MA 750 - Final Project

Nate Josephs Matthew Wiens Ben Draves

December 3, 2017

Abstract

One of the most fundamental tasks in Statistics is to understand the relationship between two random variables, X, Y, via an unspecified function $Y = f(X)$. Typically, $f(\cdot)$ is unknown and must be estimated from data relating X and Y. Estimating $f(\cdot)$ using maximum likelihood yields no meaningful solution when we consider *all* functions. Hence statisticians turn to estimating $f \in \mathcal{F}$ where $\mathcal{F}$ is a function space with some structure that provides closed form solutions. In most cases however, these function spaces are fixed, with no regard to the sample from which we are trying to infer f. In order to utilize all information inherent in the data while still imposing structure on $\mathcal{F}$, *Sieve Estimation* allows $\mathcal{F}$ to grow in complexity as n increases. Heuristically, as n increases, we attain a more robust understanding of f and should allow our modeling procedure to consider more complex forms of f. Sieve Estimation achieves this by introducing more complex functions to $\mathcal{F}$ as n increases. Here, we consider the function space

$$\mathcal{F}_n = \left\{ g(x) : g(x) = \sum_{d=0}^{D} \alpha_d \phi_d(x) \right\}$$

for basis function ϕ_d and where $D \to \infty$ as $n \to \infty$. We focus our efforts on estimating D as a function of the data. We show that the optimal choice of D is asymptotically equivalent to the estimate chosen via cross-validation. We also show the importance of correct choice of D in an intensive simulation study and conclude with applying sieve estimation to a real dataset.

1 Introduction

In multivariate statistical analysis, statisticians often try to related two unknown quantities X,Y by an unspecified function $f(\cdot)$. In maximum likelihood settings, this problem has no closed form solution when we maximize over a general function space. Instead, we search over reduced function spaces with known structure and therefore impose that functional structure on our estimate $\hat{f}$. A lot of the statistical literature focuses on properties of these estimates when the function space is given. For example, the theory of linear models focuses

on estimates of the form

$$\mathcal{F} = \left\{ g(x) : g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \right\}$$

where $E[Y|X]$ is linear in the parameters of the model. While this class of estimators provides several nice properties, the function space is generally quite restrictive as it does not allow any complex nonlinear interactions. In nonparametric settings, statisticians enlarge the function space $\mathcal{F}$ by allowing the function to be locally adaptive. An example of a function space of this form is given by the Nadaraya-Watson regression estimators

$$\mathcal{F} = \left\{ g(x) : g(x) = \frac{\sum_{i=1}^{n} K_h(X_i - x) y_i}{\sum_{i=1}^{n} K_h(X_i - x)} \right\}$$

which is parameter free and instead relies on the sample data. It is clear that changing $\mathcal{F}$ greatly affects the estimates $\hat{f}$ and that the complexity of $\mathcal{F}$ can vary dramatically. Statisticians typically make *a priori* choices of $\mathcal{F}$ and only consider estimates of that form throughout their analysis of (X, Y). However, as one collects more data of the form (x_i, y_i), our understanding of $f(\cdot)$ should improve. That is, we should allow our estimates to become increasingly complex as our sample grows. *Sieve Estimation* enables this behavior by constructing a sequence of function spaces $\{\mathcal{F}_n\}_{n=1}^{\infty}$ such that $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots$. Then, depending on the sample size n, the estimation procedure is performed over the optimal space $\mathcal{F}_{opt}$.

This report is organized as follows: In Section 2 we summarize some of the foundational results on Sieve Estimation and introduce notation used throughout the report. In Section 3 we introduce the theoretically optimal estimate of D and a provide a data-dependent solution to this problem. In Section 4 we provide a simulation experiment and apply this estimator in Section 5 with a real data application.

2 Development of Sieves Estimation

2.1 Series Estimators

Sieve estimators are a general class of estimators that are applicable to a number of nonparametric problems such as density estimation and regression. Within this framework, several models from the econometric literature have been developed, addressing tasks such as logistic regression, imputation techniques, and measurement error. While sieve estimation is a general framework, we will focus on sieves for nonparametric regression.

A sieve estimator for the unknown mean function $g(x)$ in a regression model is defined by a *sequence* of functions approximating $g(x)$. With the additional constraints that the models should have finite dimension and the complexity (dimension) should increase as n increases, the most natural choice for the estimator is given by the *series estimator*

$$\hat{g}(x) = \sum_{d=0}^{D} \alpha_d \phi_d(x)$$

This formulation leads to traditional problems in linear models, so given the dimension of the model and a choice of basis function, finding the coefficients α has well known solutions. The constraints in the formulation are light. For instance, the choice of basis function must approximate functions in L^2. Though we will focus exclusively on linear sieves, examples of non-linear sieves include Neural Networks and penalized sieve estimators [1].

