Project Proposal
Scalable Machine Learning

Haitang Hu : hthu@jhu.edu
Tianyi Chen: tchen59@jhu.edu
Huizhan Lu : hlu27@jhu.edu
Jian Jin : jjin20@jhu.edu

September 26, 2014

Abstract

We want to build a scalable learning engine in order to find the features of dataset,and the relationship between different datasets. In order to realize this goal, we will use the java to build Hadoop Mapreduce Framework, and implement several parallel machine learning algorithms. Some mechanisms will be used, like dimensionality reduction, and some evaluation will be offered as well.

1 Problem Definition

We want to build a scalable learning engine in order to find the features of dataset,and the relationships between different datasets.Since the data we got from last view engine are the big scale ones with selection-feature distribution, which means only several features of these data are useful, others are not. Hardly can we deal with such data directly without any simplification, thus, some mechanisms will be used to simplify them, like PCA, CCA to reduce their dimensionality. During these techniques, we should also trade off between lower dimensionality and low information loss. It will depend on specific precision of our aimed optimization. By using such simplification, and the input data, we will output the features of datasets, and the low-dimensionality data.

2 Timeline of Project

As the timeline of our project, we have three milestones. The first milestone is from week 5 to week 6. We plan to use these two weeks to complete the system construction. We will finish the background research, build infrastructure and complete software selection. From week 7 to week 10,, we will use the platform we built in the last two weeks to implement several learning algorithms in order to find the features of input data and output the features and low-dimensionality data. Then, from week 10-week 11, we will also complete the evaluation of our project, we will focus on the reconstruction accuracy comparism, learning algorithm efficiency, and learning rate tuning.

3 Infrastructure

3.1 Map-reduce Framework

Map-reduce is a common programming model to deal with large data sets with a parallel algorithm on a cluster. In the first step, the map-reduce engine receives and splits the data set

from the last engine into shards. The master node in the system will assign the tasks and the shards to other mappers. The mappers then pass the processed data back to the master node. Finally, the master node invokes the reducer to use the data and calculate the final result. Some algorithms, like PCA, will need to use some information, particular some scalar products, in the process of learning. To achieve this we will need to store this information and make it possible for the mappers and reducers to access.

3.2 Application

On language we prefer to use Java to construct the system and the algorithms. We'd like to use Apache's Hadoop framework for analyzing datasets.

4 Dimensionality Reduction Algorithm

There are two main methods for reducing dimensionality: feature selection and feature extraction. We apply feature extraction, thus are interested in finding a new set of k dimensions that are combinations of the original d dimensions. These methods may be supervised or unsupervised depending on whether or not they use the output information.

4.1 PCA

Principal components analysis (PCA) is an unsupervised method in that it does not use the output information; the criterion to be maximized is the variance. Consider a data set of observations $\{x_n\}$ where $n = 1, ..., N$. Our goal is to project the data onto a space having dimensionality $M < D$ while maximizing the variance of the projected data. The learning task is to choose how many dimensions we want to project to and then pick a projection vector w_i for each.

PCA defines M vector, w_i where $i = 1, ..., M$, each of which is $N-$dimensional. For a sample x_n, the i th element of the projection is the projection of x_n on the direction of w_i, $z_i = w_i^T x_n$. The principal component is w_1 such that the sample, after projection on to w_1, is most spread out so that the difference between the sample points becomes most apparent. For a unique solution and to make the direction the important factor, we require $||w_1|| = 1$, or $w_1^T w_1 = 1$. The mean of the projected data is $w_1^T \overline{x}$ where $\overline{x}$ is the sample set mean given by

$$\overline{x} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{1}$$

and the variance of the projected data is given by

$$\frac{1}{N} \sum_{n=1}^{N} \{w_1^T x_n - w_1^T \overline{x}\}^2 = w_1^T S w_1 \tag{2}$$

where S is the data covariance matrix defined by

$$S = Cov(x) = \frac{1}{N} \sum_{n=1}^{N} (x_n - \overline{x})(x_n - \overline{x})^T \tag{3}$$

