Feature Selection Using Evidence Function on Bayesian Linear Regression
-- Course Project Report for COMP-136, Fall 2011

Mengfei Cao MCAO01@CS.TUFTS.EDU

Abstract

The work presents a feature selection method using evidence maximization for linear regression model. This method works for any given number as the size of selected feature subset and gives the MAP solution to regression problem that also maximizes the evidence function. Experiments are conduct by looking at the optimality through comparing two variants of this feature selection method to Bayesian regression with a shared prior; another experiment is conduct by comparing the feature selection method using evidence function and that using correlational co-efficient.

1. Introduction

Feature selection is a critical issue in various research areas such as computational biology, where datasets are provided with several features but it is unknown which features are relevant or there is the problem of curse of dimensionality. Typically these two purposes are also the general goals in other fields; in addition improving the prediction performance, enhancing model describability and exalting the understanding of the data's generation are also vital and will benefit from the solution to feature selection. Many methods have been brought forward such as wrapper method and been used widely; this paper mainly concentrates on the feature ranking in Bayesian linear model for regression.

Various methods to measure the correlation between features and target have been studied such as Pearson correlation coefficient, performance of single feature classifiers and so on. However, there are issues such as complexity, redundancy and optimality that would provide compromise between measurements on each method.

Next the Bayesian linear model for regression is introduced briefly and the evidence approximation is followed. What is interesting is that this evidence approximation not only offers choices of models, but also offers the opportunity to conducting feature selection by ranking features according to correlational values.

Dept. Computer Science, Tufts University, Medford, MA 02155, USA

1.1 Bayesian Linear Regression

The Bayesian linear model for regression is used to predict the value of one or more continuous target variables t given the value of a D-dimensional vector x of input variables. This paper focuses on the situation where there are N observations $\{x_n\}$, where $n=1,\dots,N$, together with single valued target values $\{t_n\}$; the purpose is to approximate the value of t for a new value of x through a linear combination of feature basis. Specifically, a basis function $\Phi(x)$ is given so as to allow non-linearity w.r.t. input features vectors x. The following computation follows:

$$y(\boldsymbol{x},\boldsymbol{w}) = \sum_{i=1}^{M} w_i \emptyset_i(\boldsymbol{x}) \tag{1}$$

where $w=(w_0,\dots,w_{M-1})^T$ and $\Phi=(\Phi_0,\dots,\ \Phi_{M-1})^T$.

In order to characterize the observations in reality, Gaussian noise is added so that the target value t is:

$$t(\boldsymbol{x}) = \ y(\boldsymbol{x},\boldsymbol{w}) + e \tag{2}$$

where $e \sim N(0, \frac{1}{\beta}\mathrm{I})$, and β is the model parameter.

Therefore t has the normal distribution $N(y(x,\boldsymbol{w}),\frac{1}{\beta}\mathrm{I})$. The task is to calculate $\boldsymbol{w}$ so as to maximize the posterior probability of $\boldsymbol{w}$. The posterior follows the form:

$$p(\boldsymbol{w}|t,\boldsymbol{x},\alpha,\beta) \sim p(t|\boldsymbol{x},\boldsymbol{w},\beta) * p(\boldsymbol{w}|\alpha) \tag{3}$$

where $p(\boldsymbol{w}/\alpha)$ is the preset prior in normal distribution $N(m_0,S_0)$, and the setting follows $m_0=0$, $S_0=\alpha$*(diagonal matrix)* for simplicity. Here α is a matrix as model parameter. As for the likelihood, it has the form:

$$p(\boldsymbol{t}|x,\boldsymbol{w},\beta) = \prod_{n=1}^{N} p(t_n|x_n,\boldsymbol{w},\beta) \tag{4}$$

where $p(t_n|x_n,\mathbf{w},\beta)$ is Gaussian probability density function for observation x_n. (The product comes from the independent identically distributed assumption)

From equation (3), we can calculate the posterior distribution of $\boldsymbol{w}$. It follows Gaussian distribution $N(m_N, S_N)$:

$$m_N = S_N(\beta\phi^T t + S_0^{-1} m_0) \tag{5}$$

$$S_N = (\beta\phi^T\phi + S_0^{-1})^{-1} \tag{6}$$

Ideally by integrating over all these parameters as well as $\boldsymbol{w}$, we can make predictions on any new examples, but this computation is intractable. Instead we need to specify these parameters.

Now instead we can derive the MAP (maximum a posteriori estimation) solution for $\boldsymbol{w_{MAP}}$ using $\boldsymbol{m_N}$ because the mean of Gaussian distribution is also the mode. By applying parameters (α, β) and feature vectors into (5) and (6), we can solve the Bayesian regression problem. Nevertheless, even though we have simplified by setting $m_0 = 0$ and $S_0$to be a diagonal matrix, still $\alpha$and β representing the model remain to be determined as they are not universal but case-sensitive to different version space.

