MODEL SELECTION WHEN THE NUMBER OF VARIABLES EXCEEDS THE NUMBER OF OBSERVATIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF STATISTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Victoria Stodden

September 2006

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

David Donoho, Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Bradley Efron, Department Chair

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Trevor Hastie

Approved for the University Committee on Graduate Studies.

Abstract

The classical multivariate linear regression problem assumes p predictor variables $X_1, X_2, \ldots, X_p$ and a response vector y, each with n observations, and a linear relationship between the two: $y = X\beta + z$, where $z \sim N(0, \sigma^2)$. This thesis find that when $p > n$, there is a *breakdown point* for standard model selection schemes, such that model selection only works well below a certain critical complexity level depending on n/p. This notion is applied to some standard model selection algorithms (Classical Forward Stepwise, Forward Stepwise with False Discovery Rate thresholding, Lasso, LARS, and Stagewise Orthogonal Pursuit) in the case where $p \gg n$.

The notion of the Phase Diagram is borrowed from signal processing and statistical physics to discover that 1) the breakdown point is well-defined for random X-models and low noise, 2) increasing noise shifts the breakdown point to lower levels of sparsity, and reduces the model recovery ability of the algorithm in a systematic way, and 3) below breakdown, the size of coefficient errors follows the theoretical error distribution for the classical linear model.

Our results are exhibited in a chemometric application using P. J. Brown, T. Fearn, and M. Vannucci's near-infrared spectroscopy data.

A Matlab$^{\text{TM}}$ toolbox, *SparseLab*, is introduced, giving open source routines for researchers in the sparse representation community, as well as providing code for the

reproduction of figures from papers in sparse representation, including this thesis. The software is publicly available and downloadable from `http://sparselab.stanford.edu` .

Acknowledgements

This thesis is not the result of my effort alone. My advisor, David Donoho, provided immense intellectual guidance with seemingly infinite patience. Dave is without question one of the most generous people I have ever known. I will forever be in his intellectual debt.

I wish to thank Brad Efron and Trevor Hastie for being unfailingly encouraging over many years and creating a supportive environment for me. I always felt I could approach either of them with absolutely any issue. I also thank them for reading this thesis. I owe thanks to each member of my orals committee: Brad, Trevor, Michael Saunders, and Rob Tibshirani. And thanks to Joe Grundfest for suddenly illuminating the end of the tunnel. I also owe a debt of gratitude to Helen Tombropoulos and the supportiveness of the entire Statistics Department.

I am so lucky to have amazing people as friends, without whose encouragement I would not be at this point. I owe an insurmountable debt to Charles, and without Fil, Kirez, Eric, and Ryan my life would be very different. Monica, Darius, and Olga have been endlessly supportive. There are many more people I wish I had the space to thank personally.

Lastly, I wish to thank my parents, Clare and Ron Stodden. They bore me, raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The classical multivariate linear regression problem postulates p predictor variables $X_1, X_2, \ldots, X_p$, a response vector y, each with n observations, and a linear relationship between the two: $y = X\beta + z$, where X is the $n \times p$ model matrix and $z \sim N(0, \sigma^2)$. The model is widely used and has a long history, and the method of least squares was developed by Legendre in 1805 and Gauss in 1809 for orbital prediction of the planets. The English translation of Gauss's work was published in 1875 [31, 25]. In the 1870's and 1880's Sir Francis Galton and Karl Pearson used the model in what would be the most recognizable form to modern researchers: predicting offspring characteristics from their parents'. Today linear regression is used nearly ubiquitously: it is the main tool of empirical analysis in the social sciences and is widely used in engineering and other applied settings.

Gauss and Legendre developed the current method of *least squares* estimation for the vector of coefficients, β, i.e. least squares solves the problem $\min_\beta \|Y - X\beta\|_2^2$. In modern matrix notation the solution can be written:

$$\hat{\beta} = (X'X)^{-1}X'y.$$

Due to the inversion of $X'X$, the classical model requires $p \leq n$. Traditionally, prediction with fewer variables than observations has been typical for most linear regression settings. Recently we have been experiencing a *data deluge*: modern computers, automated data collection, huge volumes of data streaming onto increasingly inexpensive disk space. It is now common to find problems in which the number of variables far outstrips the number of observations. For example, DNA microarray data is time-consuming to collect per subject but often yields thousands of variables (genes). An example of a typical study is the well-known analysis of prostate cancer patients by Singh et al [28]. There were 50 men in the control group (no prostate cancer) and 52 in the experimental group (diagnosed with prostate cancer). Here, $p = 6033$ and $n = 102$. Also typical is Van't Wout et al.'s study of HIV genetic expression [47]. This study has 4 patients in the control group, 4 in the treatment group and measured 7680 genes. In a simple image processing experiment, the number of variables can easily outstrip the number of observations: 100 modest 256×256 greyscale images, with each image extended into a row, could create a data matrix with over $65,000$ variables but only 100 rows. There are perhaps over 8000 stocks currently traded in the United States (including NASDAQ, AMEX, NYSE, OTCBB); the number of tick-level data points outstrips this after a small number of trading days. For example, Bin Zhou [49] analyses a tick level dataset of only DEM/USD exchange rates: $1,472,032$ data points over one year. Standard statistical techniques, designed for situations with a large number of observations over a small number of variables, were designed in the pre-computer era where such circumstances were inconceivable.

In this thesis I examine several common statistical methods that select variables

and build a tractable, or *sparse*, model from an overabundance of variables. I study a setting in which there are large numbers of variables and observations, both proportional to each other, and most of the variables are irrelevant nuisance variables. I find that an idea from statistical physics, the phase diagram, helps organize simulation results, and shows that model selection methods begin to fail when the model complexity exceeds a critical level – a so called *phase transition*. I develop a tool to study variable selection in this setting, studying how the performance of model selection behaves as a function of the ratios {# potential variables / # observations} and {# actual variables / # observations}.

The layout of this thesis is as follows. The first chapter defines the problem and the solutions that the field of statistics has offered. The second chapter investigates the performance of these algorithms in random model matrix settings, varying the model noise level and the number of variables. The next chapter studies the results of the performance analysis and draws general conclusions.

The fifth chapter applies these findings to a 2004 study of component prediction using near-infrared reflectance spectra. Chapter Six introduces our Matlab^{TM} toolbox, *SparseLab*. The final chapter discusses the findings and possible useful directions for future research.

Chapter 2

Estimating the Model when $p \gg n$

2.1 Historical Background

In 1986 George Box coined the term *Factor Sparsity* [3] to describe a model where the vast majority of factors have zero effect – only a small fraction actually affect the response. His context was fractional factorial experimental design in industrial settings with the hypothesis that "a large proportion of process variation is explained by a small proportion of the process variables" ([3] p. 11). He called those explanatory factors *active*, and the remaining ones *inert*.

If the coefficient vector β is *sparse*, i.e. containing a few nonzero elements, then $y = X\beta + z$ can still be modeled successfully by exploiting sparsity, even when the problem is underdetermined in the classical sense. A sparse model, with fewer than n active variables, can be estimated in the traditional least squares way, since $(X'X)$ is now invertible. But how to select the active variables?

Parsimony is an attractive feature for a potential model: aside from ease of interpretability, a simple model with the same explanatory power as a larger model is

always preferable – Occam's Razor. A number of strategies are commonly used in statistics to extract a parsimonious linear model from a large number of potential predictors: all subsets regression (fit all possible linear models for all levels of sparsity), Forward Stepwise regression (greedily add terms to the model sequentially, by significance level), Lasso [45] and Least Angle Regression [26] ('shrinks' some coefficient estimates to zero). The performance of each of these model selection algorithms will be analyzed by simulation methods. Our results will be organized and interpreted using the phase diagram, a concept from statistical physics.

2.2 Ideas from Sparse Representation in Signal Processing

We introduce some ideas from signal processing that will allow us to see that, in certain situations statistical solutions such as LASSO or Forward Stepwise, are just as good as all-subsets regression and will find the true underlying model.

2.2.1 ℓ_0/ℓ_1 Equivalence

When estimating a sparse *noiseless* model, we would ideally like to find the sparsest solution. We can define the ℓ_0 quasi-norm as the number of nonzero entries in the solution:

Definition 1 ℓ_0-norm *For a vector* $x = [x_1, x_2, \ldots, x_n]$*:*

$$||x||_0 = \#\{nonzeros\ in\ x\}$$

Using this to define the sparsest solution gives the following problem:

$$min_\beta ||\beta||_0 \; s.t. \; y = X\beta. \tag{2.1}$$

This is intuitively compelling – literally choosing the sparsest solution. Unfortunately it is not computationally feasible since it is not a convex problem and requires an all-subsets search (i.e. all subsets regression) and so is NP-hard [37]. In 1994, Basis Pursuit [10, 9], was pioneered for sparse representation in signal processing, and solves a similar problem:

$$min_\beta ||\beta||_1 \; s.t. \; y = X\beta; \tag{2.2}$$

this is a convex optimization problem [10]. As will be elaborated below, Donoho and collaborators have shown the surprising result that under certain circumstances the solution to Equation (2.2) is also the solution to Equation (2.1) [8, 15, 14, 16, 17, 27, 29, 35, 46].

2.2.2 Performance Analysis by Phase Transition

The Problem Suite

Following [24], to measure the performance of an algorithm in our context we define a *Problem Suite* $S\{k, n, p\}$ as a collection of problems with sparse solutions. Each problem has an $n \times p$ model matrix X and a k-sparse $p - vector$ of coefficients β. In this thesis, each element of the model matrix has been sampled iid from $N(0, 1)$, then the columns of X have been subsequently normalized. The k non-zero elements of the true solution vector, β, are drawn from a $Unif(0, 100)$. For each k, n, p combination we can run the algorithm in question multiple times, and measure its success according to a quantitative criterion. We chose the normalized root mean square error (nRMSE)

$nRMSE = ||\hat{\beta} - \beta||_2/||\beta||_2$, since results are then comparable across models built with different problem sizes and noise levels. To indicate algorithm performance we report the median $nRMSE$ over instances at given values for k, n, p.

The Phase Diagram

In the context of ℓ_0/ℓ_1 equivalence, Donoho [15, 14] introduced the notion of a *Phase Diagram* to illustrate how sparsity and indeterminacy affect success of ℓ_1 optimization, i.e. over a sequence of problem suites.

The Equivalence Result

More precisely, there is a threshold phenomenon such that, provided the sparsest solution is sufficiently sparse, the ℓ_1 minimizer is precisely that sparsest solution. This suggests we may be able to solve the tractable convex optimization problem Equation 2.1 in place of Equation 2.2 in some cases, and find the sparsest solution. This phenomena is labeled ℓ_1/ℓ_0 *equivalence*. For previous literature see [15, 14]

In [15], Donoho discusses the notion of *neighborliness*:

Definition 2 k-neighborliness *Every $k+1$ vertices of the convex hull of data points span a k-face.*

Restricting the analysis to convex hulls P with the property that for every $x \in P$, $-x$ is in P as well (called *centrosymmetric polytopes*), gives a slightly different definition:

Definition 3 Central k-neighborliness *Every set of $k+1$ vertices, no two of which are antipodes, span a k-face of P.*

We require one further definition:

Definition 4 Quotient polytope *A quotient polytope is formed by taking the convex hull of the* $2p$ *points in* $\mathbf{R}^p$.

In our context, X is $n \times p$ with $n \ll p$ and we can form the centrosymmetric polytope P by $P = XC$ where C denotes the ℓ_1 ball in $\mathbf{R}^p$. We wish to solve Equation (2.1) and we conjecture this sparsest solution can be found by solving Equation (2.2) provided that:

1. a solution with at most k non-zeros exists,

2. P is centrally k-neighborly.

Many examples of matrices with $n < p$ satisfy this condition, including those generated for experiments in this thesis. For example, [14] illustrates this with a column-wise concatenation of the $n \times n$ identity matrix with an $n \times n$ Hadamard matrix, to create $X = [IH]$ with $p = 2n$.

Theorem 1 (from [14][1]) *Let* X *be an* $n \times p$ *matrix,* $n < p$. *These two properties of* X *are equivalent:*

- *The quotient polytope* P *has* $2p$ *vertices and is* k*-neighborly,*
- *Whenever* $y = X\beta$ *has a solution* $\hat{\beta}$ *having at most* k *nonzeros,* $\hat{\beta}$ *is the unique optimal solution of the optimization problem (2.2).*

Theorem 2 (from [14][1]) *Consider a sequence of problem sizes* (k_n, n, p_n) *where* $k_n/n \to \rho \in (0,1)$ *and* $n/p_n \to \delta \in (0,1)$. *Let* X_{n,p_n} *be an* $n \times p_n$ *matrix with with Gaussian iid entries and* P_{n,p_n} *the corresponding quotient polytope.*

Suppose $\rho \leq \rho_{\ell_1}(\delta)$. *Then*

[1] for the proof see [14]

$$Prob\{P_{n,p_n} \; is \; k_n neighborly\} \to 1, \; as \; n \to \infty,$$

and if $\rho > \rho_{\ell_1}(\delta)$, *then*

$$Prob\{P_{n,p_n} \; is \; k_n neighborly\} \to 0 \; as \; n \to \infty.$$

This shows that it is useful to consider (δ, ρ) as a phase space and to consider that this space has two regions – one where ℓ_1 minimization is successful and one where it is unsuccessful.

If P is centrally k-neighborly, there will be a region of the Phase Diagram over which the true underlying model is recovered. We also expect there to be a transition region, and a region over which the model is not recovered, using the ℓ_1 formulation of the problem (Equation 2.2).

Figure 2.1 has been adapted from [15]. Each point on the plot corresponds to a statistical model for certain values of n, p, and k. The abscissa runs from zero to one, and gives values for $\delta = \frac{n}{p}$. The ordinate is $\rho = \frac{k}{n}$, measuring the level of sparsity in the model. Above the plotted phase transition curve, the ℓ_1 method fails to find the sparsest solution; below the curve the solution of (2.2) is precisely the solution of (2.1).

Inspired by this approach,the following recipe is used to study several statistical model selection algorithms:

1. Generate an underlying model, $y = X\beta + z$ where β is sparse i.e. has $k < p$ nonzeros.

2. Run a model selection algorithm, obtaining $\hat{\beta}$,

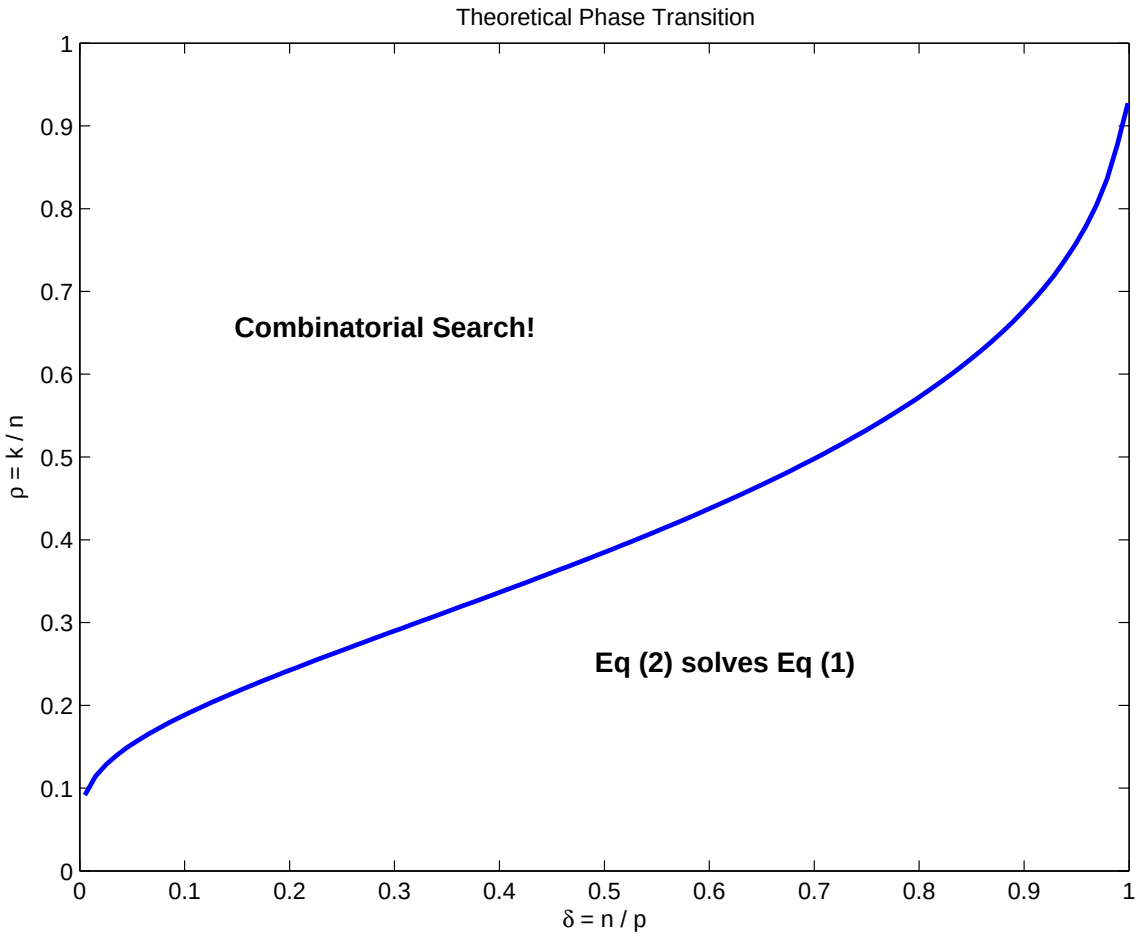

Figure 2.1: Theoretical Phase Transition Diagram: theoretical threshold at which equivalence of the solutions to the ℓ_1 and ℓ_0 optimization problems breaks down. The curve delineates a phase transition from the lower region where the equivalence holds, to the upper region, where it seems to require combinatorial search to recover the optimal sparse model. Along the x-axis the level of underdeterminedness decreases, and along the y-axis the level of sparsity of the underlying model increases.

3. Evaluate performance: $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2} \leq \gamma$.

The median value of the $nRMSE$ performance metric is displayed as a function of $\rho = k/n$ and $\delta = n/p$. Since both these variables $\rho, \delta \in [0, 1]$ the phase diagram is displayed over the unit square.

In 2005, Donoho, Drori, and Tsaig discovered empirical phase transitions for the Lasso and LARS algorithms in the noiseless setting that closely resembled the theoretical phase transition. This poses a question: is it possible such phase transitions exist for the noisy models ubiquitous in statistical research? This seems not to have yet been investigated.

Various model selction methods have been proposed within the statistics literature: All Subsets Regression, Forward Stepwise and Backward Elimination (see [48, 30]), Forward Stagewise, Least Angle Regression, and Lasso (see [26, 45]), Ridge Regression (see [36]), and Stochastic Search Variable Selection (SSVS) (see [34, 33, 32]). These methods usually yield different sparse models, contain tuning parameters, and no consensus exists on the circumstances under which one method might be superior to the others. In this thesis I study a subset of these variable subset selection methodologies (Lasso, LARS, two variants of Forward Stepwise, and Stagewise Orthogonal Matching Pursuit) and utilize a uniform basis of comparison to establish how well each can be expected to perform under specific sparsity levels and underdeterminedness levels.

2.3 Statisical Solutions to the Model Selection Problem

2.3.1 LARS and LASSO algorithms

Problem (2.2) is similar to the problem solved by LASSO and LARS. The LASSO model selection algorithm solves:

$$(L_t) \qquad min_\beta ||y - X\beta||_2^2 \; s.t. \; ||\beta||_1 \leq t \tag{2.3}$$

for a fixed t. For an appropriate choice of t, Problem (2.3) describes the same problem as equation (2.2), for a fixed choice of λ. The Least Angle Regression (LARS) algorithm provides a stepwise approximation to LASSO.

The LARS algorithm proceeds as follows:

1. Initialize each entry of $\hat{\beta}$, the vector of estimated coefficients, to be equal to zero.

2. Find the predictor X_j most correlated with y,

3. Increase the coefficient $\hat{\beta}_j$ in the direction of the sign of its correlation with y.

4. Calculate residuals $r = y - \hat{y}$ as you go in that direction. Stop when some other predictor X_k has as much correlation with r as X_j. Increase $(\hat{\beta}_j, \hat{\beta}_k)$ in their joint least squares direction, until some other predictor X_m has as much correlation with the residual r.

5. Continue to add further terms into the model until the correlation of the selected variables with the current residuals meets a pre-chosen threshold of correlation.

This suggests a course of investigation: we may be able to cross-apply the equivalence of equations (2.1) and (2.2) to the statistical model selection setting. Perhaps we can observe a threshold in behavior such that for sufficiently sparse models, traditional model selection works, while for more complex models the algorithm's ability to estimate the underlying model closely breaks down. For further details on the LARS and LASSO algorithms, please see [26, 45].

2.3.2 Forward Stepwise Algorithms

We also examine the Forward Stepwise algorithm. Forward Stepwise is a heuristic approach that greedily adds variables to the model, in order of their significance (measured by the absolute size of their t-statistic), up to some preset significance threshold T [48]. A $\sqrt{2\log(p)}$ threshold, or *Bonferroni* threshold, will admit terms provided the absolute value of their t-statistic is not less than $\sqrt{2\log(p)}$, about 3.25 when $p = 200$.

The reason for using the $\sqrt{2\log(p)}$ threshold can be motivated as follows [10]. When the predictor variables are orthonormal a number of papers [13, 21] have studied an approach to coefficient estimation by "soft-thresholding in an orthonormal basis". Suppose we define empirical coefficients via:

$$\tilde{y} = X^T y$$

and define the soft threshold nonlinearity function,

$$\eta_\lambda(y) = sgn(y) \cdot (|y| - \lambda)_+,$$

now we can define thresholded empirical coefficients by

$$\hat{\alpha} = \eta_\lambda(\tilde{y}).$$

The papers just cited show that setting $\lambda = \sqrt{2\log(p)}$ has a number of optimal and near-optimal properties with regard to the mean-squared error. There is a relationship to the Lasso problem formulation seen in Problem (2.2). The soft-thresholding nonlinearity solves the following problem:

$$\eta_\lambda(y) = \frac{1}{2}\, argmin_\xi\ (y-\xi)^2 + \lambda|\xi|. \tag{2.4}$$

Because X is orthogonal, $||y - X\beta||_2 = ||\tilde{y} - \alpha||_2$ and we can rewrite Problem (2.2) as

$$\min_\alpha\ \frac{1}{2}\ \sum_\gamma (\tilde{y}_\gamma - \alpha_\gamma) + \lambda \sum_\gamma |\alpha_\gamma|.$$

Applying equation 2.4 componentwise establishes that the solution $\hat{\alpha}$ is also the solution to Problem (2.2).

Algorithmically, Forward Stepwise can be described as follows:

1. Initialize each entry of $\hat{\beta}$, the vector of estimated coefficients, to be equal to zero.

2. Find the predictor, the j^{th} column of X, that is most correlated with y, and add it into the model. Find the vector of residuals $r = y - X\hat{\beta}$.

3. Repeat Step 2, each time adding to the model the predictor most correlated with the updated residual vector r.

4. Stop adding variables to the model when none of the t-statistics associated with

each of the remaining candidates for entry exceed some preset threshold, T (in our case $T = \sqrt{2\log(p)}$).

We also make a modification to the Stepwise Algorithm – instead of a strict cutoff T we consider threshold solutions using the *False Discovery Rate* [1, 2, 19]. In this case a term is only entered in the model if it has both the largest absolute t-statistic and the associated p-value is less than the FDR statistic, where the FDR statistic is defined as:

$$FDR_{stat} \equiv \frac{q \cdot k}{p},$$

where $q = E(\frac{\{\#FalseDiscoveries\}}{\{\#TotalDiscoveries\}})$, k is the number of terms in the current model, and p is the total number of variables for potential inclusion in the model. The number of terms in the model is not known in advance, and the algorithm will stop adding terms to the model when the p-values of the remaining terms are all greater than the FDR statistic.

