Large Scale Image Classification Based on a Multi-Class Logistic Regression Model

Tianlong Song SONGTIA6@MSU.EDU
Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, 48824, USA

Abstract

The multi-class logistic regression model is adopted to perform the large scale image classification. Its computation cost is significantly reduced by performing a random training example sampling. The experimental results demonstrate that the proposed approach can successfully achieve an acceptable prediction accuracy, and has a considerable reduction on implementation cost. In addition, a tradeoff between prediction accuracy and implementation cost can be made simply by choosing an appropriate number of iteration.

1. Introduction

Classification is one of the most interesting topics in the field of machine learning, and it has a wide application in Internet search, image and speech recognition, etc. The most applicable while simultaneously challenging classification is multi-class classification, especially when there are massive and high-dimensional data which are spread over a wide collection of different classes. There are two major concerns on the multi-class classification models: 1) Prediction accuracy; 2) Implementation cost, typically time and memory consumption. Usually these two factors are contradictory and a tradeoff is often needed to make between them.

K Nearest Neighbors (kNN) (Cover & Hart, 1967) is a very intuitive approach in classification, but it suffers from a high computation cost, especially when the number of the training examples is significantly large. Probabilistic generative models (e.g. naive bayes) (Bishop, 2006) usually have lower computation cost, while finding themselves difficult to achieve high accuracies. The Supporting Vector Machine (SVM) (Hsu & Lin, 2003) is a model aiming at finding the maximum margin between different classes; its accuracy, to some extent, relies on the chosen kernel functions, and it also cannot handle the sharply increasing computation cost with large scale data classification.

Compared with the low-accuracy models (e.g., naive bayes), the logistic regression model (Greene, 1993) can usually achieve a higher accuracy, certainly with an increased complexity. When applied to the multi-class case, it will also have to face up with the complexity problem. In this project, the multi-class logistic regression model is employed to achieve an acceptable accuracy, simultaneously with its computation cost suppressed by training example sampling.

2. Problem Formulation

We have N_{train} training examples, and each of them contains D features which characterize a specific image. The training examples are distributed over K classes, and the class assignments have already been known for all the training examples. Simultaneously, there are N_{test} testing examples, whose class assignments are to be determined. We are supposed to build a multi-class classifier which can learn its coefficients efficiently from the training examples and work well in determining the classes which the testing examples belong to.

Specifically for this task, $N_{train} = 1,000,000, D = 900, K = 164$ and $N_{test} = 262102$. The high volume and complexity of the data definitely make the task challenging in the following aspects:

- *Massive training examples:* massive data usually means more time will be needed to finish training; however, when the amount of data exceeds our tolerance, we have to derive an approach whose training cost does not increase, or at least does not increase sharply with an increasing number of training examples.

This is a report on the CSE847 course project. If interested, please find the corresponidng MATLAB codes via www.dropbox.com/sh/azy73yz2morkhpw/nER0bPx7OC.

- *High feature dimension:* this brings not only more computation, but also uncertainty in class determination, because the characteristics and importance probably vary much from feature to feature. In the worst case, some of the features may be misleading rather than helpful.
- *High class diversity:* many classifiers work well for binary classification, but tend to be computationally expensive with also significant performance decrease when extended to their multi-class counterparts; the considerably high class diversity makes this situation even more difficult to handle.

Logistic regression model is one of most prevalent classifiers because of its high accuracy and acceptable computation cost, although it sometimes suffers from a non-converging problem. One of the important features of the logistic regression model is that the training examples involved in each iteration can be either the whole training set or one single training example. This motivates me to consider using part of the training examples (neither the whole training set nor one single training example) in each individual iteration, i.e., training example sampling. This turns out to be very useful in suppressing the computation cost under a large-scale classification, and this advantage also holds when the classification is extended to multi-class case.

3. Multi-Class Logistic Regression Classification

In binary logistic regression model, we suppose that given a training example $\mathbf{x}_i$, the logarithm of the probability of $\mathbf{x}_i$ associating to class 1 ($y_i = 1$) over that to -1 ($y_i = -1$) is a linear combination of the features in $\mathbf{x}_i$, i.e.,

$$\ln \frac{p(y_i = 1|\mathbf{x}_i)}{p(y_i = -1|\mathbf{x}_i)} = \mathbf{w}^T \mathbf{x}_i \tag{1}$$

where $\mathbf{w}$ is the parameter which is supposed to be learned by the regression model. While when the model is extended to multi-class case, the assumption need to be somehow revised. An intuitive extension is to assume that the probability of $\mathbf{x}_i$ associating to class y_i is proportional to a linear combination of the features in $\mathbf{x}_i$, that is,

