Machine Learning - Assignment 1

Dongwen Lin

<2016-02-28 >

Contents

1 Machine-learning

1.1 Question 1

Discuss how to formulate this task as a classification problem – describe what are the inputs (x) and what are the outputs (y). Discuss how you would convert the input data into vector representations using the approach discussed in class.

1.1.1 Answer

There are many ways to reprensent the features of an article(email). However the easiest way is to just count the frequency of the words of on article regarding to a dictionary. And this approach is generally called **The bag of words** (BOW).

To make it simple, what I will do is just split all the strings in to words in all the article, and union them into a *set* of words (*set* means *no replication*) as the bag. Then counting the frequency of each words in an article and represent it as a vector.

Then each article would be converted into a column vector, as the input ($\vec{x}$). As for the output(y), for binary cases when there are only two categories (A and B), then y will only be $+1$ or -1 to represent two different categories. If $y = +1$, the article would belongs to category A, while if $y = -1$, it would belongs to category B.

1.2 Question 2

Consider only the documents that appeared in these two sub-folders: atheism and sports (i.e., ignore documents from science and politics for now). Implement the perceptron algorithm to perform binary classification based on the data from these two sub-folders. Evaluate the model's performance on the training and test set respectively. Discuss clearly in your report when to stop the algorithm and whether this has any effect on the performance on the test set. Try to use the averaged perceptron algorithm discussed in class, and report the performance when such an algorithm is used. Compare its performance with that of the standard version of the perceptron.

1.2.1 Answer

Table 1: Perceptron (Acuracy = 93.00%)

	Atheism	Sports	
Right	763	725	1488
Wrong	37	75	112
sum	800	800	93.00%

Table 2: Perceptron Averaged

	Test Data			Train Data		
	Atheism	Sports	Sum	Atheism	Sports	Sum
Right	753	704	1457	194	196	390
Wrong	47	96	143	6	4	10
	800	800	91.06%	200	200	97.50%

Although an academic literature [1] point out that averaged perceptron is almost always better than perceptron, our case is not. As is shown in table 1 and 2[2] the performance of original perceptron is slightly better.

As for the iterations, since perceptron require the training example to be linear separable, and we did not involve any randomness into the training set. The result will always be the same after convergence.

However, if we stop the algorithm earlier before convergence. The result would be different. Sometimes, the result would be better because it avoid over-fitting.

[1]:See `http://www.ciml.info/dl/v0_8/ciml-v0_8-ch03.pdf`

[2]:Not all testing result are in this report, but all the essential result related are showed here. If you interested in other result, check the full Original Result HERE : `https://goo.gl/8ihA5B`

1.3 Question 3

Implement the stochastic gradient descent algorithm that minimizes the empirical risk involving the hinge loss to tackle the same binary classification problem. Evaluate your model's performance using the test data. Discuss its learning behavior: the effect of using different learning rates, how fast the algorithm converges. Discuss when to stop the algorithm and whether this has any effect on the performance on the test set. Also compare its performance with that of the perceptron.

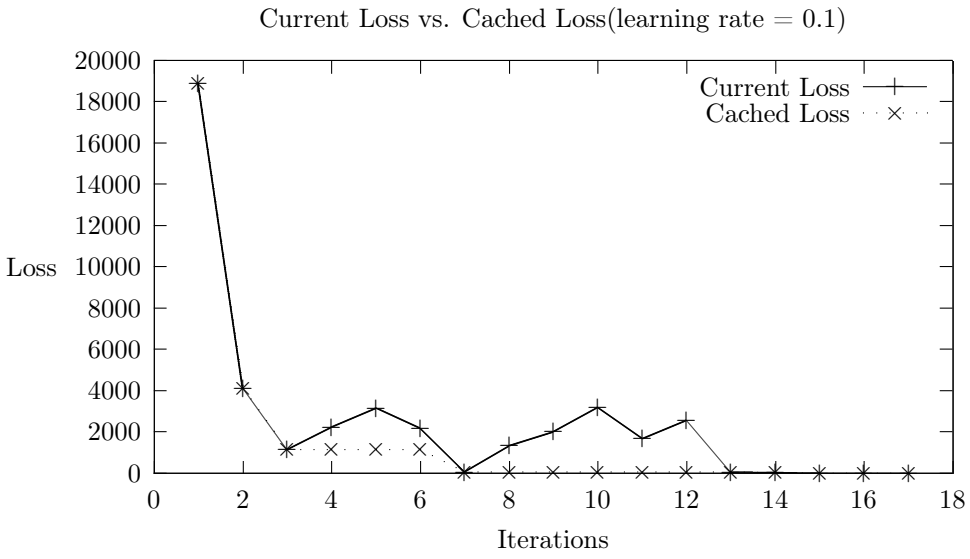
1.3.1 Answer

Table 3: Gradient Decent

iter	learning rate	Test Data	Train Data	Best Loss	Final Loss
10	0.1	93.50%	99.75%	17.2	1411.0
10	1	87.44%	96.75%	2281.0	10412.0
10	10	93.63%	99.75%	23504.0	23504.0
10	100	93.13%	99.25%	13802.0	273007.0
20	0.1	92.25%	100%	0	0
20	1	86.82%	100%	77	77

We can see that the test result it sligtly better than the perceptron algorithm. In our case, using small learning rate would encourage the algorithm to converge faster.

The graph below showes that how the loss finally converges and become zero.



As for when to stop the algorithm and whether it has any effect on the performance. The answer here is similar to perceptron algorithms: If we stop before converge, the result would be usually less good. And after the loss function become 0, no matter when we stop, the result would be the same. (Because the weights stop to update.) However, even if we stop at the same iteration every time, regardless of whether the loss is 0, the result are different in SGD because we involve randomness when training.

