

# The LPASSO method for regression regularization

Anshul Nigam, Varun Aggarwal  
{nigam, varun\_ag}@mit.edu

December 7, 2005

## 1 Introduction

Linear models are often built to understand how a set of input data affects output data and predict its value. A linear regression model assumes that that regression function  $E(Y|X)$  is linear in the inputs  $X_1, \dots, X_p$ . The output is represented as a weighted sum of the input variables (or features), where the weights are called the regression coefficients. The regressions coefficients are ascertained by minimizing the mean square error between the output and the predicted output for a given data. This technique to find the regression coefficients is called Ordinary least Squares (OLS).

OLS is regularized to make these linear models more accurate. In [7] and [3], a detailed treatment of the subject of regularization of OLS is given, whose salient points are discussed here. OLS regression is regularized for the following reasons:

1. To ensure the existence of a solution, even with highly correlated features
2. To improve prediction accuracy. OLS has lower bias, but higher variance. By constraining the size of the regression coefficients, the variance is reduced. An increase in bias and a reduction in variance may give

better prediction accuracy; and

3. To improve interpretability, by determining which features contribute most to the output.

The most popular methods to regularize regression currently are ridge regression, LASSO (Least Absolute Shrinkage and Selection Operator) and subset selection. Ridge regression offers an analytical solution and a low-variance estimate, however it does not set any coefficient to zero, which results in low interpretability. Subset selection gives high interpretability but has a high variance because of its discrete nature. The LASSO technique[7] offers a balance between the two approaches. It pushes weights linearly towards zero and may also set some weights exactly to zero. This ensures good interpretability (or feature selection). Since it uses continuous constraints, the variance is not increased as in case of subset selection.

These three methods can be defined by the form:

$$(\beta^0, \beta^j) = \arg \min_{\beta^0, \beta^j} \sum_{i=1}^N \left( y_i - \beta^0 - \sum_{j=1}^p \beta^j x_{ij} \right)^2$$

subject to  $\sum_{j=1}^p |\beta^j|^q \leq t$  (1)

For ridge regression,  $q = 2$ , for lasso,  $q = 1$  and for subset selection,  $q = 0$ . One may use a  $0 < q < 1$  in the regularization term. This approach shall have the following features.

1. As  $q$  decreases, the size of the coefficients is constrained more than that in LASSO and thus the variance of the solution decreases. However, the bias in the solution is increased and thus the net affect on prediction accuracy cannot be ascertained.
2. The approach is biased to push more coefficients set to an exact zero than LASSO. For good feature selection, one needs *appropriate* number of coefficients to be set to zero. Thus it is unclear, whether this new approach shall be better than LASSO on this criteria.
3. For  $q > 0$  we still have a continuous shrinkage process that avoids the high-variance problem of subset selection, which is a discrete process.

Figure 1 shows a geometric depiction of regression for a two dimensional input variable [7]. The diamond (with dotted lines) depict the constraint imposed by LASSO. The LASSO solution cannot lie outside this diamond. The figure with sides 'concave up' depict the constraint imposed by the new technique proposed above. The elliptic contours of the Squared error term centered at the OLS coefficients is also shown. The solution for the respective techniques exist at the point where the contours first intersect the shape depicting the technique. It is observed that the new technique constrains the coefficients more than LASSO (first point above). Secondly, it is shown that for a given OLS coefficient and contour shape, LASSO doesn't result in a zero coef-

ficient while the new technique does. This is in support of the second point mentioned above.

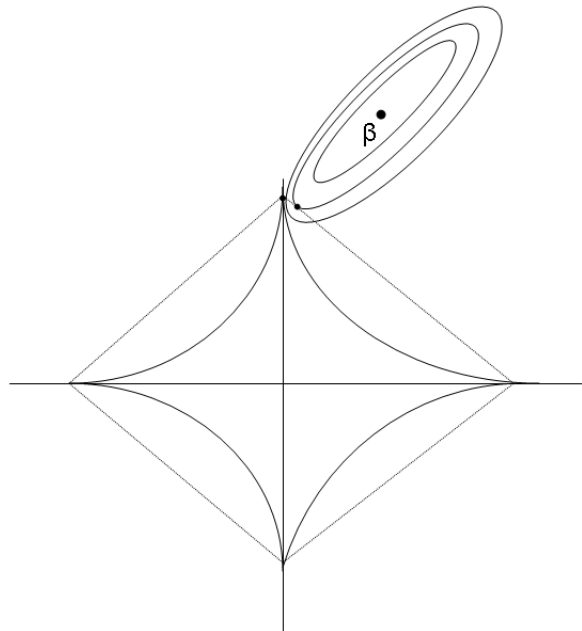


Figure 1: Geometric description of regression

We call this new approach to regularize regression (using  $0 < q < 1$ ) as the Least-Powered Absolute Shrinkage and Selection Operator (LPASSO). Each of Ridge, LASSO and Subset Selection outperform all the others given a particular characteristic of data-set [Section 11 of 1]. LPASSO lies somewhere in-between Subset Selection and LASSO. We seek to ascertain whether LPASSO outperforms these techniques (esp. LASSO) in some of the data-settings as mentioned above.

*Thus, the first question of interest is which of these two models (LASSO and LPASSO) is better to fit data for performing linear regression? The foundation of this study is based on the optimism that LPASSO is a better technique than*

*LASSO.*

Though a method exists to exactly fit the data to one of the models (LASSO), there is no way to fit the data accurately to the other (LPASSO). In LASSO, we need to solve a convex optimization problem to fit the model to data. The global optimum can thus be found accurately. On the other hand, LPASSO results in a non-convex problem and there is no method to find the global optimum for it. This suggests that we can fit the LPASSO model only approximately.

*Thus, the second question of interest is whether an approximate fit of the second model (LPASSO) is better than the exact fit of the first model.*

The first question stated above cannot be answered with surety. This is so because we can never fit the LPASSO model accurately. We seek to answer the second question. The answer to the question depends on:

1. The answer to the first question;
2. How well we fit the LPASSO model: We want to design optimization algorithms to find the *best possible* fit of the LPASSO model.

The answer to this question will be strongly suggestive to the answer of the first question as well. Given that the fit of LPASSO is approximate and its accuracy depend on data and the computational resources, only an indicative answer can be given based on benchmark data.

