Convex Optimization Project Report: Prediction of movie revenues.

Hechuan Wang 109811150

1 Introduction

1.1 Description of the problem

It is very common in the real world that people want to predict the performance of some uncompleted projects based of the knowledge of previous projects. The same thing happens in the movie industry. The problem discussed in this project is to predict the revenues of some new movies according to some records of the previous movies.

The question is given in the form of two sets of data which describe some properties and revenue of movies. In the first set of data, ‘Tr’, revenue and other 7 properties are given. In the second data set, ‘New’, revenue is not given, but the properties, which is the same as the properties in ‘Tr’, are known. In this project, this question is considered as an approximation problem, where the revenue is considered as a *target variable (or target)*, and other 7 properties as *input variables (or inputs)*. The goal is to build up a model according to target and the inputs in the data set ‘Tr’, and predict the corresponding target variable according to the new input variables in the data se ‘New’.

1.2 Metric for measuring the performance

The real values of targets according to the inputs in the set ‘New’ is given at the end of the project, so it is possible to evaluate the performance of the prediction comparing the real targets and the predicted targets. 3 ways to quantify the performance of the prediction is used in the project.

In the project, one of the metric of performance is measured by a value R^2, which is defined by

$$R^2 = 1 - \frac{R_r}{R_t},$$

where

$$R_r = \frac{1}{1000}\sum_{n=1}^{N}(t_n - \widehat{t_n})^2,$$

and

$$R_t = \frac{1}{1000}\sum_{n=1}^{N}(t_n - \bar{t})^2,$$

where

$$\bar{t} = \frac{1}{1000}\sum_{n=1}^{N} t_n.$$

R^2 is a commonly used metric to compare how an estimated target set $\widehat{t_n}$ is close to the real target set t_n. It is a value less than 1, and the closer it is to 1, the better the estimation is. It can even be negative.

Maximizing R^2 is the same as minimizing the least-square-error of the prediction, but in many cases, minimizing least-square-error is not enough. The distribution accuracy of predictions is also very important: 1. what is the proportion of accurate predictions? 2. Are the worst cases too bad that make the model unacceptable?

A metric to determine the accuracy is designed in this project, which is named 'Relative inaccuracy':

$$R \triangleq abs\left(log_{10}\frac{y}{t}\right)\ y,t > 0,$$

where R is Relative inaccuracy; y is predicted value; t is target value. R=0 means prediction is exactly the same as real target. The higher R is the more inaccurate the prediction is. Then the proportion of accurate prediction can be defined as the proportion of predictions that have $R < log_{10}2$. The worst case can be defined as the largest R.

1.3 Organization of the report

Because the data, which describe movies, are recorded in real life, some of the number are not ready to use. Before the learning process, preprocessing is necessary, which is discussed in section 2. Computer simulation is used to test 3 methods for prediction, which is discussed in section 3. The result is discussed in section 4.

