Extend Statistical Algorithms to Large Scale Data Analysis

Rui Wang

301253067

Abstract. This report introduce the Alternated Direction Multiplier Method[1], an effcient parallel opitmization framework. This framework has been widely applied to large scale statistical learning, achieves robust good enough results for many learning models. For this report, I organize this topic as three parts, 1. The motivation of ADMM; 2. The main structure of ADMM, Dual Decomposition and Augmented Lagrangian 3. Application to distributed statistical learning, mostly L1 norm related. To Follow this procedure, I still utilized two ADMM implementations, to illustrate its fast speed and good enough residual on large multidimensional dataset with 700MB regression and classfication data.

Keywords: ADMM Statistical Learning Optimization

1 The Motivation

The large data analysis has long history, from 1960s approximately, where the model is to solve large scale of linear programming, like danzig wolfe decomposition. Nowadays, the mordern methods of statistical learning are more diversified, like figure 1. Normally, for most statistical learning method, we could formulate the objective as:

$$\min_{x} \quad loss(x) + regu(x)$$

where the objective is composed of loss function and regularization norm. The loss function measures how well the model fitting on trainning data, the regularization term measures the complexity of the model. The loss function is mostly convex and continuous, while regularization is convex, non-differentiable when use L1 norm.

If the data is very large, with many instances and many features, it's very hard to solve it efficiently through traditional optimization methods. The most straigt way is to divide the data to smaller blocks, and distribute them to processing agents. There are two ways to divide the data, 1. Divide data according to instances; 2. Divide the data according to features. The figure 2

However, this decomposition style is normally embedded to primal dual algorithm. Since the distributed gradients caculated in each agents is calcualated independently, we need to collect these gradients together, to update the dual variable, called Gather Process. So the Convergence, the most important criterion in optimization algorithms, could be guranteed to a fast style.

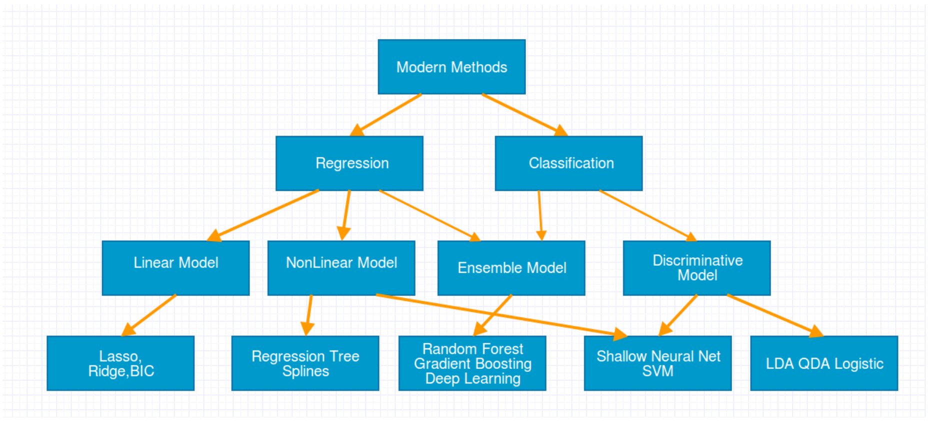
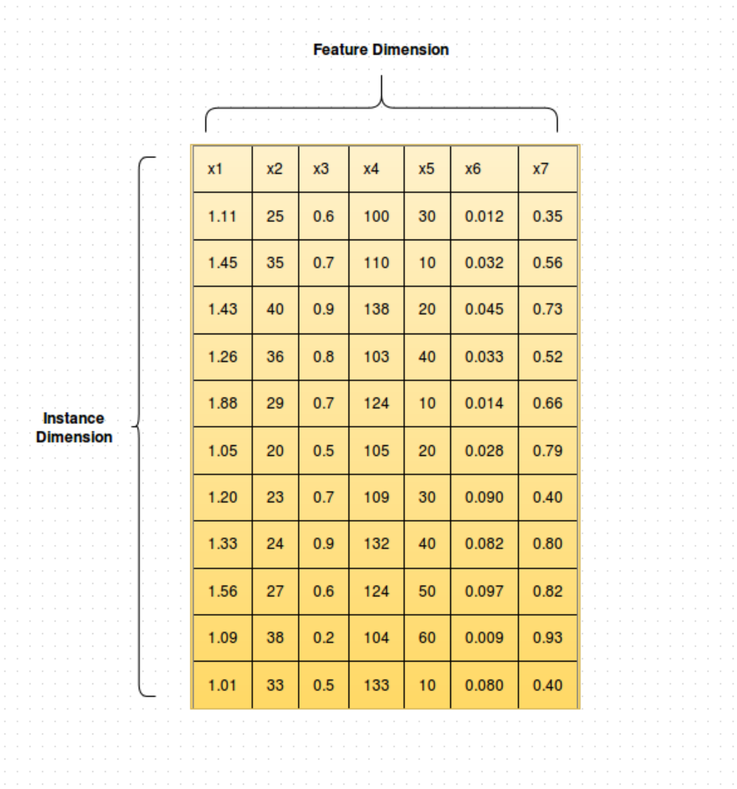


Fig. 1: A roughly sketch of statistical models

2 Decomposition and Convergence

In this section, I introduce two components of ADMM, dual decomposition and Augemented Lagrangian Multiplier. The Dual decomposition guarantees parallel optimization, and Augmented Lagrangian Multiplier guarantees fast convergence in primal updates. ADMM is a nice combinations of these two methods.