Next an evidence approximation method (Bishop [Chap 3.5], 2006) is introduced so as to unravel the model selection problem.

1.2 Evidence Maximization

Instead of integrating over all parameters and ***w***, an evidence function is defined by integrating ***w*** only and then a maximization routine is imposed. The evidence function ***E(*)*** is defined as the marginal probability of target **t**:

$$\mathrm{E}(\mathrm{x}, \alpha, \beta) = p(t|\boldsymbol{x}, \alpha, \beta)$$
$$= \int p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{w}, \beta) p(\boldsymbol{w}|\alpha) \mathrm{d}\mathbf{w} \qquad (7)$$

Then maximize this evidence by equaling the derivative to zero w.r.t α and β. The solution to these equations gives the optimal parameter values to α and β assuming that evidence characterizes model optimality.

Due to lack of closed form solutions, an iterative routine is applied and the convergent solution to α and β is obtained.

2. Feature Selection

It seems to be always true that the more information is utilized, the more accurate the model should be over training. However there are issues such as curse of dimensionality, or irrelevant features. Feature selection facilitates high-dimensional calculation in that it reduces the dimensionality. Also, under some circumstances the selected features can also avoid over-fitting problem in a lot of learning tasks by removing redundant or irrelevant features. Further, it enhances the interpret-ability of the model.

Naively, exhausting all subsets of the feature space will give the best one by learning on each subset. However, it requires exponential calculation and thus is computa-tionally intractable. In addition, different measurement of feature set might give different result, which adds up the difficulty in finding the general optimum but also gives the flexibility of finding optimal solution in specific case.

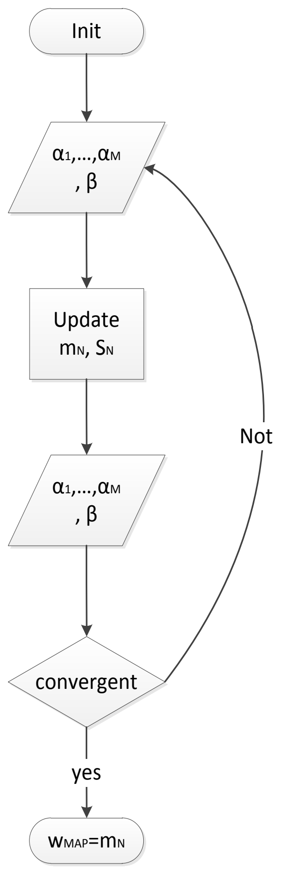

Figure 1. Model Selection Framework

Under the Bayesian linear regression situation, the prior gives an alternative to approximating optimal Recall that in the Bayesian regression setting, a conjugate prior is used for ***w***, where the mean values are zero and the covariance matrix is diagonal, such that every feature dimension is independent from each other and has variance $1/\alpha_i$, where *i=1,…,M*. Intuitively, when $\alpha_i = \infty$, this feature should have 0 variance and thus be totally irrelevant with target *t*; on the other hand, when α_i is close to 0, this feature should have values distributed widely in *R* on various different examples. Even though the latter case doesn't guarantee high dependency of target on this particular feature, yet it is sufficient to show that there is the correlation between target value and this feature. Therefore, by deriving $(\alpha_1, \ldots, \alpha_M)$ s.t. maximizing the evidence function, we can set a threshold to separate the relevant features from irrelevant features. The advantage is that this feature selection strategy gives the flexibility that for any given number *k*, it can result in a feature subset with size *k*, by cutting off at the k_{th} value among α vector.

Again, by taking derivation of (7) w.r.t α and β and setting to zero, we derive the iterative solutions to all parameters:

$$m_N = \beta S_N \phi^T t, \tag{8}$$

$$S_N = (diag(\alpha) + \beta\phi^T\phi)^{-1}, \tag{9}$$

$$\gamma_i = 1 - \alpha_i S_{N_{ii}}, \tag{10}$$

$$\alpha_i^{Iter+1} = \frac{\gamma_i}{m_i^2}, \tag{11}$$

$$\beta^{Iter+1} = \frac{\|t-\phi m_N\|^2}{N-\Sigma\gamma_i}, \tag{12}$$

where m_i is the i_{th}entry in m_N and $S_{N_{ii}}$is the element at i_{th} row and i_{th} column.

By setting initial values to α and β, then updating m_N, S_N, γ sequentially, we can calculate the new α and β, finally converging to optimal α^* and β^*.

Thus there is the feature selection algorithm based on evidence approximation method.