2 Preprocessing

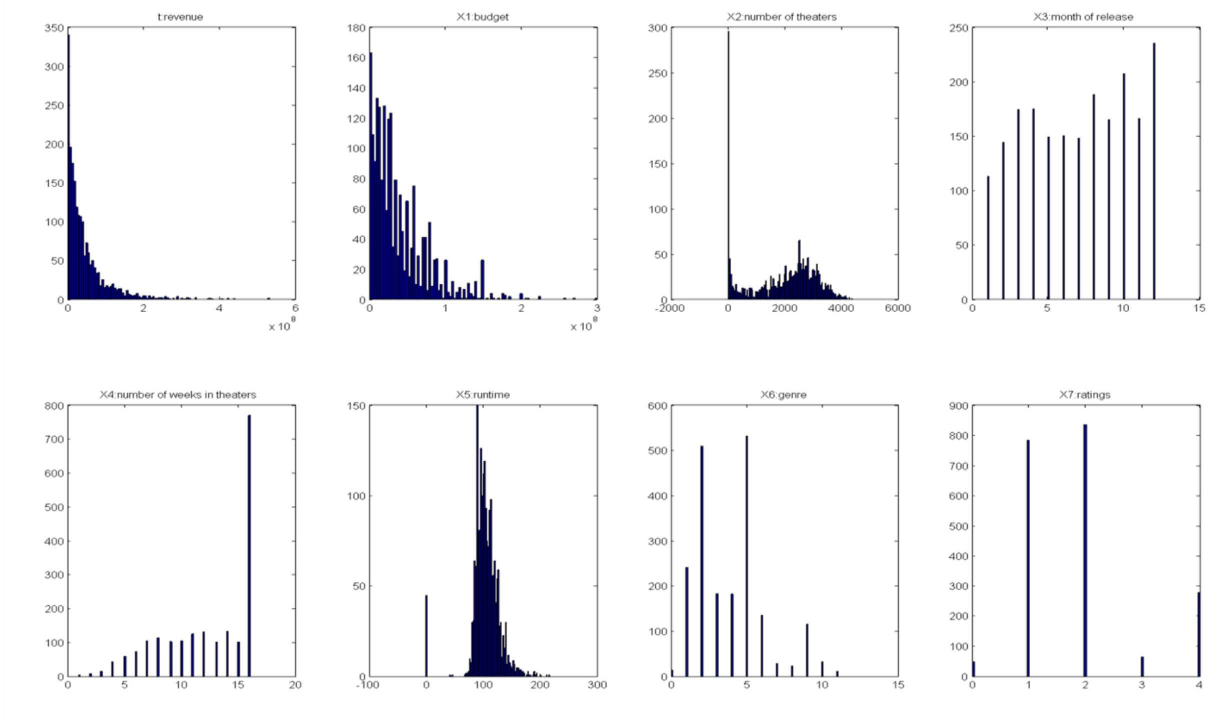


Figure 1. Histogram of target and input variables

The data is recorded in real life, and the target and input variables are the revenue and some properties which can be viewed in figure 1. According to the histogram and information given, several problems in the data set appear.

First, in the information given, X3(month), X6(Genre), X7(Rating) are categorical inputs, which mean the value of the inputs are only labels of categories. The preprocessing of categorical variables is discussed in 2.1. The second problem is, there are several missing information in X5(Runtime). Two method to obtain the missing information is discussed in 2.2. Last, the range of different dimension varies very. The problem caused by this fact and its solution is discussed in 2.3

2.1 Categorical inputs

The numbers in the categorical inputs does not have numerical meaning, which makes this kind of inputs not suitable for calculation directly. Here is a simple example about inputs which do not or do have numerical meaning. X6 is genre. If the first movie is love movie, X6(1) =1 is recorded, because '1'is the label of love movie. If the second movie is historical movie, X6(2) =2 is recorded. 2 is the label of historical movie. However, the value of the label is not calculable, because the conclusion cannot be drawn that the historical movies are twice as profitable as love movies according to the number of the labels. However, if the information is given that the average revenue of a love movies is $100,000 and that of historical movies is $120,000, the average revenues can be compared by their values and the numbers can be take into calculation.

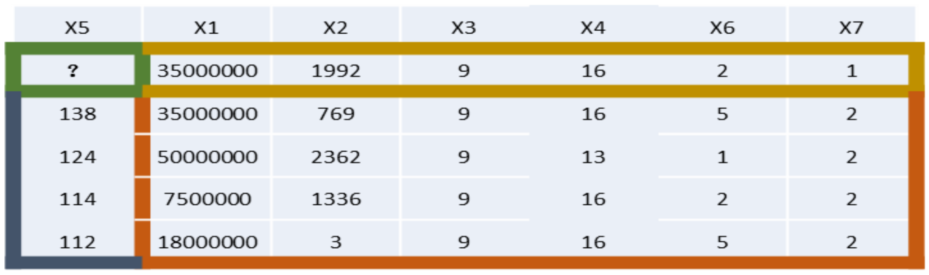
Thus, the basic idea of dealing with categorical inputs is replace the labels with numerical inputs. In this project the each categorical input is replaced by the average revenue of that category. The steps are shown below:

1. Record the number of entries of each category, calculate the sum of revenue of each category.
2. Calculate the average revenue of each category according to the result of the first step.
3. Replace the category labels with the average revenue of that category.