Given the series estimator, there are two choices to be made: the dimension of the model and the choice of basis function. D must be data dependent to scale the complexity of the estimator with the data. Therefore, for now we will assume there is an algorithm to choose D which produces a D_{opt}, for some sense of optimality. There are a number of choices for the basis functions and we will highlight just a few of the possibilities. We will discuss finding D_{opt} in the Section 3.

2.2 Basis Functions

One popular choice of functional estimator is given by the series estimator with polynomial basis functions. Within the polynomial class, choices of coefficients uniquely determine the exact form for the estimate. For example, the coefficient vector $\mathbf{c} = [0, 0 \dots, 1]$ would correspond to a basis of $\{x^k : k = 0, 1, 2, \dots\}$. The Hermite Polynomials would correspond to another choice of $\mathbf{c}$, which have desirable theoretical properties such as othronormality. While a powerful class of estimators, polynomial basis functions are not a natural choice when $g(x)$ goes to zero at $\pm\infty$. Using a Fourier Series resolves this issue, which again highlights the importance of an appropriate choice of basis function in a series estimator. With several applications in signal processing, the Fourier basis is proven to be a natural choice to approximate periodic functions. Lastly, the Gaussian basis set, given by $\{\varphi^{(d)} : d = 0, 1, 2, \dots\}$ where $\varphi^{(d)}$ is dth derivative of the Normal density, has several similar properties of the Fourier basis. Moreover, we see that these functions have a natural probabilistic interpretation (as do the Hermite Polynomials) which could be quite useful in inferential settings. The functional forms of these basis sets are given in Table 1.

Basis Functions	Functional Form
Polynomaials	$\sum_{i=0}^{d} c_d x^d$
Fourier	$a\ cos(\pi dx) + b\ sin(\pi dx)$
Gaussian	$\varphi^{(d)}(x)$

Table 1: Proposed series estimators with basis function ϕ for $\hat{g}(x)$

Other common choices for the basis function are Splines and Wavelets. Different splines can be used as the sieve estimator, however the exact form and behavior depends on the choice of spline and the number of free parameters it has. This result can be shown by considering a constrained optimization problem over the squares of the Dth derivatives of the class of potential sieve estimators. Also note that the choice of basis in the univariate case extends naturally to the multidimensional case, where the multivariate basis is constructed as a tensor produce of the univariate basis [1].

Under the setup of a series estimator with a choice of dimension and basis function, the sieve estimator has a number of similarities to the Kernel Density estimation problem. In both cases there are two choices to be made: one parameter that controls the bias-variance tradeoff and a second parameter that is related to underlying beliefs about the model. The bandwidth in the Kernel estimation problem is analogous to the choice of dimension. Intuitively, as the dimension increases or the bandwidth decreases the estimator is more sensitive to local behavior and produces a rougher estimate. Similarly, the choice of the Kernel is analogous to the choice of basis function. This choice impacts the final model and exact statistical properties of the estimator, yet is less interesting because any reasonable choice of basis function or kernel function produces a similar estimate.

2.3 Analysis of Series Estimators

To begin our formal analysis of the series estimator suppose we have some function $f(x)$ that can be well approximated by a series estimator. That is, we can write

$$y_i \equiv f(x_i) + e(x_i) = \sum_{d=0}^{\infty} \phi_d(x_i)\alpha_d + e(x_i)$$

where $\{\phi_d\}_{d=0}^{D}$ is a set of orthogonal basis functions and α_d are coefficients. Then the *sieves estimator* for this function is given by

$$\hat{f}(x_i) = \sum_{d=0}^{D} \phi(x_i)\alpha_d = \phi^T(x_i)\alpha$$

where $\phi^T(x_i) = (\phi_0(x_i), \phi_1(x_i), \ldots, \phi_D(x))^T$ and $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_D)$. Now organizing our matrices as follows

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, P_D = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \ldots & \phi_D(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \ldots & \phi_D(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \ldots & \phi_D(x_n) \end{bmatrix}, \alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_D \end{bmatrix}, e = \begin{bmatrix} e(x_1) \\ e(x_2) \\ \vdots \\ e(x_n) \end{bmatrix}$$

we can write our model as

$$Y = P_D\alpha + e$$

Recognizing this as a linear function in P_D gives rise to our OLS estimator of α

$$\hat{\alpha} = (P_D^T P_D)^{-1} P_D^T Y$$

From here we see that our estimates are given by

$$\hat{f}(x_i) = \phi^T(x_i)\hat{\alpha}$$

Due to the arsenal of statistical techniques for the linear regression model, we can extend this estimation procedure to account for correlated errors, weighted error structure, or even random effects. Notice that in the linear model framework these estimates would result in unbiased best estimates of the coefficients α *given that we specify* D *correctly*. In the case where we do not specify the mean function correctly, our estimate become unbiased and the variance also increases. This again highlights the importance of correctly choosing the dimension of the series estimator, which we discuss in the proceeding section.