Algorithm 1 Model Selection Using Evidence

```
Input: feature vectors $\phi(\boldsymbol{x_i})$ ,i=1,…,N,  in $\boldsymbol{R}^M$; initial values $(\alpha_1^0, \dots, \alpha_M^0, \beta^0)$;
Init: $\beta = \beta^0$, $\alpha_i = \alpha_i^0$, i=1,…,M
repeat
    Update $\boldsymbol{m_N}, S_N, \gamma$ using (8), (9), (10);
until No changes between two iterations of parameters
Output: $\boldsymbol{w_{MAP}} = \boldsymbol{m_N}$ and $\alpha_1, \dots, \alpha_M$
```

Algorithm 2 Feature Selection Using Evidence

```
Input: feature vectors $\phi(\boldsymbol{x_i})$ ,i=1,…,N,  in $\boldsymbol{R}^M$; k ∈ $\boldsymbol{N}$, indicating the number of features to be selected;
  Step 1: apply Algorithm 1 and obtain $\alpha_1, \dots, \alpha_M$;
  Step 2: choose k columns $col_1, \dots, col_k$ s.t $\alpha_{col_i} \le \alpha^k$, where $\alpha^k$is the $k_{th}$value among $\alpha_i$ non-decreasing order;
  Step 3: apply Algorithm 1 on the subspace by selected k features and obtain $\boldsymbol{w_{MAP}} \in \boldsymbol{R^k}$;
Output: $\boldsymbol{w_{MAP}} \in \boldsymbol{R^k}$
```

Possible Problems:

- Convergence Condition: intuitively the concavity of evidence function guarantees that the iteration will terminate but it remains to be theoretically verified.
- Convergence Speed: even though compared to exhaustive search the feature selection using evidence function outperforms at speed, it might end up with extremely low speed when the iteration progresses slowly; thus it remained to be studied which optimization algorithm should be used so as to find parameters to maximize the evidence function.
- Global optimum V.S. local optimum: it is not guaranteed to obtain global optimum especially when multiple peaks are present; also it relies on the choice of initialization.
- Stability: regarding the computation issue, since equation (9) requires computing inverse of matrix, near-singular matrices make it difficult to progress, which in the experiment turns out to be a critical issue.
- Optimality: this algorithm doesn't distinguish repetitive features explicitly; namely if some feature is highly correlated with target but exists in multiple observations, therefore they will fill in more quota among the pre-defined feature set size k, than it is supposed to be. However it doesn't have to be true because the model has the assumption that features are independent from each other due to the zero co-variance in the prior. Therefore implicitly this algorithm might work on this problem but remains to be theoretically verified. Further this effect can be moderated by sequential feature selection through cross validation or other methods such as wrapper.

A specific issue is that during feature selection stage, two different variants can be applied so as to obtain final weight vector $\boldsymbol{w_{MAP}}$: one normal strategy is to train $\boldsymbol{w_{MAP}}$in the k-dimensional selected feature space using model selection module; the other faster way is just using the corresponding w_i w.r.t. selected column from $\boldsymbol{w_{MAP}}$ trained on the whole feature space. The former strategy (noted as *Retrain After Selection*) should suffice so as to give best model on the selected feature space as long as the iterative routine works. The latter strategy (noted as *Fast Train*), however, seems to degrade because the entry comes from the maximization on all feature space. Unless the removed features have no any contribution to predicting targets, the weight vector $\boldsymbol{w_{MAP}^*}$trained from the selected features should outperform because it has looser constraints in maximizing evidence function, resulting in higher evidence. Nevertheless, in the experiment it is shown that the latter strategy doesn't really output awkward performance but can save huge amount of time.

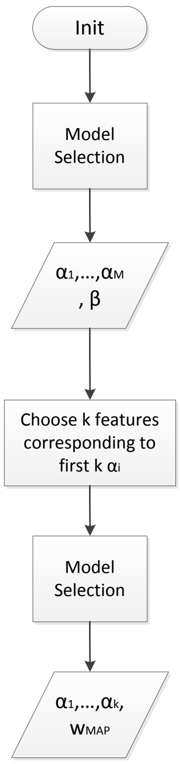

Figure 2. Feature Selection Framework

3. Experiments

3.1 Dataset

Three datasets are used with different number of features and examples so as to test the feature selection utility (Table 1), among which both training data and test data are provided.

3.2 Comparison Between Two Variants of Feature Selection Using Evidence Maximization

On each dataset, firstly three sets of weight vector $\boldsymbol{w}_{\boldsymbol{MAP}}$ are trained from the whole training data: two identical sets for feature selection, and one for model selection with fixed α. Then by setting k from 1 to M, where M is the dimensionality on whole training data, separately apply two strategies to obtain the final $\boldsymbol{w}^*_{\boldsymbol{MAP}}$, one using the original entry in $\boldsymbol{w}_{\boldsymbol{MAP}}$(Fast Train), and the other using model selection again to train $\boldsymbol{w}^*_{\boldsymbol{MAP}}$ (Retrain After Selection).

