A Novel Framework for Classification using Regression routines

Sauptik Dhar

December 12, 2008

Abstract

This project presents a novel framework for classification using different regression routines.In this project we build a framework to represent such problems as an optimization problem.Using Sequential Quadratic Programming we solve the optimization problem to obtain the parameters of the parametric model.We obtain the final decision boundary and provide a comparison with the different classifiers based on their classification error.Further we also provide some insights from Predictive Learning perspective.

1 Introduction

Most existing implementations of the classification methods based on regression routines mainly minimize the squared error or cross-entropy error for optimal parameter selection. Although these loss functions are highly correlated to the classification error,there are situations when a reduction in the squared error or cross-entropy error does not minimize the classification error[1]. This motivated the design of new algorithms which adopts early stopping criteria to avoid any increase in the empirical classification error. However, most of these algorithms use some heuristic methodology to ensure such a condition [1]. In our project we present a better framework for dealing with such issues. Moreover for the sake of simplicity we use simple parametric regression routines and provide a comparison with the present methodologies used.

This report is organised as follows.In section 2 we present the basic problem formulation.Section 3 deals with the problem analysis and converting the problem to a new form.In section 4 we provide an algorithm for parameter tuning.Section 5 presents the results of the proposed methodology with the prevalant classifiers.Finally we conclude the report in section 6.We provide some open issues with the framework in section 7.

2 Problem Formulation

Consider a finite training sample $(\mathbf{x}, y)_n$.The training set consists of m-dimensional input vectors $\mathbf{x}$ and a univariate output y.The total number of samples is n.Using regression routines we can fit a model to this data minimizing the squared error. And then threshold the model to some threshold level. However, our ultimate goal is to classify the data and hence to reduce the misclassification error. So a proper way of doing this could be:

$$min\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

$$s.t\sum_{i=1}^{n} I(y_i \neq sign(f(\mathbf{x}_i, \mathbf{w}))) \leq \xi$$

where I(.) is an Indicator function and ξ is some predefined value equivalent to an acceptance level.We may also view it as a measure of the maximum classification error allowed.Moreover it can also be considered as a user tunable parameter.This shall be discussed in details in the section 4.

3 Problem Analysis

The Problem statement in (1) can be further rewritten as

$$min\sum_{i=1}^{n}(yi - f(\mathbf{x}_i, \mathbf{w}))^2$$

$$s.t\frac{1}{2}\sum_{i=1}^{n}(1 - y_i sign(f(\mathbf{x}_i, \mathbf{w}))) \leq \xi$$

where sign(x) function is defined as:-

$$\text{sign(x)} = \begin{cases} -1 & x \leq 0 \\ 1 & x \geq 0 \end{cases}$$

A number of methods have been proposed to provide a continuous representation of the sign function with different form of asymptotics for the best approximation [3],[4],[5]. Rather we select the most frequently used sigmoid function to provide the continuous form as seen in [6],[7].So we replace the sign(x) function using a tanh(x) function.The problem statement resolves to:

$$min\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

$$s.t\frac{1}{2}\sum_{i=1}^{n}(1 - y_i tanh(f(\mathbf{x}_i, \mathbf{w}))) \leq \xi$$

For the sake of simplicity in this report we shall only consider the Linear Estimators with a 1st order polynomial basis function.The final simplified problem thus resolves to:

$$min \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T * \mathbf{w})^2$$

$$s.t \frac{1}{2} \sum_{i=1}^{n} (1 - y_i tanh(\mathbf{x}_i^T * \mathbf{w})) \leq \xi$$

Moreover we observe that the objective function in our case is convex.But the constraint,in this case though continuous and differentiable,lacks convexity.This is due to the tanh function which is convex in the R_- region but is concave in the R_+ region.A good choice for solving such an optimization problem could be using the Sequential Quadratic Programming(SQP) method.In such a case we solve a Quadratic Programming(QP)subproblem in each step by expanding the Lagrangian in Qudratic terms and linearizing the constraint.