3 Estimating the Dimension D

3.1 Mean Integrated Squared Error

In this framework, we introduce two types of error. The first is given by the error of the approximation. This is how well the series estimator matches the true underlying function. We will define this error by

$$r(x) = \phi^T(x)\alpha - f(x)$$

(*Notice that these are the true coefficient values*). The other source of error is of course the random variation around the true regression function. We will call this the residual error, $\epsilon(x_i)$. Then we see that from the original model $y_i = f(x_i) + e_i(x_i)$ that we can decompose the error as

$$e_i(x_i) = \epsilon(x_i) + r(x_i)$$

As we vary D we greatly reduce the variance in $r(x_i)$ as the more complex model will always account for more variance. But we need to be careful to ensure that the model is only reducing $r(x_i)$ and not $\epsilon(x_i)$. In order to find an analytical choice of D, we will find the Mean Integrated Squared Error (MISE). First, consider the following expansion.

$$\begin{aligned}
\hat{f}(x) - f(x) &= \phi^T(x)\hat{\alpha} - f(x) \\
&= \phi^T(x)\hat{\alpha} - \phi^T(x)\alpha + \phi^T(x)\alpha - f(x) \\
&= \phi^T(x)\hat{\alpha} - \phi^T(x)\alpha + r(x) \\
&= \phi(x)\phi^T(x)\left(\hat{\alpha} - \alpha\right) + r(x)
\end{aligned}$$

Assuming that the underlying X distribution is g, we have

$$\begin{aligned}
MISE(D) &= \int (\hat{f}(x) - f(x))^2 g(x)dx \\
&= \int \left(\phi^T(x)\left(\hat{\alpha} - \alpha\right) + r(x)\right)^2 g(x)dx \\
&= \int r(x)^2 g(x)dx + 2(\hat{\alpha} - \alpha)\int \phi^T(x)r(x)g(x)dx + (\hat{\alpha} - \alpha)^T \int \phi(x)\phi^T(x)g(x)dx(\hat{\alpha} - \alpha) \\
&= E[r^2(x)] + 2(\hat{\alpha} - \alpha)E[\phi^T(x)r(x)] + (\hat{\alpha} - \alpha)^T E[\phi(x)\phi^T(x)](\hat{\alpha} - \alpha)
\end{aligned}$$

Now recall that $r(x)$ was a projection error during our OLS estimate of α. Therefore, $r(x)$ and $\phi^T(x_i)$ exists in orthogonal spaces. Hence $E[\phi^T(x)r(x)] = 0$. Moreover, since ϕ is a

collection of orthogonal functions, the off-diagonal elements of $E(\phi(x)\phi(x)^T)$ are all zero. This gives

$$\begin{aligned} MISE(D) &= E[r^2(x)] + \text{tr}\Big[(\hat{\alpha}-\alpha)^T E[\phi(x)\phi(x)^T](\hat{\alpha}-\alpha)\Big] \\ &= E[r^2(x)] + tr\Big[E[\phi(x)\phi(x)^T]E\left((\hat{\alpha}-\alpha)(\hat{\alpha}-\alpha)^T\right)\Big] \end{aligned}$$

If $E[e(x)^2|x] = \sigma_x^2$, $\mathcal{Q} = E[\phi(x)\phi^T(x)]$ and $\Omega = E[\phi(x)\phi^T(x)\sigma_x^2]$, then [2] showed that

$$MISE(D) \simeq E[r^2(x)] + \frac{1}{n} tr\left(\mathcal{Q}^{-1}\Omega\right)$$

due to the asymptotic behavior of $MISE(D)$. Now, if we assume homoscedasticity, we see that $\Omega = \sigma^2\mathcal{Q}$. So plugging into the equation above we get

$$MISE^*(D) \simeq E[r^2(x)] + \frac{1}{n} tr\left(\sigma^2\mathcal{Q}^{-1}\mathcal{Q}\right) = E[r^2] + \frac{\sigma^2}{n}\dim(\mathcal{Q}) = E[r^2(x)] + \frac{\sigma^2 D}{n}$$

[2] also demonstrated that

$$MISE^*(D) = MISE(D)(1 + o(1))$$

so $MISE^*$ is close to $MISE$. But notice, even in this simple case we still require estimation of $r(x)$, which is directly related to the estimation of the true regression mean function $f(x)$, which is what we started to estimate in the first place. Therefore, while an excellent theoretical tool, using $MISE^*$ in practice is infeasible. We will, however, use this theoretical exercise to inform our data dependent choices of D which we give in the following sections.