Same operation is done for X3(Month), X6(Genre), X7(Rating).

2.2 Not given inputs

In X5(Runtime), different form other common values which show the length of movies, some of the entries have X5(i) = -1. This number means the information here is not given. So, the problem here is asymmetry: some entries have 7 inputs while some only have 6 inputs given. Solution to this problem is to find out what these values are before going to the learning process.



X5	X1	X2	X3	X4	X6	X7
?	35000000	1992	9	16	2	1
138	35000000	769	9	16	5	2
124	50000000	2362	9	13	1	2
114	7500000	1336	9	16	2	2
112	18000000	3	9	16	5	2

Form 1. The input variables are divided into 4 parts, which shows the known X5s can be acquired by pattern recognition.

A precise way to acquire the unknown value is pattern recognition method. As is shown in form 1, the data can be divided into 4 parts: The entries that have 7 inputs can be divided into known target value X5 and known inputs

X1~X4, X6, X7; these entries can be used as the training data of a learning process. The entries that have 6 inputs can be used as new input values and the unknown X5s are the unknown target variables and can be predicted by a pattern recognition method. In this project, the unknown X5 are predicted by equivalent kernel method with Gauss kernel.

One intuitive way to explain the Equivalent kernel is the following. If the input of the new entry is similar to the input of one of the recorded entries, the predicted target should also be similar to the corresponding recorded target, the more similar the inputs are the more similar the targets are. And in this explanation the similarity of inputs is measured by the kernel function. The steps of the process is shown as below:

1. Calculate $k(x, x')$
2. $\tilde{k}(x, x') = \frac{k(x,x\prime)}{\sum_x k(x,x\prime)}$
3. $y = \boldsymbol{t}^T \times \tilde{\boldsymbol{k}}$

$k(x, x')$ is a column vector which consist of all the kernels calculated from one new input and all the recorded inputs. y, which is the predicted target value, is a convex combination of all the recorded target value, and the weights are proportional to the kernels. The kernel used here is Gauss kernel, which is in the form:

$$k(x, x') = e^{\frac{||x-x'||^2}{-2\sigma^2}}$$

The closer x is to x', the higher the value of the kernel is. Gauss kernel has a bell-like shape which is not to sharp, so it is mild enough to accept noise by consulting a few neighbors for prediction.

In these kernels, σ are adjustable, the best prediction is acquired at p=4 and σ=0.13 by enumeration.

2.3 Normalization

If linear regression is used for learning process, it is safe to conclude that the effect of the wildly diverse ranges of data can be canceled by the weight of each features, if the features are linear. However, even if the features is linear, the badly scaled matrix may cause a calculation issue in software such as MATLAB. Because all the value are non-negative in the dataset after the preprocessing discussed in 2.1 and 2.2, a max normalization is enough to change the size and avoid the calculation issue. The operation is shown below:

```
X( :,m ) /= max( X( :,m ) );
```

For some non-linear features such as Gaussian features or sigmoid features, the shape of the features is based on the range of data. If every dimension has the same range, the design process of features can be simplified.

Normalization is also done to target values if they are used in the learning process. At the end of the prediction, the target values should go back to the original range by multiplying the normalization factor.

3 Learning process

3.1 Simulation Environment

3.1.1 Use of data

In the whole project, everything is done with the data set ‘Tr’, which is the same as real life scenario: the revenue that is to be predicted is known, the only thing to work with is find out the pattern of history records of other movies. However, because of the need of cross validation to choose the best method for prediction, the history record need to be separated into training data, by which the model is built, and test data, by which the performance is quantized. One naïve way of data separation is to divide the data set into halves, using the first half as training data, and the second half as test data. However, if a model is trained in this way, the model can over-fit the test data and depend on the way of division greatly, which means even if the model might give a rather high performance in the second half of the data set, it might not for a more general random case.

