# Smooth Regression

February 17, 2025

In this homework, we'll briefly review Bounded Variation Regression and then explore Lipschitz Regression, another form of smooth regression. We will focus on the one-dimensional case, although it extends very naturally to higher dimensions. Then we'll look into rates of convergence, comparing this new method to the stuff we've been using.

In my code, I'll be using a few libraries.

```
library(CVXR)
CVXR::add_to_solver_blacklist('OSQP')
# OSQP claims some feasible problems aren't
```

And some functions we've been using in labs.

```
invert.unique = function(x) {
  o = order(x)
  dup = duplicated(x[o])
  inverse = rep(NA, length(x))
  inverse[o] = cumsum(!dup)
  list(elements=o[!dup], inverse=inverse)
}

prediction.function = function(model) {
  function(x) { predict(model, data.frame(X=x)) }
}
```

## 1 Review of Bounded Variation Regression

In class, we talked about using least squares regression to fit a function of *bounded total variation*. If we are fitting $\mu(x) = \mathrm{E}[Y_i \mid X_i = x]$ for covariates

$X_i \in [0,1]$, this estimator is

$$\hat{\mu} = \operatorname*{argmin}_{\substack{m \\ \rho_{TV}(m) \leq B}} \frac{1}{n} \sum_{i=1}^{n} \{Y_i - m(X_i)\}^2 \qquad \text{where}$$

$$\begin{aligned} \rho_{TV}(m) &= \int_0^1 |m'(x)| dx && \text{for differentiable } m \\ &= \sup_{\substack{\text{increasing sequences} \\ x_1 \leq x_2 \leq \ldots \leq x_k \\ x_1 \ldots x_k \in [0,1]}} \sum_j |m(x_{j+1}) - m(x_j)| && \text{generally .} \end{aligned} \tag{1}$$

The set of functions we're optimizing over, those with $\rho_{TV}(m) \leq B$, is a set of functions that doesn't vary too much in total. It does, however, include both functions that vary slowly throughout the interval $[0,1]$ and those that vary quickly for a small part of it.

**Exercise 1** *To get a sense of what the constraint $\rho_{TV}(m) \leq B$ means, calculate $\rho_{TV}(m)$ for the following functions on $[0,1]$. These are repeats from class.*

1. $m(x) = x$
2. $m(x) = x^2$
3. $m(x) = e^x$
4. $m(x) = \sin(\pi x)$
5. $m(x) = \begin{cases} 0 & \text{if } x < 1 \\ 1 & \text{if } x = 1 \end{cases}$
6. $m(x) = \sin(1/x)$

**Solution 1**

1. *$m(x) = x$ increases from $0$ to $1$ on $[0,1]$, so its total variation is $1$.*
2. *$m(x) = x^2$ increases from $0$ to $1$ on $[0,1]$, so its total variation is $1$.*
3. *$m(x) = e^x$ increases from $1$ to $e$ on $[0,1]$, so its total variation is $e - 1$.*
4. *$m(x) = \sin(\pi x)$ increases from $0$ to $1$ on $[0,1/2]$ then decreases back to zero on $[1/2,1]$, so its total variation is $1 + 1 = 2$.*
5. *This step function increases from $0$ to $1$ on $[0,1]$, so its total variation is $1$.*
6. *$\sin(1/x)$ oscillates from $-1$ to $1$ on the interval $[1/x_0, 1]$ as $\sin(x)$ does on the range $[1, 1/x_0]$, i.e., approximately $(1/x_0 - 1)/(2\pi)$ times. It follows, taking $x_0$ to zero, that it passes from $-1$ to $1$ infinitely many times on $[0,1]$, so its total variation is $\infty$.*

**Exercise 2** *For the following, which are a bit subtler, give an upper bound on $\rho_{TV}(m)$. If it's infinite, explain why.*

1. $m(x) = x\sin(1/x)$
2. $m(x) = x^2\sin(1/x)$
3. $m(x) = x^{3/2}\sin(1/x)$

**Hint:** It might be hard to find the upper bound by looking at the graph of some of these functions. Instead, find the derivative and a corresponding upper bound for it, if possible. Your upper bound doesn't have to be tight. When you are bounding a sum, use the triangle inequality by adding the upper bound for each term.

**Solution 2** *We'll take a more calculus-based approach here. For the first, $\rho_{TV}(m) = \infty$.*

$$m'(x) = \{x\sin(x^{-1})\}' = \sin(x^{-1}) + x\cdot\cos(x^{-1})\cdot -1/x^2 = \sin(x^{-1}) - \cos(x^{-1})/x.$$

*In a neighborhood $[0,\epsilon]$ of zero, $\cos(x^{-1})/x$ oscillates between $-1/x$ and $1/x$ infinitely many times. Meanwhile, the sine is in $[-1,1]$, so it'll have negligible impact on the absolute value of $m'(x)$—it'll be roughly $|\cos(x^{-1})|/x$. And because $|\cos(x^{-1})|$ is going to be bounded away from zero for most $x$ near zero, this means its integral is going to be at least a constant multiple of the integral $\int_0^\epsilon 1/x$. And that integral is infinite.*

*The second is a bit different. $\rho_{TV}(m) \le 3$.*

$$m'(x) = \{x^2\sin(x^{-1})\}' = 2x\cdot\sin(x^{-1}) + x^2\cos(x^{-1})\cdot -1/x^2 = 2x\sin(x^{-1}) - \cos(x^{-1}).$$

*It follows, via the triangle inequality, that*

$$|m'(x)| \le 2x|\sin(x^{-1})| + |\cos(x^{-1})| \le 3 \quad \text{on} \quad [0,1].$$