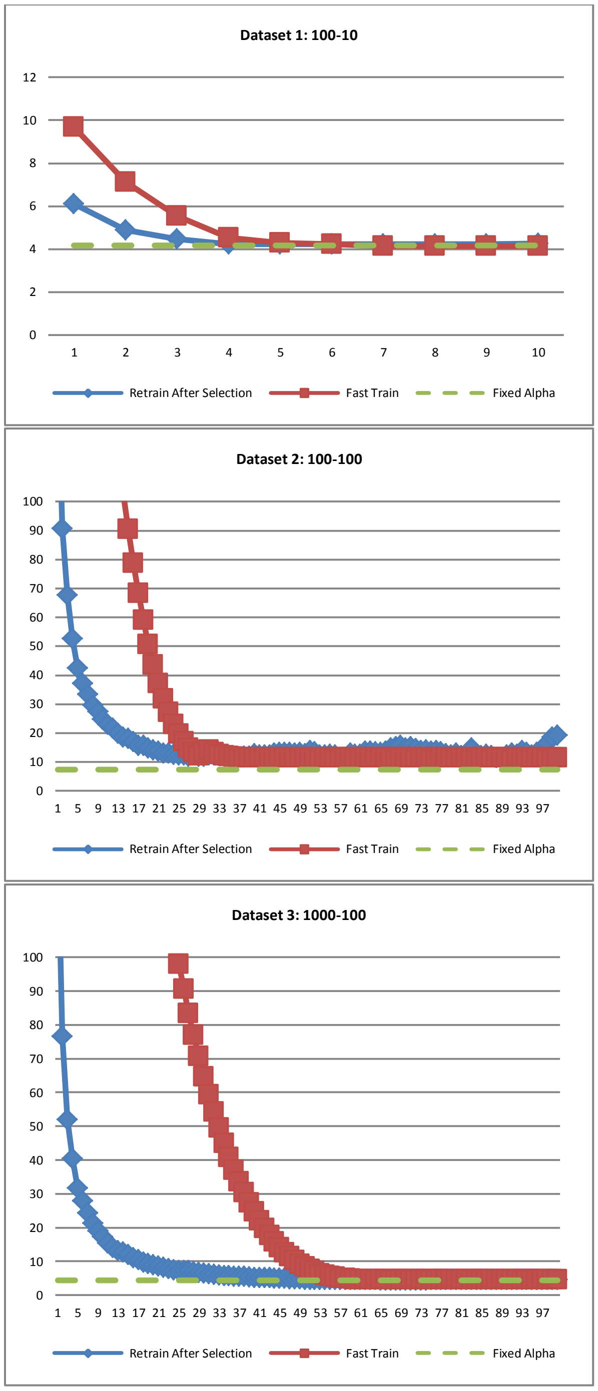

Figure 3. Comparison Between Retrain After Selection (namely train new ***w*** after feature selection) and Fast Train (namely use w entry of selected feature from first ***w***) as well as model selection using fixed α. (x-axis is the number of features selected, namely k; y-axis is the mean-square-error on test dataset)

As indicated in Fig. 3, on the first dataset, when the selected 5 features are used, the Bayesian linear model with different α_i can perform equally to that with the same α_i; this also holds for the other two datasets and the percentage of selected features so as to perform equally is at most 50%.

In addition, compared to Fast Train, the Retrain After Selection achieves earlier convergence and thus needs even less features. However, the running time degrades a lot; in particular after the first model selection, the Fast Train strategy only requires $O(MlogM)$ so as to select the k columns and corresponding k entries in w_{MAP}; but the Retrain After Selection requires another set of iterations so as to obtain the MAP solution, which is $O(Iter * MlogM)$ where *Iter* is the number of iterations before convergence. Typically one for the total 100 rounds of experiments on the second dataset, the time spent on Fast Train is less than 1 second while the time spent on Retain After Selection is more than 2 minutes.

Finally, actually it was expected that by applying different priors on each feature, the optimal MSE on test dataset should be lower than that using the same prior on all features. However, only on the first dataset the optimal MSE using different features is obtained when 7 features are selected and the value is 4.1536; the method using the same prior and all features gives MSE as 4.1801. On the other two datasets, the performance of model selection using the same prior has slightly better MSE than that with different priors. Intuitively by using different priors on features, it expands the weight vector space and thus may potentially obtain better solution; however according to the observations in the experiment, the issue of calculating matrix inverse is critical. When setting the convergence condition, it can't be two precise; otherwise there will be a cyclic process where it never terminates. Also the least-square based approximation is applied by using pinv() function in MATLAB. Therefore the precision limits the performance of both model selection and feature selection when different priors are added because from equations (9) and (10) the inverse has to be calculated explicitly and precisely.