In 2006, Abramovich, Benjamini, Donoho, and Johnstone [1] showed the optimal procedure is just such a data-adaptive thresholding scheme driven by the false discovery rate. Their treatment was for orthogonal predictors, and in this case we have non-orthogonal stochastically independent variables, but it is plausible that these good properties carry over to some degree.

Forward Stepwise with FDR thresholding proceeds as follows:

1. Initialize each entry of $\hat{\beta}$, the vector of estimated coefficients, to be equal to zero.

2. Find the predictor, the j^{th} column of X, that is most correlated with y, *and verify the p-value from the t-statistic associated with this predictor is less than*

the FDR statistic, and add it into the model. Find the vector of residuals $r = y - X\hat{\beta}$.

3. Repeat Step 2, each time adding to the model the predictor most correlated with the updated residual vector r, provided its p-value is less than the FDR-statistic.

4. Stop adding variables to the model when the p-values associated with each of the remaining candidates for entry exceed the FDR-statistic.

2.3.3 Stagewise Orthogonal Matching Pursuit

In March of 2006, Donoho et al. [24] introduced *Stagewise Orthogonal Matching Pursuit* (StOMP), modifying the heuristic algorithm Orthogonal Matching Pursuit (OMP) [38, 41], known as Forward Stepwise in the statistical community. The goal of OMP is to sucessively reduce the residual from the estimated model fit to a negligible quantity. The OMP algorithm differs from Classical Forward Stepwise in that the stopping criterion in OMP is that the size of the residual vector falls below a preset level, rather than the size of the remaining variables' t-statistics exceeding a preset threshold. When Forward Stepwise considers adding another term to the model, it finds the t-statistic associated with each potential term and chooses the highest (provided it is above a predetermined threshold-to-enter). In contrast, OMP calculates the correlation of each candidate variable with the residuals from the current model, and selects the variable with the maximum correlation (until the norm of the current model residual vector is less than some preset value). The linear algebra of the reduction in the RSS for each algorithm is the same.

The OMP algorithm proceeds as follows:

1. Begin with an initial residual vector $r_0 = y$,
2. At each stage identify the variable with the greatest correlation with the residual vector,
3. Solve the least squares problem using the selected coordinates (the newly selected variable as well as previously selected variables),
4. Subtract the least squares fit to produce a new residual,
5. Loop through the previous three steps until the resulting residual vector shrinks below some desirable threshold.

StOMP modifies this algorithm in two ways: firstly, it allows any number of variables to enter to model at any stage and secondly, it fixes the number of stages in advance. The algorithm becomes:

1. Begin with an initial residual vector $r_0 = y$,
2. At each stage identify *the variables with amplitude exceeding a pre-chosen threshold*. Specifically, find the correlations of all the candidate variables with the residual vector and include those variables that have correlations above a preset threshold level.
3. Solve the least squares problem using the selected coordinates (the newly selected variable as well as previously selected variables),
4. Subtract the least squares fit to produce a new residual,
5. Loop through the previous three steps for *a fixed number of stages*.

Throughout this thesis, the hard thresholding of the correlations at each stage is done using FDR thresholding. If a variable was not a component of the true underlying model, and yet was chosen by StOMP for inclusion in the final estimated model, we call this a *False Discovery*. FDR thresholding in the StOMP setting attempts to arrange that the number False Discoveries cannot exceed a fixed fraction q of all discoveries, and to make the maximum number of discoveries possible subject to that constraint. As in the Forward Stepwise with FDR thresholding case, q is set to 0.25. Note that we do not need to know the number of nonzeros in the final solution vector, $\hat{\beta}$, in advance.

Analyzing under what circumstances these algorithms perform well, in settings accompanied by noise, is the subject of the following chapters.

Chapter 3

Performance Analysis: Empirical Phase Diagrams

The first step in understanding the circumstances under which each of the algorithms (Lasso, LARS, two variants of Forward Stepwise, and Stagewise Orthogonal Matching Pursuit) performs well, is to fix the number of variables, p, and build an underlying model for a level of underdeterminedness $\delta \equiv \frac{n}{p}$ for $\delta \in [0, 1]$ by varying n, and a sparsity level $\rho \equiv \frac{k}{n}$ for $\rho \in [0, 1]$. Then it is possible to evaluate algorithm performance over this grid in a systematic way.

To apply the Phase Diagram methodology, a course of study is suggested as follows:

1. Generate $X_{n \times p}$, with $X_{ij} \sim N(0, 1)$ and orthogonalize columns,
2. Create $y = X\beta + z$ where β has k nonzeros drawn from $Unif(0, 100)$, and z is drawn iid from $N(0, 1)$,
3. Run full solution path, label the final solution $\hat{\beta}$,

4. Evaluate a "success" property, i.e. compute the indication of a defined property such as $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2} = \gamma$, for γ small,

5. Plot the fraction of successes on the phase plane (δ, ρ) where $\delta = \frac{n}{p}$,$\rho = \frac{k}{n}$.

We specified the underlying model to be $y = X\beta + z$ where y is $n \times 1$, z is $n \times 1$ with entries drawn independently from $N(0, 16)$, β is zero except for k entries drawn from $Unif(0, 100)$, and each $X_{ij} \sim N(0, 1)$ with columns normalized to unit length.

We display the median of the $nRMSE = \frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$ metric for a number of estimated models for each (underdeterminedness, sparsity) combination and a fixed number of variables, p, for the Lasso, LARS, both Forward Stepwise algorithms, and for StOMP.

3.1 The Lasso and LARS Algorithms

3.1.1 Setting tuning parameters

The algorithmic description of LARS found in Section 2.3.1 indicates the use of a threshold at which LARS will cease entering terms into the model. This threshold was chosen to be a function of the noise level, σ, and the sparsity level of the true underlying model, k, and is set to $\frac{\sigma}{k}$. The heuristic argument for this threshold is that it seems sensible to be more restrictive about letting terms enter the model both as the noise level increases and as the true underlying model becomes more sparse (k smaller). Models estimated with lower signal to noise ratios tend to be more easily overfitted so we hope to counteract this.

The Lasso requires a choice of tuning parameter, λ, since it seeks to solve:

$$\min_{\beta} \ \frac{1}{2}||Y - X\beta||_2^2 + \lambda||\beta||_1. \tag{3.1}$$

This is equivalent to Equation (2.3) – the convex optimization formulation – for an appropriate choice of λ. To see this notice in Equation (3.1) as λ increases $||\beta||_1$ decreases and the RSS increases. So we can find a corresponding t^* from Equation (2.3) that yields the same solution, $\hat{\beta}^*$ for a certain λ^*. There will exist an elbowed curve that traces out the RSS for increasing values of $||\beta||_1$, and the slope of the tangent at t^* yields a corresponding λ^* – each problem finding the same solution: $\hat{\beta}^*$. See also [37, 39].

In the Lasso, the simple choice of $\lambda = \frac{\sigma}{k}$ was made for the same reasons as in the implementation of LARS.

Figures 3.1 and 3.2 show the performance of the Lasso and LARS algorithms with the number of variables fixed at 200. These Phase Diagrams have been overlaid with the theoretical phase transition curve from Figure 2.1. Both LARS and LASSO perform similarly, as expected due to the close relationship of the two algorithms, although the $nRMSE$ is higher for the LARS algorithm - $nRMSE_{LARS}$ runs from 0 to 1.1 whereas $nRMSE_{Lasso}$ runs to 1. This is not surprising since LARS is approximating the Lasso algorithm in a stagewise way.

There are several observations to be made on the resulting Phase Diagrams:

- For all the figures, the empirical phase transition occurs remarkably close to the predicted theoretical phase transition (the overlaid curve in black). That is, the true underlying model is estimated with very small error below the curve – in the deep blue region.

- The empirical phase transition is not abrupt, but rather the beginning of a change to gradual estimate degradation.

- As the underlying model becomes less sparse both algorithms have more trouble

finding the correct model. As the number of observations approaches the number of variables, both algorithms find it easier to estimate the true underlying model.

- Over the entire diagram, LARS shows the same model estimation degradation pattern as Lasso, but everywhere has a slightly higher error rate.
- LARS exhibits an increase in $nRMSE$ for less underdeterminded models with low sparsity levels.
- At the very smallest sparsity levels there is a slight increase in $nRMSE$ for both algorithms.

All these observations will be made more concrete subsequently. Before doing so it would be interesting to note the impact of changing the number of variables, and changing the noise level.

3.1.2 Increasing the Number of Variables and Varying the Model Noise Level

In the previous Phase Diagrams, p was fixed at 200. When the appropriate tuning parameter remains set to $\frac{\sigma}{k}$, but p increased to 1000 for both LARS and Lasso, there is little change in the Phase Diagrams: see Figures 3.3 and 3.4. The Phase Diagram shows the same intrinsic pattern of degradation $nRMSE$ but is sharper and clearer. Increasing p emphasizes the gradual change in model estimation error above the phase transition. The phase transition does not seem to have shifted at all with p, compared to Figures 3.1 and 3.2, for either algorithm.

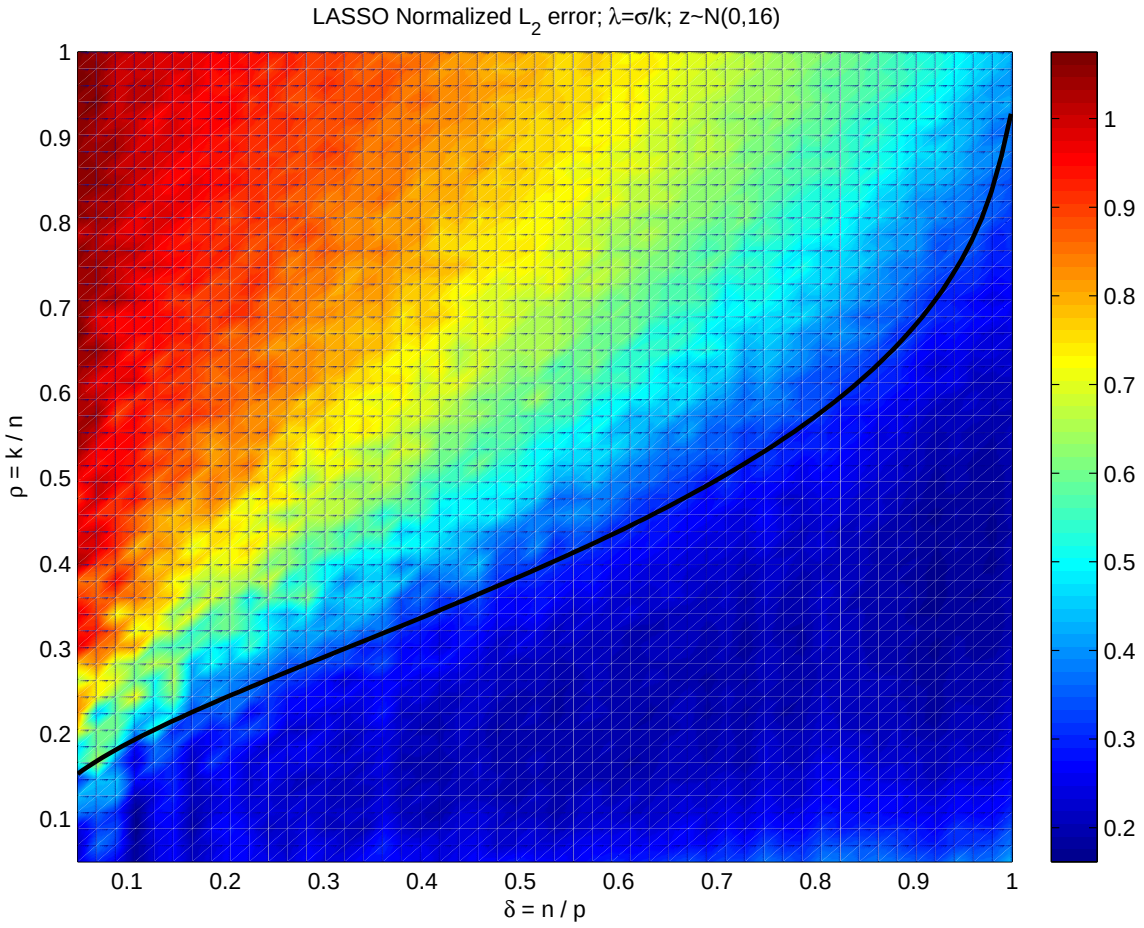

Figure 3.1: Empirical Phase Diagram for the Lasso algorithm with tuning parameter $\lambda = \frac{\sigma}{k}$ (see Section 3.1.1): the $median(nRMSE) = median(\frac{||\hat{\beta}-\beta||_2}{||\beta||_2})$ is displayed at each point on the plane, with color denoting its value (blue denotes $nRMSE$ close to 0, red close to 1). The number of variables, p, is fixed at 200. The median is taken over 40 replications. The theoretical phase transition curve from Figure 2.1 is overlaid in black. The empirical phase transition is close to the theoretical prediction, even finding the underlying model with low noise in regions above the theoretical curve.

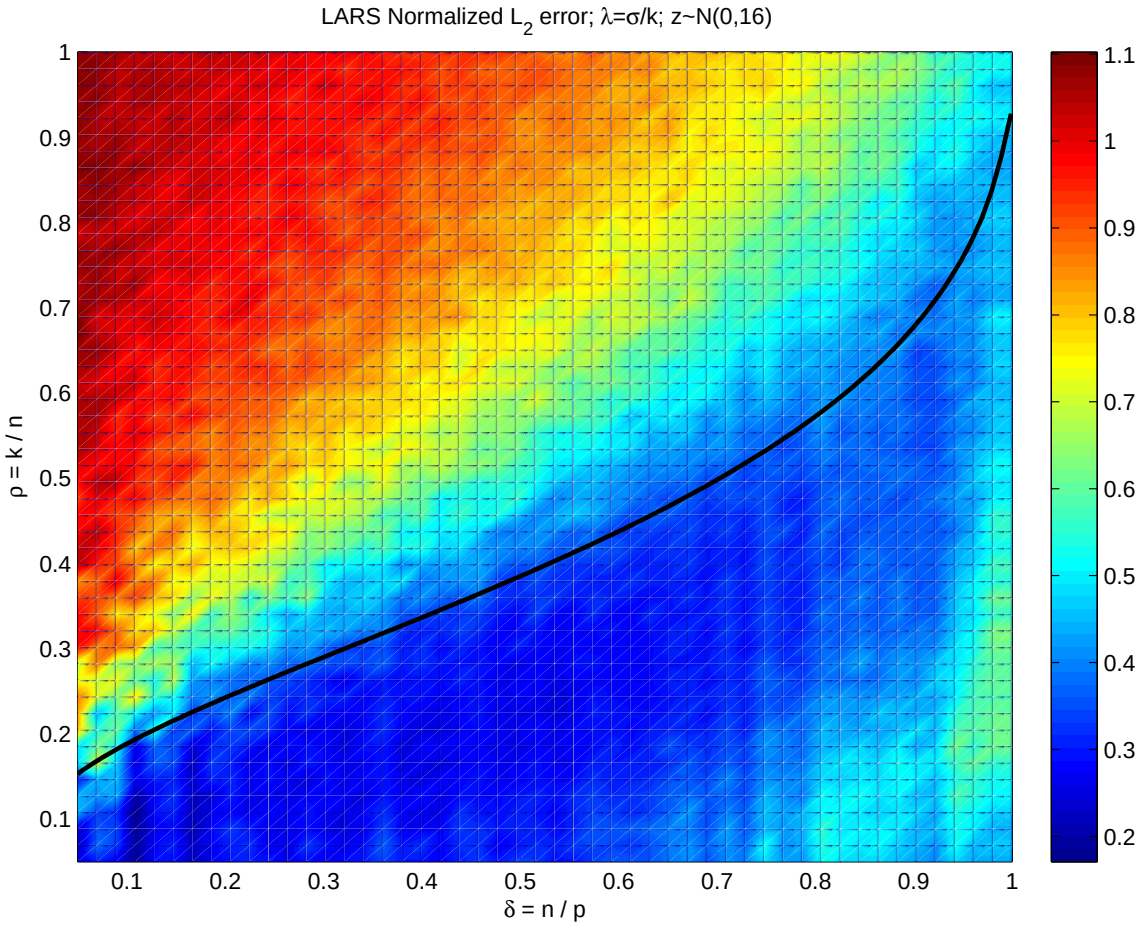

Figure 3.2: Empirical Phase Diagram for LARS algorithm with stopping criterion $\frac{\sigma}{k}$ (see Section 2.3.1): the $median(nRMSE) = median(\frac{||\hat{\beta}-\beta||_2}{||\beta||_2})$ is displayed at each point on the plane, with color denoting its value (blue denotes $nRMSE$ close to 0, red close to 1). The number of variables, p, is fixed at 200. The median is taken over 40 replications. The theoretical phase transition curve from Figure 2.1 is overlaid in black. The empirical phase transition is close to the theoretical prediction, even finding the underlying model with low noise in regions above the theoretical curve.

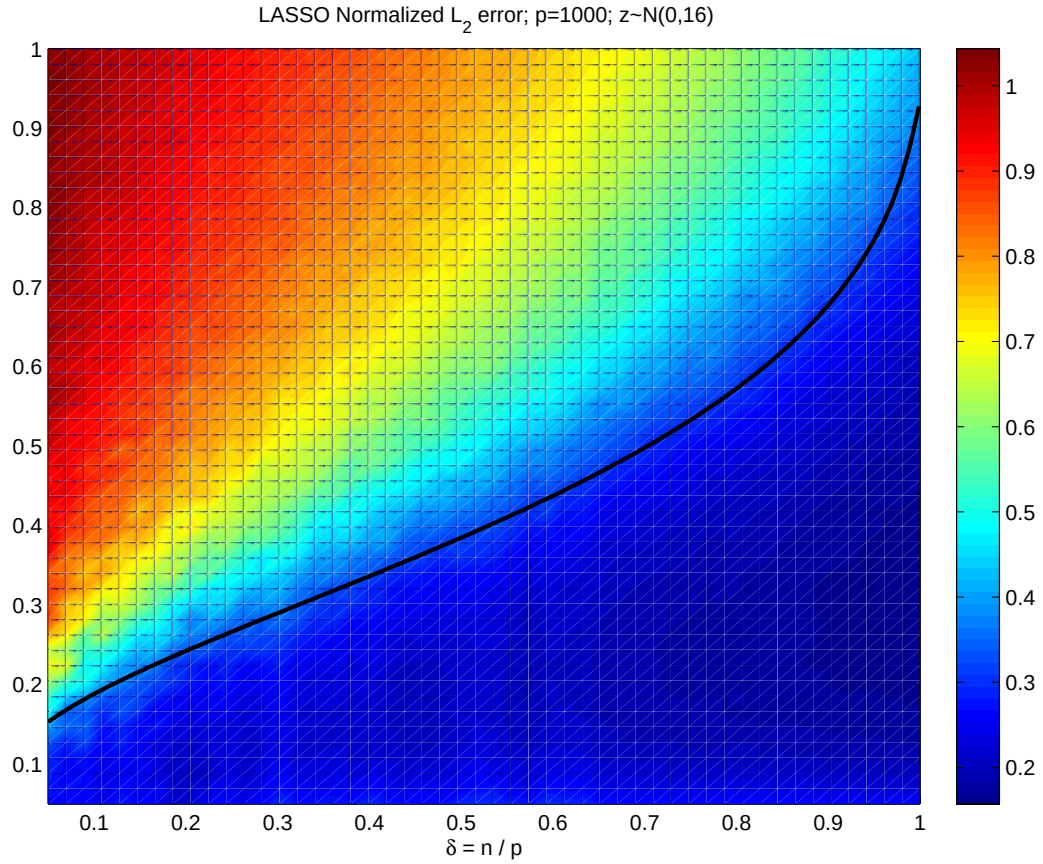

Figure 3.3: Empirical Phase Diagram for Lasso algorithm: the number of variables, p, fixed at 1000, $\lambda = \frac{\sigma}{k}$. Each color indicates a different median normalized ℓ_2 error of the coefficients, $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$, over 40 realizations. The theoretical phase transition curve from Figure 2.1 is overlaid in black. With the increase in p from Figure 3.1, the transition is sharper, but substantially unchanged.

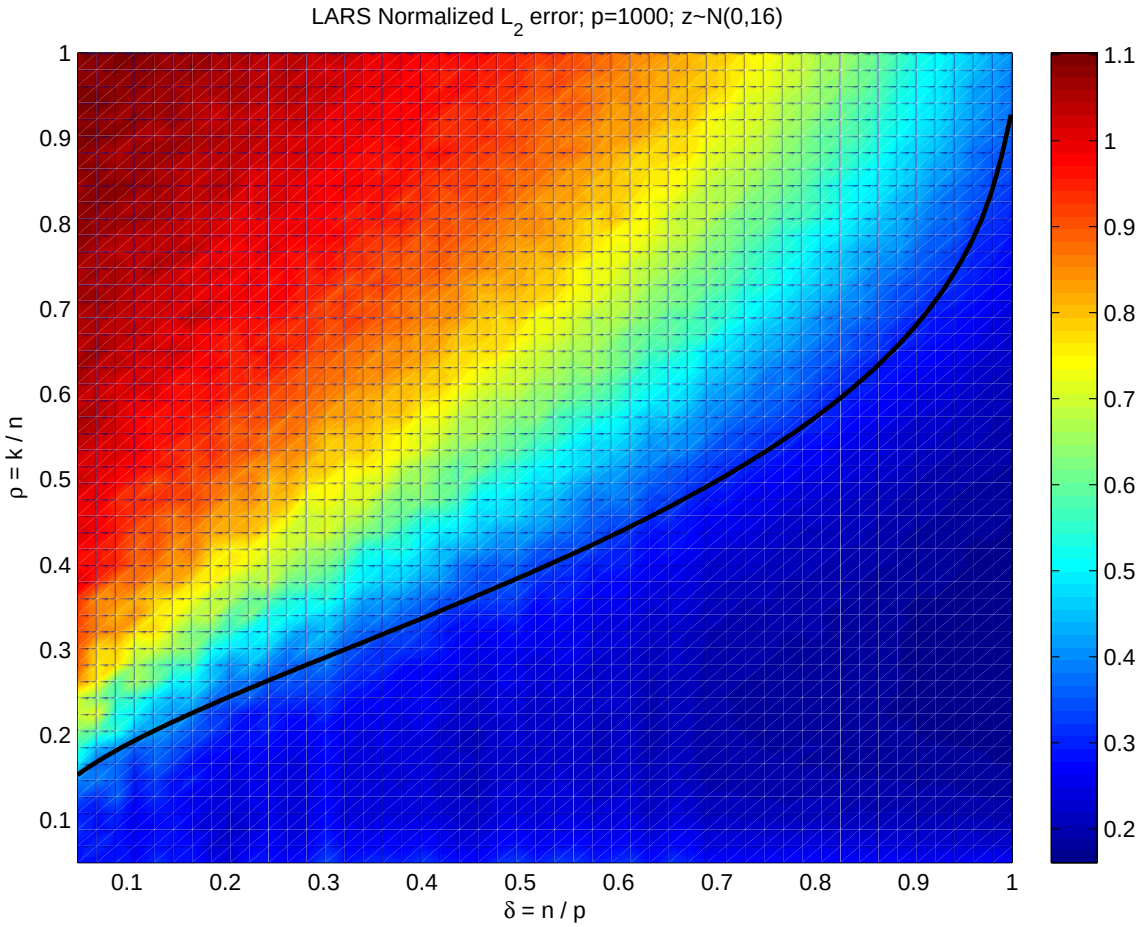

Figure 3.4: Empirical Phase Diagram for LARS algorithm: the number of variables, p, fixed at 1000, and correlation threshold remains $\frac{\sigma}{k}$. Each color indicates a different median normalized ℓ_2 error of the coefficients, $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$, over 40 realizations. The theoretical phase transition curve from Figure 2.1 is overlaid in black. With the increase in p from Figure 3.2, the transition sharpens, but remains substantially unchanged. The phase transition still remains largely above the theoretical prediction.