*Integrating, we get the bound $\rho_{TV}(m) \le \int_0^1 3 = 3$.*

*The third is fairly similar to the second. $\rho_{TV}(m) \le 5$.*

$$\begin{aligned} m'(x) = \{x^{3/2}\sin(x^{-1})\}' &= (3/2)x^{-1/2}\sin(x^{-1}) - x^{3/2}\cos(x^{-1})\cdot -1/x^2 \\ &= x^{-1/2}\{(3/2)\sin(x^{-1}) - \cos(x^{-1})\}. \end{aligned}$$

*If follows, via the triangle inequality, that*

$$|m'(x)| \le x^{-1/2}\{(3/2)|\sin(x^{-1})| + |\cos(x^{-1})|\} \le (5/2)x^{-1/2}.$$

*Integrating, we get the bound $\rho_{TV}(m) \le (5/2)\int_0^1 x^{-1/2} = (5/2)\cdot 2x^{1/2}\,|_0^1 = 5$.*

## 2 Lipschitz Regression

In some cases, it may be implausible that $\mu(x)$ varies quickly anywhere. In that case, we may prefer to fit a *Lipschitz function*, for example by solving the following least squares problem.

$$\begin{aligned} \hat{\mu} &= \operatorname*{argmin}_{m:\, \rho_{Lip}(m) \leq B} \frac{1}{n} \sum_{i=1}^{n} \{Y_i - m(X_i)\}^2 \quad \text{where} \\ \rho_{Lip}(m) &= \sup_{x \in [0,1]} |m'(x)| \qquad\qquad \text{for differentiable } m \\ &= \sup_{\substack{x_1, x_2 \in [0,1] \\ x_1 \neq x_2}} \frac{|m(x_2) - m(x_1)|}{|x_2 - x_1|} \qquad \text{generally .} \end{aligned} \tag{2}$$

We call $\rho_{Lip}(m)$ the *Lipschitz constant* of the function $m$. Let's interpret the general definition visually. It's a maximum of the absolute value of the slope of the functions's secants.

**Equivalence.** Our definition for differentiable functions is equivalent because (i) derivatives are included in the set of slopes we're maximizing over, as they are the slopes of tangents, which are just very short secants (ii) every slope in this set is equal to a derivative, as the mean value theorem tells us that the slope of the secant drawn from $x = a$ to $x = b$ is equal to the derivative of the function at some point between $a$ and $b$.

### 2.1 Finding Lipschitz Constants

**Exercise 3** *To get a sense of what this new type of constraint $\rho_{Lip}(m) \leq B$ means, calculate $\rho_{Lip}(m)$ for the examples from Exercise 1. Bound it or explain why it's infinite for the examples from Exercise 2. Is $\rho_{TV}(m) \leq \rho_{Lip}(m)$ for all of these examples? If so, either prove that it's true for all functions $m$ on $[0, 1]$ or find a counterexample.*

**Solution 3** *In the examples, the Lipschitz constants are as follows, with* $\max_x$ *taken over $x \in [0, 1]$.*

1. *The Lipschitz constant of $m(x) = x$ is $\max_x |1| = 1$.*
2. *The Lipschitz constant of $m(x) = x^2$ is $\max_x |2x| = 2$.*
3. *The Lipschitz constant of $m(x) = e^x$ is $\max_x |e^x| = e$.*
4. *The Lipschitz constant of $m(x) = \sin(\pi x)$ is $\max_x |\pi \cos(\pi x)| = \pi$.*
5. *The Lipschitz constant of this step function is $\infty$.*
   *As $x \uparrow 1$, $|m(1) - m(x)|/|1 - x| \to 1/0$.*

6. *The Lipschitz constant of* $\sin(1/x)$ *is* $\infty$.
   *Let* $x_k = 1/(k\pi)$ *and* $x'_k = 1/\{(k+1/2)\pi\}$ *and take* $k \to \infty$.

   $$\sin(1/x'_k) - \sin(1/x_k) = \sin(\{k+1/2\}\pi) - \sin(k\pi) = 1 - 0 = 1 \quad \textit{and} \quad x'_k - x_k \to 0.$$

*and*

1. *The Lipschitz constant of* $x\sin(1/x)$ *is infinite, as its derivative is* $-1/x$ *for some points* $x$ *arbitrarily close to zero. In particular, that happens at* $x = 1/(2\pi k)$ *for any integer* $k$, *where* $\cos\left(x^{-1}\right) = 1$ *and* $\sin\left(x^{-1}\right) = 0$.

2. *The Lipschitz constant of* $x^2\sin(1/x)$ *is less than or equal to* 3 *by the same argument we used to bound its total variation.*

3. *The Lipschitz constant of* $x^{3/2}\sin(1/x)$ *is infinite, as its derivative is* $-x^{-1/2}$ *for points* $x = 1/(2\pi k)$ *that are arbitrarily close to zero.*

*All of these where we've done the exact calculation are greater than or equal to the corresponding total variation. Furthermore, we've found the Lipschitz constant to be infinite where total variation is and sometimes where it is not. In fact, this is true for all functions* $m$. *To see this, consider any increasing sequence* $0 = x_1 < x_2 \ldots < x_k = 1$. *Then, since*

$$\frac{|m(x_{j+1}) - m(x_j)|}{x_{j+1} - x_j} \leq \rho_{Lip}(m) \quad \textit{for all } j, \textit{ it follows that}$$

$$\sum_j |m(x_{j+1}) - m(x_j)| \leq \sum_j \rho_{Lip}(m)(x_{j+1} - x_j) = \rho_{Lip}(m)\sum_j (x_{j+1} - x_j) = \rho_{Lip}(m).$$

*As* $\rho_{Lip}(m)$ *is an upper bound on this sum for all increasing sequences, it follows that it is at least as large as the* least upper bound, $\rho_{TV}(m)$.