3.3 Comparison Between Method Based On Correlational Co-efficient And That Using Evidence Function

Another intuitional measurement of correlation between features and target is by calculating the correlational co-efficient:

$$cc(\phi_i, t) = \frac{\sum_j(\phi_i(x_j)-\phi_i^*)(t_j-t^*)}{\sum_j(\phi_i(x_j)-\phi_i^*)^2 \sum_j(t_j-t^*)^2} \quad (14)$$

where ϕ_i is the i_{th} column in the training data, **t**=($t_1,\ldots,t_N$) is the label vector, $\phi_i(x_j)$ is the j_{th}example's i_{th}feature value, ϕ_i^* is the mean value of i_{th} feature, t* is the mean value of label vector.

Then apply this co-efficient vector to obtain the ranking of features. Thereafter for each given number *k*, select first ranked-*k* features as the subset. Followed is the routine where model selection is executed based on the selected feature space.

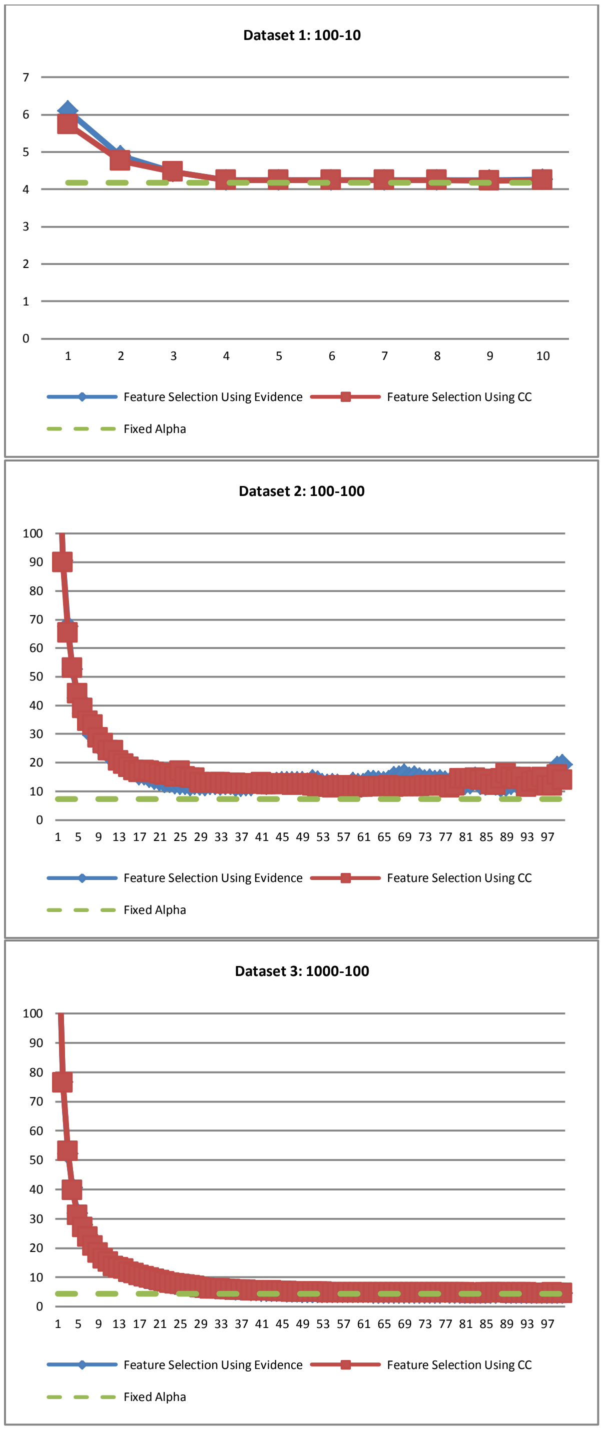

Figure 4. Comparison Between Feature Selection Using Evidence Function and Feature Selection Using Correlational Co-efficient. (x-axis is the number of features selected, namely k; y-axis is the mean-square-error on test dataset)

As shown in Fig. 4, these two feature selection methods give almost the same performance for different sizes of feature sets. Even though by visually checking the subset for each size, the resulting two subsets are not identical, however they give almost the same MSE on the test dataset.

Consider the complexity: the correlational co-efficient based method requires *O(M*N)* time to calculate the co-efficient for each feature; the evidence function based method requires *O(Iter*(M^3+M*N))*, where Iter is the number of iterations required for convergence, O(M^3) is used for calculating inverse and O(M*N) is used for matrix multiplications. Therefore in this case it seems that the correlational co-efficient based method outperforms.

However, consider the redundancy removal capability: it is for sure that the correlational co-efficient based method can't distinguish multiple repeated features at all; yet the evidence function based method might work implicity from the discussion in the previous sections.

