Data 401
Feature Engineering

Dennis Sun

October 3, 2016

1. Review of Last Time

2. Basis Expansions

3. Interactions

1. Review of Last Time

2. Basis Expansions

3. Interactions

What We Learned Last Time

- Linear algebra can simplify things!

What We Learned Last Time

- Linear algebra can simplify things!
- If we stack our data into matrices and vectors, the linear regression coefficients are

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

What We Learned Last Time

- Linear algebra can simplify things!
- If we stack our data into matrices and vectors, the linear regression coefficients are

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

- Numerical linear algebra tricks:

What We Learned Last Time

- Linear algebra can simplify things!
- If we stack our data into matrices and vectors, the linear regression coefficients are

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

- Numerical linear algebra tricks:
  - Although they give the same answer, $A(B\mathbf{v})$ is more efficient to calculate than $(AB)\mathbf{v}$.

What We Learned Last Time

- Linear algebra can simplify things!
- If we stack our data into matrices and vectors, the linear regression coefficients are

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

- Numerical linear algebra tricks:
  - Although they give the same answer, $A(B\mathbf{v})$ is more efficient to calculate than $(AB)\mathbf{v}$.
  - Any time you need $A^{-1}\mathbf{b}$, don't calculate A^{-1}. You can get this value by solving $A\mathbf{v} = \mathbf{b}$.

What We Learned Last Time

- Linear algebra can simplify things!
- If we stack our data into matrices and vectors, the linear regression coefficients are

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

- Numerical linear algebra tricks:
  - Although they give the same answer, $A(B\mathbf{v})$ is more efficient to calculate than $(AB)\mathbf{v}$.
  - Any time you need $A^{-1}\mathbf{b}$, don't calculate A^{-1}. You can get this value by solving $A\mathbf{v} = \mathbf{b}$.
- Categorical variables need to be expanded into a bunch of binary variables.

Where we left off...

We wrote a function that fits a linear regression model to the autos data. It worked when we tried it on a smallish number of predictors. But it failed when we tried it on all of the predictors.

Where we left off...

We wrote a function that fits a linear regression model to the autos data. It worked when we tried it on a smallish number of predictors. But it failed when we tried it on all of the predictors.

What happened?

Where we left off...

We wrote a function that fits a linear regression model to the autos data. It worked when we tried it on a smallish number of predictors. But it failed when we tried it on all of the predictors.

What happened?

There is no problem **in theory** with calculating

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

Where we left off...

We wrote a function that fits a linear regression model to the autos data. It worked when we tried it on a smallish number of predictors. But it failed when we tried it on all of the predictors.

What happened?

There is no problem **in theory** with calculating

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

But there are lots of problems in practice. $X^T X$ may be close to singular (non-invertible).

Where we left off...

We wrote a function that fits a linear regression model to the autos data. It worked when we tried it on a smallish number of predictors. But it failed when we tried it on all of the predictors.

What happened?

There is no problem **in theory** with calculating

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

But there are lots of problems in practice. $X^T X$ may be close to singular (non-invertible).
One way to test this is to calculate the condition number of X:

$$\text{cond}(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|}.$$

Where we left off...

We wrote a function that fits a linear regression model to the autos data. It worked when we tried it on a smallish number of predictors. But it failed when we tried it on all of the predictors.

What happened?

There is no problem **in theory** with calculating

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

But there are lots of problems in practice. $X^T X$ may be close to singular (non-invertible).
One way to test this is to calculate the condition number of X:

$$\text{cond}(A) = \frac{|\lambda_{max}(A)|}{|\lambda_{min}(A)|}.$$

(`np.linalg.cond(A)` in Numpy.)

How many predictors are there?

The autos data set has:

- 25 predictors...

How many predictors are there?

The autos data set has:

- 25 predictors...
- ...but many of them were categorical. When we expanded them using `pd.get_dummies`, we end up with 54 variables.

How many predictors are there?