Feature Dimension
Instance Dimension

x1	x2	x3	x4	x5	x6	x7
1.11	25	0.6	100	30	0.012	0.35
1.45	35	0.7	110	10	0.032	0.56
1.43	40	0.9	138	20	0.045	0.73
1.26	36	0.8	103	40	0.033	0.52
1.88	29	0.7	124	10	0.014	0.66
1.05	20	0.5	105	20	0.028	0.79
1.20	23	0.7	109	30	0.090	0.40
1.33	24	0.9	132	40	0.082	0.80
1.56	27	0.6	124	50	0.097	0.82
1.09	38	0.2	104	60	0.009	0.93
1.01	33	0.5	133	10	0.080	0.40

Fig. 2: The Data Orgnization

2.1 Dual Decomposition

Before we introduce Dual Decomposition, we first look at the Dual Ascent, the original formulation of Dual Decomposition. Consider the equality constrained problem

$$minimize \quad f(x) \tag{1}$$
$$subject \quad to \quad Ax = b \tag{2}$$

The Lagrangian of problem is

$$L(x, y) = f(x) + y^T(Ax - b)$$

and the dual function is

$$g(y) = \inf_x L(x, y) = -f^*(-A^T y) - b^T y$$

The Dual Ascent is to Maximize $g(y)$ through primal and dual upate

$$x^{k+1} := arg \min_x L(x, y^k) \tag{3}$$
$$y^{k+1} := y^k + \alpha^k(Ax^{k+1} - b) \tag{4}$$

The Dual Decomposition is to assume the objective f(x) could be decomposed as

$$f(x) = \sum_{i=1}^{N} f_i(x_i)$$

Here the f is decomposed as bounded **functionals** f_i, data x is decomposed to **subvectors** $x_i \in R^{n_i}$, while the lagrangian could also be decomposed as

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} (f_i(x_i) + y^T A_i x_i - (1/N)y^T b)$$

The new parallel update in primal x update and the dual update to gather subgradients is like:

$$x_i^{k+1} := arg \min_{x_i} L_i(x_i, y^k) \tag{5}$$
$$y^{k+1} := y^k + \alpha^k(Ax^{k+1} - b) \tag{6}$$

Here the dual decomposition could divide original model to several sub-models, then the parallel optimization on different models could be parallelized. The Dual update is a collection operation, which collect all the subgradients in primal update. The procedure is illustrated at figure 3. At this stage, we are quite sure that the broad and gather process is a stable procedure which could always improve the model at each iteration. But, it needs strong assumptions on the objectives, also, the variables of each model imposed on have to be decoupled, to remove the interactions of each model.

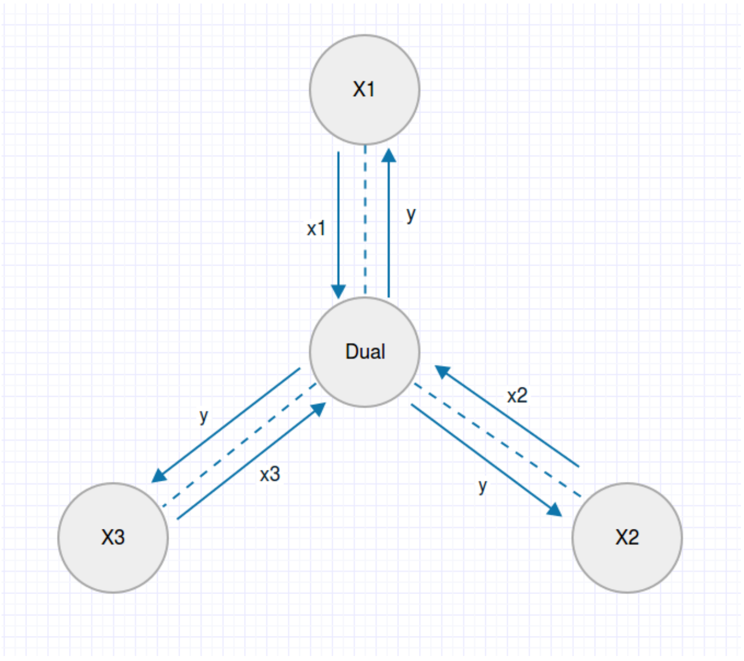


Fig. 3: The BroadCast and Gather of Dual Decomposition

2.2 Augmented Lagrangian Multiplier

Although the dual step in dual decomposition is trying to collect the subgradients, it is still not fast enough in Primal X update, since the original lagrangian is not fast enough to make each subgradient converge fast. So here the augmented lagrangian, which scaled the

penalty of contraints to second order, shows its power.
The augmented lagrangian is formulated as

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)\|Ax - b\|_2^2$$

The penalty parameter ρ is a key prameter which is set as the step size of dual variable update. This method makes the opitmization converges under more general conditions that f is not strictly convex.

The disadvantage of augmented lagrangian multiplier is that it introduces L2 norm, which makes the decomposition not available, since there would be interactions between different sub-functions.