In the study of algorithm performance, it is crucial to understand how model selection ability is affected by changing the noise level σ in the underlying model. By examining a δ-slice of the Phase Diagram and varying σ we can directly observe the resulting changes in $nRMSE$ for both LARS and the Lasso, over different sparsity levels.

With the number of variables fixed at 200, I choose to set $\delta = \frac{1}{2}$, and vary σ over $\{0, 1, 2, 4, 6, 9, 12, 16\}$. Figures 3.5 and 3.6 show the $nRMSE$ levels across sparsity levels $\rho = \frac{k}{n} \in [0, 1]$, for the Lasso and LARS respectively.

These diagrams display several interesting properties of the algorithms:

- For a fixed ρ, there seems to be a approximately linear relationship between σ and $nRMSE$, for both algorithms,
- Lasso seems to break down slightly earlier than LARS,
- Across ρ, $nRMSE$ flattens as σ increases,
- At the lowest sparsity levels the Lasso exhibits a sharp degradation in its ability to estimate the underlying model well, for $\sigma > 4$.
- When there is no noise in the underlying model, both algorithms exhibit perfect model recovery for small ρ, then beyond a specific ρ_0 the model is estimated with dramatically increased $nRMSE$,
- In the region where the model is estimated accurately, $nRMSE_{LARS}$ is higher than $nRMSE_{Lasso}$ at all noise levels except $\sigma = 0$, and $nRMSE_{LARS}$ increases more with increases in the noise level,
- When the model is not estimated correctly (at larger ρ – less sparse models),

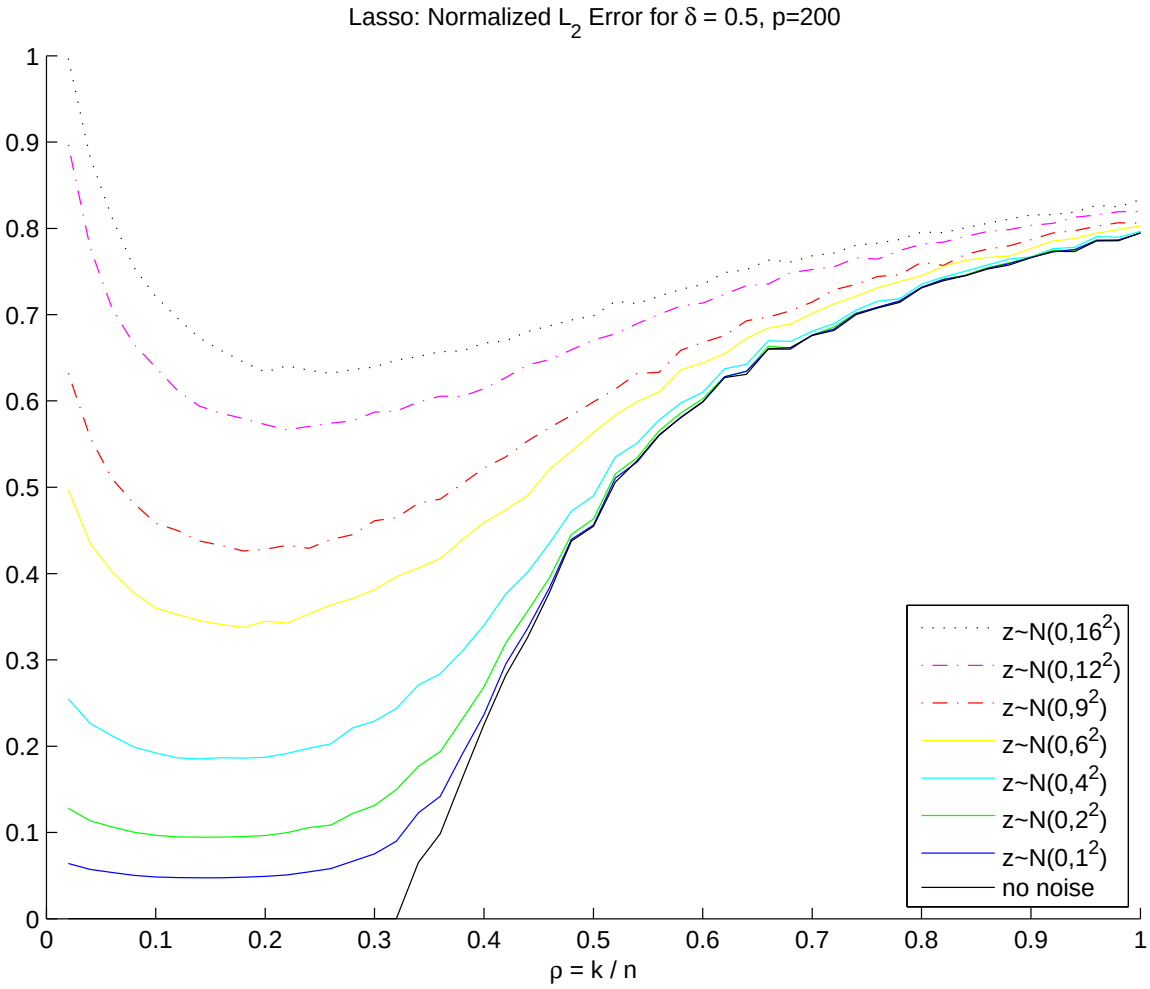

Figure 3.5: Lasso noise levels, $p = 200$, $\lambda = \frac{\sigma}{k}$, and the number of replications at each point is 1000. At low noise levels there is a point at which the equivalence of the ℓ_0 and ℓ_1 problems breaks down, but at higher noise levels the transition is gradual. In the breakdown region, $nRMSE$ levels remain at approximately 0.8. For extremely sparse models, there is a sharp increase in $nRMSE$. When a very small proportion of the variables are in the true model, $nRMSE$ levels increase for all noise levels.

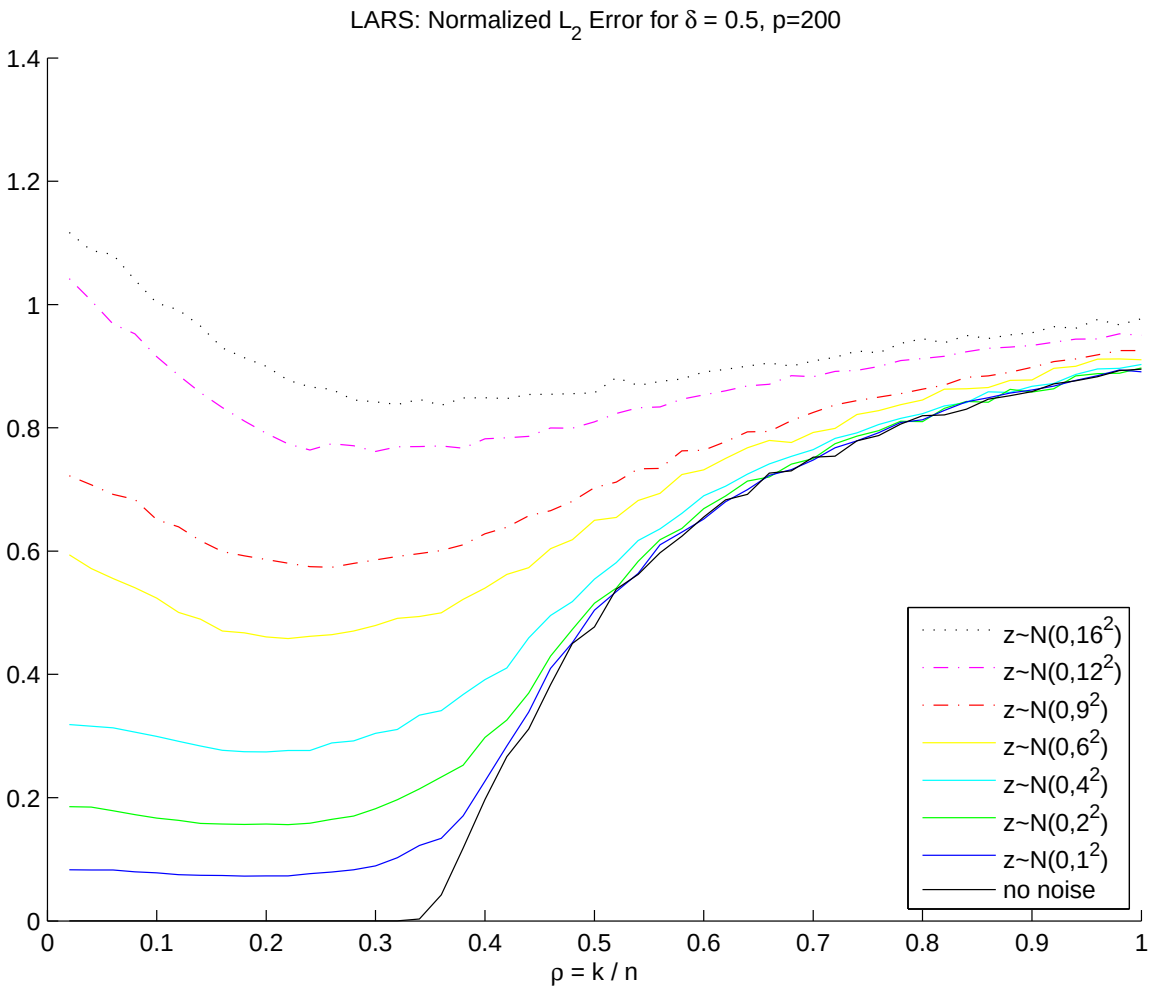

Figure 3.6: LARS noise levels, $p = 200$, the correlation threshold remains $\frac{\sigma}{k}$, and the number of replications at each point is 1000. LARS exhibits the same breakdown characteristics at the Lasso in Figure 3.5: a smoothing of the breakdown point with higher noise levels. In the region where the underlying model is not recovered, $nRMSE$ seems to converge to about 1.

$nRMSE_{LARS}$ seems to converge to close 1, whereas $nRMSE_{Lasso}$ appears to be converging close to 0.8.

3.2 Forward Stepwise

The Classical Forward Stepwise and Forward Stepwise with FDR Thresholding algorithms were run in the same test setting as the Lasso and LARS.

The threshold-to-enter for Classical Stepwise was set at $\sqrt{2\log(p)}$, implying that the absolute value of the t-statistic for a variable under consideration for inclusion in the model must be greater than $\sqrt{2\log(p)}$. If the number of variables is fixed at 200, the threshold-to-enter value is 3.25.

Forward Stepwise with FDR Thresholding requires setting the False Discovery rate parameter q. Based on work by Donoho and Jin [19], q was chosen to be 0.25.

Each Forward Stepwise algorithm produces a strikingly different phase transition diagram. Figures 3.7 and 3.8 each show a clear boundary below which the algorithm recovers the correct model, and above which it does not. Surprisingly, classical Forward Stepwise appears nearly-everywhere worse than either the Lasso and LARS, with a constant sparsity level beyond which it fails dramatically, regardless of the level of underdeterminedness. Stepwise with FDR Thresholding produces another interesting result: the algorithm matches the predicted breakdown nearly perfectly (the theoretical curve from Figure 2.1 has been overlaid on the Stepwise with FDR thresholding Phase Diagram). The FDR threshold-to-enter depends on the number of terms currently in the model which implies that as the sparsity level increases so does the threshold, and more terms are included. More precisely:

- Both Forward Stepwise algorithms exhibit a sharp boundary below which they

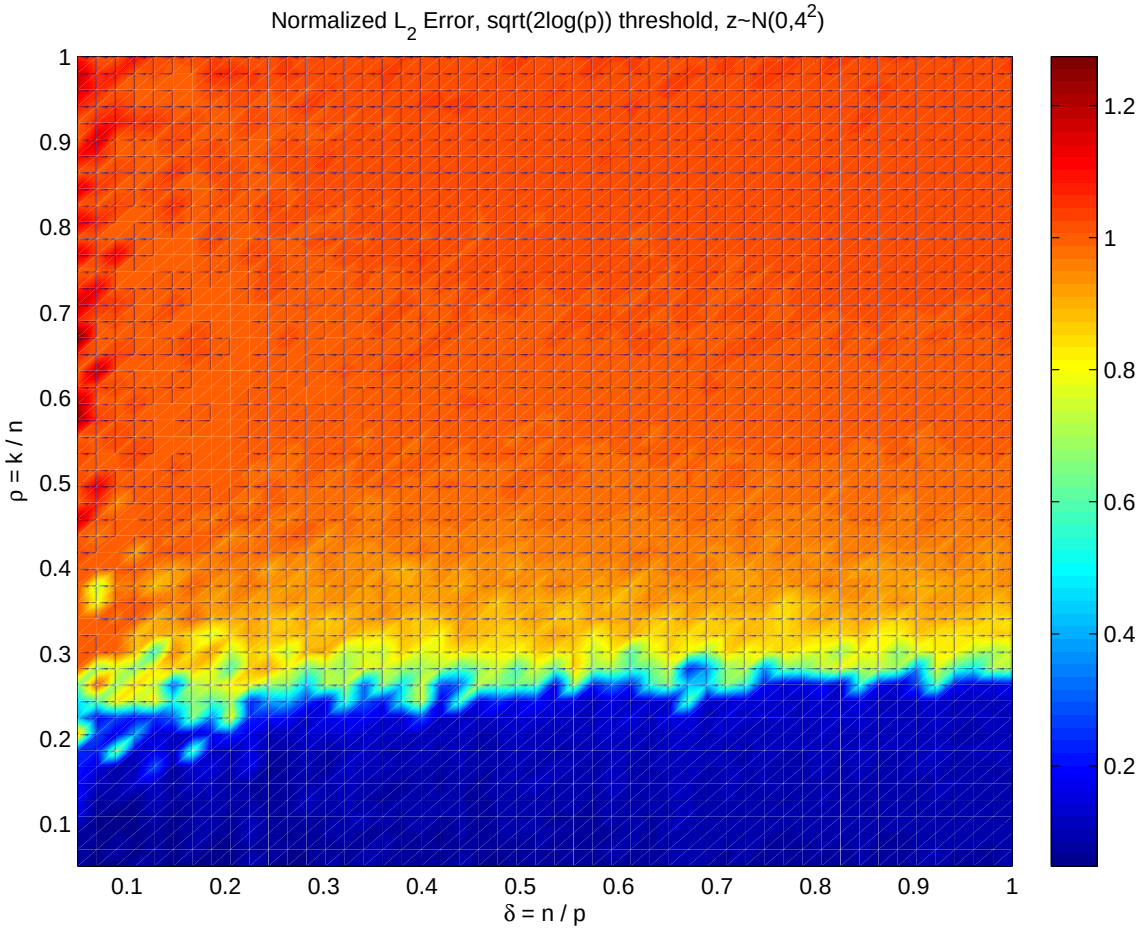

Figure 3.7: Empirical Phase Diagram for Forward Stepwise: illustrates region where the underlying sparse model is recovered using the Forward Stepwise Algorithm, with the number of variables, p, fixed at 200 and model noise $z \sim N(0, 16)$. Variables were greedily added to the model until no remaining t-statistic was greater than $\sqrt{2\log(p)}$ in absolute value. The phase transition is striking: there is a very sharp dropoff below which the algorithm recovers the model with near zero error, and above which the model is unrecoverable. As with the theoretical phase transition diagram in Figure 2.1, along the x-axis the level of underdeterminedness decreases, and along the y-axis the level of sparsity of the underlying model increases. Each color indicates a different median normalized ℓ_2 error of the coefficients $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$ over 30 realizations.

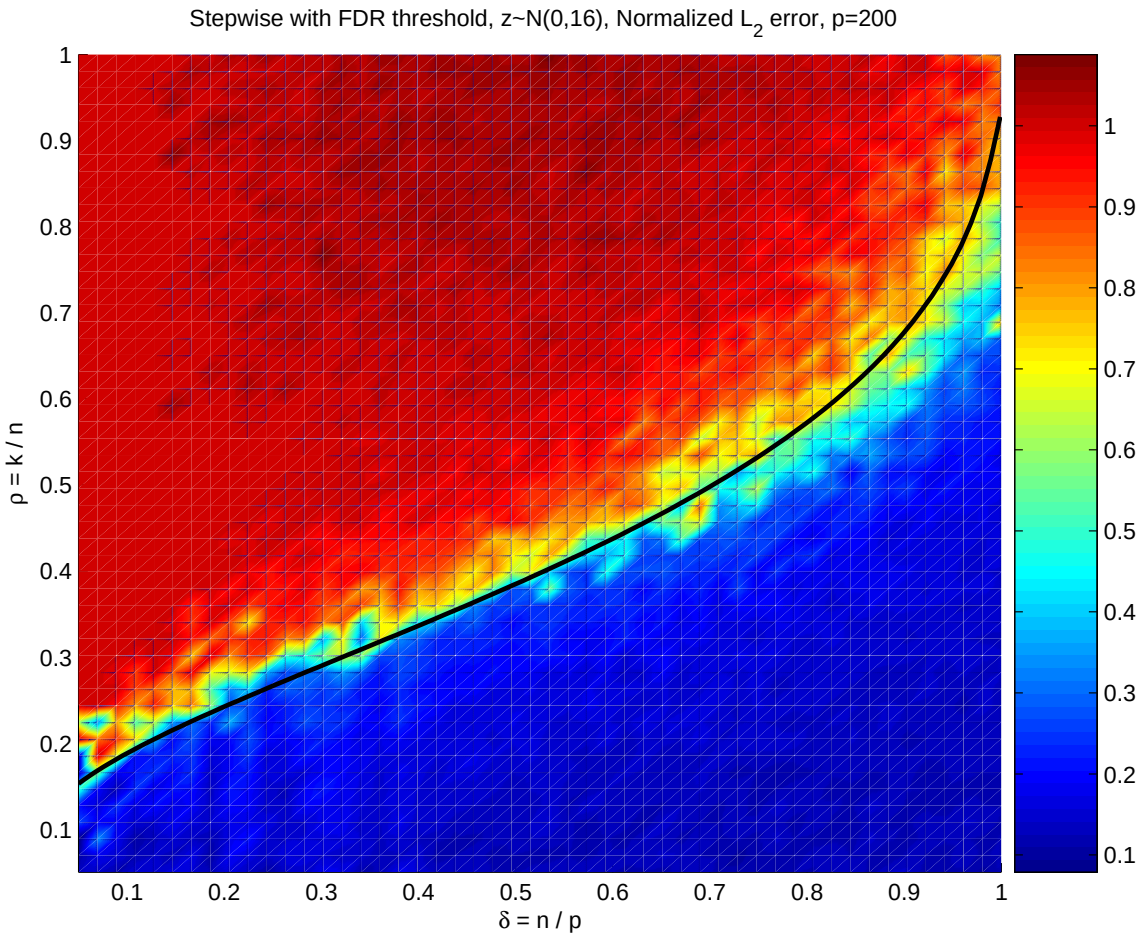

Figure 3.8: Empirical Phase diagram for Forward Stepwise-FDR Thresholding: each color indicates a different median normalized ℓ_2 error of the coefficients $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$ over 10 realizations. A term is added to the model if it has the largest t-statistic of all candidate terms and its corresponding p-value is less than the FDR value, defined as (.25×(number of terms currently in the model)/(total number of variables)). The number of variables is fixed at 200, and model noise $z \sim N(0, 16)$. This version of Forward Stepwise has a Phase Transition similar to the theoretical curve from Figure 2.1 (overlaid) rather than the steep dropoff of classical Forward Stepwise seen in Figure 3.7.

recover the underlying model with nearly zero $nRMSE$, and above which $nRMSE$ is approximately 1 – the transition is abrupt;

- Classical Forward Stepwise has a phase transition at a fixed sparsity level – below which it estimates the underlying model with great precision, above which is does not – *regardless of the level of indeterminacy*;

- Stepwise with FDR Thresholding has a sharp phase transition that matches the theoretical curve nearly perfectly.

3.2.1 Increasing the Number of Variables and Varying the Model Noise Level

For both previous Forward Stepwise Phase Diagrams, Figures 3.7 and 3.8, the number of variables was fixed at 200. Now, we can examine changes in $nRMSE$ by increasing p to 500, and varying σ through $\{0, 1, 2, 4, 6, 9, 12, 16\}$. As in the Lasso and LARS analysis, attention is restricted to a δ-slice of the phase diagram at $\delta = \frac{1}{2}$. The first figure, Figure 3.9, shows the $nRMSE$ for Classical Forward Stepwise while keeping p at 200 and varying σ. Figure 3.10 repeats the analysis with $p = 500$.

A number of observations on these two δ-slice diagrams leap out:

- the transition point from accurate model estimation to inaccurate is very sharp for both values of p,

- As p increases, Classical Stepwise fails to recover the underlying model at increasingly small sparsity levels i.e. when p increases from 200 to 500 the algorithm breaks down at a lower value of ρ (when $\sigma = 0$, this value is 0.28 for $p = 200$ and 0.25 for $p = 500$),

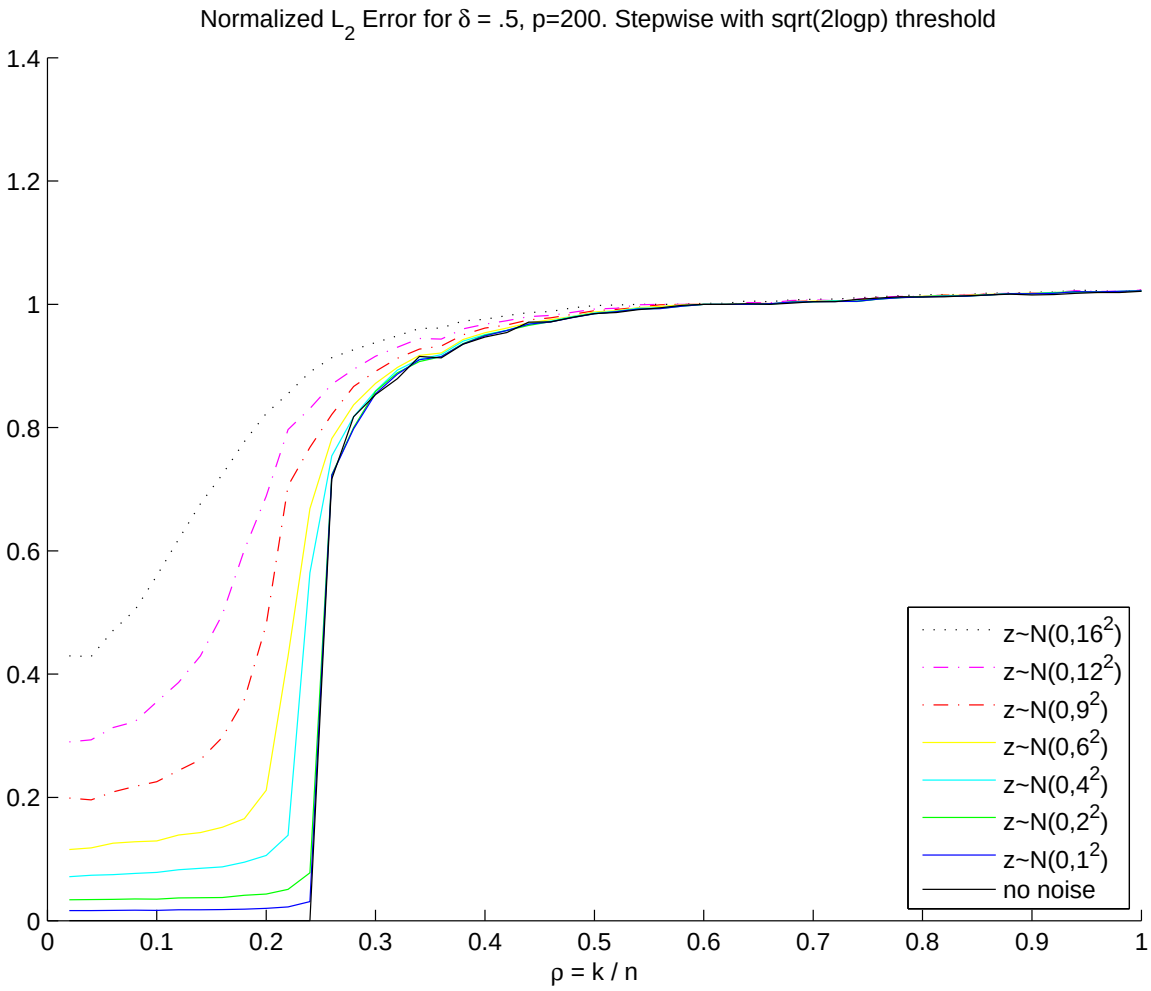

Figure 3.9: Vertical slice of the Empirical Phase Diagram for Forward Stepwise (Figure 3.7): with varying noise levels, $\delta = \frac{n}{p}$ fixed at .5 and the number of variables fixed at 200. Each increase in model noise (from no noise to $N(0, 16^2)$), causes the algorithm to break down at higher sparsity levels. The median of the normalized ℓ_2 error for the coefficient estimates is shown, over 1000 replications.