There is a very elegant way to solve the question about data division: leave-one-out. In this method the number of iterations is the number of entries, which is denoted N. At each iteration, 1 of the entries is left for test and other N-1 for training. Then each time one prediction is made, and after N iterations N prediction is made. Comparing the N predictions with the recorded N targets, the performance of such a model can be determined. This can avoid the over-fitting problem described above in the half-and-half division method, because the data set is fully used while calculating the performance. However, leave-one-out is slow when the data set is huge. 'Tr' has 2014 entries which means there must be 2014 iterations to acquire one performance, which take too much time.

Another way to generate the data is discovered during the project, which is given a name 'Shuffle-Average'. The steps of this method is shown as blow:

1. Change the order of the entries in 'Tr' randomly.
2. Divide the data into 2 parts, one for training, and one for test.
3. Use the training data for prediction and evaluate the performance by the test data. Record the performance.
4. Calculate the average performance by the records.
5. Repeat step 1~4 for several times until the average performance converge.

The first 3 steps are called shuffle, and the 4th step is average. 5th step is the stop condition of the iteration which is actually not used in the projects. The number of iterations is fixed to 200 for simplicity. Because of the randomness, this method can also avoid the problem in the half-and-half division method. The result converges within dozens of iterations, so this method is faster than leave-one-out method, and this is the method used in the project.

3.1.2 Simulation structure

At the beginning of the simulation, 200 pairs of training and test data is generated from the set 'Tr' by shuffle method talked in subsection 3.1.1. These 200 pairs are saved in a data file for future use in the 200 iterations.

In the simulation, 4 methods are used. Each method is written as a function which calculates the predicted targets by training data and test input data. One method is tested at one time. In each test, the function is called 200 times, and at each iteration one pair of the 200 pairs is used as the input, and the performance is measured in each iteration. After 200 iterations 200 sets of performance are acquired. At the end of the test, the average performance is calculated, as is discussed in subsection 3.1.1.

In the end of the simulation, the results of all the methods are displayed and saved in the disk. By comparing the performance of each method, the characteristics of the method would be known.

3.2 Linear regression

Linear regression is a commonly used method for prediction, the model can be calculated (trained) by the formula below:

$$w = (\boldsymbol{x}^T \times \boldsymbol{x})^{-1} \times \boldsymbol{x}^T \times t$$

And the predictions can be acquired by the formula below:

$$y = \boldsymbol{x}_{new} \times w$$

Before the training process, the preprocessing and normalization is done.

One problem in the result is that several prediction is negative, which is not acceptable, because the worst revenue would be 0 and cannot be lower. So whenever a prediction is below 0, that prediction is fixed to be zero.

3.3 Find the features by RVM

3.3.1 Mechanism of RVM

RVM (Relevance Vector Machine) is a method that is able to decide which features are important and which are not. The steps of this method is shown as below:

Initialization:

$$\beta, \alpha_i = 1;$$

Iteration:

$$\boldsymbol{\Sigma} = (\boldsymbol{A} + \beta \boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{\phi})^{-\mathbf{1}}$$

$$\boldsymbol{m} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{t}$$

$$\gamma_i = 1 - \alpha_i \Sigma_{ii}$$

$$\alpha_i^{new} = \frac{\gamma_i}{m_i^2}$$

$$\beta^{new} = \left(\frac{||\boldsymbol{t} - \boldsymbol{\phi m}||^2}{N - \sum_i \gamma_i}\right)^{-1}$$

In the calculation, the result converge within a few iterations, so this method is fast. α_i is the preciseness of a corresponding feature, if α_i goes very high, that means the mean and variance of the weight of that feature goes to 0. This can be observed from the calculated weight m, the weights of high α_i are very small which disables the corresponding bad features.

3.3.2 Find the features by RVM

For high dimension of the inputs, it is very hard to design a set of multidimensional features directly. Because RVM has the ability to disable bad features, it is a promising solution to find out a good feature set. This nice property provided a huge freedom in the choice if features, because even though there are lot of bad features in the model, the weight of such bad features will be low and result not deteriorated.