4 The Tunable Parameter

We observe that a good solution for the classification problem is mainly dictated by appropriately setting the constraint ξ.We consider this as the user tunable parameter.So the ulterior goal of this framework is to attain an optimum value for the constraint.A very large value of ξ is equivalent to putting no constraint at all.Again a very small ξ value may result in an infeasible system.For instance putting ξ=0 in case when the data is inseperable seems to be an overstatement.So a good ξ parameter is necessary for a good classifier.However as a least requirement we expect our Constrained Least Square Classifier(CLS) to perform better than Ordinary Least Square(OLS) without any constraints.So we may initialise ξ= Classification error due to an OLS solution.
We present a naive yet effective algorithm for selecting (rather searching) the optimal value of ξ.

```
Initialise
     preverr=currerr
     currconstr=Classification error for the OLS
     prevconstr=currconstr
     for k=1 to MaxIter
        const=$(0.5)^k$
        if (Previous Error $\prec$ currerr)
          Reduce the Previous constraint by a step of const.
        else
          Reduce the Current constraint by a step of const.
        end if
        Solve the Constrained Nonlinear LS problem using SQP
        preverr=currerr
        currerr=Classification Error due to the new model
     end for
     Select the best model obtained so far
```

The algorithm that is presented here is not the optimization algorithm.It is simply a heuristic search algorithm for the optimal constraint for a particular dataset.

5 Results

In this section we compare the performance of the newly introduced method with the prevalant classification methods.For comparison we use the following.
Performance Metrics:
1.Classification Error:Lower the Classification error,better is the model.
2.Time Complexity:This is the CPU Time taken for evaluation of the model.

We use the following methods.
1.Ordinary Least Squares(OLS) Classifier.(No constraints)
2.Constrained Least Square(CLS) Classifier.(with the prescribed constraint search method)
3.Fischer's Linear Discriminant Classifier.(FLD)
4.Linear Perceptron Classifier(LPC)(with maximum iteration=5000)
5.Linear Support Vector Classifier(SVC).
For the SVC the C parameter is chosen based on [11]within the range of [2^{-5},2^{10}]and the optimal model is selected based on the minimum classification error on the training data.Moreover we use the Sequential Minimal optimization algorithm[9] in this report.Most of these tools are available in [12].
The SQP problem for the CLS is solved by using the TOMLAB Optimization toolbox[13].Here we use the Schittkowski SQP [10].For this case we use a numerical difference method to find the Hessian rather than using the BFGS update.

5.1 Experiment Setup 1

We begin by using the two Dimensional Ripley's data set.This data set contains 250 Samples.The results are provided in the following Table.

Table 1: Performance of different methods on Ripleys dataset

Methods	Performance	
	Classification Error	CPU TIME(in sec)
OLS	0.144	0.11
FLD	0.144	0.04
LPC	0.412	0.511
SVC(Linear)	0.132	2.885
CLS(MaxItr=100)	0.124	35.15
CLS(MaxItr=500)	0.132	208.81

Here the best SVC model is obtained for C=2

5.2 Experiment Setup 2

In this case we provide the results obtained on the Haberman's dataset[15].The dataset contains 306 samples.Only two input variables Age and Number of nodes are used.The input variables are scaled in the range of [0,1].The results are provided in the following table.
Here the best SVC model is obtained for C=0.03125

Table 2: Performance of different methods on Haberman's dataset

Methods	Performance	
	Classification Error	CPU TIME(in sec)
OLS	0.2549	2.784
FLD	0.2549	0.04
LPC	0.69935	0.551
SVC(Linear)	0.2647	84.351
CLS(MaxItr=100)	0.22549	40.819
CLS(MaxItr=500)	0.22549	273.64

5.3 Experiment Setup 3

In this case we provide the results obtained on the Parkinson's dataset[14].In this dataset we omit the information of the nonlinear measures and the signal fractal exponent.The data set contains 195 samples.The data set used contains 15 Attributes and 1 Class label.The dataset is prescaled to the range of [0,1] before applying to the different classifiers.The overall performance of the different methods are provided below.