It is imperative to also examine regression regularization with  $q = 1.5$  and compare its performance with the rest of the methods. We call this operator the Modified Ridge operator. While the objective function to minimize with the Modified Ridge operator is convex, there is no known analytical solution for obtaining a minima. Given

that we can show that LPASSO and modified Ridge outperforms the other techniques, these can emerge as competitive regression techniques.

The rest of this report is organized as follows. We present an overview of the methods we use to minimize the various regularized regression loss functions in Section 2. Section 3 presents in detail the implementation details for the various methods including synthetic data generation, cross-validation and fine-tuning of parameters. We present our results in Section 4 and analyze them in Section 5. Section 6 presents our conclusion and suggestions for future work are presented in Section 7.

## 2 Methods

In this section we provide formulations of various methods of regression regularization and briefly describe the methodologies used to obtain their solutions. Details of implementation for the methods we propose for LPASSO are left for Section 3.

### 2.1 Subset selection

In this approach only a subset of the features of the input data are retained and ordinary least squares estimates are used to estimate the parameters of these features. We employ *best subset regression*, which finds the best subset of size  $k \in \{1 \dots p\}$  for each value of  $k$ . The leaps and bounds method[2] efficiently computes this for datasets with 30 features or lesser. We used the R *leaps* package for subset selection. The value of  $k$  that is used is decided using five-fold cross validation as recommended by Breiman and Spector[1]. The implementation of cross-validation is identical to that used for other methods and we discuss it in Section 3.3.

## 2.2 LASSO

Minimizing the Lasso criterion can be represented as a quadratic programming problem with linear constraints. The loss function is the squared error, which is quadratic. The constraint  $\sum_j |\beta_j| \leq t$  for  $p$  features can be represented as  $2^p$  linear constraints, of the form  $(\pm 1, \pm 1, \dots, \pm 1)\beta$ . However, the number of linear constraints can be exponential. Fortunately, Tibshirani suggests[7] a more efficient way to solve the problem – we initially only add the constraint represented by the OLS estimates and run the quadratic programming problem. If the resulting weights satisfy the overall constraint, we are done and if not, we add the violated constraint and recompute the quadratic programming solution. We iterate until a solution is found that matches the overall constraints. Although in the worst case we could still end up with having to solve a quadratic problem with a large number of constraints, in practice the average number of iterations lie in the range  $(0.5p, 0.75p)$ [7]. Another efficient method to address this problem has been suggested which involves representing each  $\beta_j$  as the difference of non-negative values, which increase the number of variables but only have  $2p+1$  constraints. We implemented only the first method and found it to perform efficiently.

## 2.3 LPASSO

Rather than model the LPASSO constraint as a squared error minimization problem with constraints, we convert it to the following equivalent form:

$$(\hat{\alpha}, \hat{\beta}) = \underset{\alpha, \beta}{\operatorname{argmin}} \sum_{i=1}^N \left( y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 + \lambda \sum_j |\beta_j|^{0.5} \quad (2)$$

We transform the problems as above, since optimization problems with non-linear constraints are difficult to solve by local methods such as gradient descent or global optimization methods like evolutionary algorithms and simulated annealing. Though, one finds methods to do constrained optimization using evolutionary algorithms, the methods are difficult to code and have not shown good results. Thus, we take advantage of this mathematical transformation to make the optimization methods efficient.

The LPASSO criterion presents us with a non-convex optimization problem, for which there is no deterministic method that will efficiently give us a good solution. We implemented three methods that are widely used for solving such problems – Multiple-point gradient descent, Simulated Annealing and Particle Swarm Optimization.

We have designed each method to take in  $(X, Y)$  data and a  $\lambda$  value and output a weight vector which minimizes the objective function. We discuss some of the generic salient issues to be kept in mind while using these algorithms.

**Behavior of local optimum.** Although they have been known to perform well in practice, each of these is a non-deterministic randomized algorithm with no guarantee of convergence on a global optima. This means that we can fit the LPASSO model only approximately to a given set of regression data. More specifically, the

weight vector that these methods return may only be a local minimum of the objective function rather than the global minimum. This can give rise to the following problem. A local optimum may not sets the weights as zeros as in the case of global optimum, which is imperative for feature selection.

**Setting the Parameters.** These optimization methods generally have many parameters to be set, which are decided according to the analysis/experience/experimentation of the designer. These parameters are notoriously called the magic knobs, since they have a large influence on the performance of the algorithm and there is no deterministic way to ascertain them. To make our methods more general and robust, we use problem-specific knowledge to automatically set the magic-knobs. We use the fact, that solution to the ordinary-least square problem can be found quickly, which is suggestive of the range and nature of the solution for LPASSO. This helps us set the parameters of the algorithm automatically. This though renders our method weak as compared to one having parameters specially tuned to the data-at-hand, it gives strength to our claim regarding the efficiency of the algorithm to fit the LPASSO model. This also relieves the user to set the parameters manually. The robustness of this automatic setting of parameters is checked over the three different experiments reported in Section 4.

**Resource-Performance tradeoff.** There is a time-performance trade-off for these optimization methods. The more the time the algorithm runs for, the better it performs. One way to decide when to stop is to observe when the incremental improvement is low. However, there

is nothing which stops the algorithm to be stagnant for a long time and then suddenly discover a large incremental improvement. Thus, the decision is largely guided by the available computational resources. The algorithms converged (gave low incremental improvement) in 10s of seconds on Pentium 4 PC. We imposed a constraint of one minute run-time (for the data considered in this study). In a practical setting, one could better our results if more computational power and time is available. On the other hand, in presence of a larger data set or for more number of features, the method may not scale given the resources. We make no general claim about the scalability of the algorithm than indicative by the benchmark data.

**Precision of solution.** One needs to decide when to consider a coefficient zero. To consider a coefficient zero, its effect on the total sum of the regression expression should be negligible. The contribution of the coefficient shall depend on the size of the feature, which in a given setting could be normalized. Thus one way to decide a coefficient to be zero could be when it is less than a factor (say  $10^{-4}$ ) less than the mean of the coefficients.

Section 3.2 describes our parameter tuning and Section 3.4 compares the performance of the three methods we chose on finding the right optimum.

