Task 1 - Text Classification

Methods and Targets

We use machine learning methods (including logistic regression, soft-max regression) to solve this problem. Within each method, we want to find out the impacts of different selected variables and different learning rates to the classification results.

0. Data Description

Sentiment Analysis on Movie Reviews, including training dataset and test dataset.

In the training dataset, each item contains a phrase and its sentiment (from 0 to 4). In the test dataset, there are only phrases without sentiment. We need to train a classifier with training dataset and predict the sentiments of the phrases in the test dataset via the trained model.

In order to get a more robust model, we divide the training dataset into two parts - training (80%) and validation (20%). i.e. the `train.tsv` can be divided into `training+validation+test`. `training` is for training the model, and `validation` is for assessing when the model is converging to the optimum, `test` is for evaluating the accuracy of the classifier.

1. We separate the phrases into words, and then use word count (uni-grams) as covariates. Use tree method to computer the importance measure of each feature. And the features with positive importance measure with be remained.
    1. Use the remained features to fit the model.
    2. Do PCA for the remained features, and use the first few principle components as covariates.
2. Use TF-IDF as covariates instead of word count. Use tree method to computer the importance measure of each feature. And the features with positive importance measure with be remained.
    1. Use the remained features to fit the model.
    2. Do PCA for the the remained features, and use the first few principle components as covariates.

1. Feature Selection

1.1 Word Counts (Term Frequencies)

As what the name tells, of each sentence or phrase, we calculate the counts of different words and use the word counts vector as features. It is somewhat like the uni-grams model. Thus, from this aspect, we can also use other N-grams model. But when the number of documents is quite large, the number of features extracted by the N-grams model will also be extremely large, which leads to the classification much more complicated. So in this part, we only focus on 'word counts' or equivalently 'term frequencies (TF)'.

1.1.1 Symbolic representation

There are N documents named $D_1, \ldots, D_N$. Each document consists of a list of words, i.e. $D_i = w_i^1, w_i^2, \ldots, w_i^{n_i}$, where $n_i = |D_i|$.

1. First form a dictionary: Dict=$set(D_1, D_2, \ldots, D_N)$=$(w_1, w_2, \ldots, w_q)$.
2. for i in range(N):

   x_i=**0** (a vector sized q)

   for j in range(n_i):

   x_i+=1*(Dict==w_i^j)
3. Then for each document with different word length, we transform them into vectors with the same length q.

1.1.2 A simple example

D_1 = I have a pen. D_2 = I am a boy.

Dict = [I, have, a, pen, am, boy]

x_1= (1,1,1,1,0,0) x_2 = (1,0,1,0,1,1)

1.1.3 Modification

There are some words like prepositions, personal pronouns, conjunctions and link-verbs containing little information and contributing little for classification. One simple way is to delete these words in the dictionary first, and then generate the word counts vectors.

The word count or term frequency method weigh each word in the dictionary as the same. This makes it hard to detect the 'Topic Words' of a specific document. To overcome this shortage, we will further talk about the TF-IDF method.

1.2 TF-IDF

TF-IDF: Term frequency–inverse document frequency. An intuitive comprehension of TF-IDF is: it puts heavier weights on those words were merely representative in some certain documents but puts lighter weights on those words appear in almost every documents.

1.2.1 Symbolic representation

TF (term frequency): the number of times that term t occurs in document d, denoted as

$$tf(t,d) = f_{t,d}$$

IDF (inverse document frequency):

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with $N = |D|$: total number of documents in the corpus, $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t,d) \neq 0$).

Then TF-IDF is calculated as

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

1.2.2 A simple example

If the total number of words in a document is 100, and the word **cow** appears three times, then the word frequency of **cow** in this document is 3 / 100 = 0.03. If the word **cow** has appeared in 1000 documents and the total number of documents is 10000000, the IDF is $\log(10000000/1000) = 4$. The final TF-IDF score was $0.03 \times 4 = 0.12$.

1.3 Tree-Based Method

One advantage of tree model over other simple statistical methods is that it can extract the importance measure of each variable.

By the bag-of-words model, the number of features is quite large. We can use tree-method to extract the importance measure of each feature, and those with importance equal 0 will be excluded.

This can be done by using `sklearn.ensemble.ExtraTreesClassifier`.

1.4 Principal Components

Principal Component Analysis (PCA) is a basic method to do dimension reduction. Dimension reduction helps us ease the calculation and storage burdens.

PCA method aims to find the directions which can explain as much variance as possible. A p-dimensional variable has p principal components. But usually, the first few components will explain most variance. By the rule of thumb, we usually first few variables so that the cumulative variance contribution rate can exceed 80% or 85%.

This can be done by using `sklearn.decomposition.PCA`

2. Classifiers

2.1 Logistic Regression

For a training set $\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$.