We now maximize the projected variance

$$Var(z_1) = w_1^T S w_1 \tag{4}$$

We seek w_1 such that $Var(z_1)$ is maximized subject to the constraint condition $w_1^T w_1 = 1$. To enforce this constraint, we make an unconstrained maximization of

$$w_1^T S w_1 - \lambda_1 (w_1^T w_1 - 1). \tag{5}$$

where λ_1 is a Lagrange multiplier. By setting the derivative with respect to w_1 equal to zero, we see that this quantity will have a stationary point when

$$Sw_1 = \lambda_1 w_1 \tag{6}$$

If we left-multiply by w_1^T and make use of $w_1^T w_1 = 1$, we see that the variance is given by

$$w_1^T S w_1 = \lambda_1 \tag{7}$$

and so the variance will be a maximum when we set w_1 equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector is known as **the first principal component**. An eigenvector of a square matrix A is a non-zero vector v that, when the matrix multiplies v, yields a constant multiple of v, the latter multiplier being commonly denoted by λ. That is:

$$Av = \lambda v \tag{8}$$

The number λ is called the eigenvalue of A corresponding to v.

The **second principal component**, w_2, should also maximize variance, be of unit length, and be orthogonal to w_1. This latter requirement is so that after projection $z_2 = w_2^T x$ is uncorrelated with z_1. For the second principal component, we have

$$\max_{w_2} \{w_2^T S w_2 - \lambda_1 (w_2^T w_2 - 1) - \lambda_2 (w_2^T w_1 - 0)\}. \tag{9}$$

Similarly we have $Sw_2 = \lambda_1 w_2$, which implies that 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.

4.2 CCA

For multiple-output linear model, the least squares estimates are simply the individual least squares estimates for each of the outputs. Other more sophisticated strategies exploit correlations in the different responses. Combining responses is at the heart of canonical correlation analysis (CCA), a dimendion reduction technique developed for the multiple output case.

CCA identify and measure the associations among two sets of variables. It determines a set of canonical variables, the orthogonal linear combinations of the variables within each set that best explain the variability both within and between sets. Given two vectors $X = (x_1, \ldots, x_n)^T$ and $Y = (y_1, \ldots, y_m)^T$ of random variables with finite second moments, the covariance matrix $\Sigma = \mathrm{cov}(X, Y)$ to be the $n \times m$ matrix that $\Sigma_{i,j} = \mathrm{Cov}(x_i, y_j)$. Canonical-correlation analysis (CCA) seeks vectors w_1 and w_2 such that the random variables $w_1^T X$ and $w_2^T Y$ maximize the correlation $\rho = \mathrm{corr}(w_1^T X, w_2^T Y)$.

The random variables $U = w_1^T X$ and $V = w_2^T Y$ are the first pair of canonical variables. Then one seeks vectors maximizing the same correlation subject to the constraint that they are to be uncorrelated with the first pair of canonical variables; this gives the second pair of canonical variables. This procedure may be continued up to $\min\{m, n\}$ times.

5 Evaluation and Performance Tuning

5.1 Accuracy Comparism

1. Learning result accuracy
   Comparing the learning result with the real data.
   Also, use recall,precision,F1 score to evaluate the accuracy of prediction.

2. Reconstruction loss
   Evaluate the information loss by comparing the reconstructed feature values to original feature vales. The less the information loss, the better the performance.
   PS: Still need to trade-off between the dimensionality reduction and information loss.

5.2 Learning Algorihm Architecture Refinement

1. Parallelize the learning algorithm through Hadhoop
   Identify the bottle neck of our implementation on Hadhoop.
   Try to deploy different mechanism (such as lock free SVD computation) and evaluate the performance.
2. Refine design/implementation of whole systems
   Try to find out limitation of the whole system design(architecture/network/language/etc.), and tuning the specific part or adjust components interfaces.

5.3 Learning Rate Tuning

Use different learning rate tuning algorithm, shuffle the initial status, to achieve a good balance between learning efficency and coverge rate.

6 Dataset

public data set: Amazon Public Dataset