4. Discussion

Further work regarding theoretical verification remaining to be done around convergence issue, optimality issue, etc. Also a typical application into computational biology is quite straightforward: distinguish positive protein families where a simulated evolution model fits from protein families where this simulated evolution model does not fit. The label of each family is represented by the difference between percentages of AUC (area under the curve) for ROC (receiver operating characteristic) on protein homology detection tasks with and without simulated evolution. Therefore it is a regression problem and the correlational values are needed so as to conduct feature selection.

Acknowledgments

It is a pleasure to thank Prof. Roni Khardon for excellent lectures in this semester. I also would like to thank Prof. Khardon for advice on the project, which made it possible.

References

Christopher M. Bishop. (2006) *Pattern Recognition and Machine Learning*. Springer..

Mitchell T. (1997) Machine Learning. McGraw Hill..

MacKay, D. J. C. (1992) The evidence framework applied to classification networks. *Neural Computation*, 4(5), 720-736.

Khardon R.,(2011) Statistical Pattern Recognition Course Webpage: http://www.cs.tufts.edu/~roni/Teaching/SPR/

Guyon I. and Elisseeff A.. (2003) An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157-1182.

Minka, T. (2000). Bayesian linear regression (*Technical Report*). MIT.

Kumar A. and Cowen L.. (2010) Recognition of beta structural motifs using hidden Markov models trained with simulated evolution. *Bioinformatics*, 26:i287-i293.

Table 1. Datasets Attributes.

DATA SET	#TRAIN-EXAMPLES	#TEST-EXAMPLES	#FEATURES
100-10	100	100	10
100-100	100	100	100
1000-100	1000	1000	100

Appendix

1. Model_Selection.m: select models, calculates model parameters.

```
function Model_Selection
%
%  Filename:      Model_Selection.m
%  Date:          Dec. 11th, 2011
%  Name:          Mengfei Cao
%  Descriptions:
%                 Take 3 datasets as input
with both training
%                 and test data, including
features and labels;
%                 1) Use iterative
algorithm to calculate the
%                 parameters that maximize
the evidence function;
%                  (so that by setting
different threshold we can
%                 select features from
alpha vector)
%                 2) Use the MAP solution
to calculate the MSE.
%
%  Code written in MATLAB 7.11.0(R2010b)
%

pnExampleCount = [100 100 1000];
pnFeatureCount = [10 100 100];

initvalue = [1000 10 500];
alpha = zeros(100,3);
mn_res = zeros(100,3);
```

```
for i=1:3
    tic;
    sTrainDataFile = sprintf('train-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestDataFile = sprintf('test-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTrainLabelFile = sprintf('trainR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestLabelFile = sprintf('testR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    X = csvread( sTrainDataFile );   phi =
X; clear X;
    t = csvread( sTrainLabelFile );
    test_data = csvread( sTestDataFile );
    test_label = csvread( sTestLabelFile );


    M = length(phi(1,:));
    N = length(phi(:,1));

    alpha1 = ones(M, 1);
    alphaI1 = diag(alpha1);
    beta1 = initvalue(i);

    alpha2 = 50000*ones(M, 1);
    alphaI2 = diag(alpha2);
    beta2 = 50000;

    MSE_test1 = 1; MSE_test2 = 0;
    mn2 = zeros(M,1);mn = mn2;
    while((abs(beta1-beta2)>0.00000001 ...
        || norm(alpha1-alpha2) >
0.00000001) ...
        && sum(isnan(alpha1))<1 &&
isnan(beta1)~=1 ...
        && max(alpha1)~=Inf &&
abs(MSE_test2-MSE_test1)>1e-5 )

        Sn_i = diag(alpha1)*eye(M) +
beta1*phi'*phi;
        if(sum(sum(isnan(Sn_i)))>0)
            break;
        end

        mn2 = mn;
        mn = beta1*(Sn_i\phi'*t);

        Sn_i = pinv(Sn_i);

        gama = 1 -
diag(diag(alpha1).*Sn_i);

        err = test_data*mn - test_label;
        MSE_test1 = MSE_test2;
        MSE_test2 = mean(err.*err);
        alpha2 = alpha1;
        beta2 = beta1;

        alpha1 = abs(gama./(mn.*mn));
        beta1 = (N-sum(gama))/((t-
phi*mn)'*(t-phi*mn));

    end
    alpha(1:M, i) = alpha2;
    beta(i) = beta2;
    MSE_test(i) = MSE_test1;
    mn_res(1:M, i) = mn2;

    tt(i) = toc;
end
save('projectres.mat', 'alpha', 'beta',
'MSE_test', 'mn_res', 'tt');
```