Table 3: Performance of different methods on Parkinson's dataset

Methods	Performance	
	Classification Error	CPU TIME(in sec)
OLS	0.17436	0.32
FLD	0.22051	0.03
LPC	0.1641	0.53
SVC(Linear)	0.1641	8.142
CLS(MaxItr=100)	0.092308	52.195
CLS(MaxItr=500)	0.076923	533.3

Here the best SVC model is obtained for C=64

5.4 Experiment Setup 4

In this case we compare the performance of the CLS(MaxItr=100) and CLS(MaxItr=500) methods.By MaxItr we refer to the Maximum iteration for the SQP algorithm to converge.So here we provide a comparison of the methods on a synthetic data.The synthetic data is a dataset with two overlapping Gaussians belonging to two classes with means(-1,-1) and (1,1) and deviation 1.The dimension of the input samples is 2.

We test by varying the number of samples from n=20,50,100.We provide the boxplots for the performance of the methods.(the boxplots have been generated by running the test 20 times)

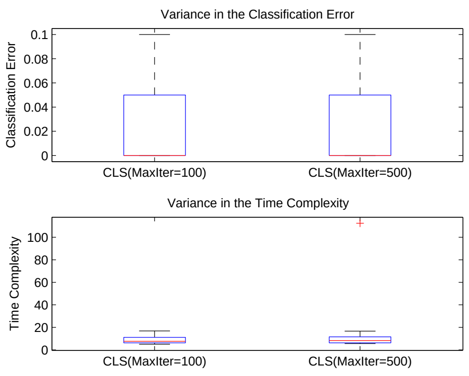

Figure 1: The comparison of the Performance of CLS for MaxItr=100 and MaxItr=500 for 20 samples

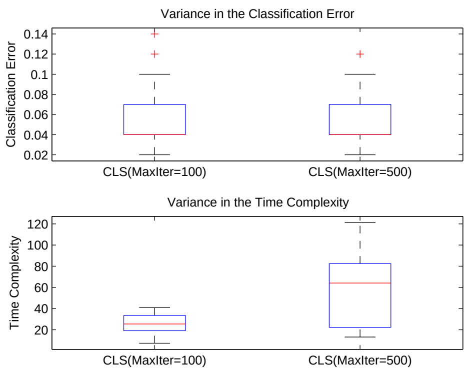

Figure 2: The comparison of the Performance of CLS for MaxItr=100 and MaxItr=500 for 50 samples

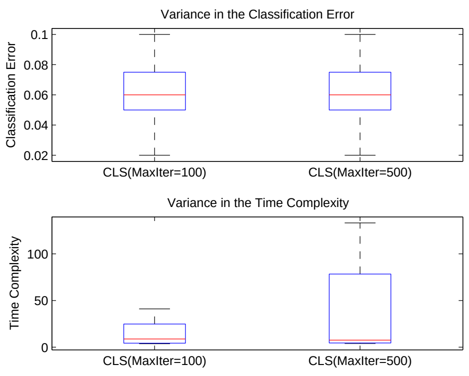

Figure 3: The comparison of the Performance of CLS for MaxItr=100 and MaxItr=500 for 100 samples

5.5 Result Analysis

We observe that the CLS outperforms most of the prevalant classifiers,but is inherently very slow.This can be attributed to the fact that the Heuristic for the selection of the appropriate constraint adopts somewhat a pseudo-exhaustive search.For the LPC,from our knowledge of the Perceptron rule we identify that the LPC model lacks convergence in case of a non-seperable dataset.This resulted to the ill performance of the LPC in the nonseperable case.As for SVC,we already know that Support Vector Machine Classifier is a universal classifier and can be used to learn a variety of representations.But here it seems somewhat limited owing to the heuristic applied for the selection of C.Optimal selection of C is very important for the performance of SVM Classifiers.However we also report that setting the value of C to a value more than 2^{11} takes a lot of time for model evaluation using SMO[9].This can be seen as a negative point for the SVC.The main point that we intend to make from the above discussion is that the CLS does perform reasonably better than most other classifiers,in the datasets used albeit the added overhead of the time complexity.A better claim could be made after a detailed study of the Problem structure and further consolidated on some more empirical results.However,one thing that still remains quite disturbingly unexplained is that in Table 1 the CLS(Maxitr=100) performs better than CLS(Maxitr=500).This seems totally confusing and needs further consideration.