3.2 Prediction Squared Error

As we have seen, MISE does not offer a feasible solution for selection D. Instead of MISE, one may be interested in calculating the Predicted Square Error (PSE) in order to find the optimal dimension with respect to PSE. If x^* is a new value from $X \sim f(x)$, then our prediction of Y given $X = x^*$ under the sieve estimator is $\hat{y}^* = \hat{f}(x^*)$. We then define PSE as the expectation of the squared error between Y^*, the actual value of the regression line at $X = x^*$, and $\hat{y}^*$. Observe that

$$
\begin{aligned}
PSE\Big(\hat{f}(x^*)\Big) &= E\Big[(Y^* - \hat{y}^*)^2\Big] \\
&= E\Big[\Big(f(x^*) + e^* - \hat{f}(x^*)\Big)^2\Big] \\
&= E\Big[\Big(e^* + (f(x^*) - \hat{f}(x^*)\Big)^2\Big] \\
&= E\big[e^{*2}\big] + 2E\Big[e^*\Big(f(x^*) - \hat{f}(x^*)\Big)\Big] + E\Big[\Big(f(x^*) - \hat{f}(x^*)\Big)^2\Big] \\
&= E\Big[(e^* - E[e^*])^2\Big] + 2E\big[e^*\big]E\Big[\Big(f(x^*) - \hat{f}(x^*)\Big)\Big] + E\Big[\Big(f(x^*) - \hat{f}(x^*)\Big)^2\Big] \\
&= Var(e^*) + 0 + \int E[(f(x) - \hat{f}(x))^2]g(x)dx \\
&= Var(e^*) + MISE\Big(\hat{f}(x)\Big)
\end{aligned}
$$

Hence, the optimal dimension for our sieve estimator with respect to PSE will be the same as the optimal dimension with respect to MISE, since minimizing PSE is equivalent to minimizing MISE. Note that in our derivation, we use the fact that the errors are zero mean and e^* is independent of the estimator.

3.3 Choosing D via Cross-Validation

If we define $\tilde{e} = Y^* - \hat{y}^*$, then $PSE\Big(\hat{f}(x^*)\Big) = E\big[\tilde{e}^2\big]$, which we may interpret as the expectation of a single leave-one-out squared prediction error, where our estimator is fit on $X_1, \ldots, X_n$ and validated against $X = x^*$. This motivates us to consider leave-one-out prediction errors for each $i = 1, \ldots, n$, which will reveal a data-driven process for choosing an optimal dimension D.

For each i, define $\tilde{e}_i = y_i - y_{\hat{(i)}}$ where $y_{\hat{(i)}}$ is fit on $X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n$. Then $PSE\Big(\hat{f_{(i)}}(x)\Big) = E\big[\tilde{e_i}^2\big]$ and we define the cross-validation (CV) criterion as

$$
CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} \tilde{e}_i
$$

.

By the linearity of expectation, we have that $E\Big[CV(\hat{f})\Big] = PSE\Big(\hat{f}(x^*)\Big)$, which we will utilize as our data-driven procedure for choosing D_{opt}. [2] demonstrated this estimate of D has the nice property that it is asymptotically equivalent to the D chosen from $MISE$.

4 Simulation Study

We simulated $N = 150$ data points from the model

$$Y = \sin^3(2\pi X^3) + \epsilon \quad \text{where} \quad X \sim N(0,1) \quad \text{and} \quad \epsilon \sim N(0,.2)$$

with the goal of fitting a polynomial series estimator to the data. From $n = 10$ to 150 in increments of 10, we fit such models with degrees of complexity varying from 1 to 8. We repeated this on 200 separate datasets and plotted the PSE against n. The results of the simulation are summarized in Figure 1. As expected, the PSE decreases as n increases.