## 2.2 Fitting the Lipschitz Model

As usual, we'll start by solving for a function $\hat{\mu}_{|\mathcal{X}}$ on the sample $\mathcal{X} = \{X_1 \ldots X_n\}$, then extend it to the real line. Just like in the bounded variation lab, we'll translate our seminorm $\rho_{Lip}$ into a seminorm on functions $m : \mathcal{X} \to \mathbb{R}$ simply by replacing the 'for all ... $\in \mathbb{R}$' (or $[0,1]$) with a 'for all ... $\in \mathcal{X}$'. Like this.

$$\begin{aligned}
\hat{\mu}_{|\mathcal{X}} &= \operatorname*{argmin}_{\substack{m \\ \rho_{Lip|\mathcal{X}}(m) \leq B}} \frac{1}{n}\sum_{i=1}^{n}\{Y_i - m(X_i)\}^2 \quad \text{where} \\
\rho_{Lip|\mathcal{X}}(m) &= \sup_{x \neq x' \in \mathcal{X}} \frac{|m(x) - m(x')|}{|x - x'|} \qquad (3) \\
&= \sup_{\substack{\text{pairs } i,j \in 1\ldots n \\ \text{with } X_i \neq X_j}} \frac{|m(X_i) - m(X_j)|}{|X_i - X_j|}.
\end{aligned}$$

This is something we can handle. This depends on the values of $m(x)$ only at the observed data points $x \in \{X_1 \dots X_n\}$, so we can implement it as an optimization over a vector $\vec{m} \in \mathbb{R}^n$ with the interpretation that $\vec{m}_i = m(X_i)$. The constraint $\rho_{Lip|\mathcal{X}}(m) \leq B$ can be expressed as a set of constraints on $\vec{m}_i - \vec{m}_j$ for pairs $i, j$.

**Exercise 4** *Rewrite this problem as a constrained optimization over the vector $\vec{m}$. Try to do it so what you've written translates straightforwardly into CVXR code.*

**Tip.** There is a smaller set of constraints that implies the full set in (3). We'll get there in Exercise 10. For now, use the full set, like we did until the section 'Optional Exercise: Optimization' in the monotone regression lab.

**Tip.** If you want to keep things simple, go ahead and assume that $X_1 \dots X_n$ take on $n$ distinct values, just like we did at the beginning of the monotone regression lab. If you want more generally applicable code, take a look at how we use `invert.unique` in the monotone regression lab to handle duplicate values.

**Tip.** CVXR seems to be having some trouble with this one if we use division in our constraint, so don't. To write your constraint without division, observe that the following set of constraints are equivalent: (i) $\max_{i \leq n} |u_i / v_i| \leq B$, (ii) $|u_i|/|v_i| \leq B$ for all $i \in 1 \dots n$, and (iii) $|u_i| \leq B|v_i|$ for all $i \in 1 \dots n$.

**Solution 4** *A function with $\hat{\mu}(X_i) = \hat{\mu}_i$ solves* (3) *if the vector $\hat{\mu}$ solves*

$$\begin{aligned} \hat{\mu} = \operatorname*{argmin}_{\vec{m} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \{Y_i - \vec{m}_i\}^2 \qquad & \textit{subject to the constraints} \\ \max_{\substack{i,j \in 1 \dots n \\ X_i \neq X_j}} \frac{|\vec{m}_i - \vec{m}_j|}{|X_i - X_j|} \leq B \quad \textit{and} \quad & \max_{\substack{i,j \in 1 \dots n \\ X_i = X_j}} |\vec{m}_i - \vec{m}_j| = 0. \end{aligned} \tag{4}$$

*Here the first constraint is our Lipschitz constraint and the second ensures that we're optimizing over functions; $m(X_i)$ and $m(X_j)$ must be equal if $X_i = X_j$. We encode these constraints more compactly in this equivalent problem.*

$$\begin{aligned} \hat{\mu} = \operatorname*{argmin}_{\vec{m} \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \{Y_i - \vec{m}_i\}^2 \qquad & \textit{subject to the constraints} \\ |\vec{m}_i - \vec{m}_j| \leq B|X_i - X_j| \qquad & \textit{for all } \; i, j \in 1 \dots n. \end{aligned} \tag{5}$$

*The subset of constraints in* (5) *on pairs $X_i, X_j$ with $X_i \neq X_j$ is equivalent to the Lipschitz constraint from* (4) *while the subset on pairs with $X_i = X_j$ is equivalent to the constraint that we're optimizing over functions.*

**Exercise 5** *Implement that optimization in* **R**. *That is, write an* **R** *function* **lipreg** *analogous to* **monotonereg** *from the monotone regression lab that solves* (3). *Then, from the six distributions described below, sample* $n = 100$ *observations* $(X_1, Y_1) \dots (X_n, Y_n)$ *and use your code to calculate predictions* $\hat{\mu}(X_1) \dots \hat{\mu}(X_1)$ *based on the solution to* (3) *with variation bound* $B = 1$. *Each time, plot your predictions on top of the data, i.e., make a single scatter plot showing both your predictions* $(X_i, \hat{\mu}(X_i))$ *and your observations* $(X_i, Y_i)$. *Turn in those six plots, labeling each with the signal used, as your solution to this exercise.*

We'll sample observations around six signals.

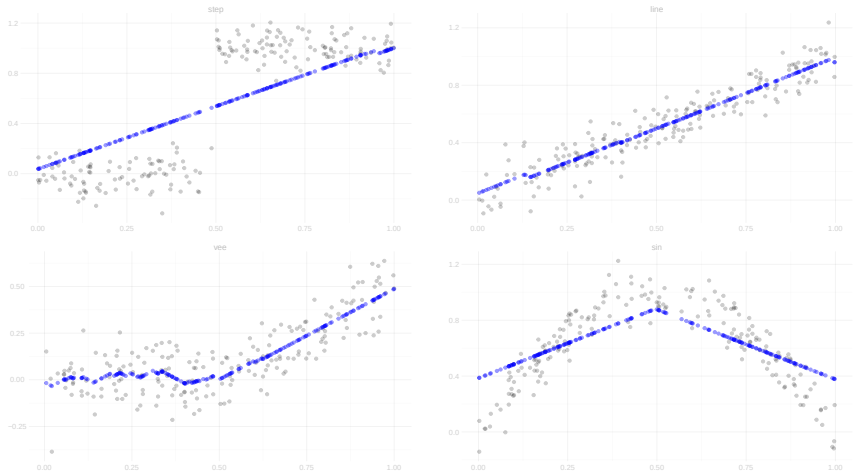
1. A step, $\mu(x) = 1(x >= .5)$.
2. A line, $\mu(x) = x$.
3. A vee, $\mu(x) = (x - .5)1(x \geq .5)$.
4. A sine, $\mu(x) = \sin(\pi x)$.
5. A damped rapidly oscillating curve, $\mu(x) = x \sin(1/x)$.
6. A more-damped rapidly-oscillating curve, $\mu(x) = x^{3/2} \sin(1/x)$.