2.3 Alterhnating Direction Method of Multipliers

The ADMM combines the dual decomposition and augmented lagrangian, formulate the original objective as two blocks of variables. The formulation is like:

$$\begin{aligned} minimize \quad & f(x) + g(z) \\ subject \quad to \quad & Ax + Bz = c \end{aligned}$$

Where the X and Z are two blocks of variables. The iterations are formulated as

$$x^{k+1} := arg \min_x L_\rho(x, z^k, y^k) \tag{7}$$

$$z^{k+1} := arg \min_z L_\rho(x^{k+1}, z, y^k) \tag{8}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \tag{9}$$

The Scaled form of ADMM is more elegant, utilize scaled dual varialbe $\mu = (1/\rho)y$,

$$x^{k+1} := arg \min_x (f(x) + (\rho/2)\|Ax + Bz^k - c + \mu^k\|_2^2) \tag{10}$$

$$z^{k+1} := arg \min_z (g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + \mu^k\|_2^2) \tag{11}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \tag{12}$$

The whole procedure of ADMM is shown in figure 4, the Z update is a block to average the collections of primal x updates, the dual ascent is the final step to guarantee to consistent covergence.

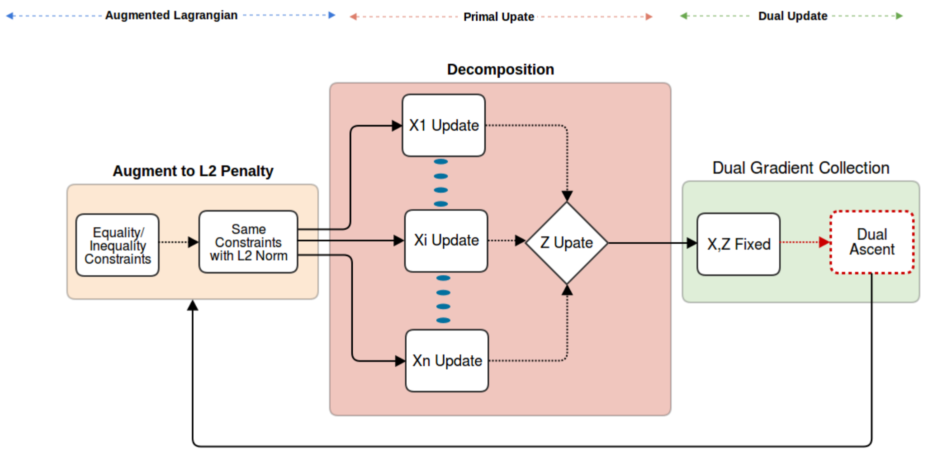


Fig. 4: The ADMM procedure

2.4 Why divide to two blocks? Not 3,4.....

The two blocks variable formulation in this work is proved to converge under two general assumptions,

- The function f is closed, proper, convex, so primal upate of x, z is solvable,
- the lagrangian L_0 has a saddle point, so dual and primal is equal

Since all is approximation in computational methods, the objective here is to get a robust and good enough result. The ADMM has been shown that if it's extended to three blocks, it would give a counter example[2] that this iteartion style is not working.

2.5 Why Dual?

- The dual update is very straight forward, since it is always a concave problem, no matter whehter the original primal is convex.
- The dual is an easier problem in most cases, if the constraints are linear and limited, then less dimension is requried
- The Dual is an lower bound of the primal solutions, there maybe gap between dual optimal and primal optimal, but it may be ignored in discontinous or discrete situation

3 Distributed Statistic learning

Turn back to the dual block representation or Dual Functional Representation,

$$minimize \quad l(Ax-b)+r(z)$$

Here the x is one block, z is another block, we can divide the x block only, or divide both blocks in consistent manner. These two methods corresponding to divide the data instances or divide data features, like figure 5 show

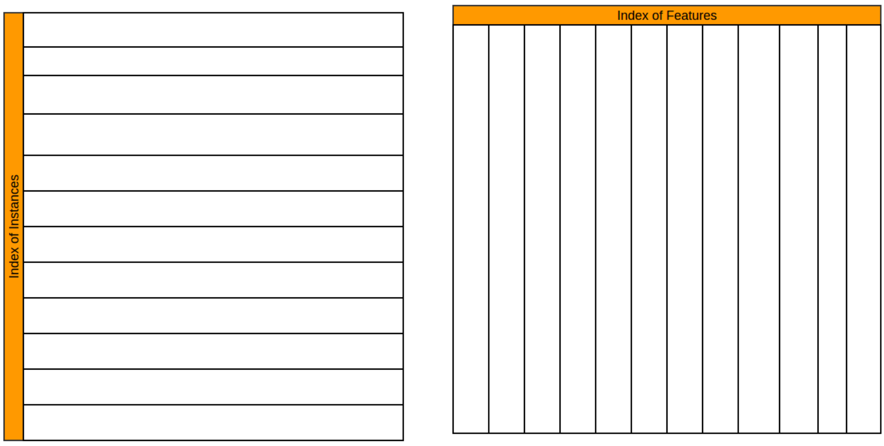


Fig. 5: Two ways of divide data

3.1 Divide across instances

The Splitting of instances is very straight forward, and the most critical part is that second iteration is to collect and average the subgradients from first iteration. The formulation of this case could be

$$minimize \quad \sum_{i=1}^{N} l_i(Ax_i-b_i)+r(z) \tag{13}$$

$$subject \quad to \quad x_i-z=0, i=1\cdots N \tag{14}$$

Iterations would be like:

$$x_i^{k+1} := arg \min_{x_i}(l_i(A_i x_i - b_i) + (\rho/2)\|x_i - z^k + \mu_i^k\|_2^2) \tag{15}$$

$$z^{k+1} := arg \min_{z}(r(z) + (N\rho/2)\|z - \bar{x}^{k+1} - \bar{\mu}^k\|_2^2 \tag{16}$$

$$\mu_i^{k+1} := \mu_i^k + x_i^{k+1} - z^{k+1} \tag{17}$$

3.2 Divide across features

The splitting across features requires to seperate the loss term and regularization term both, in contrast to splitting across instances that only requires seprate the loss term. Here the Z update is still formulated as a average over the parallel computation of x update.