2. SelectedFeature.m: select features using alpha, retrain after selection

```
function SelectedFeature
%
%  Filename:      SelectedFeature.m
%  Date:          Dec. 11th, 2011
%  Name:          Mengfei Cao
%  Descriptions:
%                 Take alpha vector as
input and
%                 select feature by
choosing features
%                 with small alpha
%
%  Code written in MATLAB 7.11.0(R2010b)
%

load('projectres.mat');
%'alpha', 'beta', 'MSE_test', 'mn_res'

M = [10 100 100];
pnExampleCount = [100 100 1000];
pnFeatureCount = [10 100 100];
MSE = zeros(100, 3);
initvalue = [1000 10 500];
for i=1:3
    sTrainDataFile = sprintf('train-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestDataFile = sprintf('test-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTrainLabelFile = sprintf('trainR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestLabelFile = sprintf('testR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    X = csvread( sTrainDataFile );   phi =
X; clear X;
```

```
    t = csvread( sTrainLabelFile );
    test_data = csvread( sTestDataFile );
    test_label = csvread( sTestLabelFile );

    alphai = alpha(:,i);
    [alphai order] =
sort(alpha(1:M(i),i)');
    alphai = alphai';
    order = order';
    N = length(phi(:,1));
    phij = phi(:,order(1));
    test_dataj = test_data(:,order(1));
    for j=1:M(i)
        if j~=1
            phij = [phij phi(:,order(j))];
            test_dataj = [test_dataj
test_data(:,order(j))];
        end

        alpha1 = ones(j, 1);
        alphaI1 = diag(alpha1);
        beta1 = initvalue(i);

        alpha2 = 50000*ones(j, 1);
        alphaI2 = diag(alpha2);
        beta2 = 50000;

        MSE_test1 = 1; MSE_test2 = 0;
        mn2 = zeros(j,1);mn = mn2;
        while((abs(beta1-
beta2)>0.00000001 ...
            || norm(alpha1-alpha2) >
0.00000001) ...
            && sum(isnan(alpha1))<1 &&
isnan(beta1)~=1 ...
            && max(alpha1)~=Inf &&
abs(MSE_test2-MSE_test1)>1e-5)

            Sn_i = diag(alpha1)*eye(j) +
beta1*phij'*phij;
            if(sum(sum(isnan(Sn_i)))>0)
                ;
            else
                mn2 = mn;
                mn = beta1*(Sn_i\phij'*t);

                Sn_i = pinv(Sn_i);

                gama = 1 -
diag(diag(alpha1).*Sn_i);

                err = test_dataj*mn -
test_label;
                MSE_test1 = MSE_test2;
                MSE_test2 = mean(err.*err);

                alpha2 = alpha1;
                beta2 = beta1;

                alpha1 =
abs(gama./(mn.*mn));
                beta1 = (N-sum(gama))/((t-
phij*mn)'*(t-phij*mn));

            end
        end

        MSE(j,i) = MSE_test1;

    end
end
save('Final.mat', 'MSE');
```

3. SelectedVector.m: feature selection and use the corresponding weight vector entry instead of retraining.

```
function SelectedVector
%
%  Filename:      SelectedVector.m
%  Date:          Dec. 11th, 2011
%  Name:          Mengfei Cao
%  Descriptions:
%                 Take alpha vector as
input and
%                 select feature by
choosing features
%                 with small alpha;
%                 fast train: directly use
the w entry
%                 from original weight
vector
%
%  Code written in MATLAB 7.11.0(R2010b)
%

load('projectres.mat');
%'alpha', 'beta', 'MSE_test', 'mn_res'

M = [10 100 100];
pnExampleCount = [100 100 1000];
pnFeatureCount = [10 100 100];
MSE = zeros(100, 3);

for i=1:3
    sTrainDataFile = sprintf('train-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestDataFile = sprintf('test-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTrainLabelFile = sprintf('trainR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestLabelFile = sprintf('testR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    X = csvread( sTrainDataFile );   phi =
X; clear X;
    t = csvread( sTrainLabelFile );
    test_data = csvread( sTestDataFile );
    test_label = csvread( sTestLabelFile );
    alphai = alpha(:,i);
    [alphai order] =
sort(alpha(1:M(i),i)');
    alphai = alphai';
    order = order';
    phij = phi(:,order(1));
    mn = mn_res(order(1),i);
    test_dataj = test_data(:,order(1));
    for j=1:M(i)
        if j~=1
            phij = [phij phi(:,order(j))];
            mn = [mn;mn_res(order(j),i)];
            test_dataj = [test_dataj
test_data(:,order(j))];
        end

        err = test_dataj*mn - test_label;
        MSE(j,i) = mean(err.*err);

    end
end
save('SelectedVector.mat', 'MSE');
```