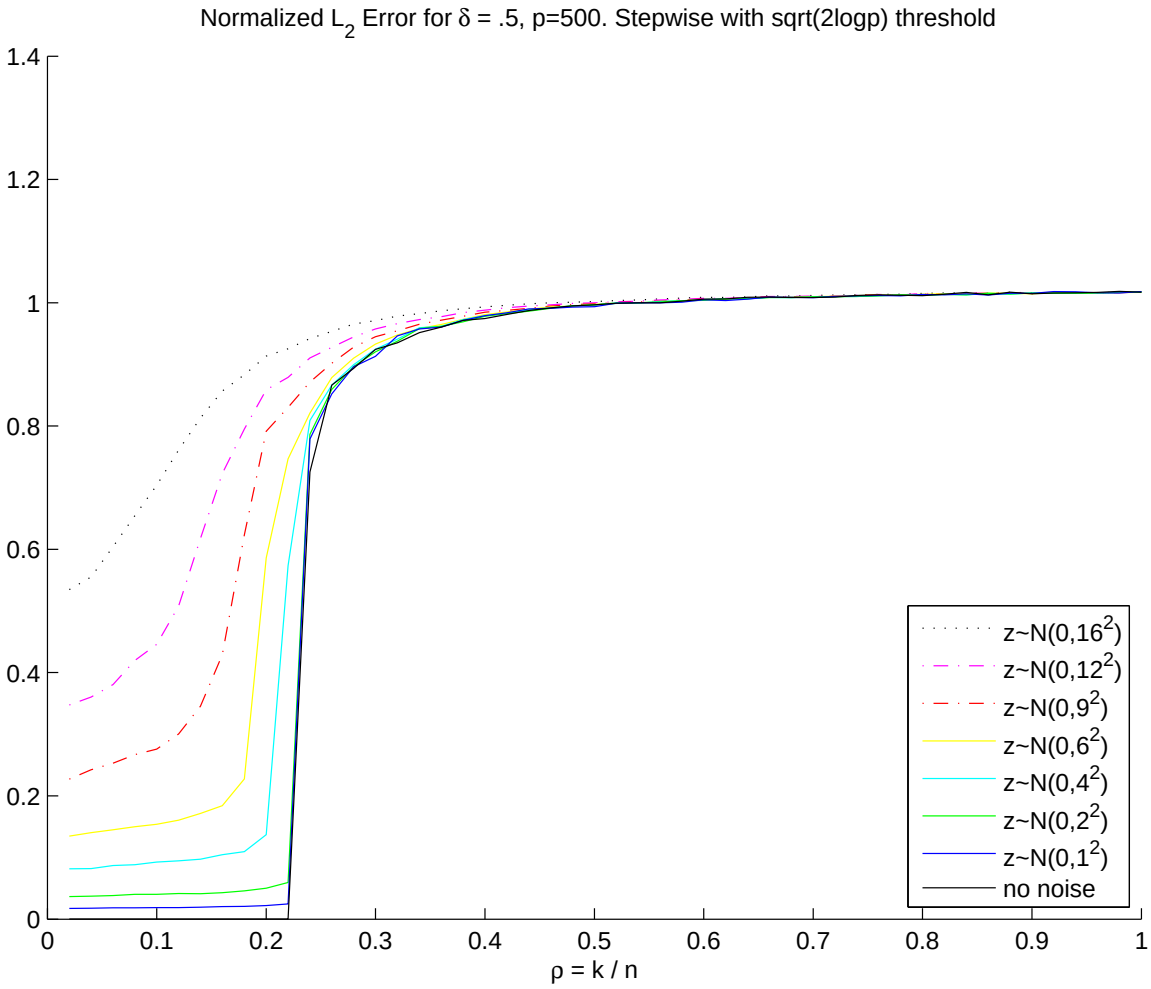

Figure 3.10: As Figure 3.9 except $p = 500$. As the noise level is increased from $\sigma = 0$ to $\sigma = 16$ the breakdown point occurs earlier, i.e. for sparser and sparser models. Notice also that with the increase in p from 200 to 500 the breakdown point, for the same level of noise, occurs at sparser underlying models.

- as ρ increases, the degradation to model recovery is always non-decreasing, regardless of p and σ,
- for either value of p, in the region where the model is being correctly recovered, $nRMSE$ increases proportional to the increase in σ,
- for either value of p, when the model is not correctly recovered, noise level makes no difference – and $nRMSE$ converges quickly to 1 for all noise levels.

This same analysis can be extended to the case of Forward Stepwise with FDR thresholding. Figures 3.11 and 3.12 do just that: p is 200, or 500 respectively, $q = 0.25$, and σ varies through $\{0, 1, 2, 4, 6, 9, 12, 16\}$.

A number of observations can be made about these two Phase Diagrams:

- as in the case of Classical Forward Stepwise, a sharp transition point is evident for each noise level, although at lower noise levels FDR thresholding induces a slightly more gradual transition,
- the ρ at which the underlying model ceases to be recovered increases with p,
- the $nRMSE$ in the region where the model is recovered is unaffected by p, for a fixed noise level,
- as ρ increases the degradation to model recovery is always non-decreasing, regardless of p and σ,
- for either value of p, in the region where the model is being correctly recovered, $nRMSE$ increases proportional to the increase in σ, and FDR thresholding causes a larger increase with ρ than Classical Forward Stepwise,

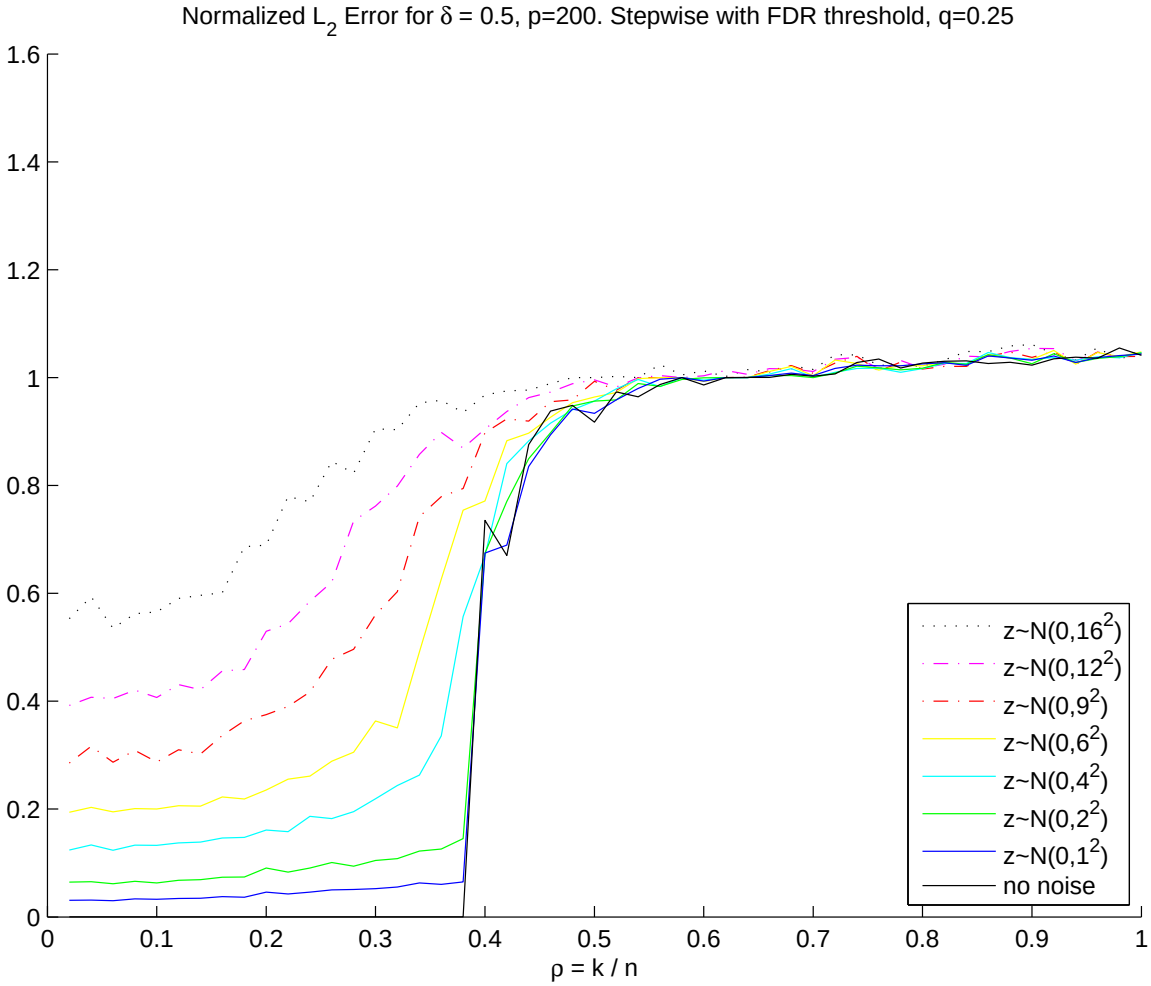

Figure 3.11: Forward Stepwise with FDR thresholding, varying noise levels, $q = 0.25$, $p = 200$, and the number of replications at each point is 100. There is sharp phase transition evident, especially at low noise levels. The $nRMSE$ seems roughly constant, for a given noise level, before the phase transition, and roughly equal to 1 after the phase transition.

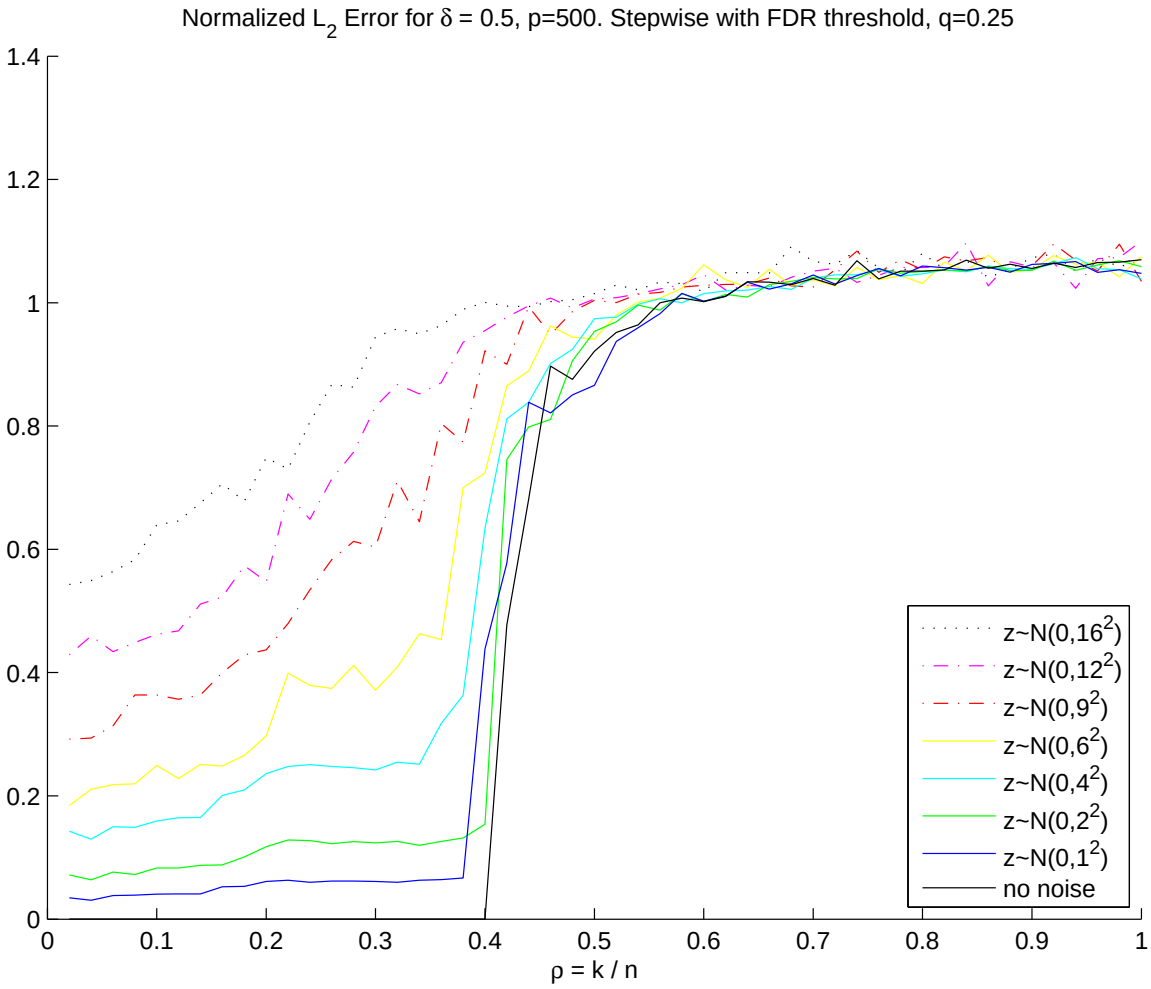

Figure 3.12: As Figure 3.11 except $p = 500$. Increasing p has caused the phase transition to occur at a lower sparsity level, ie. when ρ is larger. As in Figure 3.11, at low noise levels the phase transition is very sharp. After the phase transition the $nRMSE$ seems to be approximately 1.

- for either value of p, when the model is not correctly recovered, noise level makes no difference – and $nRMSE$ converges quickly to 1 for all noise levels.

Because of the computational complexity of this algorithm, we were able to conduct only 100 replications for each ρ, and this accounts for the 'noisiness' in the δ-slices for the Forward Stepwise with FDR Thresholding plots (Figures 3.11 and 3.12) relative to the Classical Stepwise plots (Figures 3.9 and 3.10).

3.3 Stagewise Orthogonal Matching Pursuit

StOMP was implemented in the same test settings as the Lasso, LARS, and the ForwardStepwise algorithms: generate underlying model, $y = X\beta + z$ where β is sparse i.e. has $k < p$ nonzeros, find $\hat{\beta}$, and evaluate performance via $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$. In this case the StOMP algorithm, as described in Section 2.3, is used to find $\hat{\beta}$.

A False Discovery Rate threshold was chosen with parameter $q = 0.25$ [19]. The *False Discovery Rate* is defined as the proportion of variables included in the final model that were not present in the original underlying model. *False Discovery Rate Thresholding* is an attempt to ensure that the number of False Discoveries in the final model does not exceed the fraction q – in our case no more than 25% of the variables chosen for inclusion in the final model will have not been present in the underlying model. This is implemented in SparseLab and is available for download (see Chapter 6).

In Figure 3.13 p was set to 200, consistent with the previous algorithms' phase diagrams. The number of stages was fixed at 10 for all StOMP calculations in this thesis. StOMP does not recover the underlying model for as great a sparsity level as LARS or LASSO (i.e. breaks down earlier), but it is a marked improvement over

Forward Stepwise. The algorithm has trouble when n is very small – when $n = 20$ and thus $\delta = .1$ the method is not working; it requires n to be large.

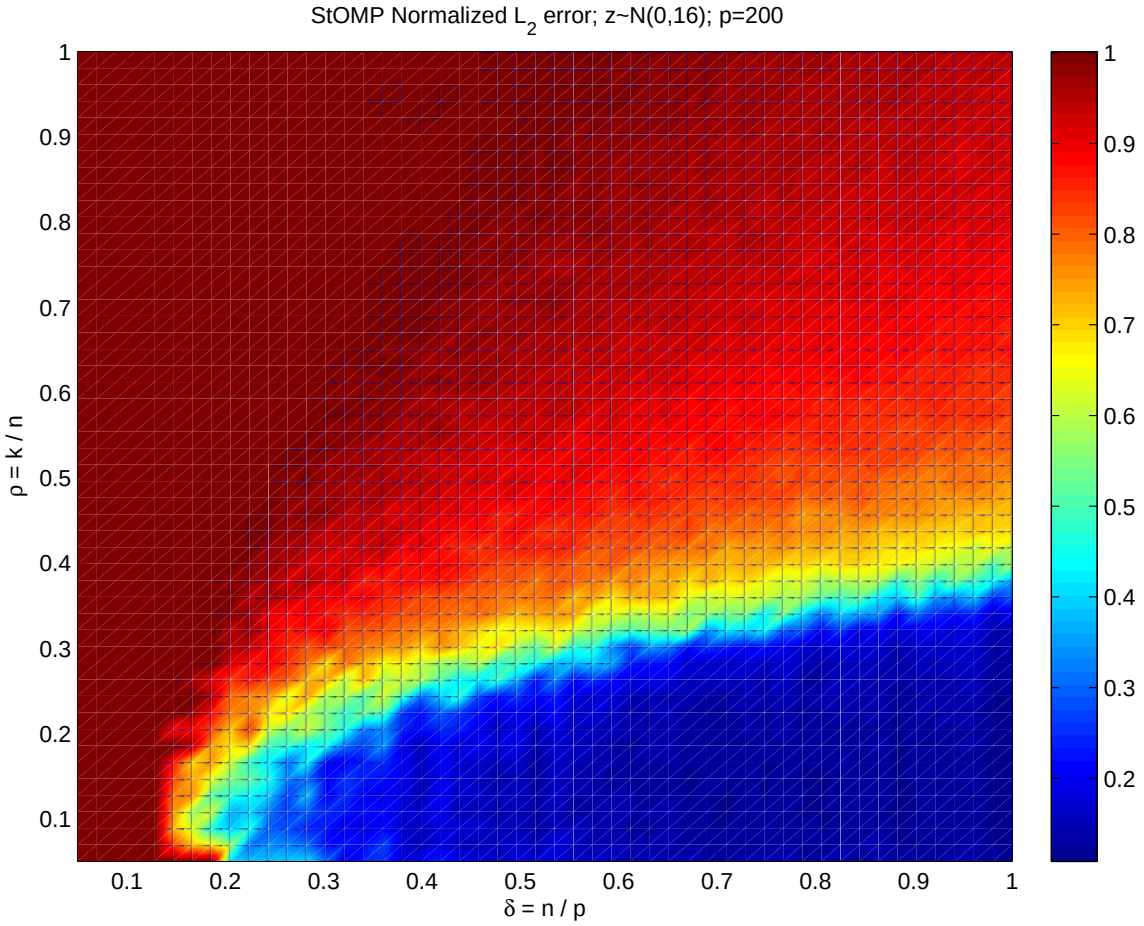

Figure 3.13: StOMP Phase Diagram with the FDR parameter set to $q = 0.25$, $p = 200$, and the number of replications at each point is 100. Notice the algorithm breakdown when n is very small. StOMP does not recover the underlying model as well as LARS/LASSO for higher sparsity levels.

3.3.1 Increasing the Number of Variables and Varying the Model Noise Level

To compare to the previous figures with $p = 200$, p was set to 1600 in Figure 3.14. StOMP now recovers the underlying model correctly for all levels of underdeterminedness, although it breaks down earlier than the Lasso or LARS cases. When the number of variables is very large compared to the number of observations, StOMP only successfully recovers the model when the underlying model is very sparse. As the model becomes less underdetermined, fewer sparse models are correctly recovered.

Figures 3.13 and 3.14 have certain important features:

- the transition from accurate model recovery to inaccurate model recovery is very sharp – and becomes sharper as p increases – although $nRMSE$ continues to gradually increase with ρ after the transition,
- for small values of δ increasing p allowed the underlying model to be recovered in that region[1],
- although it becomes more abrupt, the transition for large δ does not shift with p,
- the region in which the model is correctly recovered is everywhere smaller than any of the other algorithms studied, with the exception of Classical Forward Stepwise.

As in the analysis of the previous algorithms, I set $\delta = \frac{1}{2}$ and varied the noise level, σ, through $\{0, 1, 2, 4, 6, 9, 12, 16\}$. The number of variables remained fixed at

[1] subsequent simulations showed the transition boundary to increase leftward gradually with p, and the region of correct model estimation did not change beyond $p = 1600$

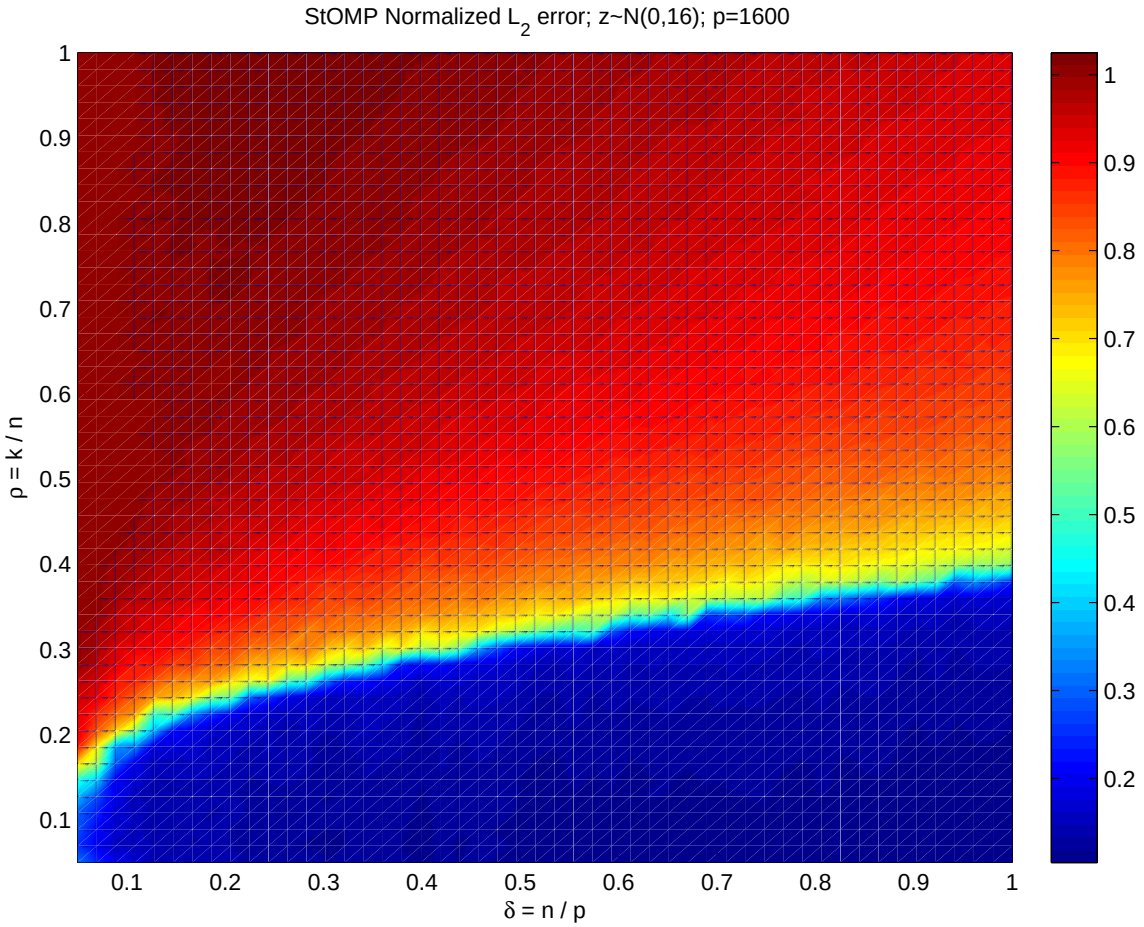

Figure 3.14: StOMP Phase Diagram with the FDR parameter set to $q = 0.25$, $p = 1600$; the number of replications at each point is 100. With a much greater p, StOMP is now able to recover the underlying model for even very small n, although it still does not recover the underlying model as well as LARS or Lasso for higher sparsity levels.

