emptyset s^{RF}(x) = \underset{k \in (1, \dots K)}{argmax} \#\{b: \emptyset s^{*b}(x) = k\}$$

Concerning about the optimization problem for each classification tree, we must compute the criterion below for all features and all possible splitting points and choose the one minimizing the criterion:

Gini index: $C(R, \bar{R}) = \sum_{\underline{x}_i \in R} p(R)\big(1 - p(R)\big) + \sum_{\underline{x}_i \in \bar{R}} p(\bar{R})(1 - p(\bar{R}))$

The complexity of Random Forest is relatively high since we have to interate the classification tree for B (number of bootstrap samples) times.

3.3 Parameters

The parameters that we tuned by GridsearchCV are as follow:

"n_estimators": This is the number of trees grown in the random forest model. The accuracy can increase with the increase of tree's number. However, it becomes stable after a certain value.

"max_features": This is the maximum number of features that we can draw randomly from the set of features. Generally, the square root of the total number of features is the best choice.

"max_depth": The max_depth of a tree in Random Forest is defined as the longest path between the root node and the leaf node. With the increase of the max_depth's value, the performance over training set increases continuously while the performance over test set increased initially but drop rapidly after some period time. This is because of overfitting. We have to prune the trees by setting a reasonable value for max_depth.

4. Deep neural network

4.1 Multilayer Perceptrons (MLP)

Multilayer Perceptrons is built based on the concept of Linear Threshold Unit (LTU) and perceptron.

We can imagine LTU as a neuron. This neuron intakes a certain amount of input x (vector) and outputs a single value y after series of calculation. Inside the neuron, there are two kinds of functions. The first one is a group of functions which calculate the linear combination of inputs with different weights; the second one is called activation function which aims to add some nonlinearity to the model and to restrict the output from the neuron to certain required limit.

Perceptron is a simple neural network with a single layer of several LTUs. Each LTU will intake all of input x and a bias b.

MLP is composed of an input layer, hidden layer which are filled by LTUs, and an output layer of LTUs.

4.2 LSTM without pre-trained embedding in RNNs

Compared to traditional neural networks, recurrent neural networks (RNNs) are able to take previous information into account. They are networks with loops which allow information to persist. However, as the gap between relevant information grows, RNNs become unable to connect the information. Long short term memory networks (LSTMs) are a special kind of RNN. It is capable of learning long-term dependencies.

Embedding layer is good for natural language processing. It is designed to learn relationships between similar words such as verbs with different tenses and then to give them a smaller distance in the space. By doing that, it helps the algorithm to better understand the meaning of text. It requires each word in input data to be coded by a unique integer.

4.3 Mathematical representations involved

1. Two calculation steps in neuron i:

Linear combination: $z_i = w_i^T X + b_i$ where

weights in neuron i: $w_i = \begin{pmatrix} w_{i1} \\ \cdots \\ w_{ij} \end{pmatrix}$, input vector: $X = \begin{pmatrix} x_1 \\ \cdots \\ x_j \end{pmatrix}$

Activation function: $y_i = Act(z_i)$

2. Activation functions used:

ReLU: $f(z_i) = \max(0, z_i)$

Softmax: $\vartheta(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$ where i=1,…,K and $z = (z_1, \ldots, z_K)$

4.4 Model description

1. MLP

We set an input layer with a x of dimension 255, which is the dimension of the second tf-idf matrix with more information; then we added our first hidden layer with 128 neurons to intake the input x; then we added the second hidden layer with 64 neurons; at last, we added an output layer with 5 neurons because here we were solving a multi-classification problem with 5 classes. The choice of number of neurons is important. Too many neurons in the layer may lead to overfitting problem and too few neurons may cause underfitting problem. So, we used GridsearchCV to determine the best number of neurons. We kept the number of neurons between the input layer size and the output layer size, around 2/3 the size of input layer plus the size of output layer.

For the hidden layers, we used Rectified Linear Unit (ReLU) as activation function which is also the default activation function. It doesn't involve complicated calculation, so it takes less time to run. It doesn't saturate when x goes large, and it doesn't suffer from vanishing gradient problem like other activation function like sigmoid. For the output layer, we used normalized exponential function (Softmax) as activation function. It is a generalized form of sigmoid and produces values in the range of [0,1] like sigmoid. It is used in multi-class classification problems and helps to convert a vector of numbers into a vector of probabilities, which is a smoother activation model. In addition, it ensures the sum of probability for each sample will be 1. So, we put it in the output layer to get a multinomial probability distribution of dimension 5.

We used Adam as optimizer in the model. Adam is an adaptive learning rate method which computes individual learning rates for different parameters. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. It is computationally efficient and proved to get good results in a shorter time than other optimizers.

2. LSTM without pre-trained embedding

For the input layer, we set the input length as 100 which is the maximum number of words per tweet considered. Then we create an embedding training layer with maximum words considered by the model 10000 as input dimension, input length as 100, and dimension of output from embedding layer as 200. The recommended size of embedding layer is 50-300. After we set a bidirectional LSTM layer with 120 neurons. Other settings are similar to the MLP above. To fit the model, we used the matrix which was filled by unique coding number for each word. Each line of the matrix represents a tweet and each unique number represent a word. As we said before, the input length was 100 for each tweet. If the number of words in tweet was less than 100, we filled the blank by 0 to make them have equal length.

4.5 Advantages and disadvantages

1. MLP

Advantages: Features can be reduced by number of neurons in hidden layers, and we can also get our desired outcome dimension by tuning number of neurons in output layer. In addition, deep learning architecture is a flexible algorithm which can be easily adopted to different problems.

Disadvantages: It requires a large amount of data, so it is expensive to train the model. There is no obvious way to find the best deep learning tools because it is a combination of topology, training method and other parameters.

2. LSTM

Advantages: It is perfect for text meaning recognization thanks for its ability of keeping long short term memory which allows it to better analyze according to the context.

Disadvantages: The cells in LSTM become complex with the additional features, they require a lot of time to get trained and occupy high memory. Thus, LSTM become quite inefficient. It is also prone to overfitting, and it is difficult to apply dropout algorithm to handle the issue.