Linear Mean Functions and Ordinary Least Squares

Mark M. Fredrickson (mfredric@umich.edu)

Computational Methods in Statistics and Data Science (Stats 406)

Smoothed Means

- Goal: estimate $\mu(x)$ from the model:

$$Y = \mu(x) + \epsilon, \quad E(\epsilon) = 0$$

- Nardaraya-Watson estimator using a kernel K and bandwidth h:

$$\hat{\mu}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right)}$$

- Trade-off in the size of the bandwidth with respect to bias and variance
- Used bootstrapping to get confidence intervals.

Mean Functions for Multiple x

Multivariate Smoothing

Suppose we want to extend our model of mean of Y to

$$Y = \mu(x_1, x_2, \ldots, x_p) + \epsilon$$

again with $\mathrm{E}\left(\epsilon\right) = 0$.

Last time we saw **bivariate kernel density estimates**

$$\widehat{f}(x_1, x_2) = \frac{1}{nh^2} \sum_{i=1}^{n} K\left(\frac{x_{i,1} - x_1}{h}, \frac{x_{i,2} - x_2}{h}\right)$$

which can be extended to d-dimensions.

In principle, we could **plug these into the Nadaraya-Watson estimator**.

Curse of Dimesionality

In a general sense, the kernel density and smoothing estimators work by **estimating $\mu(x)$ using Y_i where x_i is close to x**.

This is expressed as picking a **bandwidth h** that describes how to weight near and far observations.

The Curse of Dimensionality: as the number of dimensions increases all points are far apart.

Illustration

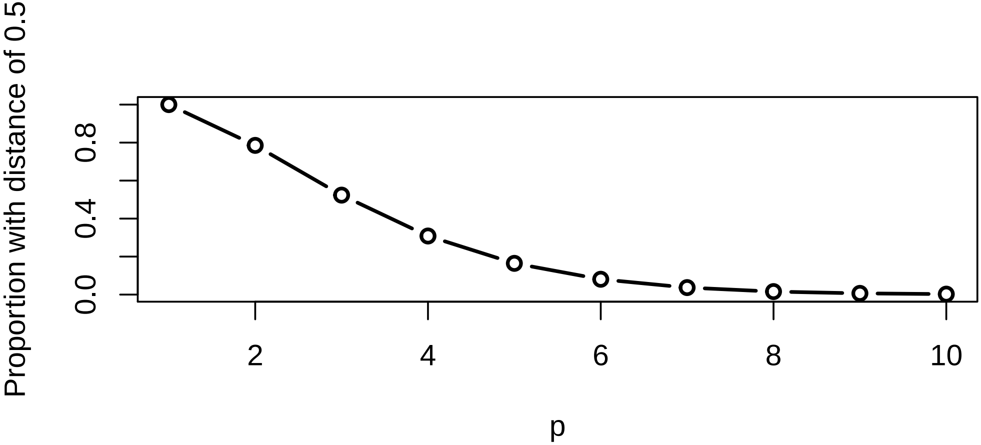
Suppose $x_j \in [0, 1]$ for $j = 1, \ldots, p$. Consider the point

$$\mathbf{x} = (0.5, 0.5, \ldots, 0.5)^T$$

(notation: $\mathbf{x}$ is a column vector of size p)

What proportion of the points in the space are within a distance of 0.5 of x?

$$\frac{\text{volume of a sphere of radius 0.5}}{\text{volume of a } p\text{-dimensional box}} = \frac{\pi^{p/2}}{p 2^{p-1} \Gamma(p/2)}$$


Proportion with distance of 0.5
0.0
0.4
0.8
2
4
6
8
10
p

Problems

- We could increase h, but this **increases bias** of the estimator.
- We could keep h the same, but we need **more observations** to give precise estimates in any region.
- Smoothing estimators require evaluating an expression **at each point of the original data** (no data compression).
- We want to **include additional structure**.
- We want to interpret the **contribution of** x_j apart from the other predictors.