1600 and the False Discovery rate set to $q = 0.25$. Figure 3.15 plots $nRMSE_{StOMP}$ at all sparsity levels for each of these noise levels.

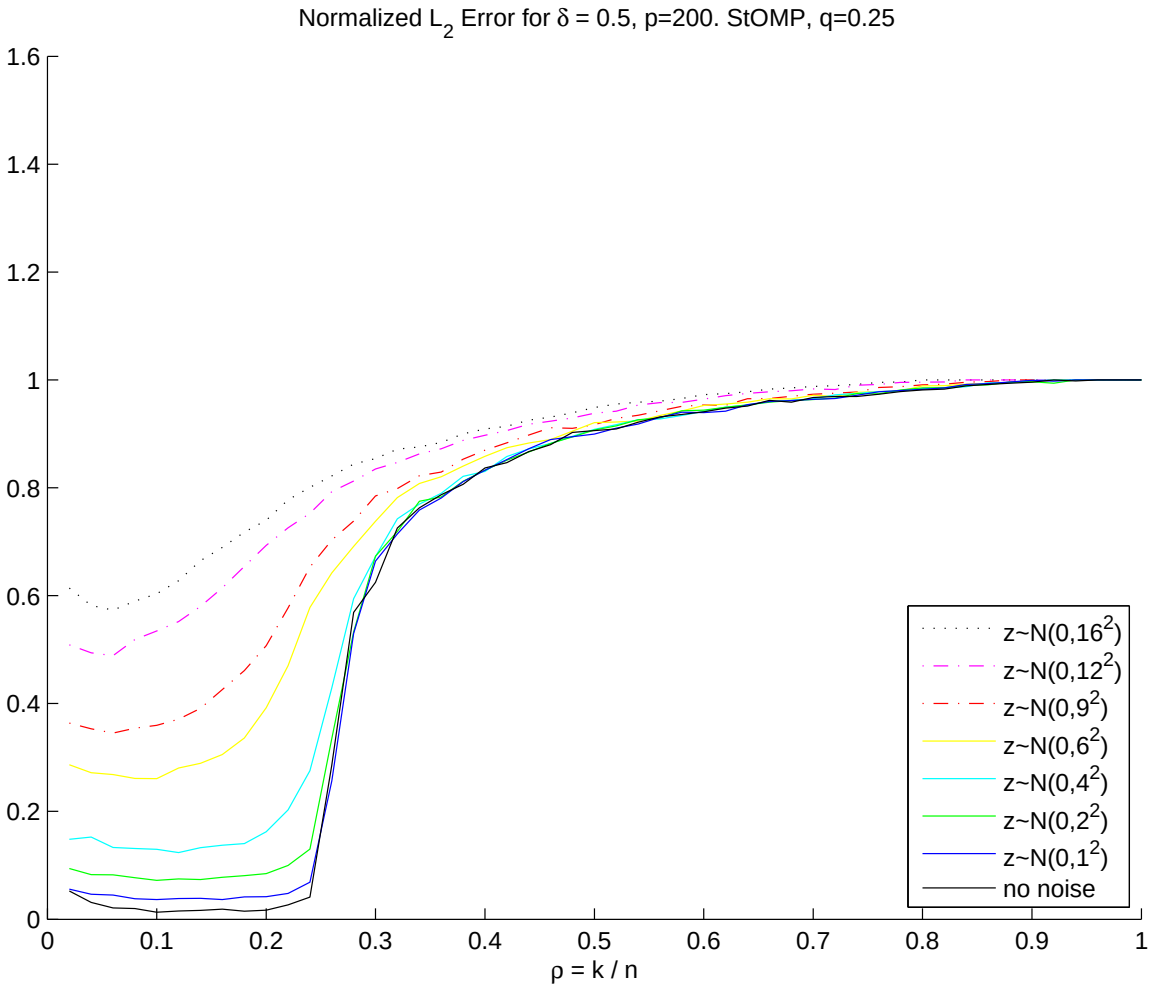

Figure 3.15: StOMP at several noise levels, $q = 0.25$, $p = 1600$; the number of replications at each point is 1000. The phase transition occurs at a lower value of ρ that for Forward Stepwise with FDR Threshold (Figure 3.12). The $nRMSE$ is approximately 1 in the region of model breakdown.

There are a number of interesting observations that can be made about Figure 3.15:

- like the two Forward Stepwise algorithms, StOMP abruptly ceases to recover

the underlying model at some critical point, then $nRMSE$ converges rapidly to 1;

- this critical point depends on the noise level: as σ increases it becomes more difficult to estimate the model with low error;
- there is a very slight increase in $nRMSE$ for the lowest sparsity levels - this is reminiscent of the Lasso and LARS algorithms;
- $nRMSE$ increases proportionally to the noise level;
- the model is not perfectly recovered in the noiseless case.

The next chapter concretizes these observations in a systematic way for all the algorithms.

Chapter 4

Algorithm Breakdown Point

The "breakdown" in the algorithm error level occurs where the algorithm stops recovering the correct underlying model i.e. when the normalized root MSE, defined as $\frac{||\hat{\beta}-\beta||_2}{||\beta||_2}$, exceeds a threshold level γ. This chapter studies the breakdown point in detail.

There exists a known function $\rho_{\ell_1}(\delta)$ [15, 14], that describes the breakdown point of equivalence between the solutions of the ℓ_1 and ℓ_0 problem formulations (Equations 2.1 and 2.2 respectively) in the noiseless cases.

The breakdown points for each algorithm were isolated as follows.

Definition 5 δ-breakdown point *For a fixed* δ, $\rho_k = \max_\rho \frac{\partial(nRMSE(\delta,\rho))}{\partial\rho}$.

We can examine each section of the Phase Diagram, with the level of underdeterminedness, δ, fixed at $\frac{1}{2}$, in turn.

4.1 When the ℓ_1 and ℓ_0 Formulations are Equivalent

Figures 4.1 through 4.5 show the $nRMSE$ up until the δ-breakdown point, with $p = 200$ and $p = 1000$ for StOMP. Based on the results in the previous chapter, for all algorithms except StOMP, increasing p past 200 seems only to sharpen the phase transition, not substantially move it. We can therefore regard the cases with $p = 200$ as representative. (Note: the phase transition for StOMP does change with increasing p, until approximately $p = 1000$).

The figures illustrate several phenomena: error levels increase as the underlying model becomes less sparse (with the exception of Lasso and LARS at the very lowest sparsity levels), and the breakdown point occurs earlier (at a lower sparsity level) with increases in σ, and the $nRMSE$ appears to increase proportionally with σ.

The increase in the error levels associated with a less sparse model is predictable. A useful experiment is to consider the existence of an all-knowing oracle, in particular with knowledge of the correct underlying model in each instance. We expect that when the algorithm estimates the model with small error (before the "breakdown"), the error will follow the *oracle MSE* [20]:

$$tr((X'X)^{-1})\sigma^2 \tag{4.1}$$

where X is an $n \times k$ matrix with each column an included variable in the final model. Setting the determinedness level to $\delta = \frac{1}{2}$ we can find the baseline normalized root MSE by simply solving directly for the true model, using ordinary least squares.

For the first 30 experiments, oracle $nRMSE$ results are displayed in Figure 4.6 for $p = 200$ and 4.7 for $p = 500$.

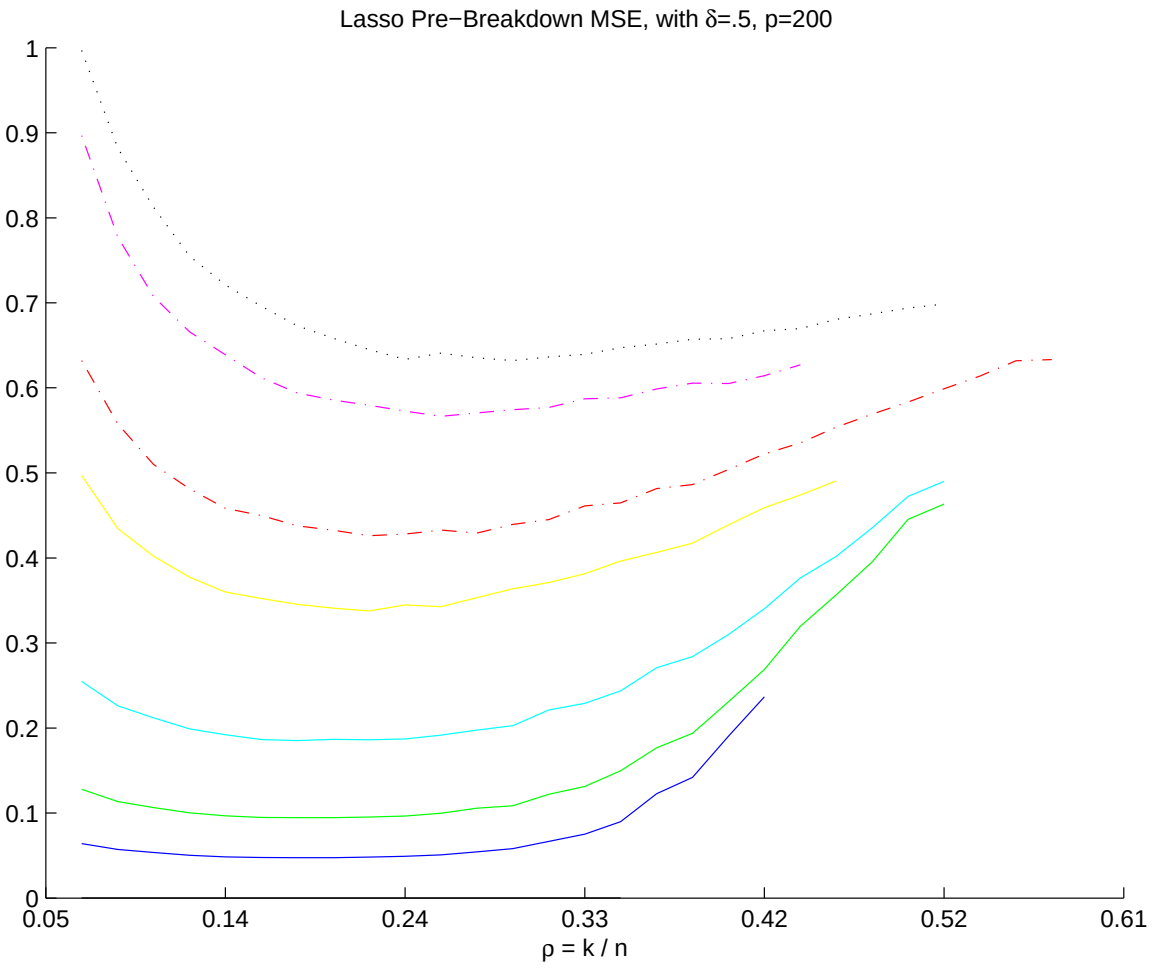

Figure 4.1: $nRMSE$ for Pre-Breakdown Region for Lasso; $\delta = \frac{1}{2}$, $p = 200$, $\lambda = \frac{\sigma}{k}$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 1000 replications. The breakdown point is less sharp with an increase in noise, and for very small ρ, the $nRMSE$ increases.

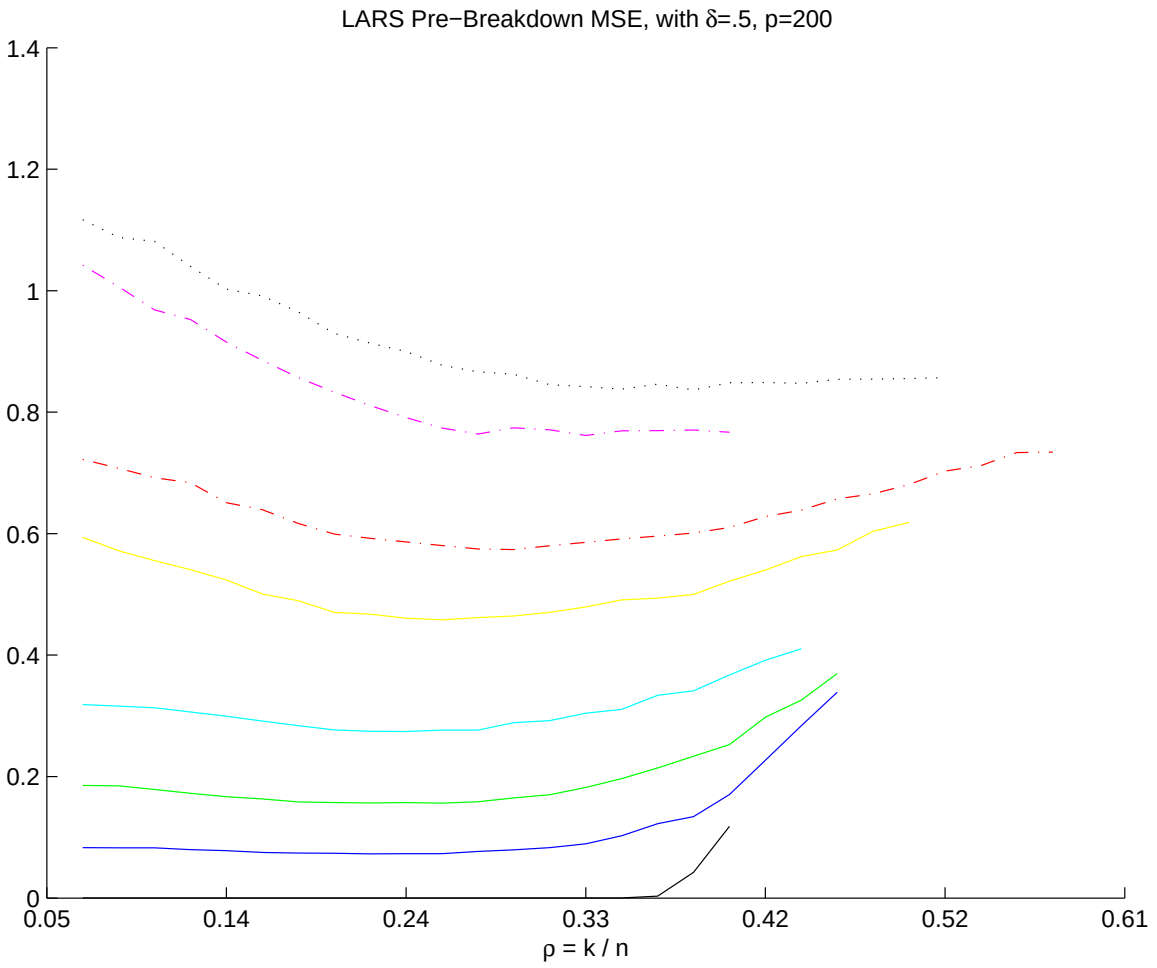

Figure 4.2: $nRMSE$ for Pre-Breakdown Region for LARS; $\delta = \frac{1}{2}$, $p = 200$, stopping criterion $= \frac{\sigma}{k}$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 1000 replications. For very small ρ, $RMSE$ increases dramatically, more than in Figure 4.1.

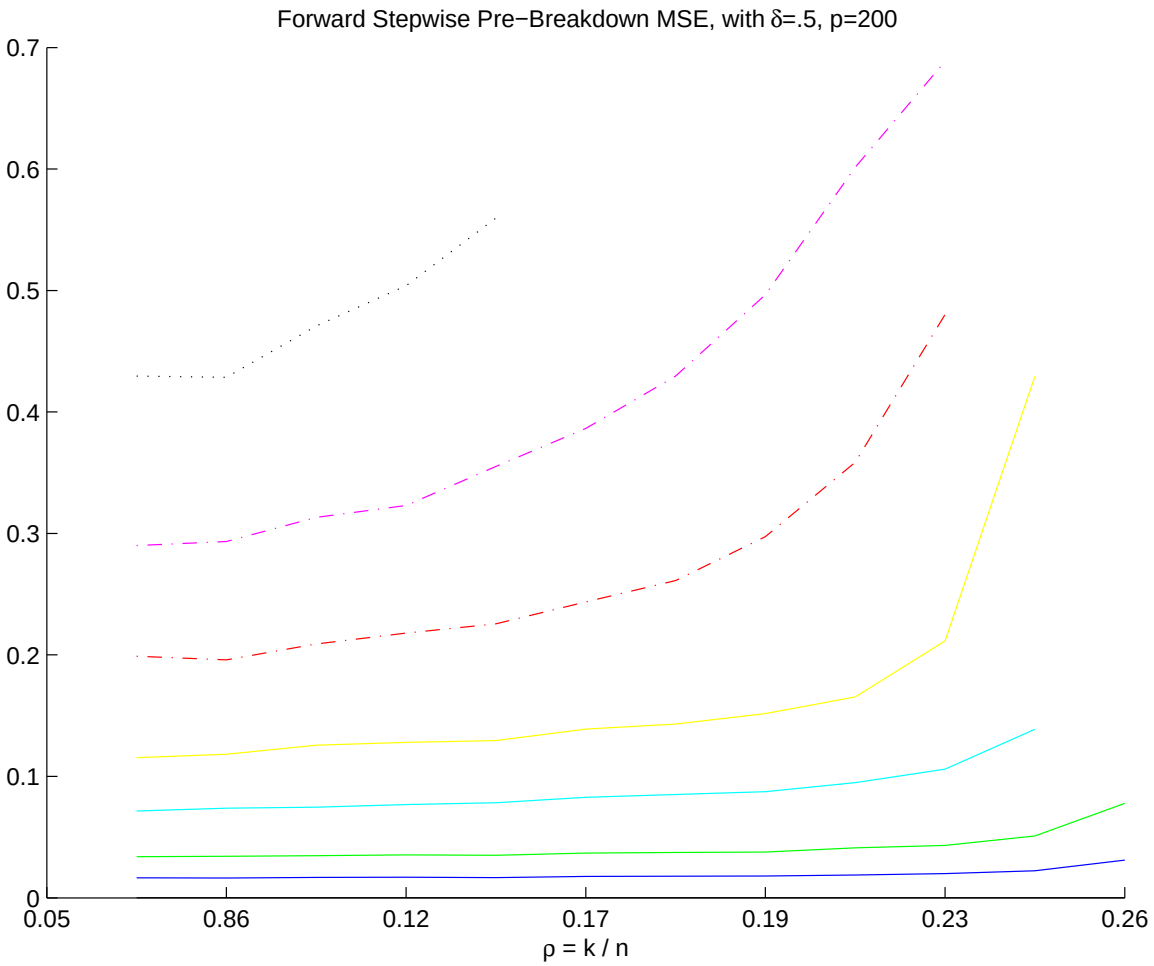

Figure 4.3: $nRMSE$ for Pre-Breakdown Region for Classical Forward Stepwise; $\delta = \frac{1}{2}$, $p = 200$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 1000 replications. Although the actual $nRMSE$ rates are low compared to the pevious algorithms, the breakdown points occurs earlier, ie. at smaller values of ρ.

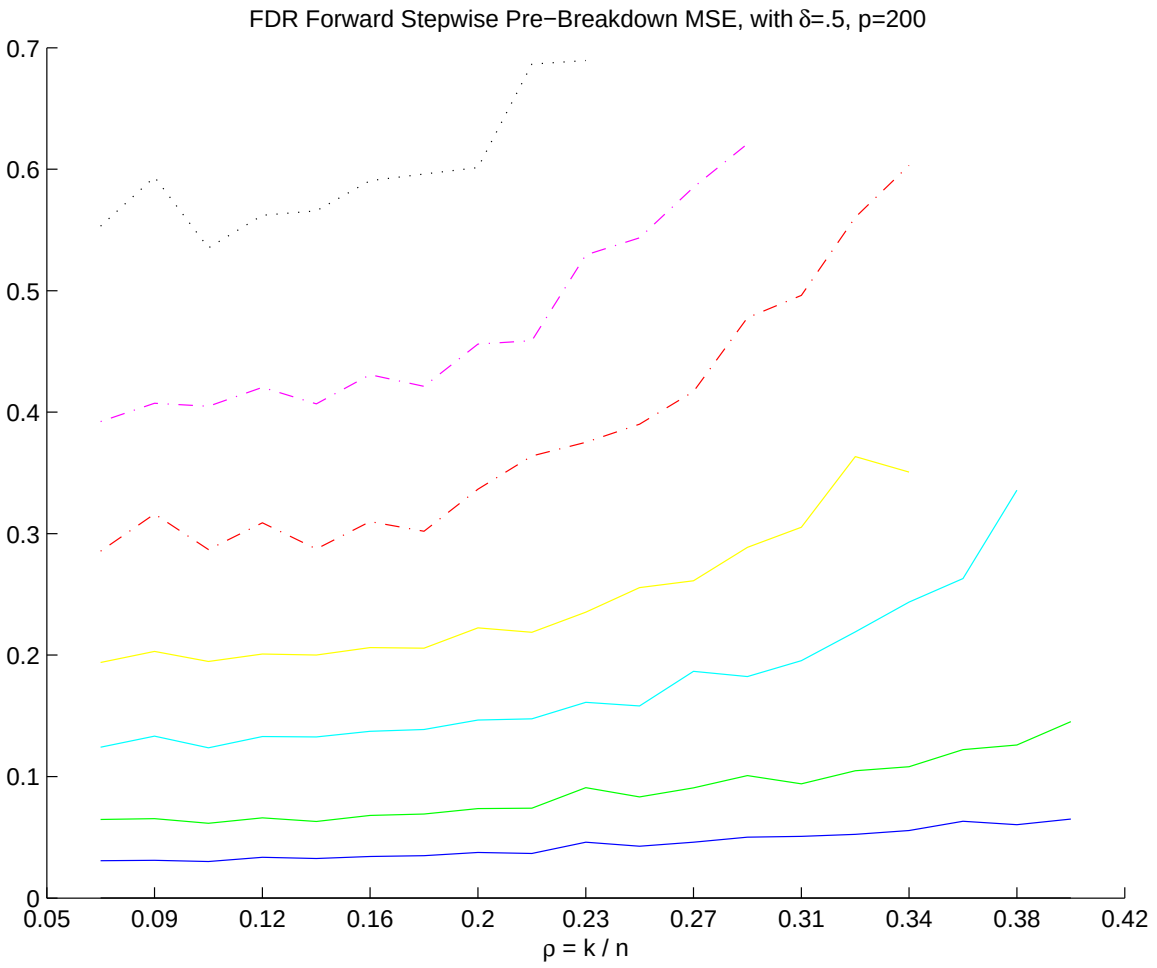

Figure 4.4: $nRMSE$ for Pre-Breakdown Region for Forward Stepwise with FDR Thresholding; $\delta = \frac{1}{2}$, $p = 200$, $q = 0.25$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 100 replications. As noise levels increase, $nRMSE$ increases proportionally.