Posterior:

$$\begin{aligned} p(y=1|\boldsymbol{x}) &= \sigma\left(\boldsymbol{w}^\top \boldsymbol{x}\right) \\ &\triangleq \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})} \end{aligned}$$

Denote the posterior probability of classifying sample $\boldsymbol{x}^{(n)}$ to 1 as $\hat{y}^{(n)}$. And we use the cross entropy loss function:

$$R(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)}\log\hat{y}^{(n)} + (1-y^{(n)})\log(1-\hat{y}^{(n)})\right)$$

Hence, the gradient is:

$$\frac{\partial R(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x^{(n)}}\left(y^{(n)}-\hat{y}^{(n)}\right)$$

Methods to solve the coefficients - Gradient Descent

Gradient Descent: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot grad$, where α is learning rate.

1. Batch Gradient Descent: in each iteration, update the coefficient by using **all** the samples.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \cdot \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x^{(n)}} \left(y^{(n)} - \hat{y}^{(n)} \right)$$

2. Mini-Batch Gradient Descent: in each iteration, update the coefficient by using **only a few** samples.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \cdot \frac{1}{K} \sum_{n=1}^{K} \boldsymbol{x^{(n)}} \left(y^{(n)} - \hat{y}^{(n)} \right), \quad K < N$$

3. Stochastic Gradient Descent: in each iteration, update the coefficient by using **only one** sample.

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \cdot \boldsymbol{x^{(n)}} \left(y^{(n)} - \hat{y}^{(n)} \right)$$

Batch Gradient Descent has the smallest variance but is the most time-consuming. Stochastic Gradient Descent is the simplest by with the highest variance. And Mini-Batch Gradient Descent is a compromise between them.

2.2 Softmax Regression

For a training set $\{\boldsymbol{x}^{(n)}, y^{(n)}\}_{n=1}^{N}$.

Posterior:

$$\begin{aligned} p(y = c|\boldsymbol{x}) &= \text{softmax}\left(\boldsymbol{w}_c^\top \boldsymbol{x}\right) \\ &\triangleq \frac{\exp(\boldsymbol{w}_c^\top \boldsymbol{x})}{\sum_{c'=1}^{C} \exp(\boldsymbol{w}_{c'}^\top \boldsymbol{x})} \end{aligned}$$

Denote

$$\hat{y}^{(n)} = \arg\max_c p(y = c|\boldsymbol{x}^{(n)}) = \arg\max_c \boldsymbol{w}_c^\top \boldsymbol{x}^{(n)}$$

Represented in vectored forms:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}^\top \boldsymbol{x}) = \frac{\exp(\boldsymbol{W}^\top \boldsymbol{x})}{\boldsymbol{1}_C^\top \exp(\boldsymbol{W}^\top \boldsymbol{x})}$$

And we use the cross entropy loss function:

$$R(\boldsymbol{W}) = -\frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{y}^{(n)})^\top \log \hat{\boldsymbol{y}}^{(n)}$$

Hence, the gradient is:

$$\frac{\partial R(\boldsymbol{W})}{\partial \boldsymbol{W}} = -\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x^{(n)}} \left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}^{(n)} \right)^\top$$

Methods to solve the coefficients - Gradient Descent

Gradient Descent: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \cdot grad$, where α is learning rate.

1. Batch Gradient Descent: in each iteration, update the coefficient by using **all** the samples.

$$\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t + \alpha \cdot \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x^{(n)}} \left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}_{\boldsymbol{W}_t}^{(n)} \right)^\top$$

2. Mini-Batch Gradient Descent: in each iteration, update the coefficient by using **only a few** samples.

$$\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t + \alpha \cdot \frac{1}{K} \sum_{n=1}^{K} \boldsymbol{x^{(n)}} \left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}_{\boldsymbol{W}_t}^{(n)} \right)^\top, \quad K < N$$

3. Stochastic Gradient Descent: in each iteration, update the coefficient by using **only one** sample.

$$\boldsymbol{W}_{t+1} \leftarrow \boldsymbol{W}_t + \alpha \cdot \boldsymbol{x^{(n)}} \left(\boldsymbol{y}^{(n)} - \hat{\boldsymbol{y}}_{\boldsymbol{W}_t}^{(n)} \right)^\top$$

Batch Gradient Descent has the smallest variance but is the most time-consuming. Stochastic Gradient Descent is the simplest by with the highest variance. And Mini-Batch Gradient Descent is a compromise between them.

For simplification, here we did not take penalty term into consideration.

3. Experimental Results

For each method, we repeat 100 times, and record the 100 prediction accuracies. In the tables, results are shown as the form `mean(std)`.