Solution

We'll consider a variety of methods that address the problems using these steps:

- **Reduce the dimension** from p to 1 using a function $\eta(x_1, \ldots, x_p; \beta)$ and a parameter β
- **Assume a structure** to $\mu(\eta(x_1, \ldots, x_p; \beta))$ (notice: one dimensional function)
- **Find "best" β** under some loss function

These techniques **may not be new to you** (e.g. OLS).

Compared to other statistics courses our emphasis will be on β as a **solution to an optimization problem** (i.e., minimizing loss).

Ordinary Least Squares

Vector and Matrix Notation

We will write:

$$\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$$

to collect all of the **predictor** or **independent variables** into a **vector of length** p.

The "T" superscript takes the transpose of a matrix (flips rows and columns).

Additionally, we observe an **outcome** or **dependent variable**, y (a scalar).

Collect all n outcomes into the vector $\mathbf{y}$ and all of the predictors in to the $n \times p$ matrix $\mathbf{X}$, with the ith row equal to $\mathbf{x}_i$.

Note: we think of y as random, but I'm writing it as lower case now that we have to deal with vectors and matrices.

A very quick linear algebra review

We can form the **inner product** of two n-vectors as:

$$\mathbf{a}^T\mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

For matrices $\mathbf{A}$ $(n \times p)$ and $\mathbf{B}$ $(p \times m)$, the matrix $\mathbf{AB} = \mathbf{C}$ $(n \times m)$ has entries:

$$c_{ij} = (i\text{th row of } \mathbf{A})^T (j\text{th column of } \mathbf{B})$$

If a matrix $\mathbf{A}$ is square $(n \times n)$ and the columns are linearly independent ("full rank"), then the **matrix inverse** $\mathbf{A}^{-1}$ is such that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$$

where $\mathbf{I}_n$ is the **identity matrix**, with 1's on the diagonal and 0's elsewhere.

Simplest μ and η

Recall that we are modeling:

$$y = \mu(\eta(\mathbf{x}; \boldsymbol{\beta})) + \epsilon, \quad \text{where } \boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_p)^T$$

(semi-colon notation indicates that while **x** will change for different observations, $\boldsymbol{\beta}$ will be the same – e.g. likelihoods)

First, we need a function η that maps our observations **x** to a scalar. The probably the simplest such function:

$$\eta(\mathbf{x}; \boldsymbol{\beta}) = \mathbf{x}^T \boldsymbol{\beta} = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Next, we need a function μ that maps scalars to scalars. The simplest function:

$$\mu(a) = a$$

Putting them together

Combining μ and η, we arrive at **a model for the mean function**:

$$\mathrm{E}\left(y \mid \mathbf{x}\right) = \mu(\eta(\mathbf{x}; \boldsymbol{\beta})) = \mathbf{x}^T\boldsymbol{\beta}$$

or equivalently

$$y = \mathbf{x}^T\boldsymbol{\beta} + \epsilon, \quad \mathrm{E}\left(\epsilon\right) = 0$$

We don't know β (in the same way we did not know the optimal bandwidth h), so we need to pick a $\hat{\boldsymbol{\beta}}$ that is “best” in some sense.

NB: We often set $x_0 = 1$ for all units so p includes $p - 1$ ”real variables”:

$$y = \beta_0 + \beta_1 x_1 + \ldots \beta_{p-1} x_{p-1} + \epsilon$$

Optimization Problems

Goal: find the best $\hat{\boldsymbol{\beta}}$. We need some **criterion** to choose between possible $\hat{\boldsymbol{\beta}}$.

We will define a **loss function** $R(\boldsymbol{\beta})$ that measures how close $\mathbf{x_i}^T\boldsymbol{\beta}$ is to y_i (for some candidate $\boldsymbol{\beta}$) and **minimize the loss**.