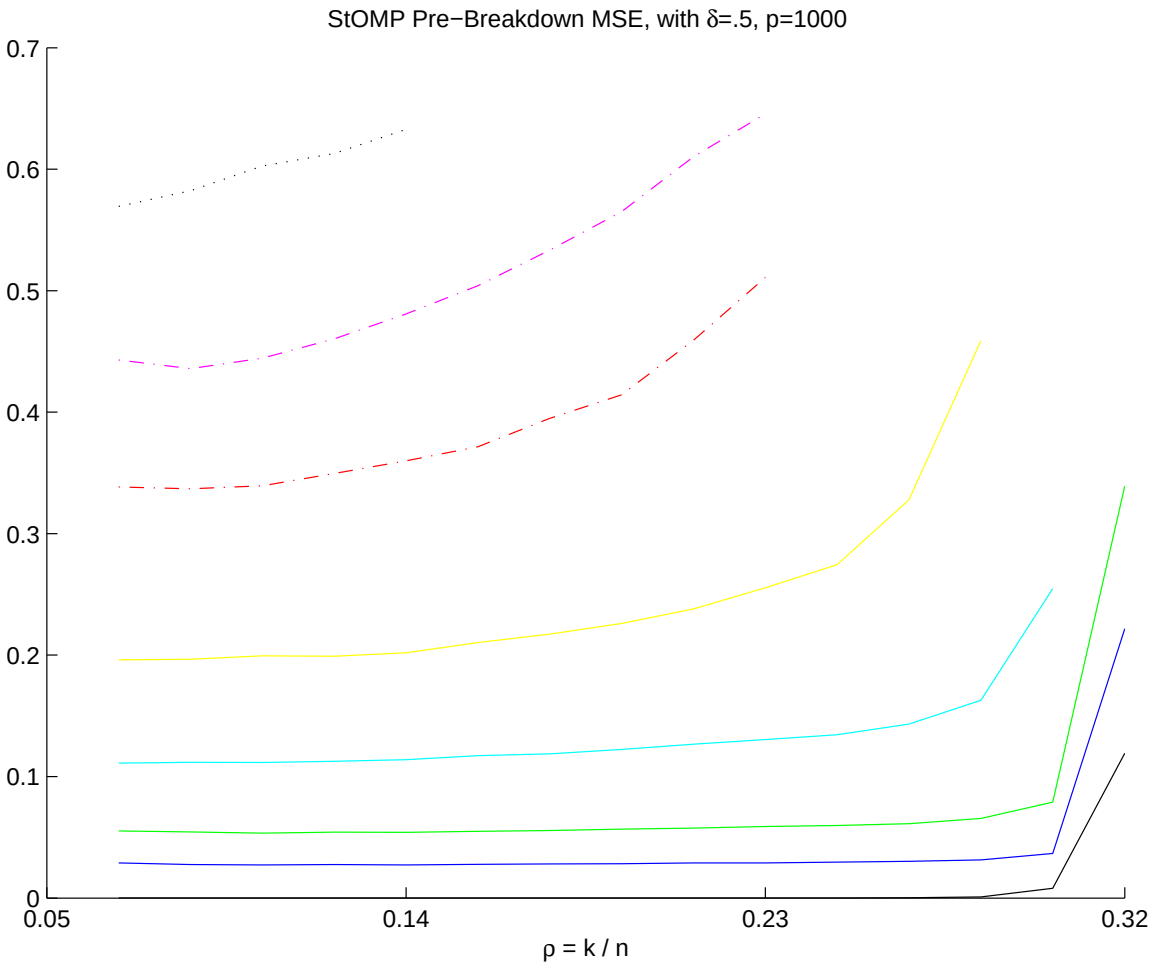

Figure 4.5: Pre-Breakdown Region for StOMP; $\delta = \frac{1}{2}$, $p = 1000$, $q = 0.25$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 1000 replications. StOMP shows the earliest breakdown points of all the algorithms, and approximately the same $nRMSE$ levels as Figure 4.4.

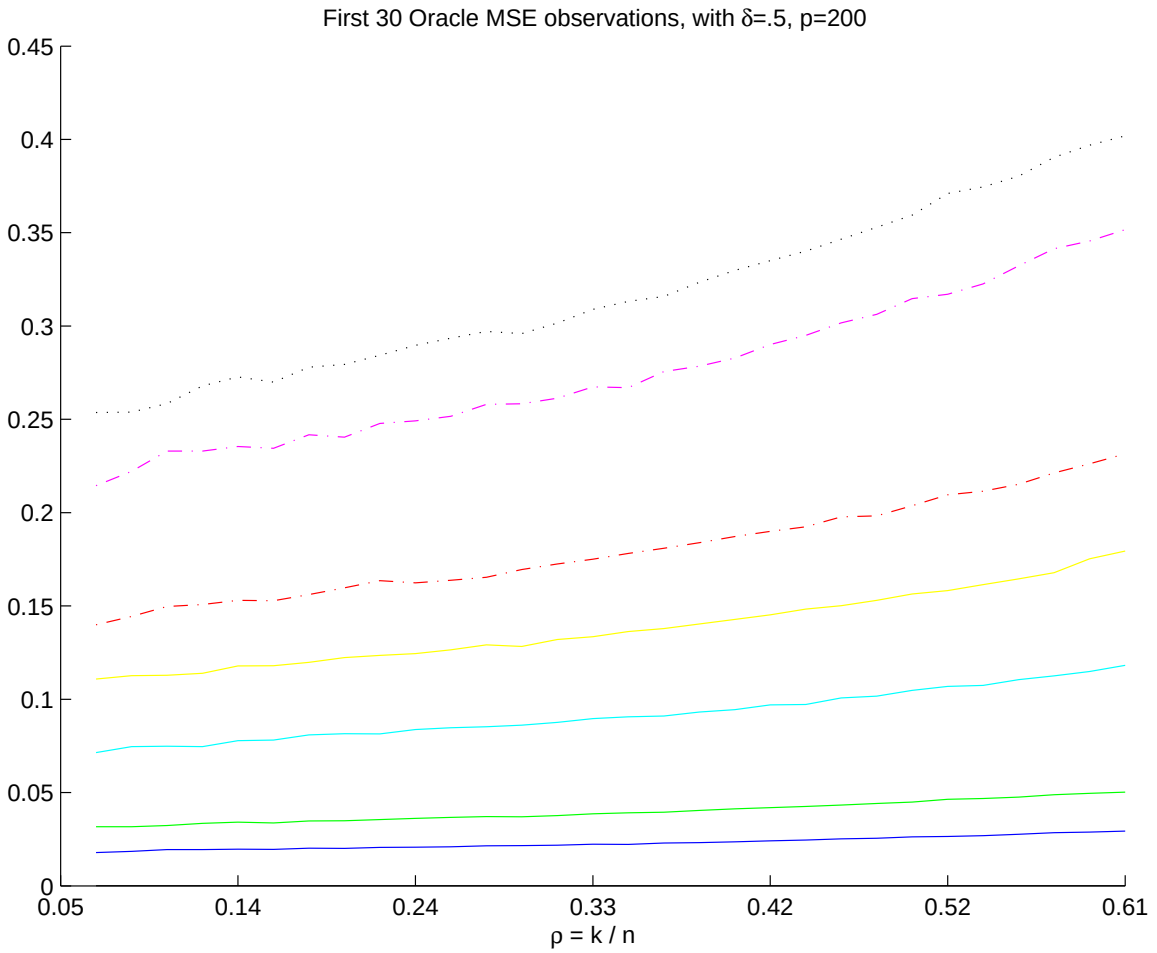

Figure 4.6: First 30 experiments for Oracle Model Selection – when the underlying model is known and estimated directly using OLS. $p = 200$; $\delta = \frac{1}{2}$, $p = 200$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 1000 replications. The $nRMSE$ increases proportionally with the noise level, as expected.

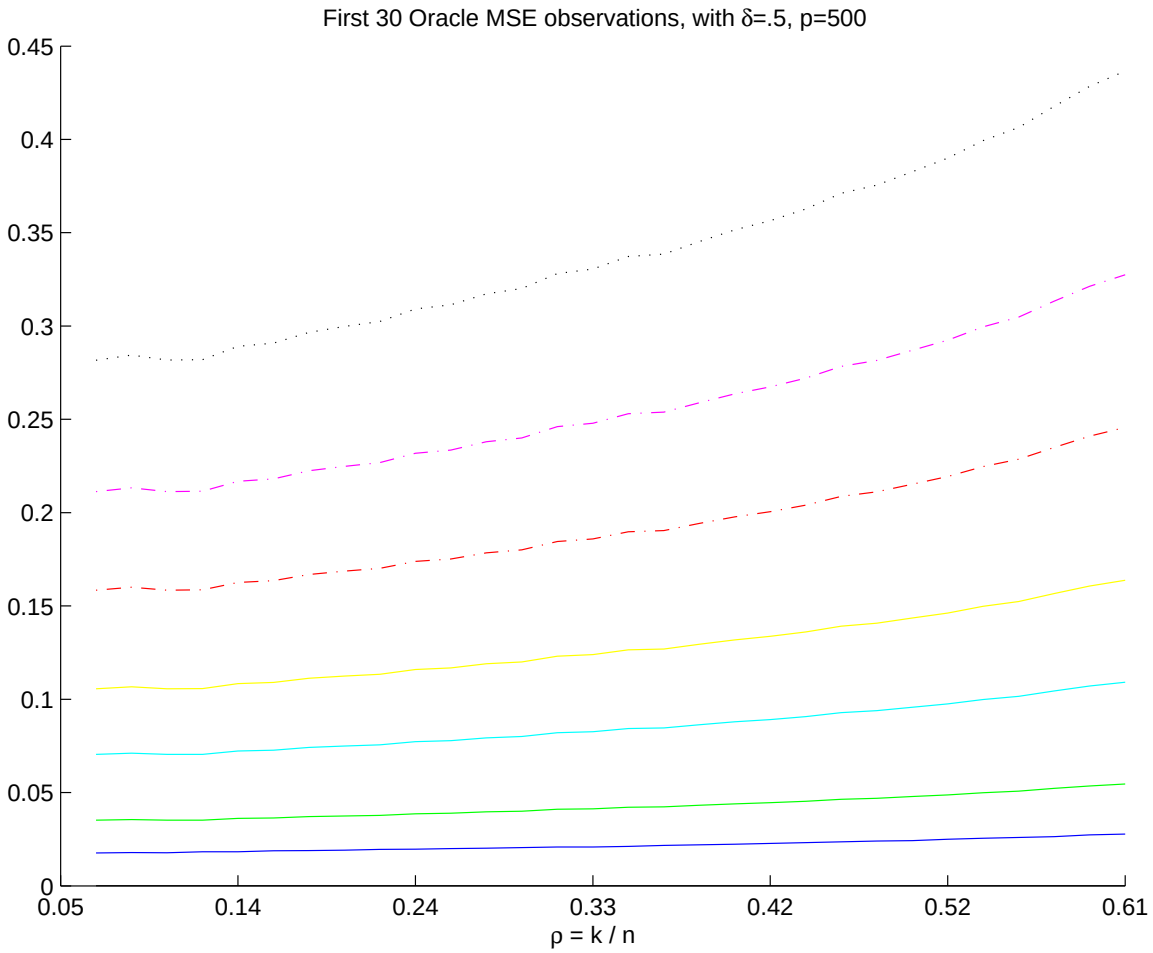

Figure 4.7: First 30 experiments for Oracle Model Selection – when the underlying model is known and estimated directly using OLS. $p = 500$; $\delta = \frac{1}{2}$, $p = 200$. Results for each of $\sigma \in \{0, 1, 2, 4, 6, 9, 12, 16\}$ are displayed. Median relative error over 1000 replications. Compared to Figure 4.6, for each noise level the $nRMSE$ is higher.

In both figures, no "breakdown" is apparent since we have oracle knowledge of the underlying model. The MSE does approach infinity as δ approaches 1 (as $p \to n$ the inverse of $X'X$ becomes increasingly unstable).

We can compare the performance of Forward Stepwise Selection with the oracle model as a baseline. Figures 4.8 and 4.9 show the ratio of show error results for this oracle model, under the same

When the Forward Stepwise algorithm *does* find the correct model, we expect the errors to be distributed according to equation (4.1); exhibiting similar behavior as the oracle MSE. Figures 4.8 and 4.9 plot the ratio of the median Forward Stepwise MSE *before the breakdown point* to the oracle MSE. The breakdown point was determined to be the point at which the first difference was maximized. The figures show the earlier breakdown point, at sparser underlying models, for the higher noise models. In both plots the noiseless model gives very nearly the same zero error rate using both algorithms, but as we add increasing levels of noise to the model, the median MSE for Forward Stepwise increases at a greater rate than that for the median Oracle MSE; roughly 1-2 times that of the oracle MSE. The effect of increasing the number of spurious variables can be seen in two ways: the Forward Stepwise MSE increases relative to the oracle MSE (for a given noise level), and it causes a breakdown point at lower sparsity levels.

4.2 Breakdown Path Analysis

Of interest is understanding precisely how the breakdown point depends on the noise level in the underlying model, and the sparsity and underdeterminedness levels. In this work, the level of underdeterminedness remains fixed at $\delta = \frac{1}{2}$ and we isolate the

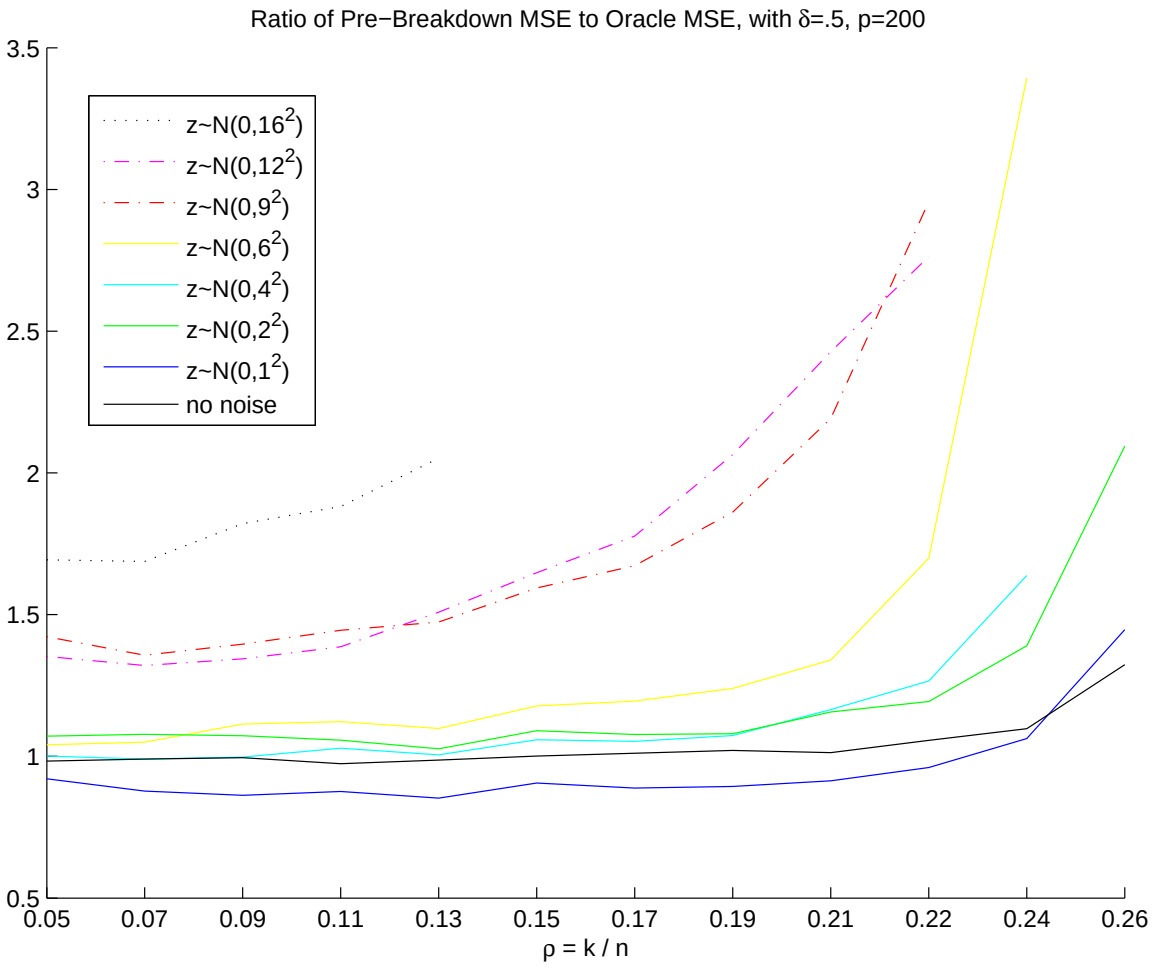

Figure 4.8: Ratio of Median MSE of Forward Stepwise to the Median Oracle MSE. The number of variables is fixed at 200, the number of observations at 100, ie. $\delta = \frac{n}{p} = .5$, and the median was taken over 1000 replications. The errors increase *more rapidly* for Forward Stepwise with increasing noise levels, than for the oracle MSE.

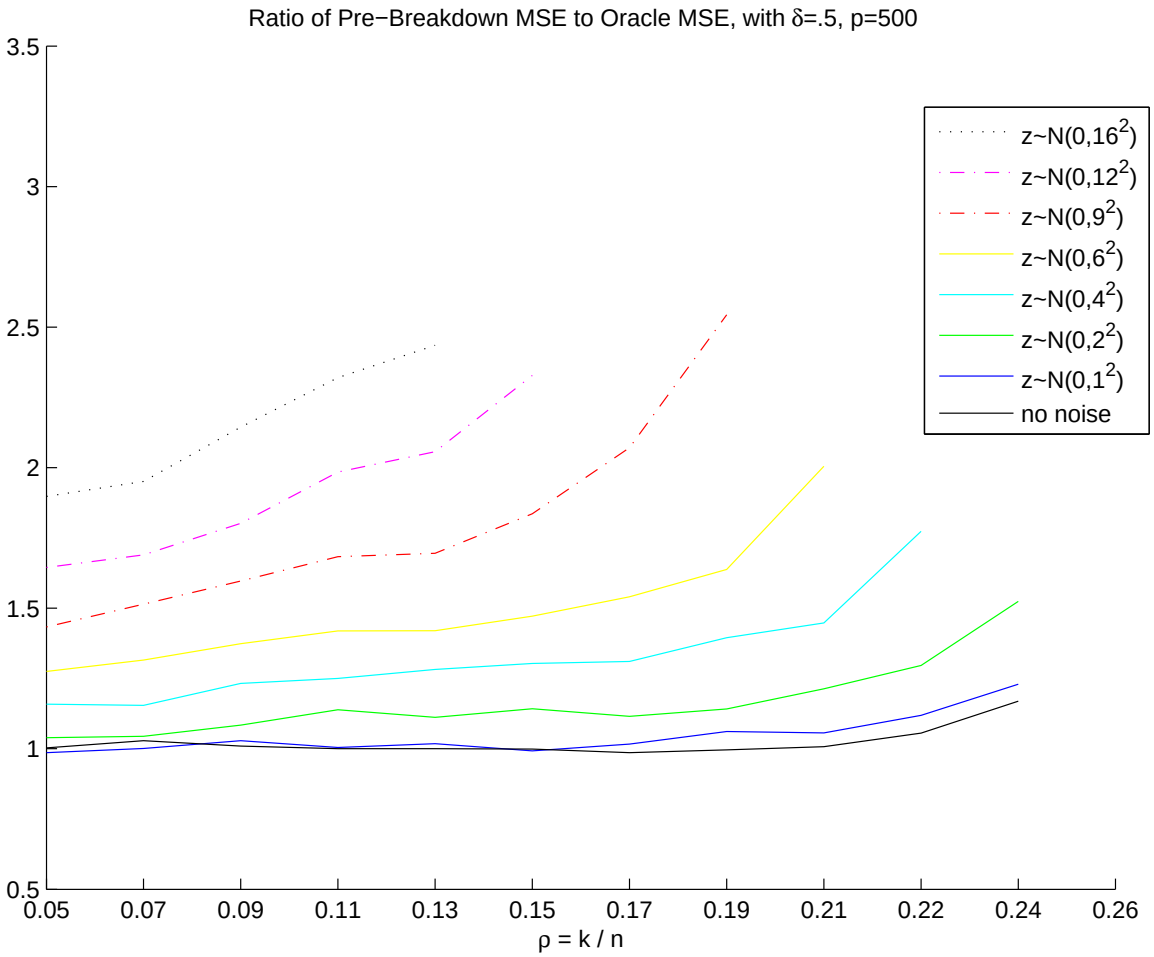

Figure 4.9: Ratio of Median MSE of Forward Stepwise to the Median Oracle MSE. The number of variables is fixed at 500, the number of observations at 250, maintaining $\delta = \frac{n}{p} = .5$, and the median was taken over 300 replications. The change in the number of variables from 200 to 500 causes the Forward Stepwise MSE to increase and implies a breakdown point at lower sparsity levels.

effects of noise and sparsity levels.

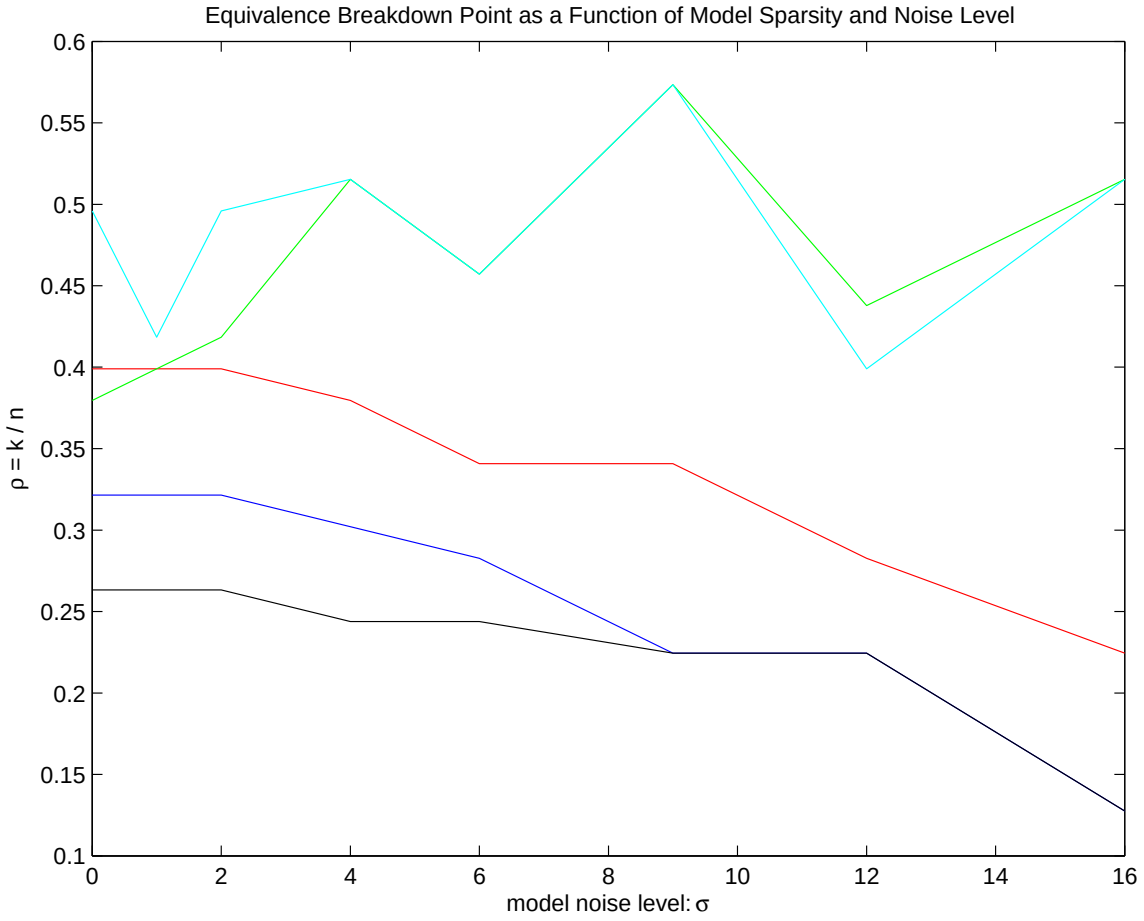

Figure 4.10: Breakdown Point for $\delta = \frac{1}{2}$ as a function of noise level and sparsity. The lowest curve, in black, shows the changes in the breakdown point for Classical Forward Stepwise, StOMP is the blue curve slightly above it, the red curve is Forward Stepwise with FDR Thresholding, the green curve is the Lasso, and the cyan curve is LARS.