3.1 Word Counts

3.1.1 Logistic Regression

sentiment $= 2$ or sentiment $\neq 2$

- Starting with random seeds:

features	GD/lr	0.01	0.05	0.1
Importance	batch	0.512 (0.058)	0.510 (0.070)	0.519 (0.076)
	mini-batch	0.512 (0.058)	0.510 (0.070)	0.519 (0.076)
	stochastic	0.511 (0.058)	0.502 (0.071)	0.522 (0.071)
PC scores	batch	0.488 (0.054)	0.501 (0.060)	0.516 (0.068)
	mini-batch	0.488 (0.054)	0.501 (0.060)	0.516 (0.068)
	stochastic	0.487 (0.053)	0.500 (0.060)	0.508 (0.064)

- Starting with zeros:

features	GD/lr	0.01	0.05	0.1
Importance	batch	0.5713 (0.0050)	0.5763 (0.0036)	0.5793 (0.0036)
	mini-batch	0.5715 (0.0197)	0.5757 (0.0041)	0.5791 (0.0031)
	stochastic	0.5047 (0.0209)	0.5199 (0.0301)	0.5311 (0.0378)
PC scores	batch	0.5611 (0.0048)	0.5710 (0.0039)	0.5733 (0.0031)
	mini-batch	0.5615 (0.0111)	0.5710 (0.0043)	0.5732 (0.0034)
	stochastic	0.5091 (0.0265)	0.5203 (0.0284)	0.5288 (0.0365)

3.1.2 Soft-max

Starting with zeros:

features	GD/lr	0.01	0.05	0.1
Importance	batch	0.510 (0.002)	0.510 (0.001)	0.510 (0.001)
	mini-batch	0.510 (0.002)	0.510 (0.001)	0.510 (0.001)
	stochastic	0.486 (0.081)	0.510 (0.001)	0.509 (0.004)
PC scores	batch	0.510 (0.001)	0.509 (0.001)	0.510 (0.001)
	mini-batch	0.510 (0.001)	0.509 (0.001)	0.510 (0.001)
	stochastic	0.492 (0.069)	0.504 (0.042)	0.510 (0.003)

3.2 TF-IDF

3.2.1 Logistic Regression

sentiment $= 2$ or sentiment $\neq 2$

- Starting with zeros:

features	GD/lr	0.01	0.05	0.1
Importance	batch	0.5100 (0.0013)	0.5034 (0.0012)	0.5099 (0.0012)
	mini-batch	0.5105 (0.0108)	0.5096 (0.0025)	0.5108 (0.0094)
	stochastic	0.5031 (0.0108)	0.5034 (0.0099)	0.5035 (0.0132)
PC scores	batch	0.5036 (0.0012)	0.5100 (0.0012)	0.5099 (0.0012)
	mini-batch	0.5111 (0.0118)	0.5089 (0.0046)	0.5116 (0.0128)
	stochastic	0.5036 (0.0129)	0.5046 (0.0105)	0.5041 (0.0153)

3.2.2 Soft-max

- Starting with zeros:

features	GD/lr	0.01	0.05	0.1
Importance	batch	0.5102 (0.0010)	0.5100 (0.0012)	0.5101 (0.0012)
	mini-batch	0.5102 (0.0010)	0.5100 (0.0012)	0.5101 (0.0012)
	stochastic	0.4961 (0.0621)	0.5026 (0.0493)	0.5041 (0.0414)
PC scores	batch	0.5103 (0.0012)	0.5102 (0.0014)	0.5098 (0.0014)
	mini-batch	0.5103 (0.0012)	0.5102 (0.0014)	0.5098 (0.0014)
	stochastic	0.5045 (0.0406)	0.4980 (0.0594)	0.4971 (0.0628)

3.3 By Toolkit (`sklearn`)

	Word Count	PCA on WC	TFIDF	PCA on TFIDF
Logistic	0.6696 (0.0013)	0.6275 (0.0013)	0.6506 (0.0015)	0.6269 (0.0015)
Soft-max	0.5662 (0.0014)	0.5285 (0.0013)	0.5543 (0.0014)	0.5322 (0.0013)

4. Conclusion

1. The influence of different features on the classification results: word counts (or equivalently term frequency) and TF-IDF, TF give the same weight to all words, while TF-IDF give different weight to words in each document. It give less weight to words that often appear in each document, and give higher weight to words that only appear in specific documents.
    1. By comparison, whether logistic or soft Max expression, the classification results based on TF are better than that based on TF-IDF.
    2. Considering the large number of features, we can reduce the dimension of features by PCA. The results show that the classification results based on PCA are worse than that based on original features. The reason is that PCA only extracts the first several principal components, resulting in the loss of information.
2. The influence of learning rate on classification results: Generally speaking, when the learning rate is small, there is no significant difference in classification effect. (we can continue to discuss it in case of higher learning rate)
3. The effects of different methods (stochastic gradient descent, batch gradient descent and mini-batch gradient descent) on classification results:

1. The results show that in most cases, the results of batch gradient descent and mini-batch gradient descent methods are better than those of stochastic gradient descent, and the results of mini-batch gradient descent and batch gradient descent method are quite similar.
2. Considering the complexity of the algorithm, the complexity of stochastic gradient descent is the lowest, batch gradient descent method is the most complex and mini-batch gradient descent is in the between. This is reflected in the results of 100 repeated experiments. The variance of prediction accuracy obtained by stochastic gradient descent is the largest, and the variance of prediction accuracy obtained by batch gradient descent method is the smallest.

5. Future Work

1. The learning rate chosen here are relatively small. The effect of larger learning rate on classification results can be observed later.
2. The selection of max iteration is small, only 100. For such a complex model, and LR is so small, when iter reaches 100, the outcome (coefficient) may not reach the optimum (even far away from optimum).