$$minimize \quad \sum_{i=1}^{N} l_i(Ax_i - b_i) + \sum_{i=1}^{N} r(z_i) \tag{18}$$

$$subject \quad to \quad x_i - z_i = 0, i = 1 \cdots N \tag{19}$$

Iterations would be like:

$$x_i^{k+1} := arg \min_{x_i}(r_i(x_i) + (\rho/2)\|A_i x_i - A_i x_i^k - \bar{z}^k + \overline{Ax}^k \mu^k\|_2^2) \tag{20}$$

$$\bar{z}^{k+1} := arg \min_{z}(l(N\bar{z} - b) + (N\rho/2)\|z - \overline{Ax}^{k+1} - \mu^k\|_2^2 \tag{21}$$

$$\mu^{k+1} := \mu^k + \overline{Ax}^{k+1} - \bar{z}^{k+1} \tag{22}$$

Here I notice that the Z update step is always defined as an average step, to average to x updates in the first step, while the dual varialbe μ is set a single varialbe update through mean function.

4 Decomposition on Functionals, Distributed learning on Data

Now we know that ADMM is a very straight forward way to divide the model to 2 blocks of functions. And the distributed learning of last section is not about dividing the model like decomposition, it is designed to divide the data. So Here I give a Comparison table of Dual Decomposition and Distributed learning as table 1. The distributed learning is like a parellel scheme embedded into the primal dual updates.

	Dual Decomposition	Distributed Learning
Objective	Decompose functions	Decompose Data
Potential Coupling	Yes	No
Dual form	necessary	not necessary

Table 1: The Comparison of Decomposition and Distributed Learning

5 Experiment

Here I utilized 3 datasets to test the ADMM model, phoneme data, Million Song data(UCI), Slice Data(UCI). The Table 2 descript the attributes of each dataset. I utilized the POGS opensourced ADMM R implementation from stanford, to do test of this optimization. Three models are built on this package, lasso, logistic, svm, to solve each dataset, like table 3

dataset	Instances	attributes	task
phoneme	4509	256	classification
slice	53500	386	regression
millison song	515345	90	regression

Table 2: The datasets details

dataset	models
phoneme	logistic, svm
slice	Lasso
millison song	Lasso

Table 3: The Models built

Since the R package of POGS doesn't provide predict function, and the main purpose here is to test this parallel optimization's performance on reducing the residuals, so here I would give three plots of primal residuals and dual residuals through iterations. All the details are provided in the appendix, with the SMSE of other implementations like glmnet and e1071. At the computational perspective, the datasets haven't been splitted to train, validation, test. We utilize all the data instances and attributes, let the method to run on the data, to test the final residuals, it is shown in table 4. The error metric of primal residual is defined by the author, shown in appendix.

dataset	model	primal residual	MSE
phoneme data	logistic	9.58e-04	0.09013
phoneme data	svm	0	0
song data	lasso	2.71e-03	96.37442
slice data	lasso	9.95e-04	126.978

Table 4: The Models Performance

6 Conclusion

This report introduce the ADMM framework, with its applications to extend the statistical learning algorithms to large scale data analysis. The experiments shows that the convergence of ADMM is good, and the final result is also good enough. The importance of ADMM is to summarize the elegant optimization methods into a new straightforward way, and what's more important is that it illustrates the difficulty at large statistical learning. If we just try to utilize more computers to handle the explosive data, it's far away from get problems done.

This report is trying to provide an explicity overview of this framework, here I also want to point out, ADMM is not good enough to handle some other problems like large linear control system, also, the two-block alternate direction update seems not possible to extend to 3 or 4 blocks[2]. Some researchers are trying to apply statistical methods to make progress in large scale optimization, I think it may be the only way to do it.

References

1. Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
2. Ruoyu Sun, Zhi-Quan Luo, and Yinyu Ye. On the expected convergence of randomly permuted admm. *arXiv preprint arXiv:1503.06387*, 2015.