Figure 4.10 plots the breakdown point for $\delta = \frac{1}{2}$ as a function of noise level and sparsity. The lowest curve, in black, shows the changes in the breakdown point for Classical Forward Stepwise, StOMP is the blue curve slightly above it, the red curve is Forward Stepwise with FDR Thresholding, the green curve is the Lasso, and the cyan curve is LARS.

Figure 4.10 shows the dramatic negative impact increasing levels of noise has on each algorithm's ability to recover the true parameters. When $\sigma = 0$, StOMP breaks down when the proportion of true variables to all variables exceeds 0.40, when $\sigma = 0$. When $\sigma = 6$, StOMP breaks down at $\rho = 0.35$, and when $\sigma = 16$ it breaks down if the true variables comprise more than 23% of the total number of variables. The pattern is more dramatic for Classical Stepwise: recovering the model effectively in the noiseless case when the sparsity level is below 27%, but only recovering models with sparsity level below 10% when σ increases to 16. Interestingly, Forward Stepwise with FDR Thresholding recovers a greater proportion of the models than Classical Stepwise, up until $\sigma = 9$, then its breakdown path is indistinguishable from Forward Stepwise.

Both Lasso and LARS occupy the top of the figure – recovering the greatest proportion of underlying models. Since both these algorithms exhibit a gradual increase in $nRMSE$ with sparsity level, their breakdown point is not sharply defined. This accounts for the erratic nature of their breakdown path. But some conclusions can be drawn: the breakdown points of the Lasso and LARS behave very similarly with respect to σ, and at low noise levels LARS seems to recover more models than the Lasso.

It is interesting to verify how the normalized root MSE at each algorithm's breakdown point changes with σ. Compared with the previously noted estimation error for an oracle model:

$$tr((X'X)^{-1})\sigma^2,$$

we expect a linear relationship between the noise level and $nRMSE$.

Figure 4.11 shows the striking linear relationship: as σ increases by 1 $nRMSE$

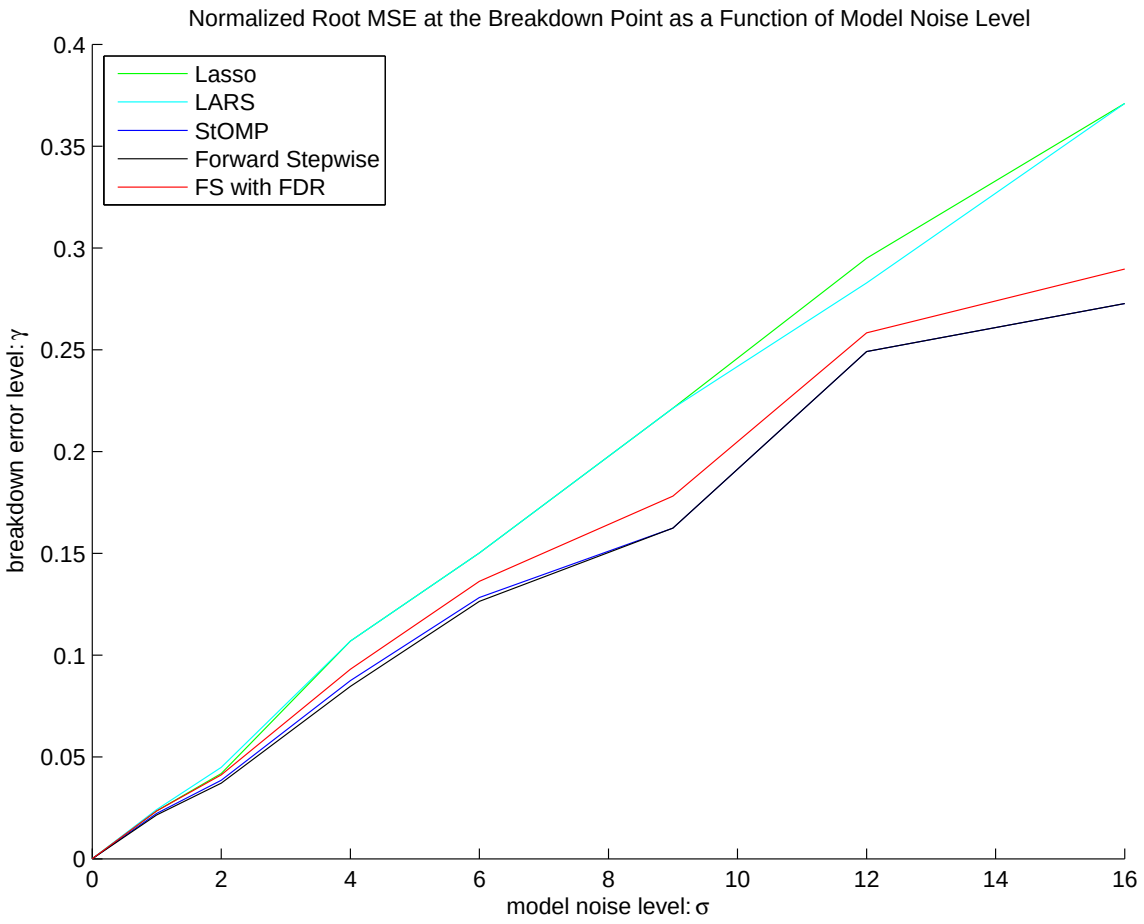

Figure 4.11: Normalized root MSE at Breakdown Point for $\delta = \frac{1}{2}$ as a function of noise level. The lowest curve, in black, shows the nRMSE for Classical Forward Stepwise, StOMP is the blue curve nearly coincident with the black curve, the red curve is Forward Stepwise with FDR Thresholding, the green curve is the Lasso, and the cyan curve is LARS.

increases by 0.025 for all σ, for the Lasso and LARS. The impact of a one unit increase in σ on the $nRMSE$ for the Stepwise algorithms lessens as σ increases.

From the path analysis there are clearly two classes of algorithms: the Lasso and LARS approach, and the Forward Stepwise family. This is very clear if we model the dependence of the $nRMSE$ at breakdown, on the noise level and sparsity level.

Dependence of $nRMSE$ on Noise Level and Sparsity

	σ	ρ
Forward Stepwise	0.0180	0.0324
FS with FDR Thresholding	0.0191	0.0237
StOMP	0.0181	0.0291
Lasso	0.0236	0.0091
LARS	0.0233	0.0101

Table 4.1: This table shows coefficients from a simple OLS fit of Noise Level, σ, and Sparsity Level, ρ, to $nRMSE$ at the breakdown point. $\hat{nRMSE} = \hat{\beta}_1\sigma + \hat{\beta}_2\rho$. The Lasso/LARS algorithms appear to behave differently than the Forward Stepwise algorithms.

Table 4.1 gives coefficients from an ordinary least squares fit of $nRMSE$ to noise and sparsity levels, for each algorithm. Increases in both sparsity and noise increase $nRMSE$ at the breakdown point, but notice how the impact of each varies by algorithm family. The Forward Stepwise family finds sparsity levels more difficult to handle, whereas the Lasso/LARS group is more affected by the noise level increases.

There is a strong positive relationship between the sparsity and model noise levels and the $nRMSE$ at the breakdown point. But what determines the breakdown point itself? We can find a function $\rho_{\ell_1,\delta} = f(\sigma)$ to describe the breakdown point of performance in the noisy case for each of the algorithm families.

Table 4.2 shows that the sparsity level at which the equivalence of the ℓ_1 and ℓ_0 problem formulations breaks down as a function of the noise level. As the noise level

Breakdown Point as a Function of Noise Level

	β_1
Forward Stepwise	-0.0071
FS with FDR Thresholding	-0.0110
StOMP	-0.0118
Lasso	0.0071
LARS	0.0010

Table 4.2: This table shows coefficients from a simple OLS fit of sparsity, ρ, to noise level, σ, at the Breakdown point. $\rho_\delta = \hat{\beta}_0 + \hat{\beta}_1\sigma$. The Lasso/LARS algorithms appear to behave differently than the Forward Stepwise algorithms.

increases the Stepwise Family of algorithms breaks down at a lower sparsity level – this is also reflected in Figures 4.3 to 4.5. No strong relationship appears between the noise level and the Lasso/LARS family - as suggested earlier this is likely due to the difficulty of selecting a breakdown point with their gradual changes in $nRMSE$ (evident in Figures 4.1 and 4.2).

4.3 When the ℓ_1 and ℓ_0 Problem Formulations are not Equivalent

The previous section focused on the portion of the Phase Diagram during which the algorithm successfully recovered the underlying model. All of the Phase Diagrams (Figures 3.1, 3.2, 3.7, 3.8, 3.14) exhibited a clear phase transition and then a convergence to a constant $nRMSE$ irrespective of the level of underdeterminedness or the sparsity level. As the number of variables, p, increased the transition sharpened, in all cases. After the equivalence breakdown the Forward Stepwise family of algorithms' $nRMSE$ converged to 1. The Lasso/LARS algorithms converged to slightly less than 1.

The width of the phase transition varied by algorithm – Classical Forward Stepwise and Forward Stepwise with FDR Thresholding had very sharp transitions and a clear ρ_{bp}. StOMP exhibited a more gradual transition, with the longest transitions belonging to the Lasso/LARS algorithms. At high noise levels, the fixed-δ curve flattens and a distinct breakdown point is difficult to discern.

4.4 Results

Several general results are clear. The algorithms studied divide themselves naturally into two classes: the *Forward Stepwise family* and the *Lasso/LARS family*.

In the Forward Stepwise family, the behavior of the $nRMSE$ at the breakdown point roughly follows:

$$nRMSE_{bp} = 0.18\sigma + 0.3\rho,$$

and the Lasso/LARS $nRMSE$ roughly follows:

$$nRMSE_{bp} = 0.23\sigma + 0.01\rho.$$

For a fixed δ, the value of ρ at which the performance breaks down can be understood as a linear function of the noise level, σ. In this case, the Forward Stepwise Family roughly follows:

$$\rho_{bp} = 0.34 - 0.01\sigma,$$

and the Lasso/LARS family can be roughly described with:

$$\rho_{bp} = 0.45 + 0.005\sigma.$$

Both families of algorithms have $nRMSE$ increasing to a constant beyond the breakdown point (1 for the Forward Stepwise family, slightly below 1 for Lasso/LARS).

Forward Stepwise with FDR Thresholding, the Lasso, and LARS all exhibit a phase transition very close to theoretical prediction. Classical Forward Stepwise breaks down at a constant low sparsity level independent of the level of indeterminacy, δ. For a very large number of variables relative to the number of observations, StOMP, Lasso, and LARS have trouble estimating the underlying model correctly.

In a practical setting, for all the algorithms except Classical Forward Stepwise and StOMP, we expect the model to be estimated with substantial error if, roughly, the sparsity level plus the underdeterminedness level is greater than one. For these two algorithms, the proportion of variables in the true model should be less than approximately one quarter, in order to estimate the model well.

Chapter 5

Component Prediction from NIR Spectra

The previous analysis suggests that it is important to consider the features of the problem at hand, such as model noise level, sparsity of the underlying problem relative to the number of variables, and number of variables relative to the number of observations. Unfortunately, all but the last of these is hidden from the researcher in most settings.

To apply our results in a practical setting, I followed the work of P. J. Brown, T. Fearn, and M. Vannucci [4]. They used Bayesian multivariate model selection techniques on wavelet transformed quantitative near-infrared reflectance (NIR) spectroscopy output in a chemometric application. Two separate batches of cookie dough were prepared from the same recipe and the task was to predict dough chemical composition from the spectroscopy readings.

The NIR spectrum of a sample is a continuous curve measured at hundreds of equally spaced wavelengths. The original experiment is fully explained in Osbourne

et al. [40]. An NIR spectrum is generated for each dough piece. The data consist of 700 points measured from 1100 to 2498 nanometers, in steps of 2 nanometers. Following Brown et al., I choose to truncate the data by removing the first 140 and the final 49 wavelengths, and subsampling the data by 2, thus being left with a dataset of 256 wavelength readings. Brown et al. ran a wavelet transform on the data then used the transformed variables for model selection.

Brown et al. then predicted 4 components: fat, sucrose, flour, and water in a multivariate linear regression setting. In contrast I did not perform a wavelet transform, and am attempting to predict only the sucrose content of the dough, rather than all 4 percentages; modeling the surcrose percentage as:

$$Y = X\beta + z$$

where Y is 39×1, X is 39×256, β is 256×1, and z is 39×1.

There are 39 dough samples in the first batch (after dropping outlier observation 23, following Brown et al.), and 31 in the second (after dropping outlier observation 21 following Brown et al.). This provides an immediately appealing situation where the number of variables is much larger than the number of predictors, $\delta = \frac{n}{p} = \frac{39}{256} = 0.1523$. In their study, Brown et al. selected 10 coefficients from the wavelet-transformed variables. If this is a rough estimate of the number of curve samples in the true model we have a situation where $\rho = \frac{k}{n} = \frac{10}{39} = 0.2564$.

I examined the Phase Diagram for each of the algorithms at $p = 200$ under study at the point $(\delta, \rho) = (0.1523, 0.2564)$ and found this to be in the region of successful model recovery for all the algorithms, with the exception of StOMP. This coordinate appears in the phase transition region for StOMP, where the algorithm performance is beginning to degrade. If the design matrix were a random Gaussian matrix of the

kind studied in this paper, we would expect algorithm performance in this example, the best estiate of the underlying model would come from Classical Forward Stepwise, the Lasso, or LARS. There is a high degree of correlation between the variables in the cookie dough example, with 85.4% of the variables having a correlation greater than 0.9, and 99.6% having a correlation greater than 0.8.

5.1 Model Selection

All five algorithms were run on these data: the 39 values giving the sucrose percentage for the first batch and the 256 untransformed NIR frequency readings for each observation in this batch. The RSS for the model selected by each algorithm is given in Table 5.1.

Residual Sum of Squares for Batch 1

	RSS	Number of Variables
Forward Stepwise	235.3027	3
FS with FDR Thresholding	33.2806	2
StOMP	0.0	21
Lasso	145.3495	1
LARS	43.0850	3

Table 5.1: This table gives the RSS for cookie batch 1 for every algorithm. $n = 39$ and $p = 256$.

Lasso and LARS were not carefully tuned. Based on the results in Section 3.1.1, λ was set to 1[1]. StOMP was used with FDR Thresholding with FDR paramter q set to 0.25 consistent with Section 3.3. StOMP appears to have found a near-perfect fit. StOMP is also the only algorithm that chose more than 3 variables, and it chose a much larger number, 21. The StOMP algorithm was run with an error tolerance of

[1]experimentation with varying λ to 2 and 3 did not improve the RSS

$1e-5$ and converged in 2 iteration, but varying the error level does not change the parameter estimates nor the number of iterations.

5.2 Algorithm Performance Analysis

In this experiment, the NIR specstroscopy was repeated on a second batch of dough. This batch was created from the same recipe but blended independently from scratch. There are 31 observations in this dataset (with observation 21 dropped as an outlier following Brown et al.) and the same preprocessing was carried out to give 256 possible variables. The model selected in the previous section was run on these 256 variables and a prediction of sucrose contents made. Table 5.2 gives the RSS between the predicted value and the true value in this test set.

Residual Sum of Squares for Batch 2

	RSS
Forward Stepwise	843.8690
FS with FDR Thresholding	60.1412
StOMP	582.5190
Lasso	159.3528
LARS	79.3956

Table 5.2: This table gives the RSS for cookie batch 2, using the model estimated from batch 1, for every algorithm. $n = 31$ and $p = 256$.

It appears StOMP has overfit the model - the RSS on the test data is the highest of all the algorithms, except Classical Forward Stepwise. Classical Forward Stepwise appears not to have fit the model well - it had the highest RSS of all the algorithms on the training data, then doubled its error on the test data. The lowest RSS, for both training and test data, belongs to Forward Stepwise with FDR Thresholding. With only 2 variables in the model it is also exceptionally sparse, much sparser than

the 10-term model obtained by Brown et al. Brown et al. obtained RSS values for their model that are not relevant here due to the fact they they implemented a wavelet transform on the data, and they predict the 4 components of the dough in a multivariate model.

LARS was close in RSS to Stepwise with FDR Thresholding, with slightly greater RSS in both training and test cases. The LARS model contains 3 terms, greater than the FDR Stepwise's choice of 2. LARS chose variables 175, 233, and 256 while Stepwise FDR chose variables 175 and 244.

StOMP was predicted not to find the underlying model accurately, and that was found to be the case. LARS was expected to do well, and that was also borne out. Surprisingly Forward Stepwise with FDR thresholding did remarkably well, and the Lasso and Classical Forward Stepwise did not perform as well as expected. These results might be due to the fact that my estimate of ρ was taken from Brown et al.'s work with wavelet transformed variables.

Improvements in model prediction might be made by transforming the variables, through a wavelet transform or principal components, and perhaps using a logistic model since sucrose is measured as a percent and thus lies in an interval.

Chapter 6

Software Implementation: SparseLab

SparseLab is an open source library of Matlab^{TM} routines for finding sparse solutions to underdetermined systems. SparseLab has two *raisons d'être*. The library provides the research community with open source tools for sparse representation, supplemented with detailed examples and demonstrations, as well as providing a vehicle for releasing open source code for reproducing figures and results in published papers. For example, all the code used to create the figures appearing in this thesis is publicly available in SparseLab and downloadable at `http://sparselab.stanford.edu`.

SparseLab provides software for several published solvers, for example Michael Saunders' Primal-Dual method for Optimization with Convex Objectives, Mallat and Zhang's Matching Pursuit, Donoho and Johnstone's Iterative Hard and Soft Thresholding, Efron et al.'s Least Angle Regression (also based on code from [44]), and a number of others.

6.1 SparseLab Defined

SparseLab has over 400 .m files which are documented, indexed, and cross-referenced in various ways.

6.1.1 Detailed Description of the Software

There are two main components of the SparseLab package. First, there are the basic "system components":

1. *Source.* There is source code, in Matlab^{TM} , TEX, Perl.

2. *Build.* The source code is assembled into a standard release. The current release is .100.

3. *Archives.* Compressed archives of the standard release available for three platforms, Mac, Unix and PC, which users can download and install on their machines.

4. *Web Documents.* A web home page and a series of postscript and pdf files which explain what SparseLab is and how to get it. The URL is http://sparselab.stanford.edu.

Next there are the basic "user components" of an installed system:

1. *SparseLab Main Directory.* A subdirectory /Sparselab100 of the `Matlab/work` directory, containing the currently released version of SparseLab software, datasets and documentation.

2. *Papers.* A directory /Sparseelab100/Papers/ in /Sparselab100 containing scripts reproducing figures in various articles and technical reports.

3. *Examples.* A directory /Sparselab100/Workouts/ in /Sparselab100 containing pedagogical examples that exercise various aspects of SparseLab.

4. *Solvers.* A directory /Sparselab100/Solvers/ in /Sparselab100 containing the various solver engines of SparseLab.

5. *Documentation.* Both pdf and Postscript files.

6. *Datasets.* Numerical and image data used in the included papers and to illustrate various aspects of sparse analysis. The largest data components are included as separate downloads: Sparselabvers_DataSupplementExtCS and Sparselabvers_DataSupplementStOMP, where vers is replaced by the current SparseLab version.

Once the actual files are downloaded, they can be decompressed and installed locally. Every folder contains a file *Contents.m*, which displays the contents and purpose of that directory. The contents file in the root folder shows the directory layout for SparseLab:

```
% SparseLab Main Directory, Version 100
%
% This is the main directory of the SparseLab package.
%
%              .m files in this directory
%
%   Contents.m           -    This file
%   SparsePath.m         -    Sets up global variables and pathnames
%
%              Subdirectories
%   Documentation        -    System-Wide Documentation
%     /About SparseLab
%     /SparseLab Architecture
%   Examples             -    Detailed examples of SparseLab finding
%                             sparse solutions
```

```
%     /nnfEX              -    Nonnegative Factorization Example
%     /reconstructionEx   -    Signal Reconstruction Example
%     /RegEx              -    Regression Example
%     /TFDecompEx         -    Time-Frequency Reconstruction Example
%   Papers                -    Scripts for reproducing figures in
%                              published articles
%     /ExtCS              -    figures for "Extensions of Compressed
%                              Sensing"
%     /HDCPNPD            -    figures for "High-Dimensional Centro-
%                              symmetric Polytopes with Neighborliness
%                              Proportional to Dimension"
%     /NPSSULE            -    table in "Neighborly Polytopes and
%                              Sparse Solutions of Underdetermined
%                              Linear Equations"
%     /NRPSHD             -    figures for "Neighborliness of
%                              Randomly-Projected Simplices in
%                              High Dimensions"
%     /SNSULELP           -    figures for "Sparse Nonnegative Solutions
%                              of Underdetermined Linear Equations by
%                              Linear Programming"
%   Solvers               -    Sparse solver packages
%   Tests                 -    Simple pedagogical workouts
%   Utilities             -    General tools for developers and users
%   shell_tools           -    Tools for use during the SparseLab build
%                              process

%
% Part of SparseLab Version:100
% Created Tuesday March 28, 2006
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail sparselab@stanford.edu
```

The Papers/ directory houses reproducible published papers, each with its own "demo". The demo allows you to reproduce the figures in the corresponding article. When you invoke the demo file in Matlab^{TM} by typing its name (without the .m extension), a new window will appear on the screen. If you mouse-click on the push button `Show All Figures` you will see, in sequence, each figure in the corresponding

article. As each figure appears in the Matlab^{TM} figure window, the command window will contain narrative explaining what you see in the figure window.