## 2.4 Modified Ridge

We use unconstrained minimization to get the weights given by the modified ridge criteria, which minimizes the following function:

$$\begin{aligned}
(\hat{\alpha}, \hat{\beta}) = \operatorname{argmin} & \sum_{i=1}^N \left( y_i - \alpha - \sum_j \beta_j x_{ij} \right)^2 \\
& + \lambda \sum_j |\beta_j|^{1.5} \quad (3)
\end{aligned}$$

We use the MATLAB *fminunc* routine to minimize this function using a Quasi Newton method with line search based on a mixed polynomial method. Since this function is convex with a single global minimum, convergence is guaranteed in a reasonably short time.

Like Ridge regression, we do not expect this method to be aggressive in pushing weights to zero, because as  $|\beta|$  gets small,  $|\beta|^{1.5}$  gets ever smaller decreasing the penalty term. The rate of decrease is not as high as  $|\beta|^2$ , so we still expect to see smaller weights in general than those given by ridge. It should also be kept in mind that though a solution to this method can be obtained fast, it is still much slower than ridge regression for which an analytical solution exists that only requires a few matrix operations.

### 3 Implementation Details

#### 3.1 Synthetic data generation

We wanted to replicate the experiments done in [7] to compare LASSO and other regression techniques. We could then compare LPASSO with others based on the same data.

50 data sets of 20 observations each are generated, with eight real-valued features and a single real-valued output. The correlation between  $x_i$  and  $x_j$  is given by  $\rho^{|i-j|}$  with  $\rho = 0.5$ . The relation between the input and output is given in (4). Here,  $\epsilon$  represents the standard normal (noise), while  $\sigma$  is a weighing coefficient.

However, no information about the distribution of  $X$  is given in [7]<sup>1</sup>. Without this knowledge, we could not replicate the LASSO results exactly and we choose to compare our results with those presented in the LASSO paper based on trends.

$$Y = X\beta + \sigma\epsilon \quad (4)$$

The correlated input data was created in the following way. Eight uncorrelated random data sets were created, whose value were spread uniformly between 0 and 1. This matrix containing the 8 vectors was then post multiplied by the upper triangular Cholesky decomposition of the correlation matrix  $R$ . The resultant matrix was standardized to have mean 0 and standard deviation unity. A Gaussian random variable with zero mean and unit variance was generated for the noise. The output values were generated as given in (4).

#### 3.2 Non-convex optimization

We now describe the methods we used to compute the LPASSO solutions along with their detailed implementation.

##### 3.2.1 Multiple point gradient descent

Multiple random points are generated and a gradient descent method is used to find the optima for each point. We used the *fmincon* function of MATLAB with zero constraints. This method uses a method similar to a quasi-Newton method with line-search based on a merit function.

We also tried using *fminunc* (unconstrained optimization, Quasi Newton method with line search based on a mixed polynomial method)

---

<sup>1</sup>We contacted the author but he was unable to provide us this information

and *fminsearch* (Simplex method). We found that *fmincon* outperformed both these methods in several runs for the LPASSO optimization problem. This is indicative that it is a better method to solve the problem at hand<sup>2</sup>. The difference between *fminunc* and *fmincon* is in the line-search method[6]. We are unaware of the exactly why *fmincon* performs better in our case.

We used the domain specific knowledge to influence the algorithm structure. The regression coefficients for the ordinary least squares can be found deterministically and very quickly. In case, they don't exist, a ridge estimate can be calculated. The OLS coefficients indicate the range where the LASSO solution may lie. In the early stages of project, we believed that the LPASSO solution shall lie somewhere between the OLS coefficients and origin. However, this is not true. Since LPASSO pushes some coefficients to zero, it may make other coefficients stronger. The addition of regularization to the OLS estimate makes it behave more irregularly than perceived intuitively. They tend to even push the coefficients in a different quadrant than the OLS coefficients[7]. Thus such a hypothesis that the solution lies between origin and OLS coefficients is wrong.

We chose the initial points as a normal random variable with mean at zero and standard deviation as the ceiling of maximum OLS coefficients. This shall put points in all quadrants and shall also put some points outside the sphere bounded by the maximum OLS coefficient.

As reported, gradient descent was implemented using the MATLAB function *fmincon*. We retained all the default parameters of *fmin-*

---

<sup>2</sup>In a different optimization problem, we could observe *fmincon* doing worse than *fminunc*, which shows that this improvement is not general.

*con* as in MATLAB Version 7.0.4.365(R14) Service Pack 2. The maximum number of function iterations allowed was equal to the number of variables multiplied by 100. The precision of the solution was kept to 1e-6. We created a total of 30 points to do gradient descent and chose the lowest error minima.

### 3.2.2 Simulated Annealing

Simulated Annealing uses a Metropolis Criterion Monte Carlo simulation to find the optimum value of a function[5]. Simulated Annealing works by exploring the search space in a random manner, and by accepting or rejecting exploration moves based on the a *temperature* parameter and the Metropolis criterion – moves that decreases the system energy are always accepted and moves that increase the system energy are accepted with a probability proportional to the current temperature.

The exploration continues for a fixed number of iterations at a given temperature, and the the temperature is gradually decreased according to a cooling schedule, until a designated final temperature has been reached. The input point that returned the lowest energy during the exploration is returned as an optimum value.

For our algorithm, we defined the metropolis criterion in a standard manner.

$$P(X_i \rightarrow X_j) = \begin{cases} 1, & \text{if } E(X_j) < E(X_i) \\ e^{-\frac{E(X_j) - E(X_i)}{T}}, & \text{otherwise} \end{cases} \quad (5)$$

where  $P(X_i \rightarrow X_j)$  is the probability that a move from  $X_i$  to  $X_j$  is accepted,  $T$  is the temperature and  $E(X_k)$  is the energy (value of the objective function) at point  $k$ .

Based on a few sample runs, we chose the following parameters. We set the initial temperature to be 100 and the final temperature to be  $10^{-5}$ . At each temperature we tried 100 moves and the we chose to decrease the temperature by a factor of 0.95 at each step (i.e. a proportional cooling schedule).