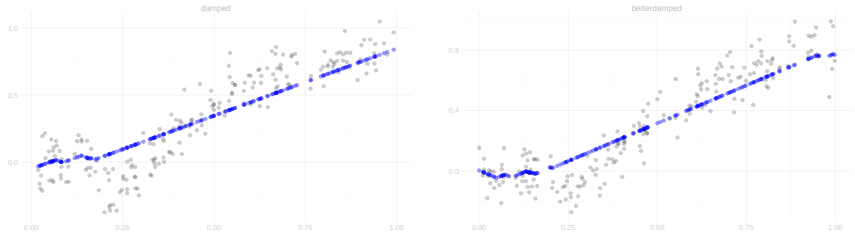
For each, we'll work with independent and identically distributed observations $(X_1, Y_1) \dots (X_n, Y_n)$ where $X_i$ is drawn from the uniform distribution on $[0, 1]$ and $Y_i = \mu(X_i) + \varepsilon_i$ for $\varepsilon_i$ drawn from the normal distribution with mean zero and standard deviation $\sigma = 1/10$.

**Solution 5**
```
lipreg = function(X,Y,B=1) {
  input = list(X=X, Y=Y)
  n = length(X)
  m = Variable(n)

  mse = sum((Y - m)^2) / n

  grid = expand.grid(i=1:n, j=1:n)
  lipschitz.constraint = abs(m[grid$i] - m[grid$j]) <=
                         B * abs(X[grid$i] - X[grid$j])

  # solve and ask for m that solves our minimization problem
  solved = solve(Problem(Minimize(mse), list(lipschitz.constraint)))
  mu.hat = solved$getValue(m)

  # now a little boilerplate to make it idiomatic R
  #  1. we record the input X and the solution mu.hat in a list
  #  2. we assign that list a class, so R knows predict should
```

```
  #     delegate to predict.convexreg
  #  3. we return the list
  model = list(X=X, mu.hat=mu.hat, input=input)
  attr(model, "class") = "lipreg"
  model
}
```

```
mus = list(step      = function(x) { 1*(x >= .5) },
           line      = function(x) { x },
     vee       = function(x) { (x-.5)*(x >= .5) },
           sin       = function(x) { sin(pi*x) },
     damped    = function(x) { x*sin(1/x) },
           betterdamped    = function(x) { x^(3/2)*sin(1/x) })

make.plot = function(mu, fit=function(X,Y) { lipreg(X,Y,B=1) }) {
    sigma = .1
    n = 200
    X = runif(n)
    Y = mu(X) + sigma*rnorm(n)
    model = fit(X,Y)
    ggplot() +
        geom_point(aes(x=X,y=Y), alpha=.2) +
        geom_point(aes(x=model$X, y=model$mu.hat),
                   alpha=.4, color='blue')
}
```



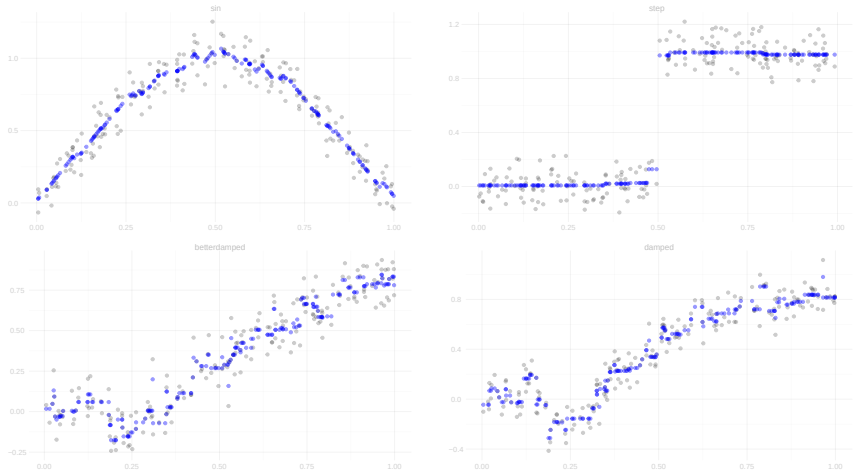



**Exercise 6** *Revisit the curves $\hat{\mu}$ you fit in the last exercise. For each, answer these questions.*

1. *Does it fit the data?*
2. *If not, what — if anything — could we do to fit the data better?*