The architecture of the `/Papers` directory is as follows. At present, it contains these subdirectories, recreating figures in the following published articles:

```
ExtCSDemo     - ‘‘Extensions of Compressed Sensing’’
HDCPNPDDemo   - ‘‘High-Dimensional Centrosymmetric Polytopes with
                Neighborliness Proportional to Dimension’’
MSNVENODemo   - ‘‘Breakdown Point of Model Selection when the
                Number of Variables Exceeds the Number of
                Observations’’
NPSSULEDemo   - ‘‘Neighborly Polytopes and Sparse Solutions of
                Underdetermined Linear Equations’’
NRPSHDDemo    - ‘‘Neighborliness of Randomly-Projected Simplices
                in High Dimensions’’
SNSULELPDemo - ‘‘Sparse Nonnegative Solutions of Underdetermined
                Linear Equations by Linear Programming’’
StOMPDemo     - ‘‘Sparse Solution of Underdetermined Linear
                Equations by Stagewise Orthogonal Matching
                Pursuit’’
```

The directory `Solvers` contains the central program solver tools; its `Contents.m` file looks as follows:

```
% SparseLab Solvers Directory
%
% This is directory houses the solvers for the SparseLab package.
%
%              .m files in this directory
%
%   Contents.m           -    This file
%   fdrthresh.m          -    Uses the False Discovery Rate to
%                             Threshold a Signal
%   HardThresh.m         -    Implements Hard Thresholding
%   lsqrms.m             -    Iterative least squares
%   pdco.m               -    Primal-Dual Barrier Method for Convex
%                             Objectives (Michael Saunders 2003)
%   pdcoSet.m            -    creates or alters options structure for
```

```
%                                pdco.m
%   SoftThresh.m          -    Soft Thresholding
%   SolveBP.m             -    Basis Pursuit
%   SolveIRWLS.m          -    Iteratively ReWeighted Least Squares
%   SolveIST.m            -    Iterative Soft Thresholding
%   SolveISTBlock.m       -    Iterative Soft Thresholding, block
%                              variant with least squares projection
%   SolveLasso.m          -    Implements LARS/Lasso Agorithms
%   SolveMP.m             -    Matching Pursuit
%   SolveOMP.m            -    Orthogonal Matching Pursuit
%   SolveStepwise.m       -    Forward Stepwise
%   SolveStepwiseFDR.m    -    Forward Stepwise with FDR Threshold
%   SolveStOMP.m          -    Stagewise Orthogonal Matching Pursuit
%

%
% Part of SparseLab Version:100
% Created Tuesday March 28, 2006
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail sparselab@stanford.edu
%
```

The user is free to browse all the source code for these sparse solvers. SparseLab also contains an Examples/ directory to facilitate the user's introduction to these routines.

```
nnfEx              Nonnegative Factorization
reconstructionEx   Signal Reconstruction
RegEx              Model Selection in Regression
TFDecompEx         Time Frequency Decomposition
```

Each software example is also worked through in detail in the Examples section of the SparseLab website. All scripts are documented to allow users to modify parameters such as noise level, number of variables or observations, algorithm used, etc.

6.1.2 Documentation

Each function in SparseLab has help documentation. For example, `SolveMP` is Mallat and Zhang's Matching Pursuit algorithm. If you are in Matlab™ and type `help SolveMP`, Matlab™ will type out the following documentation:

```
% SolveMP: Matching Pursuit (non-orthogonal)
% Usage
%   [sol iters activeHist] = SolveMP(A, b, maxIters, NoiseLevel, verbose)
% Input
%   A           dictionary (dxn matrix), rank(A) = min(d,n) by assumption
%   y           data vector, length d.
%   maxIters    number of atoms in the decomposition
%   NoiseLevel  estimated norm of noise, default noiseless, i.e. 1e-5
%   verbose     1 to print out detailed progress at each iteration, 0 for
%               no output (default)
% Outputs
%    sol         solution of MP
%    iters       number of iterations performed
%    activeHist  Array of indices showing elements entering
%                the solution set
% Description
%   SolveMP implements the greedy pursuit algorithm to estimate the
%   solution of the sparse approximation problem
%      min ||x||_0 s.t. A*x = y
% See Also
%   SolveOMP
% References
%   Matching Pursuit With Time-Frequency Dictionaries (1993) Mallat &
%   Zhang, IEEE Transactions on Signal Processing
%
```

Code submitted for release in SparseLab follows a strict formatting guideline: a one-line *help header*, and sections for *Usage*, *Inputs*, *Outputs*, *Side Effects*, *Description*, *Examples*, *Algorithm*, *See Also* and *References*.

1. *Header.* The first line of the help header is called the `H1` line by Matlab™. It is special to Matlab™, and to SparseLab. When you use the `lookfor` command,

Matlab™ examines this line for each .m file in its path to find text matching the request. When a release of SparseLab is built, these lines are compiled and sorted in alphabetical order to make files in the documentation directory. The format is: a percent sign, a single blank, the name of the function, a blank followed by double hyphens and a blank, and a short description of the function. The description should contain as many helpful keywords as possible.

2. *Usage.* The appropriate call to the function is indicated. Format: the output argument(s) (enclosed within square brackets if there is more than one output argument), an equals sign, the function name followed by the input argument(s) enclosed within parentheses. Optional input arguments are enclosed within square brackets.

3. *Inputs.* One input variable per line, indicating the name of the variable, the formal data type, and the interpretation.

4. *Outputs.* Again, one output variable per line, indicating the name of the variable, the data type, and the interpretation.

5. *Description.* A description of what the function does in as much detail as possible.

6. *Examples.* Examples of how the function is called in practice. This field is optional.

7. *Algorithm.* Here, describe the algorithm used by the function. This field is optional.

8. *See Also.* Here, mention other routines which this routine calls or which call this one, or routines with a special relationship to this function. This field is

optional.

9. *References*. Here, list references from which the user may obtain further information about the function. This field is optional.

The SparseLab system also has extensive built-in documentation about the system itself. If you look in the directory `Documentation`, you will find several files of general interest:

```
% ADDINGNEWFEATURES       -  How to Add New Features to SparseLab
% BUGREPORT               -  How to report bugs about SparseLab
% COPYING                 -  SparseLab Copying Permissions
% DATASTRUCTURES          -  Basic data structures in SparseLab
% FEEDBACK                -  Give feedback about SparseLab
% GETTINGSTARTED          -  Ideas for getting started with SparseLab
% INSTALLATION            -  Installation of SparseLab
% LIMITATIONS             -  SparseLab known limitations
% PAYMENT                 -  No Charge for SparseLab Software
% REGISTRATION            -  SparseLab Registration
% SUPPORT                 -  SparseLab Support
% THANKS                  -  Thanks to contributors
% VERSION                 -  Part of SparseLab Version v090
% WARRANTY                -  No Warranty on SparseLab software
%
%              Subdirectories
%
%  AboutSparseLab         -  documentation for AboutSparseLab document
%  SparseLabArchitecture  -  documentation for SparseLabArchitecture
%                            document
```

Each file explains SparseLab's policy on each of the topics. There has been extensive concern for the documentation of the code in SparseLabto produce a coherent, understandable system. Two technical reports are included on the SparseLab webpage detailing SparseLab – its motivation and its architecture, *About SparseLab* and *SparseLab Architecture*.

6.2 Motivation

The philosophy adopted by SparseLab can be traced to two articles: *Electronic Documents Give Reproducible Research New Meaning* (`http://sepwww.stanford.edu` by Jon Claerbout and Martin Karrenbach and *Making scientific computations reproducible* [42] by Jon Claerbout, Martin Karrenbach, and Matt Schwab `http://sep.stanford.edu/research/redoc/cip.html`). In 1991, they developed a uniform set of standards for electronic documents made a paper's results and their reproduction readily accessible (labeled ReDoc) with details located at `http://sep.stanford.edu/research/redoc/`.
As a direct consequence in their lab, junior researchers were able to build on work previously accomplished by senior lab members.

To paraphrase their thoughts:

> *A traditionally published article is not the end product of scholarship; it is the advertisement for the scholarship. The working software environment that produced the figures in the article is the actual end product of the scholarship.*[1]

What this means to computational statisicians is that computational experiments in Matlab^{TM} must be structured in a such a way to allow reproducibility. This approach was pioneered by David Donoho and Johnathan Buckheit with Wavelab in 1994 [6]. Their idea was that, when doing research, long before writing an article,

> "we prepare ourselves with the thought that *what we do on the computer*

[1] espoused as *Claerbout's Principle* in [12]

> *will ultimately be made available to others, for their inspection, modification, re-use and criticism.* This implies that the work product which we are aiming to create will be a series of scripts that will generate, from scratch, all the figures of the corresponding article. The work product is the underlying algorithms and code which generate the figures, and which will be made available to others. The deliverable is not a figure itself, but instead the software environment that, applied in the right way, produces the image, and which, hopefully, could be applied to other datasets to produce equally nice images."[5]

With this as background, reproducibility of experiments requires having the complete software environment available in other laboratories and the full source code available for inspection, modification, and application under varied parameter settings. This approach has been fully implemented in the software packages *Wavelab*[2], *BeamLab*[3], and *SymmLab*[4][5, 23, 22, 6, 11, 7, 18, 42].

Within the statistics community reproducible research seems to be a growing phenomena. In 2001 Jan de Leeuw, at the UCLA Department of Statistics, wrote a technical report *Reproducible Research. The Bottom Line* [12] calling for further reproducibility of published articles in statistics (including all code). Increasing numbers of researchers are releasing code associated with published articles in R, an open source statistics programming language (GNU 'S'). Packages can be made publicly available as `shar` files (archived unix packages) at the Comprehensive R Network website `http://cran.r-project.org/` and/or incorporated into TEX files directly

[2]see `http://www-stat.stanford.edu/~wavelab`
[3]see `http://www-stat.stanford.edu/~beamlab`
[4]see `http://www-stat.stanford.edu/~symmlab`

using *Sweave*, a package for reproducible research, or *literate programming*[5], in R[6]. A presentation was given by L. Philip Schumm and Ronald A. Thisted of the University of Chicago at the North American Stata Users' Group Meetings in 2005 on reproducible research using the Stata programming language[7].

In the most recent issue of the Journal of the American Statistical Association (Volume 101, Number 474, June 2006), 3 of 35 articles actually made their code publicly available. All the articles outlined a statistical procedure the authors had applied. Four mentioned the software package they had used, and one mentioned that their code was available upon request. Although all articles described a new statistical methodology, possibly algorithmic, and nearly all included empirical examples, the article seemed to imply coding was to be repeated by the reader. Bear in mind that 10 years ago, in the June 1996 issue of JASA, none of the published articles released their code, or even mentioned it, and none mentioned what software package they had used to produce their results. Although, interestingly, 11 of these 20 were theoretical papers without a computer component, whereas in June 2006 a mere 2 of the 35 were.

SparseLab code is made available to allow other researchers to reproduce the figures in our articles and to study the exact parameter settings and algorithms which were used in those articles. Given availability of a Matlab™ environment[8] the toolbox provides the interested reader with everything necessary to rebuild and modify the figures in the included articles.

[5] a term originally coined by Donald Knuth, author/inventor of TEX. See `http://www-cs-faculty.stanford.edu/~knuth/lp.html`

[6] see `http://www.stat.umn.edu/~charlie/Sweave/`

[7] `http://repec.org/nasug2005/Schumm_NASUG-presentation.pdf`

[8] The Matlab™ programming language is not available free, and our toolbox does require access to Matlab. Until freely available Matlab™ alternatives, such as *Octave* for GNU/Linux, have the capability of Matlab™ we continue to use Matlab™ . Sorry, Richard Stallman.

6.3 Methodology

This body of software is under continuing development by a team of researchers, including David Donoho, Iddo Drori, Yaakov Tsaig, and myself. We conduct our research with the idea, from the beginning, that we will implement our tools in SparseLab. We believe that the discipline this entails makes our research of a higher quality than otherwise possible.

The code is compiled together with the separate directories:

```
%   Documentation        -    System-Wide Documentation
%   Examples             -    Detailed examples of SparseLab finding
%                             sparse solutions
%   Papers               -    Scripts for reproducing figures in
%                             published papers
%   Solvers              -    Sparse solver packages
%   Tests                -    Simple pedagogical workouts
%   Utilities            -    General tools for developers and users
%   shell_tools          -    Tools for use during the SparseLab build
%                             process
```

Each subdirectory in the Papers directory houses one published paper and the files associated with its Demo. The process of building a "standard" release involves:

1. Appending copyright notices and date-of-modification information to all files in the library;

2. Adding the Matlab^{TM} source files to a zip file. This .zip file is the final version that will be made available on the WWW sites.

The copyright and date-of-modification information is appended automatically using the UNIX shell script and text file:

```
append_footer.sh        -   Appends a footer to all non-Contents .m files
SparseLab_Footer.txt    -   the footer that gets appended
```

An account named `SparseLab` is maintained on the leland system at stanford.edu, as is the website. The account serves several purposes:

1. The sub-directory `WWW` holds the files used to maintain our Web page.

2. The current version of SparseLab is always present on this account in the sub-directory `SparseLab`.

3. Feedback – questions, comments, suggestions, etc. – may be sent to the development team by e-mailing sparselab@stanford.edu.

4. Registration – there is an online web facility for users to register their use of SparseLab so that we can send e-mail about upgrades and enhancements.

The SparseLab package, version 100, is currently available online at `http://sparselab.stanford.edu` along with our papers, examples, documentation notes, contributor information, and collaborator acknowledgments.

Chapter 7

Discussion and Future Work

The notion of the Phase Diagram was borrowed from studies of sparse underdetermined equations and used to document the existence of a well-defined breakdown point for model selection algorithms, for linear regression with noise in the $p > n$ case. A Matlab^{TM} software toolbox was created housing open source routines for the reproduction of figures from papers in sparse representation.

7.1 Summary of Results

When the true underlying model is sufficiently sparse – less than the breakdown point – Classical Forward Stepwise regression, Forward Stepwise with False Discovery rate Thresholding, the Lasso, LARS, and Stagewise Orthogonal Matching Pursuit (StOMP) can find the true underlying model, up to a tolerance level γ. When the true underlying model uses a number of terms close to the number of observations, such model selection methods do not work well. We find that the Stepwise algorithms (Classical, FDR thresholding, and StOMP) exhibit different breakdown behavior than do the Lasso and LARS.

Depending on the threshold-to-enter used the Stepwise algorithms found different breakdown thresholds. Classical Stepwise with Bonferroni ($\sqrt{2\log(p)}$) thresholding, the algorithm ceased to recover the underlying model correctly at a particular sparsity level, regardless of the level of underdeterminedness. Using the False Discovery Rate threshold gave breakdown results similar to the theoretical threshold. Forward Stepwise with the Bonferroni threshold breaks down earlier (at sparser models) with an increase in model noise, and an increase in the total number of variables. Before the breakdown point, the size of coefficient errors follows the theoretical error distribution for the classical linear regression model. For a fixed level of underdeterminedness, $\delta = \frac{1}{2}$, the behavior of the root Mean Squared Error at breakdown is more heavily dependent on the sparsity level than the model noise level. The sparsity level at which algorithm performance breaks down can be predicted as a decreasing function of the model noise level. Increasing p reduces the sparsity level at which breakdown occurs.

The Lasso and LARS algorithms had a phase transition very similar to the theoretical breakdown curve. For the same fixed level of underdeterminedness, $\delta = \frac{1}{2}$, the root Mean Squared Error at breakdown was impacted more by changes in the model noise level than changes in the sparsity level.

7.2 Future Work

This thesis could be built upon in a number of ways. There are many more algorithms that could be studied using the methodology of the Phase Diagram: Backward Elimination, Forward Stagewise, Stochastic Search Variable Selection, and Bayesian Model Averaging for example.

The experimental setup used in this thesis ensured stochastic independence of the columns of the model matrix, and hence the variables were not correlated. In many empirical settings variables are likely to be correlated, so it would be useful to extend this analysis to the setting where the variables have specific levels of correlation, and then determine how each correlation level impacts the breakdown point for various algorithms.

All analyses of Forward Stepwise with False Discovery Rate Thresholding and StOMP set the False Discovery Rate $= q \equiv E(\frac{\{\#FalseDiscoveries\}}{\{\#TotalDiscoveries\}}) = 0.25$. Preliminary work suggests altering q has an impact on the breakdown point, with severe deterioration in algorithm performance when $q > 0.5$. A more thorough examination at different values of q could improve the overall performance of these algorithms.

Expanding analysis beyond $\delta = \frac{1}{2}$ would be useful since the nature of the breakdown in performance, for all algorithms except Classical Stepwise, depends on the level of indeterminacy δ. Extending the experiments to much larger p would also allow us to draw empirical conclusions about how the level of indeterminacy affects the breakdown point.

Recent work by T. Simila and J. Tikka [43] extend Efron et al.'s LARS algorithm to the case of multiresponse regression. A new multiresponse experimental setting could be developed to learn more about the breakdown characteristics of this algorithm. It would also be possible to carry out full component prediction (sucrose, fat, flour, water) in the near-infrared prediction example.

At this point no established theory exists to explain the convergence of the algorithms to different levels after the breakdown in performance occurs. Theory could also be developed to provide insight on the bavior of the breakdown point with respect to changes in the model noise level, the spasity of the true underlying model, and

the underdeterminedness of the model matrix. My work sheds light on these issues but would benefit from a fuller theoretical explanation. A theoretical explanation for the degradation in performance for the Lasso and LARS at very high indeterminacy levels would also help complete the body of work.

The breakdown point of Classical Forward Stepwise seems to depends on the number of variables, and the level of indeterminacy, but not the sparsity of the underlying model. This is unlike every algorithm studied, whose breakdown point depended on all three parameters. Further theoretical research is needed to explain this difference.

7.3 Implications for Practical Estimation Problems

The Lasso and LARS algorithms are able to recover the underlying model successfully at every determinacy/sparsity combination that every other algorithm can and some at which no other algorithm can. If your data design matrix approximates a random Gaussian matrix of the kind studied in this paper, these algorithms have the best chance at finding the true underlying model. The other algorithms (the Stepwise group) can recover the model with slightly smaller error when the true underlying model is very sparse.

The matlab toolbox, SparseLab, has dual aims. The first is to facilitate truly reproducible research by making publicly available research papers on sparse representation and all routines necessary for the regeneration of the paper's results. The code is provided fully open source, and so parameter settings and experimental conditions may be modified. The second goal is to package Matlab^{TM} tools for sparse representation together, with a uniform use and documentation presentation. A number

of solvers are included that are not explicited used in the associated research documents. It is hoped that ease of use of this code will facilite greater research into sparse systems of equations.

Bibliography

[1] F. Abramovich, Y. Benjamini, D. Donoho, and I. Johnstone. Adapting to unknown sparsity by controlling the false discovery rate. *Annals of Statistics*, 34(2):584–653, 2006.

[2] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. Roy. Statist. Soc. Ser. B*, 57:289–300, 1995.

[3] G. E. P. Box and R. D. Meyer. An analysis for unreplicated fractional factorials. *Technometrics*, 28(1):11–18, February 1986.

[4] P. Brown, T. Fearn, and M. Vannucci. Bayesian wavelet regression on curves with application to a spectroscopic calibration problem. *Journal of the American Statisical Association*, 96(454):398–408, July 2001.

[5] J. Buckheit and D. Donoho. About wavelab. Technical report, Stanford University, http://www-stat.stanford.edu/~wavelab, 1994.

[6] J. Buckheit and D. Donoho. Wavelab and reproducible research. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, pages 55–81. Springer-Verlag, 1995.

[7] J. Buckheit and D. Donoho. Wavelab architecture. Technical report, Stanford University, http://www-stat.stanford.edu/~wavelab, 1995.

[8] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Info. Theory*, 52(2):489–509, 2006.

[9] S. Chen and D. Donoho. On basis pursuit. Technical report, Stanford University, 1994.

[10] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[11] J. Claerbout. Hypertext documents about reproducible research. Technical report, Stanford University, http://sepwww.stanford.edu, 1994.

[12] Jan de Leeuw. Reproducible research. the bottom line. Technical report, UCLA, 2001.

[13] D. Donoho. De-noising by soft thresholding. *IEEE Transactions on Information Theory*, 41:613–627, 1995.

[14] D. Donoho. Neighborly polytopes and sparse solution of underdetermined linear equations. Technical Report 2005-04, Stanford University, February 2005.

[15] D. Donoho. High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension. *Discrete and Computational Geometry*, 35(4):617–652, 2006.

[16] D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proc. Natl. Acad. Sci. USA*, 100(5):2197–2202, 2003.

[17] D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Transactions on Information Theory*, 47(7):2845–2862, 2001.

[18] D. Donoho and X. Huo. Beamlab and reproducible research. *International Journal of Wavelets, Multiresolution and Information Processing*, 2(4):391–414, 2004.

[19] D. Donoho and J. Jin. Higher criticism for detecting sparse heterogeneous mixtures. *Annals of Statistics*, 32(3):962–994, 2004.

[20] D. Donoho and I. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[21] D. Donoho, I. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymtopia. *Journal of the Royal Statistical Society Series B*, 57:301–369, 1995.

[22] D. Donoho, V. Stodden, and Y. Tsaig. About sparselab. Technical report, Stanford University, http://sparselab.stanford.edu, 2006.

[23] D. Donoho, V. Stodden, and Y. Tsaig. Sparselab architecture. Technical report, Stanford University, http://sparselab.stanford.edu, 2006.

[24] D. Donoho, Y. Tsaig, I. Drori, and J. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions of Information Theory*, 2006. submitted.

[25] J. Dutka. On gauss' priority in the discovery of the method of least squares. *Archive for History of Exact Sciences*, 49(4):355–370, December 2004.

[26] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[27] M. Elad and A. Bruckstein. A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Trans. Inform. Theory*, 48(9):2558–4567, 2002.

[28] D. Singh et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, 1:203–209, March 2002.

[29] J. J. Fuchs. On sparse representations in abitrary redundant bases. *IEEE Trans. Info. Theory*, 50(6):1341–1344, 2004.

[30] G. M. Furnival and R. W. Wilson. Regression by leaps and bounds. *Technometrics*, 16:499–511, 1974.

[31] K. F. Gauss. *Theoria Motus Corporum Coelestium (Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections)*. Little, Brown, and Co., 1857. republished by Dover 1963.

[32] E. I. George and D. P. Foster. Calibration and empirical bayes variable selection. *Biometrika*, 87(4):731–747, 2000.

[33] E. I. George and R. E. McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

[34] E. I. George and R. E. McCulloch. Approaches for bayesian variable selection. *Statistica Sinica*, 7:339–373, 1997.

[35] R. Gribonval and M. Nielsen. Sparse representation in unions of bases. *IEEE Trans. Inform. Theory*, 49(12):3320–3325, 2003.

[36] A. E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58:54–59, 1962.

[37] X. Huo and X. Ni. Rsbn: Regression with stochastically bounded noise. Technical report, Georgia Institute of Technology, 2005.

[38] S. Mallat and S. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[39] X. Ni. *New Results in Detection, Estimation, and Model Selection.* PhD thesis, Georgia Institute of Technology, May 2006.

[40] B. Osbourne, T. Fearn, A. Miller, and S. Douglas. Application of near infrared reflectance spectroscopy to compositional analysis of biscuits and biscuit dough. *Journal of Science and Food Agriculture*, 35:99–105, 1984.

[41] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In A. Singh, editor, *Proc. 27th Asilomar Conference on Signals, Systems and Computers*. IEEE Comput. Soc. Press, 1993.

[42] M. Schwab, N. Karrenbach, and J. Claerbout. Making scientific computations reproducible. *Computing in Science & Engineering*, 2(6):61–67, 2000.

[43] T. Simila and J. Tikka. Common subset selection of inputs in multiresponse regression. pages 3574–3581. International Joint Conference on Neural Networks (IJCNN), July 2006.

[44] J. Sweetkind-Singer. Log-penalized linear regression. *Annals of Statistics*, 2005. forthcoming.

[45] R. Tibshirani. Regression shrinkage via the lasso. Technical report, Stanford University, June 1994.

[46] J. Tropp. Greed is good algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50(10):2231–2242, 2004.

[47] A. van't Wout et al. Cellular gene expression upon human immunodeficiency virus type 1 infection of cd4+-t-cell lines. *Journal of Virology*, 77(2):1392–1402, January 2003.

[48] S. Weisberg. *Applied Linear Regression*. John Wiley & Sons, 2 edition, 1985.

[49] B. Zhou. High frequency data and volatility in exchange rates. *Journal of Business and Economics Statistics*, 14(1):45–52, January 1996.