$$\ln p(y_i|\mathbf{x}_i) \propto \mathbf{w}_{y_i}^T \mathbf{x}_i \tag{2}$$

This can be equivalently written as

$$p(y_i|\mathbf{x}_i) = \frac{1}{Z} e^{\mathbf{w}_{y_i}^T \mathbf{x}_i} \tag{3}$$

where Z is a normalization factor. Subjecting to the following constrain:

$$\sum_{k=1}^{K} p(y_i|\mathbf{x}_i) = 1 \tag{4}$$

Z can be calculated as

$$Z = \sum_{k=1}^{K} e^{\mathbf{w}_k^T \mathbf{x}_i} \tag{5}$$

Note that K is the total number of classes. Based on maximum likelihood estimation (MLE), the optimization problem becomes:

$$\begin{aligned} \max_{\mathbf{W}} \mathcal{L}(\mathbf{W}) &= \max_{\mathbf{W}} \sum_{i=1}^{N} \ln p(y_i|\mathbf{x}_i) \\ &= \max_{\mathbf{W}} \sum_{i=1}^{N} \ln \frac{e^{\mathbf{w}_{y_i}^T \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k^T \mathbf{x}_i}} \end{aligned} \tag{6}$$

Note that $\mathbf{W}$ includes all $\mathbf{w}_k$ for $k = 1, 2, ..., K$. Applying gradient descent method, the gradient for each individual $\mathbf{w}_k$ can be obtained as:

$$\frac{\partial \mathcal{L}(\mathbf{W})}{\partial \mathbf{w}_k} = \begin{cases} \sum_{i=1}^{N} \mathbf{x}_i (1 - \frac{e^{\mathbf{w}_{y_i}^T \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k^T \mathbf{x}_i}}), & k = y_i \\ -\sum_{i=1}^{N} \mathbf{x}_i \frac{e^{\mathbf{w}_{y_i}^T \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k^T \mathbf{x}_i}}, & k \neq y_i \end{cases} \tag{7}$$

To obtain the optimum $\mathbf{W}$, iteratively adjust every $\mathbf{w}_k$ independently by the following expression:

$$\mathbf{w}_{k,t+1} = \mathbf{w}_{k,t} + \eta_t \frac{\partial \mathcal{L}(\mathbf{W})}{\partial \mathbf{w}_k} \tag{8}$$

where η_t is a step size which decreases with an increasing time t, specifically $\eta_t \propto \frac{1}{\sqrt{t}}$.

4. Training Example Sampling

As illustrated in equation (7), in each individual iteration, all the N training examples are supposed to be considered. Nevertheless, this is infeasible in large-scale training examples, i.e., with a significantly large N. As a result, performing sampling to form smaller sets of training examples for each iteration turns to be an effective way to address this problem.

The major idea is to randomly choose N_s training examples out of the entire set with N training examples, involve them in the gradient calculation and adjust each $\mathbf{w}_k$ according to equation (8) in every iteration. Note that ideally N_s can be any integer between 1 and

N, but practically some integers may be much more preferred considering both the performance and computation cost.

There are two major concerns regarding this sampling approach: 1) how many iterations do we need to derive an acceptable result? 2) how to implement fast samplings under a memory limit? For the first one, an intuitive answer would be the number of iterations needed probably will increase with a smaller N_s. Some extra information will be provided in the experimental results (Section 5) concerning the relation between the accuracy and the number of iterations. Here the second question is primarily focused on.

From the perspective of saving memory, sampling itself avoids reading the entire training example set into the memory; while at the same time it invokes the problem of randomly choosing training examples and efficiently reading them into the memory. It would be time-consuming to directly operate on the text documents, since sequential access wastes a lot of time on meaningless reading. To save time, the text documents are first converted into binary ones to enable random access, which significantly accelerates the random training example sampling.

5. Experimental Results

Based on the proposed approach, a cross validation experiment is conducted to analyze and evaluate the performance of the proposed classification approach. In each iteration, $N_s = 250$ training examples are randomly sampled from the entire set to participate the calculation. Table 1 reports the cross validation accuracy, mean average precision (MAP) and the consumed time under different number of iterations.