Another important parameter in a Simulated Annealing algorithm is the sampling strategy - a random process by which we generate new weights to explore the search space. For each weight vector  $\beta$  with  $p$  features, we sampled new values for  $\beta$  by picking a random index  $j$  from 1 to  $p$  and then picking a new value for  $\beta_j$  according to (6).

$$\beta_j^{new} = \beta_j^{old} + \Phi(rand() - 0.5) \quad (6)$$

where  $rand()$  returns a uniform random number from 0 to 1 and  $\Phi$  is a constant. We chose to not change all weights simultaneously because at later stages of the optimization we will want to move toward lower weights in general and a random reassignment of all the weights will probably not generate good moves.

The constant  $\Phi$  was chosen as the smallest weight in the OLS estimate bigger than 0.1. This is based on the observation that if we choose a stepsize that is too large, getting weights to close in to zero will be difficult simply because we will not generate samples at a fine enough resolution to do so.

The initial point for simulated annealing was chosen to be the OLS estimate.

We found that the final solution for simulated annealing was pushing coefficients to as low as  $10^{-4}$ . Further decreases in weight value are not performed due to the step size. To take care of this problem, we applied gradient descent starting with the final solution of simulated anneal-

ing which does not change the solution much but pushes the smaller coefficients to  $10^{-9}$  or lower.

### 3.2.3 Particle Swarm Optimization

PSO was invented by Kennedy and Eberhart in 1995[4]. It falls in the class of evolutionary algorithms, where the search begins with a set of points which are guided towards the optima over generations.

The algorithm works in the following way. A set of points called particles are initialized as the initial population. The algorithm iterates over the population for a number of generations to find the solution to a given optimization problem. Each particle has two properties: position and velocity. As the name suggests, position is the coordinate of the particle in the search space. Velocity represents a value which is added to the position of the particle to find its position for the next generation (considering time factor to be unity). The initial positions for the particles are random, while the velocity is initialized to zero.

In each generation, the fitness of the particle is calculated which is equal to the value of the function being optimized. Lower the value of the function, higher the fitness of the particle. The velocity of the particle is updated according to the best position the particle has come across in its life-time (all generations) and the best point the whole swarm (all the particles) has ever seen. The best point here refers to the point of best fitness. Thus all the points are updated and a new population is formed, which is again iterated upon in a similar manner.

The reader is referred to [4] for equations used to perturb and update the particle positions.

The search is guided by biasing it to move in the direction of the best point it has seen and

that seen by any particle. A parameter of the algorithm is maximum velocity, which determines the maximum amount by which a point can be perturbed in a single iteration.

PSO has been successfully used to optimize the extremely non-linear Schaffer f6 functions and train a 13-dimensional Neural network[4]. They compare well to genetic algorithms in reference to the number of function evaluations, but are extremely simple to code and understand.

To suit our problem, we had to modify the PSO algorithm. The conventional algorithm was unable to solve the optimization problem at hand. As stated before, the parameter, maximum velocity is equivalent to the maximum value by which any variable is perturbed in a single iteration. In the conventional PSO, a maximum velocity is set which decides how wildly the particle flies in the search space. A high maximum velocity relates to a good global search, while a low maximum velocity leads to a good local search around a point.

In our problem, we required both a global search and a local search to the precision of  $1e-4$  or more to push coefficients to zero (with the same precision). To facilitate this, we used a time adaptive maximum velocity (MV).

The MV was initialized to the ceiling of the maximum OLS coefficient. This gave the particles an initial independence to fly around vigorously in the search space. The maximum velocity was scaled by half every  $n$  (40 in our case) generations, till it became less than  $1e-7$ . It was then fixed to  $1e-7$ . This way, in the initial stages, the algorithm is biased towards doing a global search and later a local search around the particles. This modification to the algorithm got it working and it started finding lower error than before and pushing coefficients to zero. According to our best knowledge, such a modification

to PSO has been first time proposed and implemented.

Parameters of PSO: The parameters of PSO were fixed as follows:

1. Total number of particles in a generation: 30
2. Initial population: A normal random population with zero mean and standard deviation as ceiling of maximum OLS coefficient. Two points in the initial generation were put at origin, while another two were put at the OLS coefficient value.
3. Number of Generations: 5000. This parameter was ascertained after a few runs and empirical decision as to when the improvement curve of the algorithm becomes nearly stagnant. A function was devised, which would automatically increase the number of generations, if the MV is unable to go down to  $1e-6$  (this can happen due to very large OLS coefficients).
4.  $c_1 = 2, c_2 = 2$  (Conventional)
5. MV update: MV was initialized as the ceiling of maximum OLS coefficient and halved every 40 generations.

The best solution found in the life time of the swarm was further optimized by gradient descent and then used as the solution.

### 3.3 Cross Validation

We used 5-fold cross validation for parameter setting in all our experiments. For LASSO, LPASSO, modified ridge and ridge regression, the parameter to be set using cross validation is  $\lambda$ , the penalty or constraint term applied to the

norm of the weight vector. For subset selection, the parameter is the  $k$ , the number of features to be selected.

### 3.3.1 Cheaper methods for cross-validation runs

For subset selection, LASSO, modified ridge and ridge regression, the methods used to train during cross-validation are the same as those used to compute the real solutions. However, for methods which perform non-convex optimization computation time becomes a limiting factor when doing cross-validation. Due to this, we experimented with cheaper alternatives for performing parameter selection using cross validation, based on timing constraints.

Two options for using cheaper methods for cross-validation exist – we can use a deterministic minimization algorithm (like fixed initial point gradient descent), or we can use a less aggressive version of the non-deterministic algorithm (e.g. multiple point gradient descent with fewer restarts). Although the former option is the cheapest and can be used to select parameters across all the methods (because it is deterministic), we found that the  $\lambda$  values given by this method do not correlate well with the values selected when we use one of our randomized algorithms for cross validation.

Figure 2 shows the parameter values selected in 20 runs of cross-validation on the same dataset (used in Experiment 1, see Section 4.1). We observe that the values between the cheaper and expensive versions of Particle Swarm Optimization (1000 vs. 5000 generations) and Multipoint Gradient Descent (10 starting points vs. 30) are highly correlated and we decided to use the cheaper alternatives for cross-validation. We did not modify the parameters for Simulated An-

nealing because its running time even when using the full strength algorithm was acceptable.