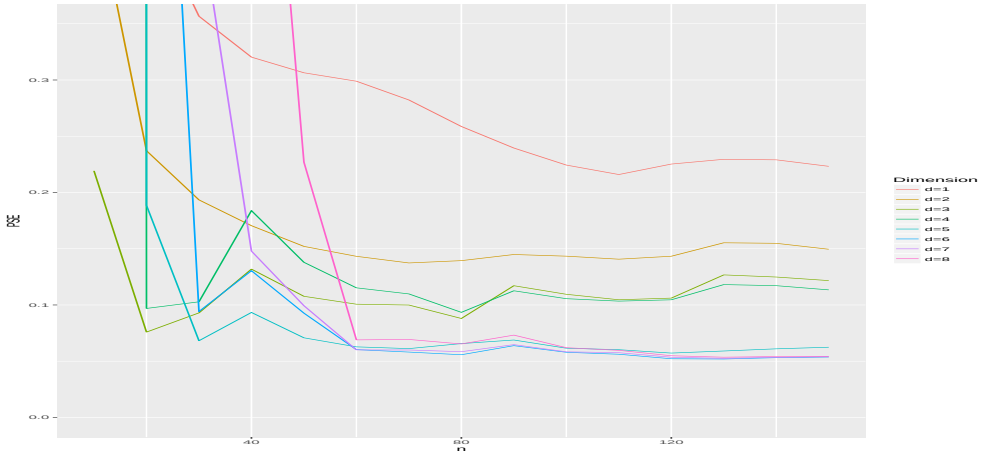

Figure 1: The effect of the n on PSE. From the graph we clearly see the optimal functional form relies on the sample size, n.

Furthermore, the optimal degree increases with n. A linear model is best only for $n = 10$. At $n = 20$, a quadratic fit minimizes the PSE. However, by $n = 30$ a fifth degree polynomial is best and remains so until $n = 50$. Ultimately, a sixth degree polynomial performs very well and is not supplanted by a seventh degree polynomial until $n = 90$. We note that an eighth degree polynomial never minimizes the PSE.

5 Real Data Application

5.1 Geyser Data Example

Consider a canonical regression problem: the waiting period of the Old Faithful geyser and a function of the previous eruption duration. Here, we demonstrate that the sieve estimator gives similar results as other nonparametric regression estimates, but with many fewer degrees of freedom. The *faithful* dataset has 272 data points, which is small, but sufficient to demonstrate the sieve estimator. As in the simulation above, a polynomial basis is used.

First, consider a subset of the geyser data with only 40 data points. Here, as expected, sieve estimators with a low dimension perform the best. However, note that polynomials of low degree still perform better than simply a linear model.