Appendix

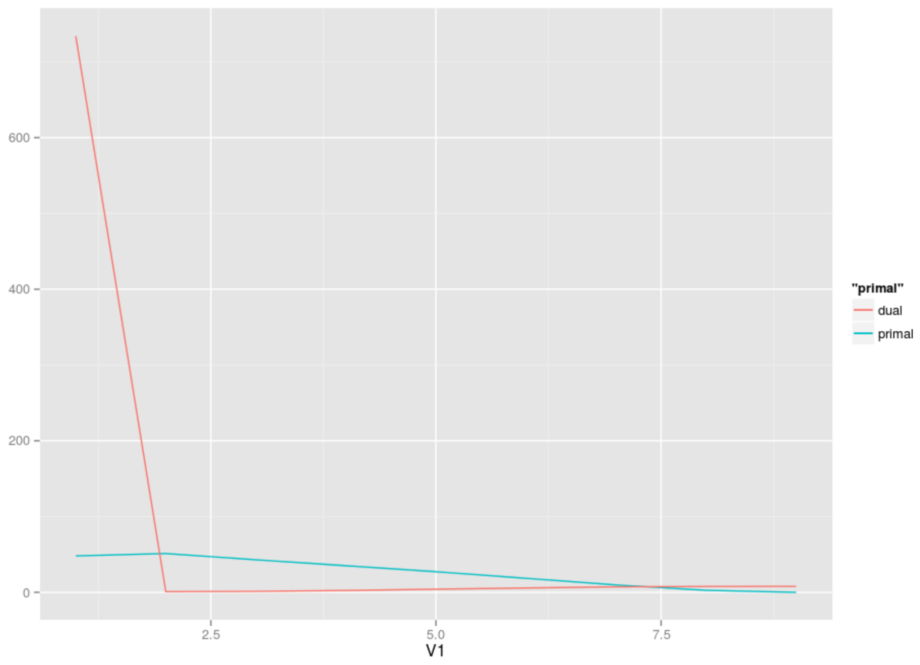


Fig. 6: The Primal and Dual residuals on phoneme

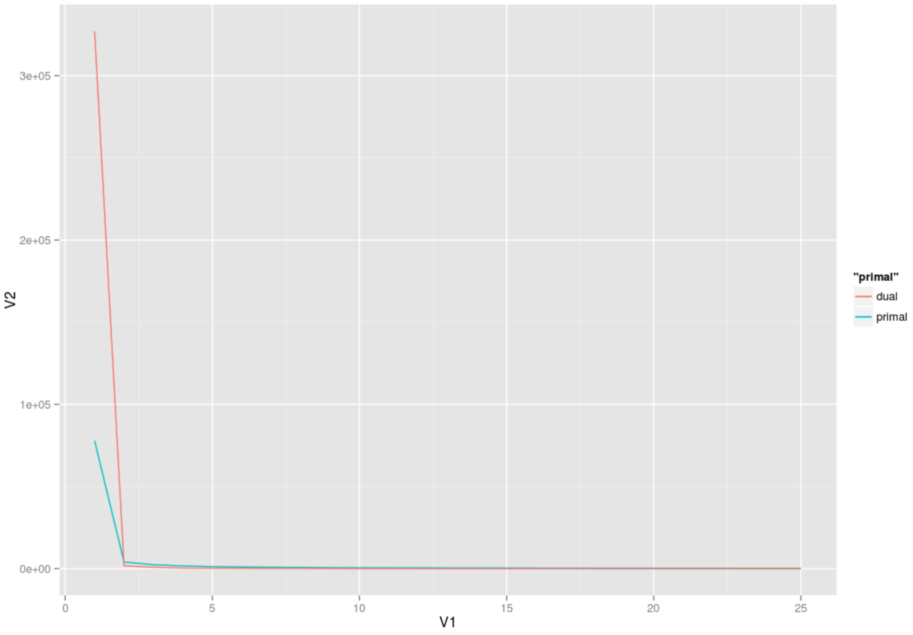


Fig. 7: The Primal and Dual residuals on million song

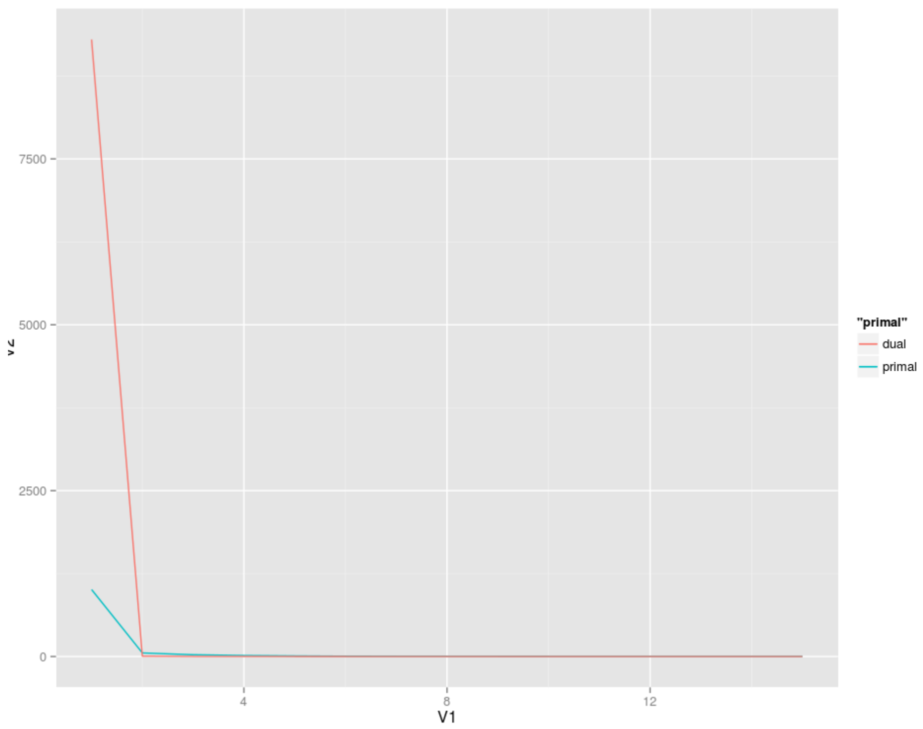
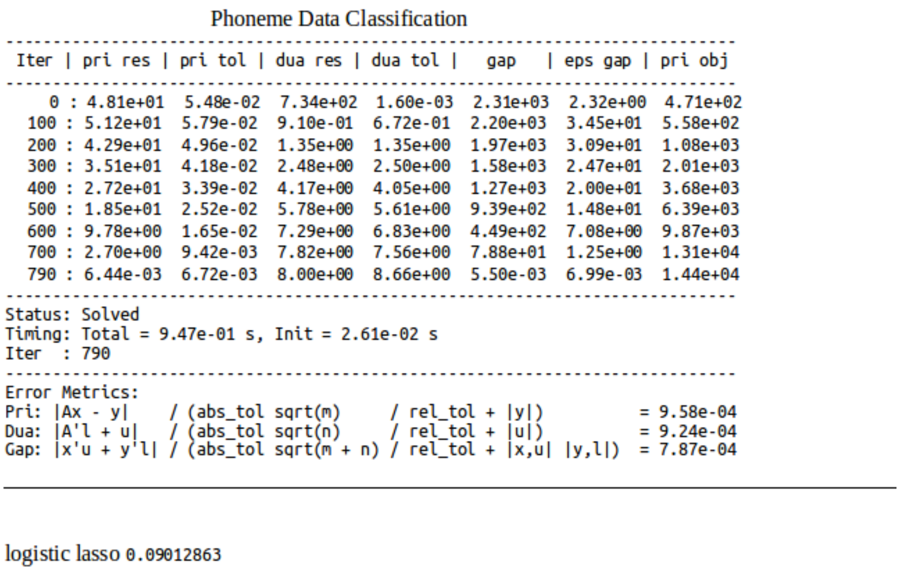