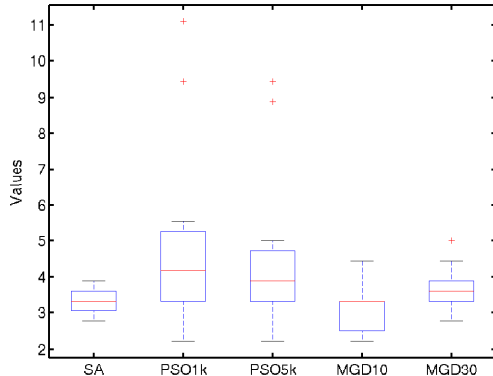


Figure 2:  $\lambda$  values returned by cross validation for various methods

### 3.3.2 Parameter ranges for cross-validation

Cross validation for subset selection is discrete, ranging from 1 to the number of features. For LASSO, we used the normalized constraint parameter  $\hat{s} = t / \sum |\beta_j|$ , where  $t$  is the constraint value, as defined in [7]. For LPASSO, modified ridge and ridge regression, we optimize using an objective function of the form:

$$f(\beta) = RSS(\beta) + \lambda reg(\beta)$$

As  $\lambda$  varies from a small to a large values the weightage given to the residuals decreases. If we let  $\lambda_{eq} = RSS(\beta^{OLS}) / reg(\beta^{OLS})$ , this value of  $\lambda$  balances the two components of the equation. We choose a range of  $\lambda$  from zero to  $4\lambda_{eq}$  for cross-validation. We perform cross-validation for 20 equally spaced values within this range.

### 3.4 Optimization performance

Before applying these non-convex methods to the LPASSO criterion, we measure their optimization performance against each other. We selected 100 datapoints from the dataset for our first experiment (Section 4.1), selected five values of  $\lambda$  over the cross-validation set and ran each algorithm 40 times for each value of  $\lambda$ . We compared the LPASSO value of the minimized solution generated in each case (a lower value is indicative of better optimization performance).

The mean LPASSO value and the best LPASSO value returned for each method is shown in Figure 3. In reference to the mean error, it is observed that all the three methods are very close. Multiple-point gradient descent performs the best for all  $\lambda$ , while PSOs performance worsens for higher lambda. SA does worse than Multiple-Point Gradient descent, but its performance is close to it. One observes that the best of run solution for all three approaches is same, except for PSO at higher values of  $\lambda$ .

### 3.5 Learning performance metrics

We compare our results from all methods using the following performance metrics.

**Model error.** It is measure of how well the generated regression coefficients predict the actual regression coefficients and can be evaluated only for synthetic data [7]. Model Error is given by (7).

$$(\hat{\beta} - \beta)^T V (\hat{\beta} - \beta) \quad (7)$$

where  $\hat{\beta}$  is the true weight vector,  $\beta$  is the regression weight vector and  $V$  is the population covariance matrix of data.

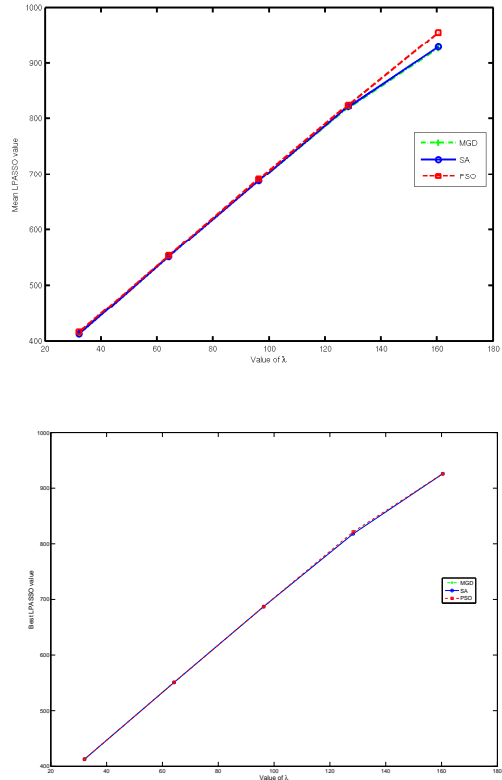


Figure 3: Performance of Multiple point gradient descent, Simulated Annealing and Particle Swarm Optimization. (a) Mean LPASSO value returned by 40 runs at different values of  $\lambda$ . (b) The best LPASSO value seen in 40 runs.

**Number of zero coefficients.** Number of zero coefficients indicates whether the regression model does any feature selection. We find the average number of zero coefficients for each technique and compare it with the actual number of zero coefficients in the model.

**Variance in solution.** We use a box-plot to compare the variance in the regression coefficients predicted by different regression models.

## 4 Results

### 4.1 Experiment 1

The data for the experiment was generated as mentioned in Section 3.1 with  $\beta = (3, 1.5, 0, 0, 2, 0, 0)$ ,  $\sigma = 1.73$ . As in [7], experiments were done on 50 data set of 20 values each. The Median ME, Mean ME, Average Number of Zeros and the most common Model selected by each regression technique is given in Table 1<sup>3</sup>.

LASSO does better than OLS, Best Subset, Ridge and Modified Ridge in case of Median ME and Mean ME. LASSO sets 2.48 coefficients to zero, while Best Subset sets 4.52 coefficients to zero. The actual model has 5 coefficients set to 0 and Subset Selection comes nearest to this number. The most common model chosen by Best Subset is 11001000 (1 represents the coefficient is non-zero and vice-versa), while the most common model selected by LASSO was 11111111. According to these results, Best Subset Selection and LASSO trade-off between Median ME and Average number of coefficients set to zero (Henceforth called Zero coefficients). These results confirm the findings in [7].

---

<sup>3</sup>Mean ME was an additional metric used by us, not used in [7]

LPASSO outperforms all other methods in case of both mean ME and Median ME in all the three cases. The Zero Coefficients for LPASSO are more than LASSO, while they are less than those in Subset Selection. The most common model selected by LPASSO is 11001000, which is indeed the correct model. In the three techniques used to implement LPASSO, SA performs best on Median ME, while PSO does best on Mean ME and Zero Coefficients.