Moreover from the boxplot analysis of the performance of the CLS(Maxitr=100)and CLS(Maxitr=500) we observe that both the methods have almost similar performance on the synthetic data.Moreover,we have seen that the CLS(Maxitr=100) is relatively faster in cases Experiment 1,2 and 3 with a small performance degradation.So a choice of smaller Maximum Iteration is desirable.But to what extent the maximum iteration needs to be small is still an open question.

6 Concluding Remarks

Although it seems a bit too early to provide conclusions,however from the empirical results that we have generated the CLS definitely seems to be a potential classifier in comparison to the other available linear classifiers.We can atleast claim that the CLS performs better than the OLS method.The framework could be further extended to nonlinear regression routines.However a possible limitation for such a framework could be that the framework remains valid only for Parametric methods.
Moreover,results direct towards using smaller Maximum Iteration counts with a slight degradation of the Classification Error.

7 Future Plans

Some of the results generated conflict our intuitions(viz.Table 1).So possible direction of future work could be:
1.Provide an explanation for the results generated in Table 1
2.Exploration of the Problem Structure could lead us to explore better Optimization methods.
3.Test the model for different kinds of data sets.
4.Development of such a framework for Nonlinear Parametric functions.

Moreover we also need to explore the implication of the approximation that we made in the constraint function.Although it seems reasonable that the present form provides a measure of the Classification error.It is not an exact representation of the Classification error.
We also need to note that the present Framework has no application from the Predictive Learning Perspective.From the problem statement provided above,it is pretty much clear that the model is likely to overfit.However a slight alteration in the constraint formulation may result in better methods for the Predictive Learning Framework.In short,for the present case this model should not be used for Predictive Learning.

APPENDIX

Listing 1: AllMethodsFINAL.m

```
%dat=load('riply_trn');    %This is available in STPRTOOL
load parkinson;            %This is a Processed data
%dat=data2(2);             %Habbermann Cancer Data Preprocessed

%*****************************************
%****USE STPRTOOL To run this module*****
%*****************************************

%#######    FISCHER LDA     ##########
t=cputime;
 model = fld(dat);
 ypred = linclass(dat.X,model);
 cerr_Fischer=cerror(ypred,dat.y);
 t_FISCH=cputime-t;



%########   PERCEPTRON CLASSIFIER  ##########
t=cputime;
options.tmax=5000;      %If it has not converged still It wont!!!
model = perceptron(dat,options);
ypred = linclass(dat.X,model);
cerr_Percep=cerror(ypred,dat.y);
t_PERCEP=cputime-t;


%############# SVM  Classifier ########

C=[];
for pwr=-5:10
    C=[C,2^pwr];
end

    options.ker = 'linear';
    options.C = C;
    options.solver = 'smo';
    options.verb = 1;
    options.arg=1;
    t=cputime;
    [model,Errors] = evalsvm(dat,dat,options);
    t_SVC=cputime-t;
    ypred = svmclass(dat.X,model);
    cerr_SVC=cerror(ypred, dat.y);
```