Table 1. Cross Validation Report on the Proposed Approach Under Different Numbers of Iterations

No. of Iteration	Accuracy /MAP	Training Time	Testing Time
100	29.11%/22.40%	4min57s	30s
200	30.29%/25.23%	9min9s	38s
500	31.53%/30.53%	18min57s	35s
1000	32.40%/35.56%	37min46s	32s
2000	32.99%/39.37%	1h7min31s	31s
5000	33.74%/44.31%	1h54min28s	30s
10000	34.03%/47.98%	3h58min20s	31s
20000	34.40%/49.94%	6h37min15s	30s

According to the cross validation report, after 100 iterations, the multi-class logistic regression model can achieve a cross validation accuracy of approximately 30%. The accuracy can be slightly increased with an increasing number of iterations, but the increasing speed decreases sharply when the number of iterations increases. Nevertheless, the MAP value increases a lot with an ascending number of iterations. This is probably because that extra iterations offer an opportunity for the classes with less training examples to become adequately trained. The training time is approximately proportional to the number of iterations, while the testing time depends only on the volume of the testing set and thus remains almost the same here.

Considering the performance can possibly be improved by regularization, the experiment with regularization is also conducted. However, the results demonstrate that no improvement can be made no matter what the regularization coefficient λ is set to be. The cross validation accuracy with different λ ($\lambda = 0$ means no regularization) is shown in Table 2.

Table 2. Cross Validation Report on the Proposed Approach Under Regularization

Regularization Coefficient λ	Cross Validation Accuracy
0	29.11%
0.01	28.56%
0.05	28.83%
0.25	28.57%
1	29.07%
5	29.08%
25	28.07%
100	26.02%

To learn the parameters for the final prediction, all the training examples (N_{train}=1000000) in the training set are involved in the training procedure. The number of iterations is set to be 5000, which roughly covers all the training examples under the sampling training scheme with $N_s = 250$. Both the time and memory cost are listed in Table 3 for this very case of training.

Table 3. Time and Memory Cost of the Final Training and Prediction

No. of Iterations	Training Time	Prediction Time	Peak Memory
5000	3h27min43s	49s	1.764Mb

6. Additional Discussions

In this section, several other methods are introduced to tentatively improve the multi-class logistic regression model. Some of them are untested, and the others probably cannot bring any promotion on the performance, at least under the current experimental conditions. However, they are summarized here to provide some helpful ideas, which are possibly useful for other classification tasks or at their revised versions, although currently they remain as untested or failed trials.

- *Smart Sampling:* Under the current sampling scheme, training examples involved in each iteration are chosen from the training example pool in a completely random way. The problem is that the numbers of training examples associating to different classes are unbalanced, and this will lead to an inadequate learning for the classes with less training examples. A smart sampling can be designed to ensure roughly identical probabilities of being chosen for training examples associating to different classes. This can be an effective way in balanced learning, especially when we have the same expectation on the prediction accuracy for each class.

- *Principal Component Analysis (PCA):* PCA is usually employed to reduce the data dimension at the minimum loss of statistical information. The decrease of data dimension can considerably cut the computational cost, whereas the performance is more or less degraded compared with the PCA-absent case. Moreover, the additional computation cost and the required data process on testing data caused by PCA also make it less preferable.

- *Bad Data Elimination:* Considering that there might be some training examples, which locate extremely far away from the concentrated zone corresponding to its associated class in the hyperplane, a bad data elimination approach based on location deviation is proposed to exclude these misleading training examples. Nevertheless, the experimental result shows that it contributes little in the accuracy promotion.

- *Feature Vector Expansion:* We are told that some of the classes in the training set are merged from multiple classes. I tried to expand the feature vectors in a polynomial way and made those classes distinguishable in the expanded hyperplane, but this also leads to little improvement on the cross validation accuracy. It is probably a good idea to first identify the special features which make those classes ambiguous, and then expand these features in a larger polynomial degree.

7. Conclusions

In this project, the multi-class logistic regression model is adopted to perform the large scale image classification. Its computation cost is significantly reduced by performing a random training example sampling. The experimental results demonstrate that the proposed approach can successfully achieve an acceptable prediction accuracy, and has a considerable reduction on implementation cost. A tradeoff between prediction accuracy and implementation cost can be made simply by choosing an appropriate number of iteration. Some extra efforts are also made to try to improve the developed approach. Some of them are experimentally proved to be ineffective, but smart sampling remains hopeful in decreasing the cost to achieve a certain prediction performance.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Singapore: Information Science and Statistics.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, *13*, 21–27.

Greene, W. H. (1993). *Econometric analysis*. New Jersey: Prentice Hall.

Hsu, Chih-Wei; Chang, C.-C., & Lin, C.-J. (2003). *A practical guide to support vector classification* (Technical Report). Department of Computer Science and Information Engineering, National Taiwan University.