Fig. 8: The Primal and Dual residuals on slice data

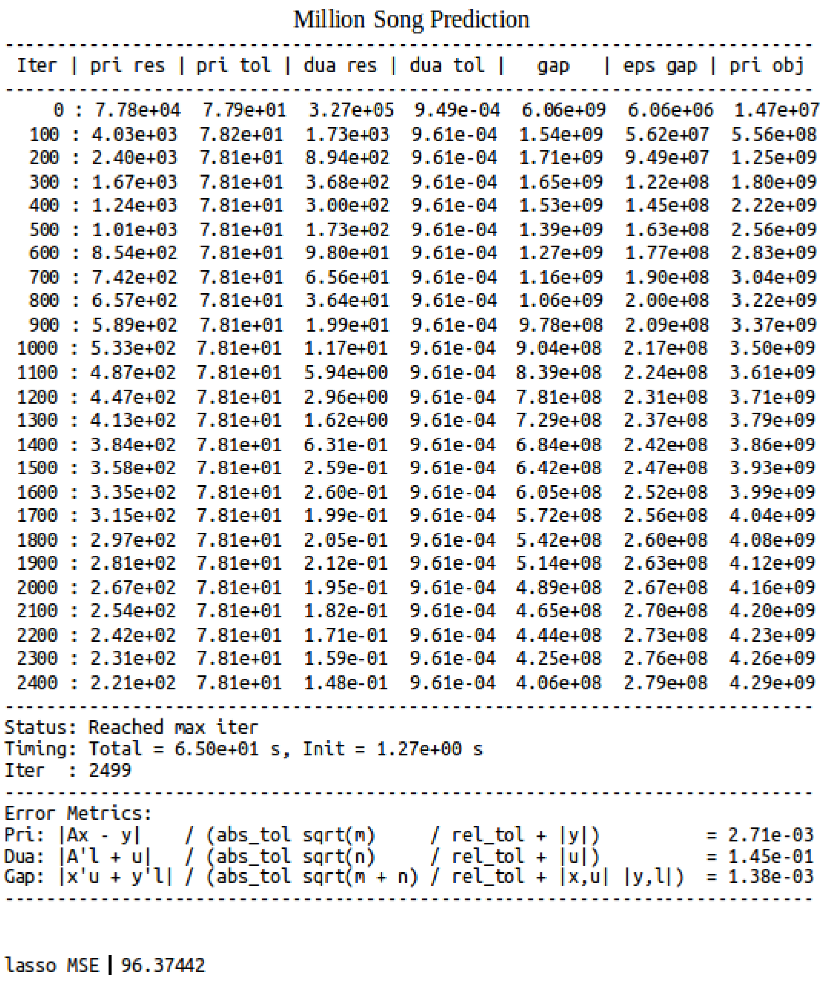


Phoneme Data Classification

```
---------------------------------------------------------------------------
 Iter | pri res | pri tol | dua res | dua tol |   gap   | eps gap | pri obj
---------------------------------------------------------------------------
    0 : 4.81e+01  5.48e-02  7.34e+02  1.60e-03  2.31e+03  2.32e+00  4.71e+02
  100 : 5.12e+01  5.79e-02  9.10e-01  6.72e-01  2.20e+03  3.45e+01  5.58e+02
  200 : 4.29e+01  4.96e-02  1.35e+00  1.35e+00  1.97e+03  3.09e+01  1.08e+03
  300 : 3.51e+01  4.18e-02  2.48e+00  2.50e+00  1.58e+03  2.47e+01  2.01e+03
  400 : 2.72e+01  3.39e-02  4.17e+00  4.05e+00  1.27e+03  2.00e+01  3.68e+03
  500 : 1.85e+01  2.52e-02  5.78e+00  5.61e+00  9.39e+02  1.48e+01  6.39e+03
  600 : 9.78e+00  1.65e-02  7.29e+00  6.83e+00  4.49e+02  7.08e+00  9.87e+03
  700 : 2.70e+00  9.42e-03  7.82e+00  7.56e+00  7.88e+01  1.25e+00  1.31e+04
  790 : 6.44e-03  6.72e-03  8.00e+00  8.66e+00  5.50e-03  6.99e-03  1.44e+04
---------------------------------------------------------------------------
Status: Solved
Timing: Total = 9.47e-01 s, Init = 2.61e-02 s
Iter  : 790
---------------------------------------------------------------------------
Error Metrics:
Pri: |Ax - y|    / (abs_tol sqrt(m)     / rel_tol + |y|)          = 9.58e-04
Dua: |A'l + u|   / (abs_tol sqrt(n)     / rel_tol + |u|)          = 9.24e-04
Gap: |x'u + y'l| / (abs_tol sqrt(m + n) / rel_tol + |x,u| |y,l|)  = 7.87e-04
```

logistic lasso `0.09012863`

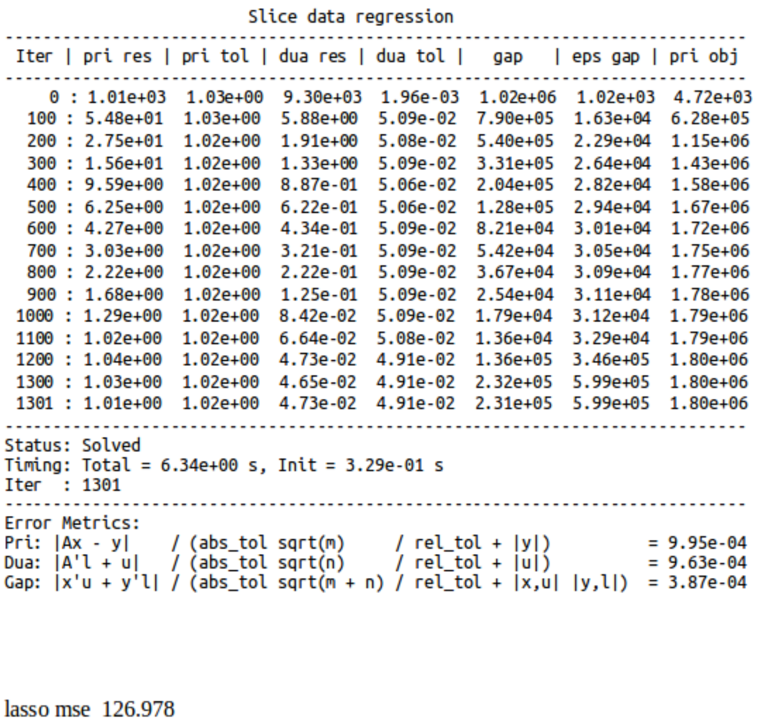
Fig. 9: The Primal and Dual residuals on phoneme



```
                    Million Song Prediction
--------------------------------------------------------------------------
 Iter  | pri res | pri tol | dua res | dua tol |   gap    | eps gap | pri obj
--------------------------------------------------------------------------
    0 : 7.78e+04  7.79e+01  3.27e+05  9.49e-04  6.06e+09  6.06e+06  1.47e+07
  100 : 4.03e+03  7.82e+01  1.73e+03  9.61e-04  1.54e+09  5.62e+07  5.56e+08
  200 : 2.40e+03  7.81e+01  8.94e+02  9.61e-04  1.71e+09  9.49e+07  1.25e+09
  300 : 1.67e+03  7.81e+01  3.68e+02  9.61e-04  1.65e+09  1.22e+08  1.80e+09