Listing 2: NewMethod.m

```
clear;
Name='Sauptik';
%***************************************************
%*******Load the Type of Data***********************
%***************************************************

%dat=load('riply_trn'); %Ripley's data
%load parkinson;        %Parkinson's Data(This relatively a new data set)
%dat=data2(2);          %Habberman Cancer data

%**************************************************
%**Preprocess the data for usage with TOMLAB*****
%*************************************************

X=dat.X';
X=[X,ones(size(X,1),1)];
y=dat.y';

for i=1:length(y)
    if(y(i)==2)
        y(i)=-1;
    end
end


%*****************************************************
%*******Initialise the Problem***********************
%*****************************************************
x_0 = ones(1,size(X,2));
%***************************************************
%From here we need the Liscenced version of TOMLAB
%***************************************************
%Note:The Liscence that I am sending is valid only upto Dec 19.

% CLSASSIGN::Assigns the Problem Structure.No Constraints

Prob = clsAssign('sauptik_r',[], [], [], [], Name, x_0, ...
y,[],[],[],[],[],[],[],[],[],[],[],[]);

Prob.user.X = X;                        %We use this variable in the Problem statement
t=cputime;
Result = tomRun('consolve', Prob, 0);   %Use SQP on the Problem.
t_OLS=cputime-t;

%**********************************************
%****We get the Model for the OLS*************
%**********************************************
```

```
model.W=Result.x_k(1:end−1);
model.b=Result.x_k(end);
ypred = linclass(dat.X,model);
cerr=cerror(ypred,dat.y);
cerr_OLS=cerr;
%*********************************************
%*********Initialise the CLS****************
%*********************************************
prev_cerr=cerr;
constr=cerr;
prev_constr=constr;
max=−7;                 %This is something that depends on the type of problem
for par=−1:−1:max
    param=2^par;
    if(prev_cerr<cerr)
        cerr=prev_cerr;
        constr=prev_constr;
        constr=constr−param*constr;
    else
        prev_constr=constr;
        constr=constr−param*constr;
    end
Prob = clsAssign('sauptik_r',[], [], [], [], Name, x_0, ...
y,[],[],[],[],[],[],[],[],'sauptik_c',[],[],[],constr);
Prob.user.X = X;
Prob.optParam.MaxIter =100;             %Here you can change the Maximum Iterations
Result = tomRun('consolve', Prob, 0);
model.W=Result.x_k(1:end−1);
model.b=Result.x_k(end);
ypred = linclass(dat.X,model);
prev_cerr=cerr;
cerr=cerror(ypred,dat.y);
if(cerr≤prev_cerr)
   final_model_CLS=model;           %Update the Final Model
end
end
%########The Final CLS Model#########
ypred = linclass(dat.X,final_model_CLS);
cerr_CLS=cerror(ypred,dat.y);

t_CLS=cputime−t;
```

References

[1] Vladimir Cherkassky and Filip Mulier. Learning From Data-2nd ed,Wiley 2007

[2] D.P. Bertsekas. Nonlinear Programming, Athena Press, 2004.

[3] Alexandre Eremenko and Peter Yuditskii. *Uniform approximation of sgn x by polynomials and entire functions.* J. d'Analyse Math., 101 (2007) 313-324

[4] F. Nazarov, F. Peherstorfer, A. Volberg and P. Yuditskii. *Asymptotics of the Best Polynomial approximations of IxIp and of the best Laurent Polynomial Approximation of sgn (x) on two symmetric intervals.*

[5] Hans H Hosenthein. *Nth order Flat Approximation of the Signum function by a Polynomial.* NASA TN D-6688,March 1972.

[6] Edward Wilson and Stephen M. Rock. *Gradient-based parameter optimization for systems containing discrete-valued functions.* International Journal of Robust and Nonlinear Control 2002, Int. J. Robust Nonlinear Control 2002; 12:10091028 (DOI: 10.1002/rnc.729)

[7] Edward Wilson. *Backpropagation Learning for Systems with Discrete-Valued Functions.* Proceedings of the World Congress on Neural Networks, San Diego, California, June 1994.

[8] Gang Ji and Jeff Bilmes. *Necessary Intransitive Likelihood Ratio Classifiers.* UWEE Technical Report Number UWEETR-2002-0014.

[9] Platt,J. *Fast training of Support Vector Machines using sequential minimal optimization,in B.Scholkopf,C.J.C Burges,and A.J.Smola(Eds.),Advances in Kernel Methods-Support Vector Learning,Cambridge,MA,MIT Press,1999,pp,185-208.*

[10] Klaus Schittkowski. *On the convergence of a Sequential Quadratic Programming Method with an Augmented Lagrangian Line Search Function*January 1982.

[11] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin *A Practical Guide to Support Vector Classfication.* http://www.csie.ntu.edu.tw/ cjlin Oct2,2008.

[12] http://cmp.felk.cvut.cz/cmp/software/stprtool/stprtool.pdf

[13] Private communication Marcus M. Edvall.Tomlab Optimization.http://tomopt.com/

[14] Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM. *'Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection'* BioMedical Engineering OnLine 2007, 6:23 (26 June 2007).

[15] Asuncion, A. and Newman, D.J. *UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.*(2007)