*Then, if there is something you can do, do it and include the resulting plot.*

**Solution 6** *The ones that fit well are the line and the vee. Both are in our regression model; they have Lipschitz constant* 1*. The Lipschitz constant of* $\sin(\pi x)$ *is* $\pi$*; relaxing our constraint* $\rho_{Lip}(m) \leq 1$ *to* $\rho_{Lip}(m) \leq B$ *for* $B \geq \pi$ *gives a good fit. On the other hand, the step and damped rapidly-oscillating curves aren't Lipschitz at all (they have* $\rho_{Lip}(\mu) = \infty$*), so we're not going to get a good fit with Lipschitz regression no matter what. We get good fits to the step and the better-damped oscillation using bounded variation regression with appropriately chosen variation budget* $B$*, taking* $B = \rho_{TV}(\mu) = 1$ *for the step and* $B = 5 \geq \rho_{TV}(\mu)$ *for the better damped oscillation (see Exercises 1 and 2). On the other hand,* $\rho_{TV}(\mu) = \infty$ *for the other damped oscillation, so based on the way we've been arguing you might think it'd be hard to fit with bounded variation regression. It isn't. This function can be approximated very well by a function of bounded variation, as most of its variation happens near* $x = 0$ *where* $\mu(x) \approx 0$*. I've shown a good fit using bounded variation regression with the total variation budget* $B = 5$*.*



*What the fact that $\rho_{TV}(\mu) = \infty$ for the damped oscillation tell us? It tells us that, no matter what variation budget $B$ we use, if we increased our sample size $n$ toward infinity without increasing the budget, at some point our fit $\hat{\mu}$ would essentially stop improving because there is* some limit *to how well we can approximate $\mu$ using any curve $m$ with $\rho_{TV}(m) \leq B$. If, on the other hand, we used cross-validation to select a budget at each $n$, our fit would keep improving. What'd happen is that the sequence of selected budgets $B_n$ would increase to $\infty$ as $n \to \infty$ so that the selected model $\mathcal{M}_n = \{m : \rho_{TV}(m) \leq B_n\}$ would include better and better approximations to $\mu$ as we got enough data to need them.*

## 2.3 Filling in the gaps

At this point, you have an estimator $\hat{\mu}_{|\mathcal{X}}$ that minimizes squared error among the functions $m$ satisfying $\rho_{Lip|\mathcal{X}}(m) \leq B$. This lets us plot some isolated points. But we want a complete curve $\hat{\mu}(x)$ for $x \in [0, 1]$ that satisfies $\rho_{Lip}(\hat{\mu}) \leq B$, and we want it to be the best-fitting such curve, i.e., we want the solution to (2).

To do this, we'll use a *piecewise-linear extension* of $\hat{\mu}_{|\mathcal{X}}$. That is, having sorted $X_i$ into increasing order, we will define $\hat{\mu}(x)$ everywhere on $[X_1, X_n]$ by drawing line segments between successive points $\{X_i, \hat{\mu}(X_i)\}$ and $\{X_{i+1}, \hat{\mu}(X_{i+1})\}$, and extend the leftmost and rightmost segment to fill the intervals $[0, X_1]$ and $[X_n, 1]$.[1] This gives us a piecewise-linear solution to (3). First, we'll implement it. Then we'll verify that it is, in fact, a solution to (2).

**Exercise 7** *Briefly explain why piecewise-constant extension would not give us a solution to* (2)*. A sentence or a sketch should do.*

**Tip.** Think about Exercise 3.

### 2.3.1 Implementation

**Exercise 8** *Write out a formula for the piecewise-linear curve $\hat{\mu}(x)$ in terms of $\hat{\mu}(X_1) \ldots \hat{\mu}(X_n)$. Then implement it and add the curve $\hat{\mu}(x)$ for $x \in [0, 1]$ to your plots from the last exercise.*

**Tip.** For coding a piecewise linear function, try to modify the function `predict.piecewise.constant` from the bounded variation lab.

**Solution 7** *I used a piecewise-linear curve in the solution to the Convex Regression Homework, so I'll borrow from there. Here's the formula.*

$$\hat{\mu}(x) = \hat{\mu}(X_i) + \frac{\hat{\mu}(X_{i+1}) - \hat{\mu}(X_i)}{X_{i+1} - X_i}(x - X_i) \quad \text{for } i = \max\{i \in 1 \ldots n-1 : X_i \leq x\} \cup \{1\}. \tag{6}$$

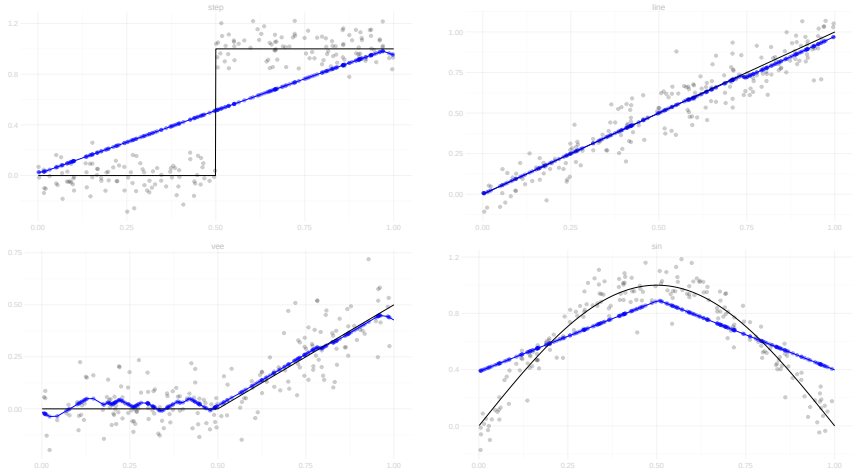
*Here's a little explanation. The formula for $\hat{\mu}(x)$ is the formula for the line through $\{X_i, \hat{\mu}(X_i)\}$ and $\{X_{i+1}, \hat{\mu}(X_i)\}$ where $X_i$ for $i \in 1 \ldots n-1$ is the largest*