1.4 Question 4

Now, consider the multi-class classification problem (i.e. consider all documents from all 4 topics). Think of a way to perform such a multi-class classification task using what you have learned so far. Describe and implement your algorithm, and report its performance on training and test set. (Hint: How to cast a multi-class classification problem into binary classification problems?)

1.4.1 Answer

There might be several ways to do this and finally I decided to use a simple approach but the performance might not be that good.

Table 4: Multi Classifier

		Data Set				
		Atheism	Sports	Politics	Science	Accuracy
Classifier 1	Athesim	715	15	23	122	92.34%
	Others	85	785	777	678	
Classifier 2	Sports	6	703	12	14	95.97%
	Others	794	97	788	786	
Classifier 3	Politics	9	16	662	33	93.88%
	Others	791	784	138	767	
Classifier 4	Science	10	34	15	534	89.84%
	Others	790	766	785	266	
	SUM	3200	3200	3200	3200	81.69%

Here is how I did this:

I would separately train 4 classifier and then using them to find out each category. Each classifier only can separate one category. For example, for classifier 1, it can only distinguish *atheism* and then the rest would be labeled *others*.

However, there is a major problem for this approach. The total number of each category add up together might not be the total number of samples, which means, there might be samples which are not labled to any of the category(only *others*). On the other hand, there might be some document which are labeled more than once.

There could be a way to solve the latest problem and I would like to discussed in question 7 (By comparing the result between each classier). However, for the first problem, I haven't find a gentle way to do it yet without altering the classifier.

1.5 Question 5

The objective function involving the hinge loss is defined as follows:

$$R'_n(\vec{\theta}, \theta_0) = \frac{1}{n} \sum_{t=1}^{n} max\{1 - y^{(t)}(\vec{\theta} \cdot \vec{x}^{(t)} + \theta_0), 0\}$$

Now let us introduce a so-called regularization term to the above objective function, leading to the following new object function:

$$R'_n(\vec{\theta}, \theta_0) = \lambda||\vec{\theta}||^2 + \frac{1}{n} \sum_{t=1}^{n} max\{1 - y^{(t)}(\vec{\theta} \cdot \vec{x}^{(t)} + \theta_0), 0\}$$

where λ is a constant. Present your new learning algorithm to optimize this new objective function. Provide necessary deriva- tions for any update equations used in your algorithm. Train and evaluate your new model based on the documents from the two sub-folders atheism and sports. Try to set to different values (such as 0.001, 0.01, 0.1, 1, 10 etc) and report the learned model's performance on both the training set and the test set. Try to explain why the performances on the training and test set have such behaviors as we change the value of .

1.5.1 Answer

Our new object function is

$$R'_n(\vec{\theta}, \theta_0) = \lambda||\vec{\theta}||^2 + \frac{1}{n} \sum_{t=1}^{n} max\{1 - y^{(t)}(\vec{\theta} \cdot \vec{x}^{(t)} + \theta_0), 0\}$$

Thus we could have

$$\begin{aligned} \vec{\theta^{(k+1)}} &= \vec{\theta^{(k)}} - \eta \nabla_\theta R'_n(\vec{\theta}, \theta_0)|_{\theta=\theta^{(k)}} \\ &= \vec{\theta^{(k)}} - 2\eta\lambda\vec{\theta^{(k)}} + \eta y^{(t)} \cdot \vec{x}^{(t)} \end{aligned}$$

We can see the performance is much better if we provide proper λ but if the λ were not provided properly, the result might not be converged. (see table 5)

Table 5: SGD with Regulation

λ	iter	learning rate	Test Data	Train Data	Loss
0.001	30	0.01	96.31%	100%	0.32
0.01	30	0.01	95.69%	100%	0
0.1	20	0.01	91.06%	100%	0.65
0.1	30	0.01	93.38%	100%	0
1	30	0.01	80.44%	82.25%	1049.17
10	30	0.1	50%	50%	3349625.7

1.6 Question 6

The approach described in class for representing documents as vectors is the so-called "bag- of-words (BOW)" approach. It is only one of the many ways to represent input documents as vectors. Think of some other ways to represent the input documents as vectors, and discuss the effect of using such alternative representations when the perceptron algorithm is used. Try to explain why the performance on the test set becomes better or worse with your alternative representations

1.6.1 Anwser

There are many ways to represent the features for an document. For example, we could reprensent the words not by their fequency but by there length. Or we can simply represent a document to a vector of its bit format.But the permermance are much worse in these cases. because we lost a lot of useful important infomation.

So two better way to represent these words into vector is **n-gram** model or **tf-idf** model. Not only they preserve the infomation of words frequency but they also preserve some relationship between words.

The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

1.7 Question 7

This is a very open question. Try to think of something else interesting to explore based on what you have learned and the data provided. For example, in class we discussed that we can use gradient descent with hinge loss to optimize the empirical risk. You can think of a different loss function to replace the hinge loss and see what performance you can obtain.

1.7.1 Anwser

If we change the classier in question 4 a little bit.

Table 6: Another version of multiclassifer

	Atheism	Sports	Politics	Science
Athesim	749	10	17	24
Sports	10	731	9	50
Politics	25	17	732	26
Science	65	18	50	667
Acuracy	93.6%	91.4%	91.5%	83.4%

Each time we choose the result which give the best score. In other words, the smallest lost in this function and we could get the result as betlow.

The perfemance is slightly better and solves some pevious problems.