A box-plot of coefficients generated by all the techniques and LPASSO using SA was generated and is shown in Figure 4. The variability of Subset is evident, while LPASSO seems to have more variability than LASSO. This goes against the hypothesis that LPASSO would reduce variability. This could be attributed not to the variability of the LPASSO model, but the method (SA) used to fit it, which induces variance in both finding the  $\lambda$  using CV and then finding the  $\beta$  for the given  $\lambda$ .

### 4.2 Experiment 2

The data is same as Experiment 1 with  $\beta = (0.85, 0.85, 0.85, 0.85, 0.85)$  and  $\sigma = 1.73$ . The results are given in Table 2. This experiment makes it clear that when none of the features are expected to be redundant, using LASSO/LPASSO or subset selection is undesirable. As expected, Ridge Regression does best on this data by a good margin with reference to all the performance criteria used. Modified Ridge comes close to Ridge Regression in case of Mean ME, however is outperformed by a good margin by Ridge in case of Median ME.

Subset selection has the worst performance on this data. LPASSO also performs worse than LASSO in terms of model error as well as number of zero coefficients; but it is better than Subset

Table 1: Results for experiment 1

Method	Median ME	Average ME	Average #zero coeffs	Most Common Model	Average runtime per dataset (sec)
Subset	1.141	1.534	4.52	11001000	10.78
LPASSO (MGD)	1.107	1.405	3.92	11001000	296.9
LPASSO (SA)	0.967	1.396	3.82	11001000	246.6
LPASSO (PSO)	0.999	1.350	3.98	11001000	564.0
LASSO	1.287	1.510	2.48	11111111	2.45
Mod. Ridge	1.339	1.590	0	11111111	6.59
Ridge	1.470	1.802	0	11111111	0.0464
OLS	1.4364	2.000	0	11111111	0.0006

Table 2: Results for experiment 2

Method	Median ME	Average ME	Average #zero coeffs	Most Common Model	Average runtime per dataset (sec)
Subset	3.080	3.446	2.66	11111111	13.96
LPASSO (MGD)	2.556	2.783	2.26	11111111	291.9
LPASSO (SA)	2.347	2.722	2.12	11111111	247.8
LPASSO (PSO)	2.174	2.564	2.10	11111111	563.9
LASSO	1.287	1.692	2.133	11111111	2.08
Mod. Ridge	1.165	1.444	0	11111111	6.83
Ridge	0.854	1.256	0	11111111	0.0464
OLS	1.938	2.555	0	11111111	0.001

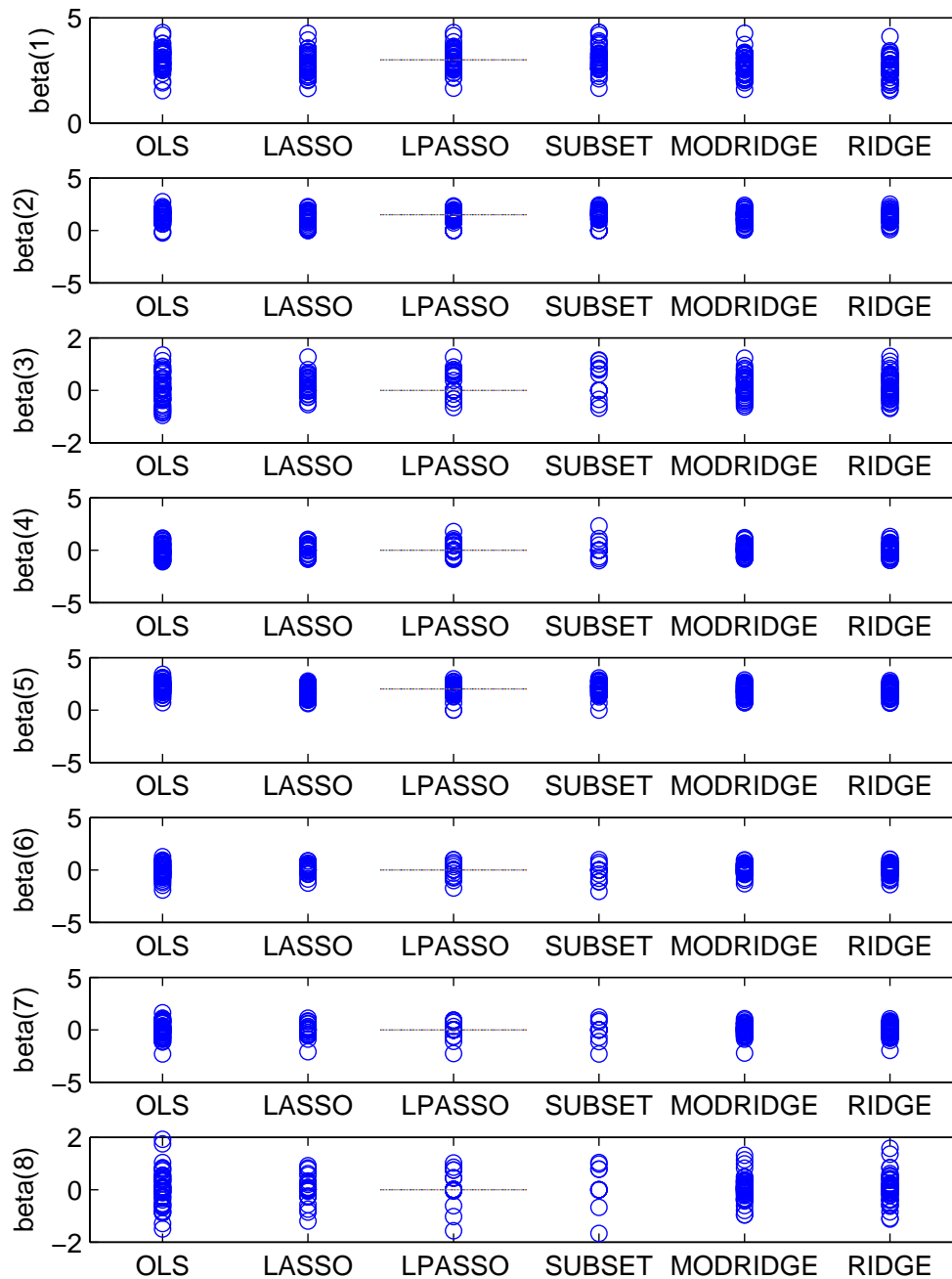


Figure 4: Estimates for 8 coefficients for experiment 1 excluding the intercept: horizontal line indicates the true coefficients.