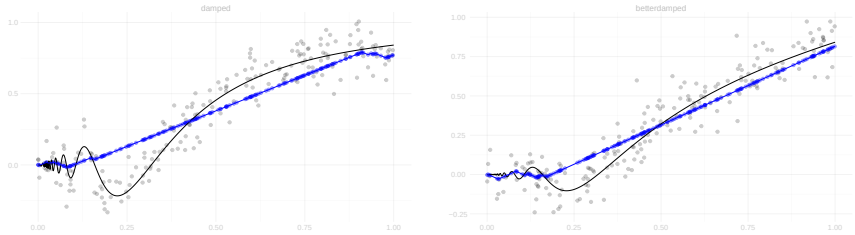
*$X_i$ to the left of our query point $x$ or, if there are none, $X_1$. This last caveat handles the case that $x$ is to the left of $X_1$; to handle the case that it's to the right of $X_n$, we've restricted the range of our search to $1 \ldots n-1$ because if we are to the right of $X_n$, we still want to use $X_{n-1}$ as our segment's starting point.*

*Here's the implementation.*

```
predict.piecewise.linear = function(model, newdata=data.frame(X=model$input$X)) {
  Y = model$mu.hat; X=model$X; x=newdata$X; n = length(X)

  # for each new data point x[k]
  # find the closest observed X[i[k]] left of x[k]
  # i.e., i[k] is the largest integer i for which X[i] <= x[k]
  i = findInterval(newdata$X, X)
  # If there is no X[i] < x[k], findInterval sets i[k]=0
  #  and we'll want to act as if we'd gotten 1 so we use the
  #  line through (X[1], Y[1])  and (X[2], Y[2])
  # If that k is n, we'll want to act as if we'd gotten n-1 so we use
  #  the line through (X[n-1], Y[n-1])  and (X[n], Y[n])
  i[i==0] = 1; i[i==n] = n-1
  # make a prediction using the formula y - y0 = (x-x0) * slope
  Y[i] + (x-X[i]) * (Y[i+1]-Y[i])/(X[i+1]-X[i])
}

predict.lipreg = predict.piecewise.linear
```

*Here are the plots. The gray dots are observations $(X_i, Y_i)$, the black line is $\mu(x)$, the blue dots are points $\{X_i, \hat{\mu}(X_i)\}$ on the fitted curves, and the blue line is $\hat{\mu}(x)$.*






### 2.3.2 Verification

**Exercise 9** *Consider any pair $x < x'$. Prove that for any piecewise-linear function $m$ with breaks at $X_1 \ldots X_n$, the secant slope $\{m(x') - m(x)\}/(x' - x)$ between these points is a* weighted average *of the slopes $\{m(X_{j+1}) - m(X_j)\}/(X_{j+1} - X_j)$ of the segments that lie between them. Briefly explain why this implies that our piecewise-linear solution $\hat{\mu}$ satisfies $\rho_{Lip}(\hat{\mu}) = \rho_{Lip|\mathcal{X}}(\hat{\mu}_{|\mathcal{X}})$ and why this implies that $\hat{\mu}$ solves* (2)*.*

**Tip.** Break the 'explain' part of this down into feasibility and optimality, like we did in the bounded variation regression lab.

**Solution 8** *What we're going to do is break down the secant slope we're interested in as a weighted average of segment slopes. Let $X_1 \ldots X_n$ be sorted in increasing order and $i$ and $i'$ be chosen as in* (6) *for $x = x$ and $x = x'$ respectively, so $x$ and $x'$ are between $X_i$ and $X_{i+1}$ and $X_{i'}$ and $X_{i'+1}$ respectively.*[2] *We'll expand the secant slope's numerator, $m(x') - m(x)$, by adding zero written in a fancy way, using a telescoping sum: $0 = -m(X_{i'}) + \sum_{j=i}^{i'-1} m(X_{j+1}) - m(X_j) + m(X_i)$. What we get is a sum of differences in the piecewise-constant function $m$ between points on the same segment. And we'll rewrite those differences as the product of the segment slope and the distance between the points, i.e., using the identity $m(b) - m(a) = \{b - a\} \times \{m(b) - m(a)\}/\{b - a\}$. That*

*gives us a weighted average of slopes. Take a look.*

$$\begin{aligned}
\frac{m(x') - m(x)}{x' - x} &= \frac{\{m(x') - m(X_{i'})\} + \left\{\sum_{j=i}^{i'-1} m(X_{j+1}) - m(X_j)\right\} - \{m(x) - m(X_i)\}}{x' - x} \\
&= \frac{m(x') - m(X_{i'})}{x' - X_{i'}} \left\{ \frac{x' - X_{i'}}{x' - x} \right\} \\
&+ \sum_{j=i}^{i'-1} \frac{m(X_{j+1}) - m(X_j)}{X_{j+1} - X_j} \left\{ \frac{X_{j+1} - X_j}{x' - x} \right\} \\
&+ \frac{x - m(X_i)}{x - X_i} \left\{ \frac{x - X_i}{x' - x} \right\} \\
&= \frac{m(X_{i'+1}) - m(X_{i'})}{X_{i'+1} - X_{i'}} \left\{ \frac{x' - X_{i'}}{x' - x} \right\} \\
&+ \sum_{j=i}^{i'-1} \frac{m(X_{j+1}) - m(X_j)}{X_{j+1} - X_j} \left\{ \frac{X_{j+1} - X_j}{x' - x} \right\} \\
&+ \frac{m(X_{i+1}) - m(X_i)}{X_{i+1} - X_i} \left\{ \frac{x - X_i}{x' - x} \right\}
\end{aligned}$$

*What's going on in the last equality? The red and blue slopes are the same in each expression. Because $\hat{\mu}$ is piecewise-linear, the slope of $\hat{\mu}$ between $X_i$ and $x$ is the same as the slope of $\hat{\mu}$ between $X_i$ and $X_{i+1}$.*

*Now observe that the ratios in curly braces that multiply these slopes are non-negative weights that sum to one, so our secant slope is a weighted average of these segment slopes. And it follows, by Hölder's inequality, that the absolute value of this weighted average can be no larger than that of the largest individual slope. That is, the Lipschitz constant $\rho_{Lip}(m)$ of a piecewise linear curve with breaks at $X_1 \ldots X_n$ like (6) is no larger than its restriction's Lipschitz constant $\rho_{Lip|\mathcal{X}}(m_{|\mathcal{X}})$.*