Can also specificy any **restrictions on** $\boldsymbol{\beta}$ (e.g., must be positive).

Problem statement:

$$\begin{aligned} \text{minimize:} \quad & R(\boldsymbol{\beta}) \\ \text{subject to:} \quad & \boldsymbol{\beta} \in \mathcal{S} \end{aligned}$$

We will begin by letting $\boldsymbol{\beta} \in \mathbb{R}^p$ (i.e., any real number).

Loss functions

One choice we could pick is **absolute error loss**:

$$R(\beta) = \sum_{i=1}^{n} |Y_i - \mathbf{x_i}^T \beta|$$

Another choice is **squared error loss**:

$$R(\beta) = \sum_{i=1}^{n} \left(Y_i - \mathbf{x_i}^T \beta\right)^2$$

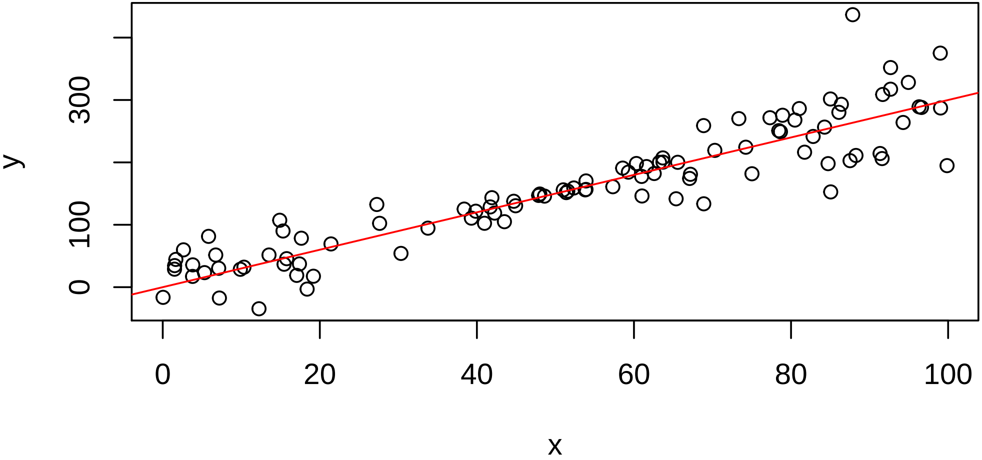
Both are examples of **convex loss functions**, which means that any **local optimum** is also the **global optimum**.

One dimensional case

Let's simulate an outcome that is **linear in x**, **E** $(\epsilon = 0)$, but **does not follow usual Normal assumptions**:

$$
\begin{aligned}
x &\sim U(0, 100) \\
\epsilon &\sim 0.25N\left(1, 2(x-50)^2+10\right) + 0.75N\left(-1/3, 2(x-50)^2+10\right) \\
y &= 3x + \epsilon
\end{aligned}
$$

```
> n <- 100
> x <- runif(n, 0, 100)
> epsilon <- rnorm(n,
+                  mean = 1 - 4/3 * rbinom(n, size = 1, prob = 0.75),
+                  sd = sqrt(2 * abs(x - 50)^2 + 10))
> y <- 3 * x + epsilon
```