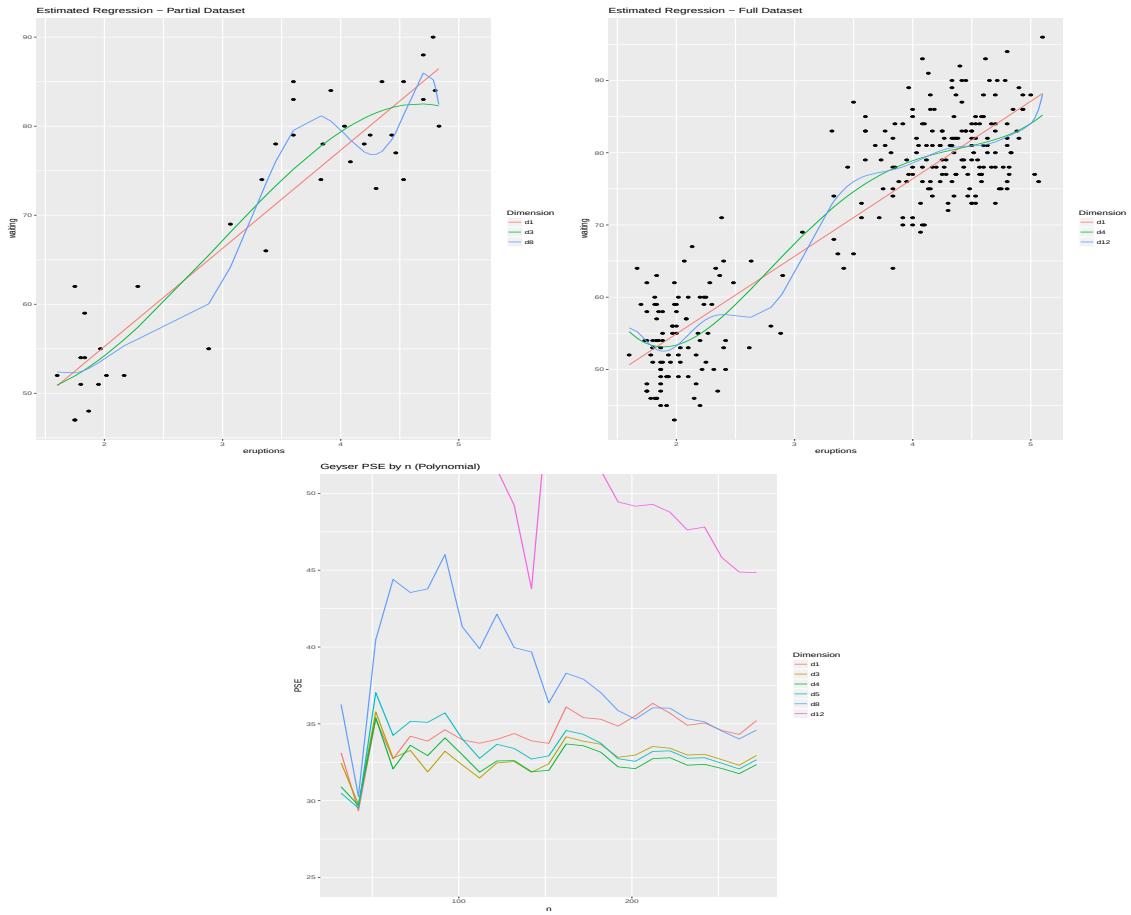

Figure 2: Sieves Analysis of Geyser dataset. We see that increasing our sample size leads to differing optimal models

With the full 272 data points, the optimal dimension is 4, which has increased from the case with only 40 data points. Note that the optimal dimension increases much more slowly than N. Again, if D grows too quickly, like $D = 11$, then the data is overfit and the variance is large.

5.2 Econometric Example

Sa and Portugal [4] apply sieve estimators to model a loss function of inflation and output gap for the Brazilian Central Bank and the Federal Reserve for their determination of interest rates. The sieve estimator is chosen because the sieve estimator allows for global computation of the derivatives of the estimator. Therefore, a polynomial basis is a natural choice for the easy interpretation of derivatives in the model. Specifically, a Chebyshev Polynomial basis

is used because the loss function is defined on $[-1, 1]$. A set of 14 variables is chosen to be the basis and a sieve estimator is fit. If any of the coefficients of degree 3 or higher are significantly different than zero, then the authors would conclude the third derivative is nonzero and therefore the loss function is asymmetric.

In the sieve estimator, the cross-term between inflation and the output gap is near zero, so the problem is separable into the inflation and output gap components. Based on the sieve estimator the Federal Reserve was more concerned about inflation than the output gap, which confirmed other studies. Furthermore, the sieve estimator does have nonzero third derivative with respect to inflation, so it is concluded that the Federal Reserve was more concerned about high inflation than deflation from 1960-2011. However, by considering only the years from 1982-2011, then the authors determined that there was not enough data to significantly conclude the presence of asymmetry in the loss function.

This example uses sieve estimators in a multivariate inferential context, which is a natural extension of the single dimensional estimator discussed in Sections 2 and 3. However, the advantage of the sieve estimator are apparent by specifying a general, closed-form, global functional form which results in intuitive determination of parameters of interest in the economic model.

6 Conclusion

In this report, we develop the methodology of a general sieve estimator, discuss the properties of a polynomial series sieve estimator, and explore data driven applications. Sieve estimators are general estimators and are able to model many different classes of mean functions. Like other nonparametric estimators, two key choices are the choice of basis function, which is determined by properties of the problem, and the choice of dimension, which controls the complexity of the estimator. We propose selecting the dimension through a cross-validation scheme. Solving the polynomial series sieve estimator, after choosing the dimension and basis, is equivalent to solving ordinary least squares. With this property, we derive the MISE and justify the use of predicted squared error as a criterion to select the dimension.

Additionally, we evaluate the empirical use of the sieve estimator through simulation and observed data sets. The simulation illustrates how the optimal dimension of the estimator increases as the amount of data increases. Moreover, the dimension grows much more slowly than N, which results in the asymptotic properties of the MISE. In the Old Faithful example, we show how a sieve estimator naturally incorporates additional data to better approximate the mean function with much less model complexity than other nonparametric regression estimators.

Further exploration could be done by considering a penalized sieve estimator as a way to control model complexity. Little work has been done on testing with sieve estimators, and the simple functional forms of the estimators may lead to useful results for global properties of the mean function.

7 References

Code for the analysis and figures is available at `https://github.com/matthewrw/Sieves_Project/tree/master/R_code` and may be used freely with proper acknowledgment.

[1] Xiaohong Chen. "Large Sample Sieve Estimation of Semi-Nonparametric Models". In: *Handbook of Econometrics, Volume 6B*. Elsevier B.V., 2007.

[2] Bruce Hansen. "Nonparametric Sieve Regression: Least Squares, Averaging Least Squares, and Cross-Validation". In: *The Oxford Handbook of Applied Nonparametric and Semi-parametric Econometrics and Statistics*. Ed. by Jeffrey Racine, Liangjun Su, and Aman Ullah. Oxford University Press, 2012.