Table 3: Results for experiment 3

Method	Median ME	Average ME	Average #zero coeffs	Most Common Model	Average runtime per dataset (sec)
Subset	0.197	0.569	6.16	10000000	10.64
LPASSO (MGD)	0.167	0.499	5.96	10000000	311.4
LPASSO (SA)	0.185	0.525	5.66	10000000	256.7
LPASSO (PSO)	0.216	0.498	5.72	10000000	576.2
LASSO	0.456	0.684	4.1	10000000	2.91
Mod. Ridge	0.819	1.002	0	11111111	6.02
Ridge	1.297	1.369	0	11111111	0.0518
OLS	1.366	1.504	0	11111111	0.0004

Selection. These results are easily explained by the fact that LASSO, LPASSO and subset selection are being increasingly aggressive at pushing coefficients to zero when in fact none of them are.

### 4.3 Experiment 3

The data is same as the other experiments with  $\beta = (5, 0, 0, 0, 0, 0, 0, 0)$  and  $\sigma = 1.414$ . The results are given in Table 3. As reported in [7], Subset Selection performs the best in comparison of all conventional techniques by having the best Median ME/Average ME and zero coefficients. The most common model is chosen is 11000000, which is closest to the actual  $\beta$ .

LPASSO performs better than Subset Selection in case of Median ME and Average ME in case of SA, MGD and all three cases (MGD, SA and PSO) respectively. Though LPASSO has lesser number of zero coefficients, the most frequent model selected by it is indeed the correct model unlike the case of Subset Selection.

The better performance of LPASSO than Sub-

set Selection maybe attributed to the variance of Subset Selection, it being a discrete process. A box-plot of the regression coefficients for subset and LPASSO is shown in Figure 5. Once again, we can see that LPASSO results have lesser inherent variance than those given by subset selection.

## 5 Discussion

We tested our new scheme LPASSO with the current regression techniques on the benchmark data given in [7]. The results matched the trends for all techniques as given in [7].

For datasets with significant number of non-predicting features (zero weights), our results show that LPASSO performs better than both LASSO and Subset selection in terms of model error. The performance benefit over LASSO<sup>4</sup>

<sup>4</sup>We note that according to the literature, generalized cross validation (GCV) gives the best performance for the LASSO criterion [7], which we did not implement. It will be interesting to compare these results with LASSO using GCV selected parameters, and also to examine whether a

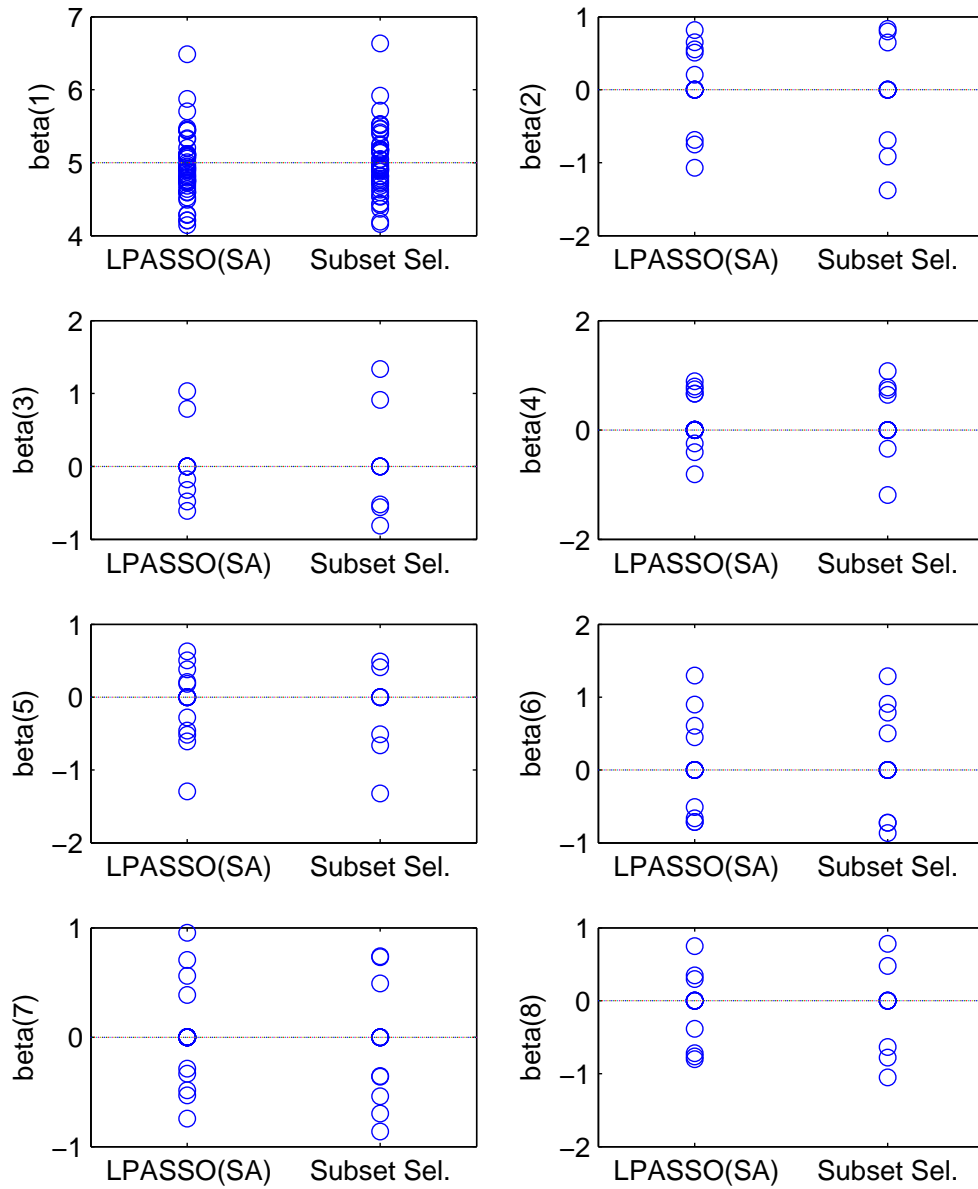


Figure 5: Estimates for 8 coefficients for experiment 3 excluding the intercept: horizontal line indicates the true coefficients.

can be attributed to the fact that LPASSO is more aggressive in pushing weights to zero, which is confirmed by the fact that we see a greater number of zero coefficients on average when compared to LASSO. However, the performance improvement over subset selection cannot be explained in this manner, since subset selection is the most aggressive method that pushes weights to zero, and our results reflect the same. We believe that LPASSO does better than subset selection because its solutions have a lesser inherent variance, as shown in Figure 5.

