Large-Scale Image Pattern Recognition Parallel Machine Learning Implementation

(According to first name's initial's order in English vocabulary)
Haitang Hu, Huizhan Lv, Jian Jin, Tianyi Chen

December 9, 2014

Abstract

Image recognition requires high workload because of its intrinsic complex feature space. When it comes to large amount of pictures, common machine learning algorithm shows obvious limitation in pattern recognition in terms of computation efficiency, resource occupation, etc. Thus techniques developed specifically to deal with large-scale data processing could be expected to have obviously superior performance. In this project we implement parallel SVM based on PCA in large-scale image pattern recognition and verify this scheme works in such kind of task with favourable performance as we have expected.

1 Background and Motivation

Pattern recognition concerning image has the mark of complex feature space. Consider a 32×32 color picture, each picture will have 3072 features taking RGB scheme into account. For a dataset of 60,000 pictures, it is quite a huge workload for the classification task.

However the ability to deal with classification in image recognition is of quite significance. For instance, when we look for images by inputting keyword "car" through a search engine, clearly it should distinguish between distinct types of objects and we don't expect a picture of a boat in the search result.

We aim to develop an algorithm which makes classification of large-scale image dataset in an efficient manner, that is to say, gives out result quickly with a high level of accuracy.

2 Approach

2.1 Preprocessing

We use PCA for preprocessing. In PCA, the goal is to project the data $\mathbf{x}$ with dimensionality D onto a space having dimensionality $M < D$ while maximizing the variance of the projected data.

Suppose S is the data covariance matrix defined by

$$S = Cov(\mathbf{x}) \tag{1}$$

Let $\{z_i\}$ represent M linear combinations of our original D predictors:

$$z_m = \mathbf{w}_m^T \mathbf{x} = \sum_{j=1}^{D} w_{jm} x_j, \qquad m = 1, 2, ..., M \tag{2}$$

We now maximize the projected variance

$$Var(z_1) = \mathbf{w}_1^T S \mathbf{w}_1 \tag{3}$$

Constraint condition is $\mathbf{w}_1^T \mathbf{w}_1 = 1$. Make an unconstrained maximization of

$$w_1^T S w_1 - \lambda_1 (w_1^T w_1 - 1). \tag{4}$$

where λ_1 is a Lagrange multiplier. The optimization result is

$$S\mathbf{w}_1 = \lambda_1 \mathbf{w}_1 \tag{5}$$

So

$$w_1^T S w_1 = \lambda_1 \tag{6}$$

which reveals that the variance will be a maximum when we set $\mathbf{w}_1$ equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector $\mathbf{w}_1$ is the first principal component.

The second principal component $\mathbf{w}_2$ should also maximize variance, be of unit length, and be orthogonal to w_1. We could find that $\mathbf{w}_2$ should be the eigenvector of S with the second largest eigenvalue. Similarly, we can show that the other dimensions are given by the eigenvectors with decreasing eigenvalues.

Since there are too many features in the task, we use PCA as preprocessing, which reduces 3072 features to 100 features. We projected all the features on the first 100 eigenvectors thus acquiring the data for classification. Also, to facilitate the computation of eigenvectors, we use SVD

$$X = U \Sigma V \tag{7}$$

where X is data matrix. Then the eigenvectors of covariance matrix of X will be the columns of the matrix V.

2.2 Processing

We use parallel SVM with Gaussian kernel for classification, which runs on four seperate machines. The standard SVMs are binary classifiers using a linear decision boundary with disciminant function

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x}^t + w_0 \tag{8}$$

of which the decision function is

$$y_{new} = \text{sign}(\mathbf{w}^T \mathbf{x}_i + w_0) \tag{9}$$

In our tasks there is no guarantee that the dataset is linear separable, so we introduce kernel function which allow non-linear dicision boundaries.

Kernel is based on transformation of original features, define the basis functions

$$\mathbf{z} = \phi(\mathbf{x}) \text{ where } z_j = \phi_j(\mathbf{x}),\ j = 1, ..., k \tag{10}$$

mapping from the d-dimensional $\mathbf{x}$ space to the k-dimensional $\mathbf{z}$ space where we write the discriminant as

$$g(\mathbf{z}) = \mathbf{w}^T \mathbf{z} \tag{11}$$

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \textstyle\sum_{j=1}^{k} w_j \phi_j(\mathbf{x}) \tag{12}$$

The dual in standard SVM problem is now

$$L_d = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \phi(x_i)^T \phi(x_j) \tag{13}$$

subject to

$$\sum_i \lambda_i y_i = 0, \quad 0 \leq \lambda_i \leq C,\ \forall i \tag{14}$$

The idea in kernel machines is to replace the inner product of basis functions, $\phi(x_i)^T \phi(x_j)$, by a kernel function, $K(x_i, x_j)$, thus the kernel function also shows up in the discriminant

$$g(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = [\sum_{i=1}^{k} \lambda_i y_i \phi(\mathbf{x}_i)^T] \phi(\mathbf{x}) = \sum_{j=1}^{k} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) \tag{15}$$

Use this kernel function, we do not need to map it to the new space at all. There are multiple types fo kernels and we use Guassian kernel

$$K(x, x') = exp(-\frac{1}{2}(x - x')^T \Sigma^{-1} (x - x')) \tag{16}$$

Besides kernel, we make another extension from binary class to $K-$classes by applying 1-of-K encoding scheme, in which y is a vector of length K containing a single 1 for the correct class and 0 elsewhere. For example, if we have $K = 5$ classes, then an input that belongs to class 2 would be given a target vector:

$$y = (0, 1, 0, 0, 0)^T \tag{17}$$

2.3 K-means

Comparing with SVM, K-means is another clustering algorithm.In we don't know the exact number of labels.

2.4 Percolation Clustering Algorithm

Besides SVM and k-means, we also realize another clustering algorithm which can be used in the case that we don't know the number of labels(SVM), and we don't have the expectation of number of labels(k-means). This algorithm derived from the percolation algorithm of Newman []. The original algorithm is realized on lattice plane. We modified it to let it suitable for our continuable space. In our derived algorithm, each data point i has other data points which are within the circle of radius R centred at data point i as its neighbors. Radius R is our parameter. After each data determining their neighbors, a directed graph will be formed. We w

3 Framework and Implementaion

3.1 Framework-Apache Spark

Apache Spark is a fast and general engine for large-scale data processing, which is could work in standalone cluster mode, on EC2, or run it on Hadoop YARN or Apache Mesos. Also, it can read from HDFS, HBase, Cassandra, and any Hadoop data source.

Spark introduces an abstraction called resilient distributed datasets (RDDs). Spark can outperform Hadoop multiple times, which is the reason we choose it as the framework of our task. It supports both python and Java. We use python for the task.

3.2 Implementations

4 Result and evaluation

5 Conclusion

Image recognition requires high workload because of its intrinsic complex feature space. When it comes to large amount of pictures, common machine learning algorithm shows obvious limitation in pattern recognition in terms of computation efficiency, resource occupation, etc. Thus techniques developed specifically to deal with large-scale data processing could be expected to have obviously superior performance. In this project we implement parallel SVM based on PCA in large-scale image pattern recognition and verify this scheme works in such kind of task with favourable performance as we have expected.

References

[1] H. Simpson, *Proof of the Riemann Hypothesis*, preprint (2003), available at `http://www.math.drofnats.edu/riemann.ps`.