0
100
300
y
0
20
40
60
80
100
x

Loss functions

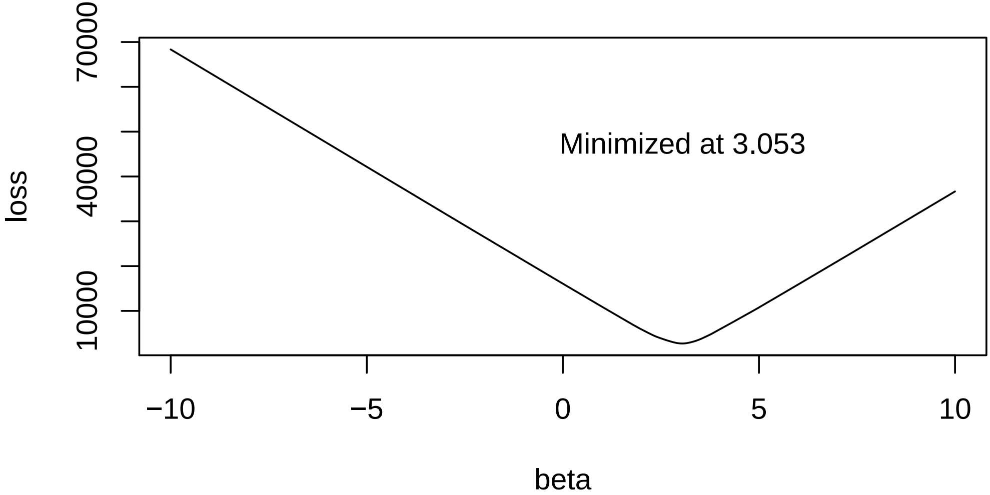
```
> loss_abs <- function(beta) {
+     sum(abs(y - beta * x))
+ }
> loss_sqd <- function(beta) {
+     sum((y - beta * x)^2)
+ }
```

Brute Force Solutions

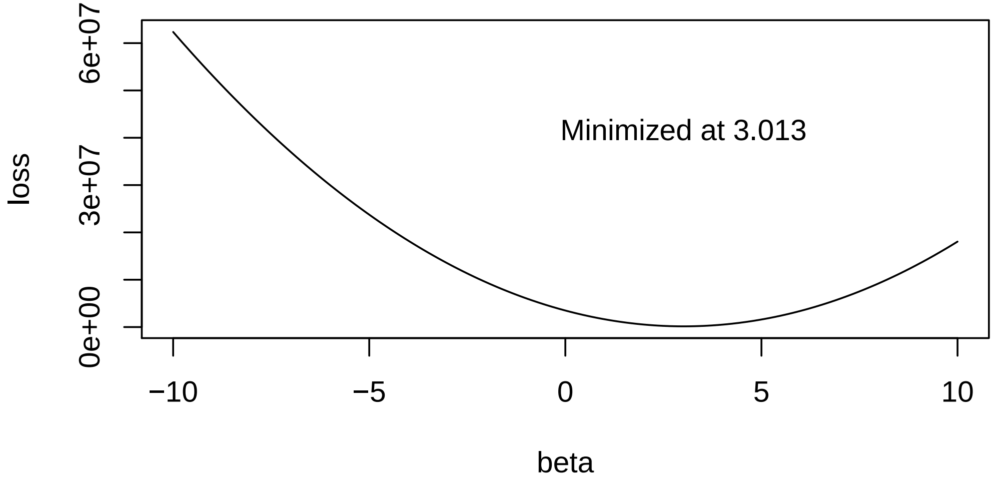
```
> betas <- seq(-10, 10, length.out = 1000)
> beta_loss_abs <- map_dbl(betas, loss_abs)
> beta_loss_sqd <- map_dbl(betas, loss_sqd)

> best_abs <- betas[which.min(beta_loss_abs)]
> best_sqd <- betas[which.min(beta_loss_sqd)]
```

Plotting Absolute Loss



Plotting Squared Loss



Brute force depends on the grid

```
> betas_coarse <- seq(-10, 10, length.out = 10)
> betas_coarse[which.min(map_dbl(betas_coarse, loss_abs))] # Absolute Loss

[1] 3.333

> betas_coarse[which.min(map_dbl(betas_coarse, loss_sqd))] # Squared Loss

[1] 3.333
```

Finding true optimal solutions

As we consider **multidimensional problems**, where $p > 1$, grid searches are even harder.

We already **simplified μ and η**, can we use some **analytical techniques** to improve our methods?

In particular, we happen to know how to minimize functions using calculus, would that be helpful?