Polynomial features are very easy to design in any multi-dimension space. Before finding a good feature set by RVM, a wild feature set can be designed by a polynomial, which is 3rd order polynomial in this project. The operation is done as below:

```
x=[x x(:,1)*0+1];% add a constant term.
M=size(x,2);
for i=1:M-2
 for j=i:M-1
  for k=j:M
   x1=x(:,i).*x(:,j).*x(:,k);
   phix=[phix x1];
  end
 end
end
```

After this operation, *phix* would have all the terms in the 7-D 3rd order polynomial, Which is a 110-D feature space. Of course, there must be some bad features which will affect the result of regression. So RVM is used in this feature space, and the weight *m* of the features can be acquired in the process. It can be observed that some of the feature have a rather small weight, which is less than 0.05 (because of normalization, the weight would be expected as a number that can be compared with 1). Then remove the features which have very small weight and repeat the RVM process, remove the features that have small weights again. Repeat the process again until every feature remains has a relative high weight. Then these remaining features are the good feature set wanted.

3.4 Find the optimal solution by Gradient Descent

After a set of good features is found, the problem can be modeled into an optimization problem as below:

$$\min_{w} P(\phi * w - t),$$

Where $P()$ is the penalty function used, ϕ is the features of input data, t is the target data, and $\phi * w - t$ is the residual of the prediction, which is a linear model of the weight w. There are various solvers available on line to solve this kind of problem, such as the CVX toolbox for MATLAB. However, the tool boxes are not reliable enough in some cases, because they are designed for general purpose. In this project, for example, the CVX tool box fails a lot even in minimizing l2-norm penalty functions. The inability to solve the optimization problem is probably caused by a rather strict stop criteria in the solver. Because this optimization problem does not have any restrictions, gradient descent would be a very simple and robust way to achieve the optimal solution.

3.4.1 Path search: Gradient or Newton

In most cases, Newton method is preferred over gradient descent. The reason is the number of iterations is smaller. Yet there are two problems of this method: 1. the method asks the penalty functions to be strict convex, so that Hessen matrix exists. 2. The method requires calculate the inverse of matrixes. The first problem appears in penalty functions like l1-norm or dead band, because they are not 2nd order differentiable, and the second problem will cause each iteration much longer than that of gradient descent. So in this project, gradient descent is used for path searching. The direction of the path is represent as a normalized vector *dir* in the weight space.

3.4.2 Step search: Bisection

The step is acquire by calculating the minimum in the path. And the minimum is acquired by bisection method. Bisection is an efficient method in 1-D optimization, but is not suitable for multi-dimensional optimization, so it is used along one path but not for the whole optimization problem.

In a general optimization problems, instead of divide the interval into halves, the interval is divided into 3 equal parts. If *up* and *low* are the upper and lower bound of the interval, and *a* and *b* represent the upper and the lower section point, they are calculated in the operation.

```
a=low + ( up-low )/3;
b=up - ( up-low )/3;
```

Then if f(x) have the smallest value at point 'a' among the 4 points (low, a, b, up), then the minimum must not in the region [b,up], which is used as the new interval in the next iteration. Similarly, when f(x) have the smallest value at 'b', the interval can be changed to [low,a]. So, each time 1/3 of the interval is excluded, and finally the iteration would converge.

In this project, the function to optimize is the penalty function $P()$, and the interval is defined in the path. The upper and lower boundary point in the path are defined as *wn+up*dir* and *wn+low*dir*, where *wn* is the current point in the weight space, *up* and *low* are scalars, and *dir* is the normalized direction vector discussed in subsection 3.4.1. The upper and lower section point are defined as *wn+xu*dir* and *wn+xl*dir*, where *xu* and *xl* are scalars, which are calculated in the same way discussed above. In each iteration, the scalars defining the upper or lower bound of the interval might be replaced by the scalars defining the upper or lower section point, in the way discussed above. When the iteration converges, the minimum point in the path in the weight space is reached, and a new iteration of gradient descent can start from this point. When gradient descent converges, the optimal solution for the penalty function in the weight space is reached.

3.4.3 Design of penalty function

Because the optimal is achieved by gradient descent, the optimal solution is achievable whenever the penalty function is unimodal, which means it is not necessarily convex. Many penalty functions are tried and the performance calculated. The performance of these penalty functions would be discussed in section 4.