*Furthermore, $\rho_{Lip|\mathcal{X}}(m_{|\mathcal{X}})$ is no larger than $\rho_{Lip}(m)$ for any function $m$, as the former is the maximum of the same function of pairs $x, x'$ over a smaller set of pairs. Summarizing, these piecewise linear curves satisfy $\rho_{Lip|\mathcal{X}}(m_{|\mathcal{X}}) \leq \rho_{Lip}(m) \leq \rho_{Lip|\mathcal{X}}(m_{|\mathcal{X}})$ and therefore $\rho_{Lip}(m) = \rho_{Lip|\mathcal{X}}(m_{|\mathcal{X}})$.*

***Why the Piecewise-Linear Extension of** $\hat{\mu}_{|\mathcal{X}}$ (3) **solves** (2).*

*What we've just shown tells us that the piecewise-linear extension of $\hat{\mu}_{|\mathcal{X}}$ gives us something feasible for (2): the constraint $\rho_{Lip|\mathcal{X}}(\hat{\mu}_{|\mathcal{X}}) \leq B$ imposed by (3) implies that its piecewise-linear extension (6) satisfies the constraint $\rho_{Lip}(\hat{\mu}) \leq B$ in (2). And if it weren't optimal, there'd have to be some feasible function $m$ with $\rho_{TV}(m) \leq B$ and therefore $\rho_{TV|\mathcal{X}}(m_{|\mathcal{X}}) \leq B$, because $\rho_{TV} \mid \mathcal{X}(m_{|\mathcal{X}}) \leq \rho_{TV}(m)$ for all $m$, with mean squared error strictly less than $\hat{\mu}$. This would imply that $\hat{\mu}_{|\mathcal{X}}$ does not solve (3), so there can exist no such $m$.*

## 2.4 Optimized Fitting

We can speed up our fitting code by simplifying our set of constraints by hand. In particular, I claim that you get the same solution if you impose the constraint $|m(X_i) - m(X_j)|/|X_i - X_j| \leq B$ for adjacent points. That is, if the points $X_1 \ldots X_n$ are sorted in increasing order, it's equivalent to impose the constraint for the pairs $(i, j = i + 1)$.

**Exercise 10** *Prove it! Then implement it and check that your solution agrees with the one you got before using the all-pairs constraint. Include the proof as your solution. No need to turn in code, but you'll want this faster implementation later.*

**Tip.** The proof should be easy. Use the weighted-average idea from the last exercise.

**Solution 9** *Consider any pair of observations $X_i, X_{i'}$ sorted so $X_i < X_{i'}$. From the previous part, taking $x = X_i$ and $x' = X_{i'}$, we know that the slope absolute value $|m(X_{i'}) - m(X_i)|/|X_{i'} - X_i|$ of the segment between these endpoints is no larger than the largest of the slope absolute values $|m(X_{j+1}) - m(X_j)|/|X_{j+1} - X_j|$ of the segments between adjacent points between them. It follows that the maximum $\rho_{Lip|\mathcal{X}}(m)$ of the absolute slope between all pairs $X_i, X_j$ is no larger than the largest of the absolute slopes for adjacent points. And, as it clearly cannot be smaller, it follows that two maxima are equal and the resulting constraints equivalent.*

```
lipreg = function(X, Y, B=1) {
  # Step 0.
  # We check that the inputs satisfy our assumptions.
  stopifnot(length(X) == length(Y))
  input = list(X=X, Y=Y)
  n = length(X)
  # and find the unique elements of X and the inverse mapping
  unique.X = invert.unique(X)

  # Step 1.
  # We tell CVXR we're thinking about a vector of unknowns m in R^p.
  m = Variable(length(unique.X$elements))
  # and permute and duplicate these into a vector mX with n elements in correspondence with
  mX = m[unique.X$inverse]

  # Step 2.
  # We tell CVXR that we're interested in mean squared error.
  mse = sum((Y - mX)^2 / n)

  # Step 3.
```

```
  # We specify our constraints.
  uX = X[unique.X$elements]
  constraints = list( abs(diff(m)) <= B * diff(uX) )

  # Step 4.
  # We ask CVXR to minimize mean squared error subject to our constraints.
  # And we ask for vector mu.hat that does it.
  solved = solve(Problem(Minimize(mse), constraints))
  mu.hat = solved$getValue(m)

  # Step 5: a little boilerplate to make it idiomatic R.
  # 1. we record the unique levels of X and mu.hat, in correspondence and sorted in increas
  # 2. we assign that list a class, so R knows predict should delegate to predict.lipreg
  # 3. we return the list
  model = list(X = X[unique.X$elements], mu.hat = mu.hat, B=B, input = input)
  attr(model, "class") = "lipreg"
  model
}
```

# 3 Rates of Convergence

Now we've got three nonparametric regression models: monotone curves, bounded variation curves, and lipschitz curves. To keep things simple, we'll be working with data sampled around one signal: $\mu(x) = x$. That is, we'll work with independent and identically distributed observations $(X_1, Y_1) \dots (X_n, Y_n)$ where $X_i$ is drawn from uniform distribution on $[0, 1]$ and $Y_i = \mu(X_i) + \varepsilon_i$ for $\varepsilon_i$ drawn independently from the normal distribution with mean zero and standard deviation $\sigma = .5$.

**Tip**. What we're doing here is taking what we did at the end of the convergence rates lab, simplifying it by using only one signal instead of four, and then adding two new regression models. Use the lab's solution as a starting point.