4. SelectedFeaturefromCoeff.m: feature selection using correlation coefficient

```
function SelectedFeaturefromCoeff
%
%  Filename:
SelectedFeaturefromCoeff.m
%  Date:           Dec. 11th, 2011
%  Name:           Mengfei Cao
%  Descriptions:
%                  Feature selection using
%                  correlation coefficient
%
%  Code written in MATLAB 7.11.0(R2010b)
%
load('coeffi.mat');
% 'R'

R = -abs(R);

M = [10 100 100];
pnExampleCount = [100 100 1000];
pnFeatureCount = [10 100 100];
MSE = zeros(100, 3);
initvalue = [1000 10 500];
for i=1:3
    sTrainDataFile = sprintf('train-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestDataFile = sprintf('test-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTrainLabelFile = sprintf('trainR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestLabelFile = sprintf('testR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    X = csvread( sTrainDataFile );   phi =
X; clear X;
    t = csvread( sTrainLabelFile );
    test_data = csvread( sTestDataFile );
    test_label = csvread( sTestLabelFile );

    [Ri order] = sort(R(1:M(i),i)');
     Ri = Ri';



    N = length(phi(:,1));
    phij = phi(:,order(1));
    test_dataj = test_data(:,order(1));
    for j=1:M(i)
        if j~=1
            phij = [phij phi(:,order(j))];
            test_dataj = [test_dataj
test_data(:,order(j))];
        end

        alpha1 = ones(j, 1);
        alphaI1 = diag(alpha1);
        beta1 = initvalue(i);

        alpha2 = 50000*ones(j, 1);
        alphaI2 = diag(alpha2);
        beta2 = 50000;

        MSE_test1 = 1; MSE_test2 = 0;
        mn2 = zeros(j,1);mn = mn2;
        while((abs(beta1-
beta2)>0.00000001 ...
            || norm(alpha1-alpha2) >
0.00000001) ...
            && sum(isnan(alpha1))<1 &&
isnan(beta1)~=1 ...
            && max(alpha1)~=Inf  &&
abs(MSE_test2-MSE_test1)>1e-5)

            Sn_i = diag(alpha1)*eye(j) +
beta1*phij'*phij;
            if(sum(sum(isnan(Sn_i)))>0)
                ;
            else
                mn2 = mn;
                mn = beta1*(Sn_i\phij'*t);

                Sn_i = pinv(Sn_i);

                gama = 1 -
diag(diag(alpha1).*Sn_i);

                err = test_dataj*mn -
test_label;
                MSE_test1 = MSE_test2;
                MSE_test2 = mean(err.*err);
                alpha2 = alpha1;
                beta2 = beta1;

                alpha1 =
abs(gama./(mn.*mn));
                beta1 = (N-sum(gama))/((t-
phij*mn)'*(t-phij*mn));

            end
        end

        MSE(j,i) = MSE_test1;

    end
end
save('FinalCorre3.mat', 'MSE');
```

5. CoefficientVector.m: calculate correlation coefficient for each feature

```
function CoefficientVector
%
%   Filename:      CoefficientVector.m
%   Date:          Dec. 11th, 2011
%   Name:          Mengfei Cao
%   Descriptions:
%                  Calculate Coefficient
%
%   Code written in MATLAB 7.11.0(R2010b)
%
pnExampleCount = [100 100 1000];
pnFeatureCount = [10 100 100];

R = zeros(100,3);

for i=1:3
    tic;
    sTrainDataFile = sprintf('train-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestDataFile = sprintf('test-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTrainLabelFile = sprintf('trainR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    sTestLabelFile = sprintf('testR-%d-
%d.csv', pnExampleCount(i),...
        pnFeatureCount(i) );
    X = csvread( sTrainDataFile );   phi =
X; clear X;
    t = csvread( sTrainLabelFile );
    test_data = csvread( sTestDataFile );
    test_label = csvread( sTestLabelFile );


    M = length(phi(1,:));
    N = length(phi(:,1));
    for j=1:M

        temp = corrcoef(t,phi(:,j));
        R(j,i) = temp(1,2);
    end

    tt(i) = toc;
end
save('coeffi.mat', 'R');
```

6. GetOrderDiff.m: look into the difference between two feature selection methods' result

```
function GetOrderDiff
%
%   Filename:      GetOrderDiff.m
%   Date:          Dec. 11th, 2011
%   Name:          Mengfei Cao
%   Descriptions:
%                  Look into the difference
%                  between two feature
selection
%                  methods
%
%   Code written in MATLAB 7.11.0(R2010b)
%
load('coeffi.mat');%R
load('projectres.mat');%alpha
alphaa = alpha(:,:);
clear alpha;

M = [10 100 100];
for i=1:3
    temp = R(1:M(i),i);
    [c d] = sort(temp);
    order_corre = d;


    temp = alphaa(1:M(i),i);
    [c d] = sort(temp);
    order_EV = d;
    diff = order_corre-order_EV;
    save(sprintf('order%d.mat',i),...
        'order_corre','order_EV','diff');
end
```