[3] Andres Munk-Nielson. *The Method of Sieves*. 2016. URL: `http://www.econ.ku.dk/munk-nielsen/notes/sieve_note.pdf`.

[4] Rodrigo de Sa and Marcelo S. Portugal. "Central bank and asymmetric preferences: An application of sieve estimators in the U.S. and Brazil". In: *Economic Modeling* 51 (2015), pp. 72–83.

[5] Liangjun Su. *Nonparametric Sieve Estimation*. 2014. URL: `http://www.mysmu.edu/faculty/ljsu/Econ713_files/Ch4.pdf`.

A Simulation Code

```
library(ggplot2)
library(gridExtra)

N <- seq(10, 150, by = 10)
D <- seq(1:8)
error_plot_avg <- data.frame(n = integer(), pse = numeric(), d = character())

## PSE
seeds <- seq(1:200)

for(seed in seeds){
  # sample data
  set.seed(seed)
  X_i <- runif(max(N), 0, 1)
  e_i <- rnorm(max(N), 0, 0.2)
  Y_i <- sin(2*pi*X_i^3)^3 + e_i

  # initialize df for plotting
  error_plot <- data.frame(n = integer(), pse = numeric(), d = character())

  # compute CV for dimension-D model on N datapoints
  for (d in D) {
    d_plot <- data.frame(n = N, pse = rep(0, length(N)), d = paste0("d=", d))
    for (n in N) {
      m_hat = lm(Y_i[1:n] ~ poly(X_i[1:n], d))
      pse = sum((residuals(m_hat) / (1 - hatvalues(m_hat)))^2) / n
      d_plot[d_plot$n == n, "pse"] <- pse
    }
    error_plot <- rbind(error_plot, d_plot)
  }
  if(nrow(error_plot_avg) == 0){
    error_plot_avg <- error_plot
  }else{
    error_plot_avg[, "pse"] <- error_plot_avg[, "pse"] + error_plot[, "pse"]
  }}

error_plot_avg[, "pse"] <- error_plot_avg[, "pse"] / length(seeds)

# plot
ggplot(data = error_plot, aes(x = n, y = pse, group = d)) +
  geom_line(aes(colour = d)) +
  coord_cartesian(ylim = c(0, 0.35)) +
  labs(x = "n", y = "PSE", color = "Dimension")

frames = 15
```

```
for(i in 1:frames){
  # creating a name for each plot file with leading zeros
  if (i < 10) {name = paste('000',i,'plot.png',sep='')}

  if (i < 100 && i >= 10) {name = paste('00',i,'plot.png', sep='')}
  if (i >= 100) {name = paste('0', i,'plot.png', sep='')}

  n <- 10*i

  min_pse = min(error_plot_avg[error_plot_avg$n == n, "pse"])
  min_d = error_plot_avg[error_plot_avg$n == n & error_plot_avg$pse == min_pse, "
      d"]
  min_d = as.integer(substr(min_d, 3, 3))

  d_sizes <- rep(.5, length(D))
  d_sizes[min_d] = 2

  #saves the plot as a .png file in the working directory
  png(name)
  p <- ggplot(data = error_plot[error_plot$n <= n, ]
              , aes(x = n, y = pse, group = d)) +
    geom_line(aes(colour = d, size = d)) +
    scale_size_manual(values = d_sizes, guide = FALSE) +
    coord_cartesian(ylim = c(0, 0.35), xlim = c(0, 150)) +
    guides(colour = guide_legend(override.aes = list(size = d_sizes))) +
    labs(x = "n", y = "PSE", color = "Dimension")
  print(p)
  dev.off()
}

## Fitted Results
# sample data
set.seed(750)
X_i <- runif(max(N), 0, 1)
e_i <- rnorm(max(N), 0, 0.2)
Y_i <- sin(2*pi*X_i^3)^3 + e_i

true_reg <- function(x){sin(2*pi*x^3)^3}

opt_D <- c(1, 2, 5, 5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7)

for(i in 1:frames){
  # creating a name for each plot file with leading zeros
  if (i < 10) {name = paste('000',i,'fit.png',sep='')}
  if (i < 100 && i >= 10) {name = paste('00',i,'fit.png', sep='')}
```

```
  n <- 10*i

  min_pse = min(error_plot_avg[error_plot_avg$n == n, "pse"])
  min_d = error_plot_avg[error_plot_avg$n == n & error_plot_avg$pse == min_pse, "
      d"]
  min_d = as.integer(substr(min_d, 3, 3))

  d_sizes <- rep(.5, length(D))
  d_sizes[min_d] = 2

  #saves the plot as a .png file in the working directory
  png(name)
  p <- ggplot(data = error_plot[error_plot$n <= n, ]
              , aes(x = n, y = pse, group = d)) +
    geom_line(aes(colour = d, size = d)) +
    scale_size_manual(values = d_sizes, guide = FALSE) +
    coord_cartesian(ylim = c(0, 0.35), xlim = c(0, 150)) +
    guides(colour = guide_legend(override.aes = list(size = d_sizes))) +
    labs(x = "n", y = "PSE", color = "Dimension")

  # fitted values
  df <- data.frame(x = X_i[1:N[i]], y = Y_i[1:N[i]])
  m_hat <- lm(y ~ poly(x, opt_D[i]), data = df)
  ix <- sort(x, index.return = TRUE)$ix
  m_hat <- approxfun(x[ix], predict(m_hat)[ix])
  grid <- seq(0, 1, by = .01)
  pred_df <- data.frame(x = grid, pred_y = m_hat(grid))

  f <- ggplot(data = df, aes(x = x, y = y)) +
    geom_point() +
    geom_line(data = pred_df, aes(y = pred_y, x = x)) +
    coord_cartesian(ylim = c(-1.2, 1.2), xlim = c(0, 1))

  grid.arrange(p, f)

  dev.off()
}

# ezgif.com for making gif
```