Norm penalty:

$$P(\boldsymbol{r}) = \left(\sum_{n=1}^{N} |r_n|^k\right)^{\frac{1}{k}}$$

Where k is the order of the norm. For l2-norm, k2, etc.. When k=infinity, linf-norm is the same as $\max_m r_m$. In the project, l1-norm, l2-norm, l4-norm and linf-norm is used.

Log barrier penalty:

$$p_n(r_n) = \begin{cases} -\log(1-|r_n|)^2 \,, |r_n| < 1 \\ \quad 0 \quad\quad\quad\quad\; , |r_n| \geq 1 \end{cases}$$

$$P(\boldsymbol{r}) = \sum_{n=1}^{N} p_n$$

This not really the real log barrier, because the residuals which is beyond the barrier actually do not have penalty. It is impossible to calculate the real log barrier because infinity cannot be compared. (Even though mathematically infinity is greater or equal to infinity, in engineering background, this trivial claim means nothing.) Instead the penalty for larger residual is simply set to zero, which is able to avoid the effect of outliers. This non-convex function is mostly unimodal, so that the convergence would be the optimal solution (or a good solution if not unimodal).

Huber penalty:

$$p_n(r_n) = \begin{cases} |r_n|^2 & , |r_n| < 1 \\ 2 * |r_n| - 1 & , |r_n| \geq 1 \end{cases}$$

$$P(\boldsymbol{r}) = \sum_{n=1}^{N} p_n$$

Dead band:

$$p_n(r_n) = \begin{cases} 0 & , |r_n| < 0.02 \\ |r_n| - 0.02 & , |r_n| \geq 0.02 \end{cases}$$

$$P(\boldsymbol{r}) = \sum_{n=1}^{N} p_n$$

L2-norm with restriction to y:

$$P(\phi * w - t) = l2norm(\phi * w - t) - \sum_{\phi_n * w < 0} \phi * w$$

The additional term in the end represents the effect of predictions that are negative, which is not practical. Adding this term to the penalty function can minimize the number of unpractical predictions.

4 Result and discussion

4.1 Learning result

method	R2
Huber	0.679
log barrier	0.677
RVM to find Basis	0.6714
l2 norm	0.6678
linear regression with feature	0.6648
dead band	0.6523
l2 norm with restriction to y	0.6485
l1 norm	0.6319
pure linear regression	0.5731
l4 norm	0.5508
linf norm	-1.9903

Form 2. Performance ranking measured by R square.

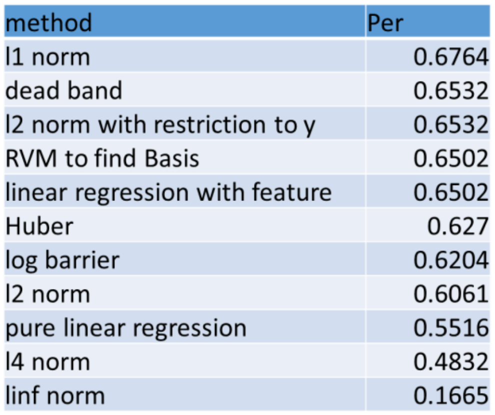
The performance measured by R^2 is shown in form 2.

Need not to say, pure linear regression in section 3.2 has the worst performance, if linf-norm and l4-norm are not took into consideration. This means the feature space is very important since it change a nonlinear data to a more linear data, so that linear model would have better performance. The highest 5 models have very close performance. As is discussed above, maximizing R^2 is the same as minimizing the square error of the prediction. L2-norm or Linear regression (regression is the analytic solutions of l2-norm optimization, so they are the same) is the method directly set least square as the goal of optimization, so it is understandable that linear regression is one of the best methods. Huber is a more robust kind of l2-norm so it has similar(even better) performance to the l2-norm method. Log barrier has a shape which is very similar to l2-norm when the residuals are small considering that the log barrier dose not really has barrier (there is no penalty for outliers) the behaver of this penalty function is very similar to Huber penalty function. Reasonably, RVM as a similar performance with the regression method. After all the features are found in this method, and RVM is also a Gaussian kind of regression, so the behavior would be similar to regression.

If the restriction is added to l2-norm method, R^2 become worse, because the goal of optimization is not that direct enough as least square. However this gives it some better performance in other perspective.