For datasets in which all features are predictors, we observe that as the regularization norm  $q$  decreases, the results get progressively worse – Ridge performs the best, followed by modified ridge, LASSO, LPASSO and subset selection performs the worst. Yet, ridge regression methods cannot do feature selection efficiently, even when a large number of features are useless for prediction. This underlines the importance of domain knowledge while implementing regression. None of the aggressive algorithms for feature selection can perform a “sanity check” to see if feature selection is feasible on a given dataset – they blindly go ahead and push weights to zero. A regularizer for regression which has the above property will be extremely useful.

An interesting observation from our experiments is the following. In terms of optimization performance, our data indicate that the performance of PSO is slightly below that of Simulated Annealing or Multiple-point gradient descent. However, in terms of learning performance (on unseen data), our results indicate that PSO performs slightly better than the other two algorithms. This could be due to two reasons – either the PSO learning algorithm got “lucky” (a

sub-optimal fit to the LPASSO criterion turned out to do better on unseen data), or the amount of data we used for measuring the optimization performance was too small.

We have seen that although there is great variability in the three optimization techniques we use, even over the same data, they produce good performance in terms of prediction accuracy and feature selection. The element of variability is pronounced due to variability in selection of  $\lambda$  and the variability in the coefficients found given a  $\lambda$ . We average the results over 50 runs on similar data-sets, which shall decrease the variability to some extent. We thought about trying multiple runs of the optimization algorithm on each data set, but we discarded the idea because of running time constraints. The confidence why these results shall generalize to other data sets comes from the fact, that the optimization methods had no knowledge of the data, aside from some sample runs used to fine-tune their parameters. Given that approximate solutions to the LPASSO model, even with their inherent variability, produce good performance is indicative of the fact that the underlying LPASSO model is a good method to regularize regression.

We also observe that the modified ridge operator does not push any coefficients to zero, though it does shrink the weights more than ridge regression. It appears that there is not much point in using this operator given that it takes much longer than ridge with a small performance increase but no increase in feature selection power.

## 6 Conclusions

These results suggest modification to the conclusions drawn in Section 11 of [7].

1. *Small number of large effects:* Subset and

---

GCV method can be developed for the LPASSO criterion.

LPASSO perform the best (trade-off between performance criteria) followed by LASSO and Ridge.

2. *Small to moderate number of moderate sized effects:* LPASSO outperform LASSO, followed by Subset Selection and then Ridge
3. *Large number of small effects:* Ridge performs best followed by LASSO, then LPASSO and Subset Selection. (Same as before)

The answer for the two questions raised in Section 1 are summarized in the following conclusion:

1. An approximate fit of LPASSO outperforms LASSO. (Question 1)
2. It is thus *indicated* that a perfect fit of LPASSO is a better model than LASSO. (Question 2)
3. LPASSO takes approximately 100 times more time than LASSO in case of MGD and SA, while it takes 200 times more time when implemented using PSO. This may become a serious concern when the method is used in a practical setting.

## 7 Future work

The present study may be complemented/improved/used in the following ways:

1. With the development of optimization methods for implementing LPASSO model and indication of its superiority, it can be now used in a practical setting for real data. The performance of the trained LPASSO

and LASSO can be compared on a Validation Set and the better model could be used.

2. A proposal to learn the value of the power of the regularization coefficient according to the data is made in [3]. Such a study can be an extension to the present project.
3. In the present study, LPASSO and LASSO have used five-fold CV. It will be a nice study to compare their performance in case of other CV schemes like Generalized Cross Validation. One can set  $q$  values between 0 and 1 other than 0.5.
4. This study indicates that a perfect fit of LPASSO shall outperform LASSO. This may yield interest developing a deterministic method to fit the LPASSO model.

## Division of labour

The project has been a result of cooperative effort. Considering this project has a research component and incorporates some original ideas led to several discussions between the group members at each stage. A division of labour has been presented, which is my all means fuzzy, since the ideas of the two members were incorporated in each others work.

1. Anshul Nigham: Implementation of LASSO, Subset Selection and Simulated Annealing.
2. Varun Aggarwal: Data Generation, Implementation of Multiple Point Gradient Descent and Particle Swarm Optimization
3. Anshul Nigham and Varun Aggarwal: Formulation of LPASSO, Strategy for Cross-Validation, Conduction of Experiments

and Result Interpretation, Report Writing (apologies, couldnt disentangle these)

## Acknowledgements

We will like to thank Prof. Leslie Kaelbling for encouraging us to try this problem as an end-term project for the course, 6.867. Thanks to TAs for course, 6.867, John and Luke for listening patiently to us and advising us. A special thanks to Prof. Una-may O'Reilly for showing interest in the project, encouragement and contribution towards the ideas presented herein. Lastly, we thank Prof. Tibshirani to answer our emails.

## References

- [1] BREIMAN, L., AND SPECTOR, P. Submodel selection and evaluation in regression: the x-random case. *International Statistical Review* 60, 3 (1992), 291–319.
- [2] FURNIVAL, G. M., AND WILSON, R. W. Regressions by leaps and bounds. *Technometrics* 42, 1 (February 1974), 69–79.
- [3] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*, 1 ed. Springer, 2001.
- [4] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *IEEE International Conference on Neural Networks, 1995.* (1995), vol. 4, IEEE, pp. 1942–1948.
- [5] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science* 220, 4598 (May 1983), 671–680.
- [6] THE MATHWORKS. *MATLAB Optimization Toolbox User's Guide*, 3 ed., 2005.
- [7] TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.* 58, 1 (1996), 267–288.