**Exercise 11** *Draw a sample of size $N = 1600$ from this distribution. To get samples of sizes $n = \{25, 50, 100, 200, 400, 800, 1600\}$, use the first 25, 50, etc. observations.*

*At all of these sample sizes, fit a line, an increasing curve, and bounded variation and lipschitz curves with budgets $B = 1$. Calculate sample MSE $\|\hat{\mu} - \mu\|^2_{L_2(P_n)}$ and population MSE $\|\hat{\mu} - \mu\|^2_{L_2(P)}$ for each. Repeat this ten times and average the results to get estimates of expected sample MSE and expected population MSE at each sample size $n$. Include plots of these as a function of $n$ as your solution.*

**Tip.** This can be slow for larger samples. Try it out for samples of size $25 \dots 400$ before adding in $n = 800$ and $n = 1600$.

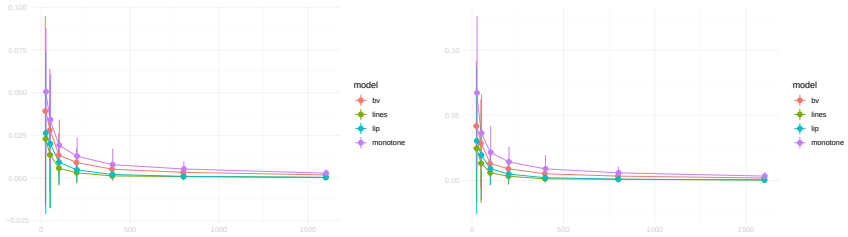



Figure 1: Expected sample (left) and population (right) MSE

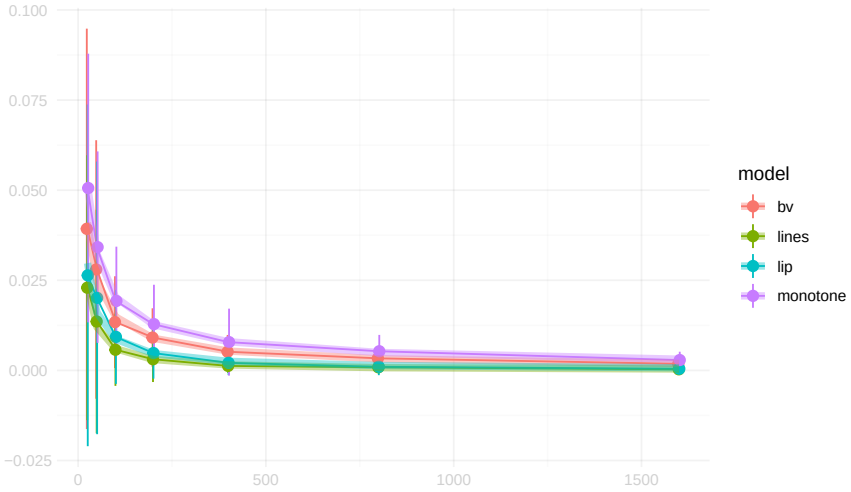



Figure 2: Expected sample MSE (thin lines) and a predictions (thick lines) based on our rate estimates

**Solution 10**

Let's try to summarize these plots by rates of convergence.

**Exercise 12** *For each of your four regression models, use* `nls` *to fit a curve of the form* $m(n) = \alpha n^{-\beta}$ *to* $RMSE = \sqrt{MSE}$ *where MSE is your estimate of expected population mean squared error from the last exercise. Repeat for expected sample mean squared error.*

*Plot the resulting predictions of MSE,* $\hat{m}(n)^2$*, on top of your actual MSE curves from the the previous exercise to check their accuracy. Include these plots and report these rates of convergence* $\hat{\beta}$ *as your solution. Briefly comment on what you see, too.*

**Solution 11** *For both sample and population RMSE, the rates I'm estimating are somewhere between* $n^{-1/3}$ *and* $n^{-1/2}$ *for everything but lines and* $n^{-1/2}$ *for lines.*

| error.measure | model | a | b |
|---|---|---|---|
| population | bv | 0.70 | 0.38 |
| population | lines | 0.81 | 0.51 |
| population | lip | 0.82 | 0.47 |
| population | monotone | 0.81 | 0.36 |
| sample | bv | 0.66 | 0.37 |
| sample | lines | 0.77 | 0.50 |
| sample | lip | 0.72 | 0.45 |
| sample | monotone | 0.67 | 0.34 |

- *The $n^{-1/2}$ rate for lines is something you may have seen in a previous class. If you haven't, you've probably seen it for* horizontal lines*, since the least squares prediction $\hat{\mu}(x)$ in that model is the constant $\bar{Y}$ which has standard deviation $\sigma/\sqrt{n}$.*
- *Later in the semester, we'll prove that the rates for the other models are, in fact, $n^{-1/3}$ or better. To get a more precise characterization of these rates using this simulation-based method, we'd want to use a larger range of sample sizes, and for that we'd want faster code.*

*Our actual error curves do agree well with the predictions we get based on these rates.*

*At most sample sizes the errors follow the pattern*

$$\textit{bounded variation} > \textit{monotone} > \textit{lipschitz} > \textit{lines}.$$

*We'd expect some of these comparisons. The signal is in all of these models, so lower error boils down to less overfitting. Our lipschitz model is contained in our bounded variation model, so we'd expect it to overfit less and therefore have lower error. And our lines model, while not contained in any of the others, is in a sense much smaller, so I'd expect the least overfitting and therefore the lowest error there.*

*If this pattern wasn't in line with your expectations, no worries. Later on in the semester, we'll see some theory that'll tell us what's going on here pretty clearly.*

# Notes

[1]I'm going to stop writing $\hat{\mu}_{|\mathcal{X}}$ all the time from here on. Since we're talking about an extension, we know that $\hat{\mu}(X_i) = \hat{\mu}_{|\mathcal{X}}(X_i)$, so I'll write that to reduce notational clutter.

[2]The 'so' clause here assumes $x$ and $x'$ are inside the range of the data. What we can say generally, and what's relevant, is that slope of $\hat{\mu}$ between $x$ and $X_i$ is the slope of the segment connecting $(X_i, m(X_i))$ to $(X_{i+1}, m(X_{i+1}))$.