L1-norm is not as directly as l2-norm. L2-norm minimize the sum of the absolute value of residuals. So that a great amount residuals should condensed in a very narrow region around zero. Yet it suffers from a lot of outliers, so in the case of R^2, it is not as high as the least square related methods. Dead band is a more robust l1-norm method, since that it widen the range where the small residual locates, so that there are less outliers. As the result the dead band method have a higher R^2 than l1-norm.

Tragedies happen in the case of l4-norm and linf-norm. Which has a very sharp shape at the tails of the penalty functions. These two functions are actually the closest to real log barrier function. The goal of this kind of function is to reduce the effect of the large residuals, so square error is not consider in the optimization. No wonder why R^2 of these functions are low.



method	Per
l1 norm	0.6764
dead band	0.6532
l2 norm with restriction to y	0.6532
RVM to find Basis	0.6502
linear regression with feature	0.6502
Huber	0.627
log barrier	0.6204
l2 norm	0.6061
pure linear regression	0.5516
l4 norm	0.4832
linf norm	0.1665

Form 3. Performance ranking measured by proportion of accurate predictions.

In form 3, the performance is expressed in the proportion of accurate predictions.

As is mentioned before, l1-norm would condense the residuals in a small area around zero, and this victory is shown in highest proportion of accurate predictions. For the same reason, dead band method also has very high proportion of accurate predictions.

Another method which has very good accuracy is the l2-norm with restriction. The restriction states that the penalty should be added to ridiculous predictions, which increase the proportion of accurate predictions.

Needless to say, l4-norm and linf –norm are tragedies in terms of accuracy, again.

method	max
l1 norm	2.2677
pure linear regression	2.3421
dead band	2.5193
l2 norm	2.5809
l2 norm with restriction to y	2.5876
Huber	2.8504
log barrier	2.8689
l4 norm	2.8882
RVM to find Basis	2.9635
linear regression with feature	3.2119
linf norm	3.5469

Form 4. Performance ranking measured by worst case.

Form shows the performance measured by worst case. This metric is not carefully designed so it does not reveal much information. However on of the most important thing is discovered here: l4-norm and linf-norm are still tragedies when measured by worst case. Because the method is designed to minimize the worst case, they should give better worst case, but they actually give even worse worst case. Thus the reason for this phenomenon is important. As can be seen in the penalty function, if the penalty function is only related to the maximum residual, every iteration is only driven by very few dimensions in a huge many entries of training data. This problem is called sparsity, which means the huge data set is not sufficiently used by the model.

4.2 Final result

At the end of the project, the real targets are given. It is important to know whether the models have the similar behavior with the real data as with the training data. So the last job of the project is to use the whole 'Tr' as training data, and the 'New' as test inputs to predict the targets by all the methods. If there is method is robust enough to overcome over fitting the performance in the new (real) data would be the same as the performance in the recorded data.

	Training			Real		
method	R2	Per	max	R2	Per	max
pure linear regression	0.5731	0.5516	2.3421	0.595	0.546	2.9439
RVM to find Basis	0.6714	0.6502	2.9635	0.6668	0.604	3.6428
linear regression with feature	0.6648	0.6502	3.2119	0.6463	0.601	3.5941
l2 norm	0.6678	0.6061	2.5809	0.654	0.59	2.9122
l1 norm	0.6319	0.6764	2.2677	0.6252	0.629	2.5984
l4 norm	0.5508	0.4832	2.8882	0.5628	0.489	3.0795
linf norm	-1.9903	0.1665	3.5469	-1.0131	0.184	3.4776
log barrier	0.677	0.6204	2.8689	0.6311	0.585	3.2924
Huber	0.679	0.627	2.8504	0.6417	0.601	3.2373
dead band	0.6523	0.6532	2.5193	0.6591	0.628	2.7686
l2 norm with restriction to y	0.6485	0.6532	2.5876	0.643	0.617	2.9284

Form 5. Performance cooperation: train data with real data .

By measuring the differences of the two set of performances, it can be found that the difference is very small, which means the models built in this project is reliable in real life.