  400 : 1.24e+03  7.81e+01  3.00e+02  9.61e-04  1.53e+09  1.45e+08  2.22e+09
  500 : 1.01e+03  7.81e+01  1.73e+02  9.61e-04  1.39e+09  1.63e+08  2.56e+09
  600 : 8.54e+02  7.81e+01  9.80e+01  9.61e-04  1.27e+09  1.77e+08  2.83e+09
  700 : 7.42e+02  7.81e+01  6.56e+01  9.61e-04  1.16e+09  1.90e+08  3.04e+09
  800 : 6.57e+02  7.81e+01  3.64e+01  9.61e-04  1.06e+09  2.00e+08  3.22e+09
  900 : 5.89e+02  7.81e+01  1.99e+01  9.61e-04  9.78e+08  2.09e+08  3.37e+09
 1000 : 5.33e+02  7.81e+01  1.17e+01  9.61e-04  9.04e+08  2.17e+08  3.50e+09
 1100 : 4.87e+02  7.81e+01  5.94e+00  9.61e-04  8.39e+08  2.24e+08  3.61e+09
 1200 : 4.47e+02  7.81e+01  2.96e+00  9.61e-04  7.81e+08  2.31e+08  3.71e+09
 1300 : 4.13e+02  7.81e+01  1.62e+00  9.61e-04  7.29e+08  2.37e+08  3.79e+09
 1400 : 3.84e+02  7.81e+01  6.31e-01  9.61e-04  6.84e+08  2.42e+08  3.86e+09
 1500 : 3.58e+02  7.81e+01  2.59e-01  9.61e-04  6.42e+08  2.47e+08  3.93e+09
 1600 : 3.35e+02  7.81e+01  2.60e-01  9.61e-04  6.05e+08  2.52e+08  3.99e+09
 1700 : 3.15e+02  7.81e+01  1.99e-01  9.61e-04  5.72e+08  2.56e+08  4.04e+09
 1800 : 2.97e+02  7.81e+01  2.05e-01  9.61e-04  5.42e+08  2.60e+08  4.08e+09
 1900 : 2.81e+02  7.81e+01  2.12e-01  9.61e-04  5.14e+08  2.63e+08  4.12e+09
 2000 : 2.67e+02  7.81e+01  1.95e-01  9.61e-04  4.89e+08  2.67e+08  4.16e+09
 2100 : 2.54e+02  7.81e+01  1.82e-01  9.61e-04  4.65e+08  2.70e+08  4.20e+09
 2200 : 2.42e+02  7.81e+01  1.71e-01  9.61e-04  4.44e+08  2.73e+08  4.23e+09
 2300 : 2.31e+02  7.81e+01  1.59e-01  9.61e-04  4.25e+08  2.76e+08  4.26e+09
 2400 : 2.21e+02  7.81e+01  1.48e-01  9.61e-04  4.06e+08  2.79e+08  4.29e+09
--------------------------------------------------------------------------
Status: Reached max iter
Timing: Total = 6.50e+01 s, Init = 1.27e+00 s
Iter  : 2499
--------------------------------------------------------------------------
Error Metrics:
Pri: |Ax - y|    / (abs_tol sqrt(m)     / rel_tol + |y|)         = 2.71e-03
Dua: |A'l + u|   / (abs_tol sqrt(n)     / rel_tol + |u|)         = 1.45e-01
Gap: |x'u + y'l| / (abs_tol sqrt(m + n) / rel_tol + |x,u| |y,l|) = 1.38e-03
--------------------------------------------------------------------------


lasso MSE | 96.37442
```