We'll look at squared error loss today. We'll return to absolute loss with more advanced techniques.

Finding minimum for squared error loss

Recall that any **local optimum** will have

$$\frac{\partial}{\partial \beta_j} R(\boldsymbol{\beta}) = 0, \quad \forall j = 1, \ldots, p$$

In the case of squared loss:

$$\begin{aligned}\frac{\partial}{\partial \beta_j} R(\boldsymbol{\beta}) &= \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} \left(y_i - \mathbf{x_i}^T \boldsymbol{\beta}\right)^2 \\ &= \sum_{i=1}^{n} -2 \left(y_i - \mathbf{x_i}^T \boldsymbol{\beta}\right) x_{ij} \\ &= 2 \left[\sum_{i=1}^{n} \mathbf{x_i}^T \boldsymbol{\beta} \, x_{ij} - \sum_{i=1}^{n} y_i x_{ij}\right]\end{aligned}$$

Expressing as a system of equations

We are seeking solutions $\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p$ such that

$$\sum_{i=1}^{n} \mathbf{x_i}^T \boldsymbol{\beta} \, x_{ij} = \sum_{i=1}^{n} y_i x_{ij}$$

for each $j = 1, \ldots, p$.

We can stack the **system of p linear equations**:

$$\mathbf{X}^T \mathbf{X} \boldsymbol{\beta} = \mathbf{X}^T \mathbf{y}$$

Where $\mathbf{X}$ is the $n \times p$ **design matrix** of stacked $\mathbf{x_i}$ vectors and $\mathbf{y}$ is the vector of outcomes

Solving systems of linear equations: Ordinary Least Squares

In a regression class, we often take the next step of writing:

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

But **computing matrix inverses** can be **numerically unstable**.

We are better off using techniques from your linear algebra course (**Gaussian elimination**!).

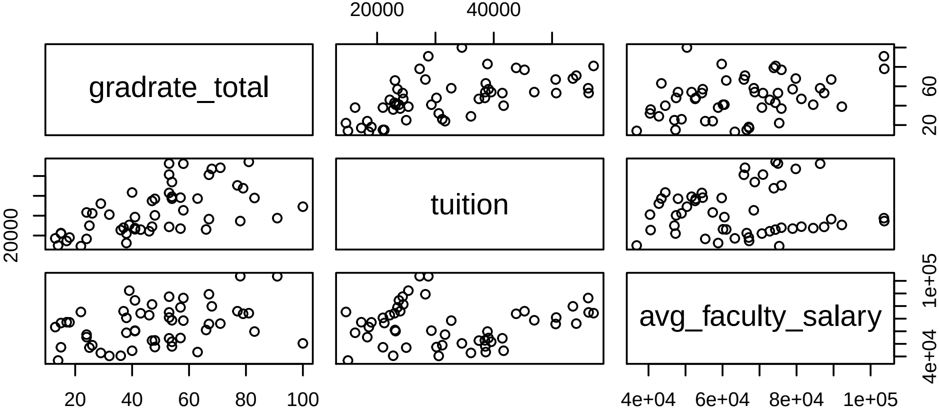
Luckily for us, R will handle actually computing solution:

```
> solve(A, b) ## solves Ax = b for x
```

We call this solution, **ordinary least squares** because it minimized **squared error loss**.

Example: Graduation Rates

How do graduation of Michigan colleges and universities vary as a function of tuition and faculty salaries? (2016 data)



```
> edu_analysis[edu_analysis$name == "University of Michigan-Ann Arbor", 2:4

    gradrate_total tuition avg_faculty_salary
165             91   28776             103605

> design_matrix <- as.matrix(
+     cbind(1,
+           edu_analysis[, c("tuition", "avg_faculty_salary")]))
```

```
> XtX <- t(design_matrix) %*% design_matrix
> XtY <- t(design_matrix) %*% edu_analysis$gradrate_total
> (beta_hat <- as.vector(solve(XtX, XtY)))

[1] -1.426e+01  1.010e-03  4.583e-04

> (mod <- lm(gradrate_total ~ tuition + avg_faculty_salary,
+            data = edu_analysis))

Call:
lm(formula = gradrate_total ~ tuition + avg_faculty_salary, data = edu_anal


Coefficients:
       (Intercept)             tuition  avg_faculty_salary
         -1.43e+01            1.01e-03            4.58e-04
```

Interpreting Results

What do the parameters β_j mean?