B Geyser Data Code

```
Y_i <- faithful$waiting
X_i <- faithful$eruptions
N  <- seq(32,272, by=10)
```

```
# initialize df for plotting
D <- seq(1,12)
error_plot <- data.frame(n = integer(), pse = numeric(), d = character())
predicted_df <- data.frame(d = character(), eruptions = numeric(), waiting =
    numeric())
# compute CV for dimension-D model on N datapoints
p <- ggplot(data =faithful[1:max(N),] , aes(x = eruptions, y = waiting) ) +
    geom_point() + labs(title = "Estimated Regression - Full Dataset", color = "
    Dimension")
for (d in D) {
  d_plot <- data.frame(n = N, pse = rep(0, length(N)), d = paste0("d", d))
  for (n in N) {
    m_hat <- lm(Y_i[1:n] ~ poly(X_i[1:n], d))
    pse = sum((residuals(m_hat) / (1 - hatvalues(m_hat)))^2) / n
    d_plot[d_plot$n == n, "pse"] <- pse

  }
  error_plot <- rbind(error_plot, d_plot)
  #col <- c("blue","blue",'red','red','orange','orange','orange','green','green
      ','green','green','green')
  m_hat_predict <- approxfun(X_i[1:n],m_hat$fitted.values)
  predicted_df <- rbind(predicted_df,data.frame(d = paste0("d", d),eruptions =
      seq( min(X_i),max(X_i),length.out=1000), waiting = m_hat_predict( seq( min(
      X_i),max(X_i),length.out=1000))))

}
p + geom_line( data = predicted_df[predicted_df$d %in% c("d1","d4","d12"),] , aes
    (y = waiting, x = eruptions, group = d, colour = d)) + labs(title = "Estimated
     Regression - Full Dataset", color = "Dimension")



# plot
ggplot(data = error_plot[error_plot$d %in% c("d1","d3","d4","d5","d8","d12"),],
    aes(x = n, y = pse, group = d)) +
  geom_line(aes(colour = d)) +
  coord_cartesian(ylim = c(25, 50)) +
  labs(title = "Geyser PSE by n (Polynomial)", x = "n", y = "PSE", color = "
      Dimension")

# Now compute estimator for small number of data points

n <- 40
D <- seq(1,11)
error_plot <- data.frame(n = integer(), pse = numeric(), d = character())
```

```
# compute CV for dimension-D model on N datapoints

predicted_df <- data.frame(d = character(), eruptions = numeric(), waiting =
    numeric())
p <- ggplot(data =faithful[1:n,] , aes(x = eruptions, y = waiting)) + geom_point
    () + labs(title = "Estimated Regression - Partial Dataset", color = "Dimension
    ")


for (d in D) {
    m_hat <- lm(Y_i[1:n] ~ poly(X_i[1:n], d))
    pse = sum((residuals(m_hat) / (1 - hatvalues(m_hat)))^2) / n
    d_plot[d_plot$n == n, "pse"] <- pse

  m_hat_predict <- approxfun(X_i[1:n],m_hat$fitted.values)
  predicted_df <- rbind(predicted_df,data.frame(d = paste0("d", d),eruptions =
      seq( min(X_i),max(X_i),length.out=1000), waiting = m_hat_predict( seq( min(
      X_i),max(X_i),length.out=1000))))

}
p + geom_line( data = predicted_df[predicted_df$d %in% c("d1","d3","d8"),] , aes(
    y = waiting, x = eruptions, group = d, colour = d))
```