Fig. 10: The Primal and Dual residuals on million song



```
                      Slice data regression
-------------------------------------------------------------------
 Iter | pri res | pri tol | dua res | dua tol |   gap   | eps gap | pri obj
-------------------------------------------------------------------
    0 : 1.01e+03  1.03e+00  9.30e+03  1.96e-03  1.02e+06  1.02e+03  4.72e+03
  100 : 5.48e+01  1.03e+00  5.88e+00  5.09e-02  7.90e+05  1.63e+04  6.28e+05
  200 : 2.75e+01  1.02e+00  1.91e+00  5.08e-02  5.40e+05  2.29e+04  1.15e+06
  300 : 1.56e+01  1.02e+00  1.33e+00  5.09e-02  3.31e+05  2.64e+04  1.43e+06
  400 : 9.59e+00  1.02e+00  8.87e-01  5.06e-02  2.04e+05  2.82e+04  1.58e+06
  500 : 6.25e+00  1.02e+00  6.22e-01  5.06e-02  1.28e+05  2.94e+04  1.67e+06
  600 : 4.27e+00  1.02e+00  4.34e-01  5.09e-02  8.21e+04  3.01e+04  1.72e+06
  700 : 3.03e+00  1.02e+00  3.21e-01  5.09e-02  5.42e+04  3.05e+04  1.75e+06
  800 : 2.22e+00  1.02e+00  2.22e-01  5.09e-02  3.67e+04  3.09e+04  1.77e+06
  900 : 1.68e+00  1.02e+00  1.25e-01  5.09e-02  2.54e+04  3.11e+04  1.78e+06
 1000 : 1.29e+00  1.02e+00  8.42e-02  5.09e-02  1.79e+04  3.12e+04  1.79e+06
 1100 : 1.02e+00  1.02e+00  6.64e-02  5.08e-02  1.36e+04  3.29e+04  1.79e+06
 1200 : 1.04e+00  1.02e+00  4.73e-02  4.91e-02  1.36e+05  3.46e+05  1.80e+06
 1300 : 1.03e+00  1.02e+00  4.65e-02  4.91e-02  2.32e+05  5.99e+05  1.80e+06
 1301 : 1.01e+00  1.02e+00  4.73e-02  4.91e-02  2.31e+05  5.99e+05  1.80e+06
-------------------------------------------------------------------
Status: Solved
Timing: Total = 6.34e+00 s, Init = 3.29e-01 s
Iter  : 1301
-------------------------------------------------------------------
Error Metrics:
Pri: |Ax - y|    / (abs_tol sqrt(m)     / rel_tol + |y|)          = 9.95e-04
Dua: |A'l + u|   / (abs_tol sqrt(n)     / rel_tol + |u|)          = 9.63e-04
Gap: |x'u + y'l| / (abs_tol sqrt(m + n) / rel_tol + |x,u| |y,l|)  = 3.87e-04



lasso mse  126.978
```

Fig. 11: The Primal and Dual residuals on slice data