Recall, **partial derivatives** tell us how a function changes **with respect to one variable**.

$$\frac{\partial}{\partial x_j}\mu(\eta(\mathbf{x};\boldsymbol{\beta})) = \frac{\partial}{\partial x_j}\sum_{i=1}^{p} x_i\beta_i = \sum_{i=1}^{p}\frac{\partial}{\partial x_j} x_i\beta_i = \beta_j$$

Classic interpretation: "a one unit change in x_j leads to β_j change in y"

Note: be careful about scale! The units of the x variables are usually not the same, so **magnitude does not mean importance**!

Changes in Grad Rate (as percentage)

From the previous slides we see that **schools that differ by $1,000 tuition** differ by

```
> 1000 * beta_hat[2]
```

```
[1] 1.01
```

in **mean graduation percentage**.

Whereas, **$10,000 change in faculty salary** differ by

```
> 10000 * beta_hat[3]
```

```
[1] 4.583
```

in **mean graduation percentage**.

In both cases, **according to our model** and **holding all other variables equal**.

Inference

Recall our basic model for the (random) y:

$$y = \mu(\eta(\mathbf{x}; \boldsymbol{\beta}) + \epsilon = \mathbf{x}^T \boldsymbol{\beta} + \epsilon, \quad \mathrm{E}\left(\epsilon\right) = 0$$

Once we get an estimate $\hat{\boldsymbol{\beta}}$ we would like to **include our uncertainty** about the estimate or **test hypotheses** about $\boldsymbol{\beta}$.

We've already seen ways to do this with the bootstrap:

- Assume nothing about ϵ and resample $(y, \mathbf{x})$
- Assume that all ϵ have the same distribution, and resample from $e = y - \mathbf{x}^T \hat{\boldsymbol{\beta}}$ to make $y^* = \mathbf{x}^T \hat{\boldsymbol{\beta}} + e^*$
- Assume ϵ follow a specific distribution and sample from that to create $y^* = \mathbf{x}^T \hat{\boldsymbol{\beta}} + \epsilon^*$

Gaussian model for ϵ

If we assumed $\epsilon \sim N(0, \sigma^2)$ in

$$Y = \mathbf{x}^T \boldsymbol{\beta} + \epsilon \Rightarrow Y \sim N(\mathbf{x}^T \boldsymbol{\beta}, \sigma^2)$$

then the **likelihood** of the data (with respect to $\boldsymbol{\beta}$ and assuming IID) would be:

$$L(\boldsymbol{\beta}) = f(y_1, \ldots, y_n; \boldsymbol{\beta}) = \frac{1}{\sqrt{2\pi\sigma^2}^n} \exp\left\{ -\frac{\sum_{i=1}^n (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2}{2\sigma^2} \right\}$$

And the **log-likelihood** is

$$l(\boldsymbol{\beta}) = n \log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2$$

MLE of β for Gaussian Model

To find **maximum likelihood estimates**, we want to **maximize the log-likelihood**:

$$l(\boldsymbol{\beta}) = n\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2$$

To do so we have to **minimize the term**:

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2$$

The **squared error loss function**!

Not only is $\hat{\boldsymbol{\beta}}$ the solution to squared error loss, but is also **the maximum likelihood estimate** when under the Gaussian model.

Additional Implications of Gaussian Model

Our MLE estimate is the solution to

$$\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{y}$$

and because $\mathbf{y}$ is Normal, $\mathbf{X}^T\mathbf{y}$ is **also Normal**.

This also implies a distribution for $\hat{\boldsymbol{\beta}}$:

$$\hat{\boldsymbol{\beta}} \sim N_p\left(\boldsymbol{\beta}, \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right)$$

which can be used for tests and intervals.

NB: Even if ϵ is not Normal, since $\hat{\boldsymbol{\beta}}$ is based on sample averages, the **central limit theorem** can also be used to justify these intervals.

```
> summary(mod)

Call:
lm(formula = gradrate_total ~ tuition + avg_faculty_salary, data = edu_analysis)

Residuals:
   Min     1Q Median     3Q    Max
-24.10 -11.20  -0.71   8.05  56.27

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        -1.43e+01   1.07e+01   -1.33   0.1898
tuition             1.01e-03   1.85e-04    5.47  1.5e-06 ***
avg_faculty_salary  4.58e-04   1.38e-04    3.33   0.0017 **
---
Signif. codes:  0
```

Adding constraints: Quadratic Programs

Recall that we were trying to **minimize the sum of squared error**, but subject to **no constraints on β**.

Quadratic programs are those that can be specified as

$$\begin{aligned} \text{minimize:} & \frac{1}{2}\boldsymbol{\beta}^T \mathbf{D}\boldsymbol{\beta} - \mathbf{c}^T \boldsymbol{\beta} \quad \text{(with respect to } \boldsymbol{\beta}) \\ \text{subject to:} & \quad \mathbf{A}\boldsymbol{\beta} \geq \boldsymbol{\beta}_0 \end{aligned}$$

OLS is a **special case** of a quadratic program.

OLS as QP

$$
\begin{aligned}
\sum_{i=1}^{n}(y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2 &= \sum_{i=1}^{n} y_i^2 - 2\boldsymbol{\beta}^T \mathbf{x}_i y_i + (\boldsymbol{\beta}^T \mathbf{x}_i)^2 \\
&\approx \sum_{i=1}^{n}(\boldsymbol{\beta}^T \mathbf{x}_i)(\mathbf{x}_i^T \beta) - 2\boldsymbol{\beta}^T \mathbf{x}_i y_i \\
&= \boldsymbol{\beta}^T \left[\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T\right] \boldsymbol{\beta} - 2\boldsymbol{\beta}^T \mathbf{X}^T \mathbf{y} \\
&\propto \frac{1}{2}\beta^T \mathbf{X}^T \mathbf{X} \beta - (\mathbf{X}^T \mathbf{y})^T \beta
\end{aligned}
$$

Non-negative least squares in R

If we want all $\beta_j \geq 0$, we could add the contraint:

$$\mathbf{I}_p \boldsymbol{\beta} \geq \mathbf{0}$$

```
> library(quadprog)
> (qps <- solve.QP(XtX, XtY, diag(3), rep(0, 3))$solution)

[1] 0.0000000 0.0008838 0.0003107
```

In the previous (unconstrained) OLS fit, we got a negative intercept, now we forced it to be non-negative.

Summary

- Smoothed mean estimators are **flexible** but **don't scale well** and **don't summarize data well**.
- We started to investigate models of the form:

$$\mathrm{E}\left(Y \mid \mathbf{x}\right) = \mu(\eta(\mathbf{x}; \boldsymbol{\beta}))$$

- In particular, we focused on:

$$\mu(a) = a, \quad \eta(\mathbf{x}; \boldsymbol{\beta}) = \mathbf{x}^T\boldsymbol{\beta}$$

- Looked at some **optimization techniques** to find $\hat{\boldsymbol{\beta}}$ under **different loss functions**.
- Found that **squared error loss** can be solved using a **system of linear equations** (easy!)
- Considered how to **interpret** and **add uncertainty** to estimates.
- Used the optimization perspective to add constraints.