The autos data set has:

- 25 predictors...
- ...but many of them were categorical. When we expanded them using `pd.get_dummies`, we end up with 54 variables.
- But we can also construct new features out of existing features. We can include square, cubic, quartic terms. We can cross two features. The possibilities are endless.

How many predictors are there?

The autos data set has:

- 25 predictors...
- ...but many of them were categorical. When we expanded them using `pd.get_dummies`, we end up with 54 variables.
- But we can also construct new features out of existing features. We can include square, cubic, quartic terms. We can cross two features. The possibilities are endless.

This feature engineering is necessary to get good prediction accuracy. In this lecture, we explore some techniques.

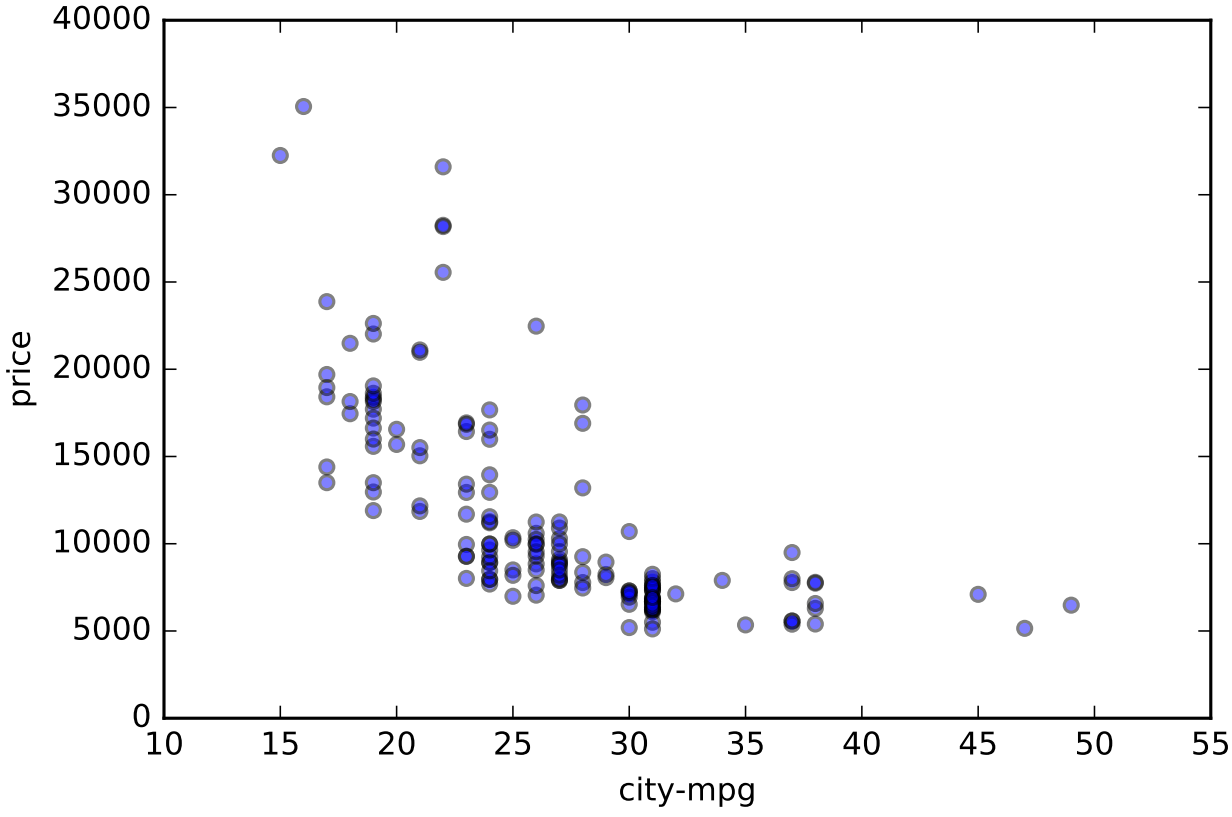
1 Review of Last Time

2 Basis Expansions

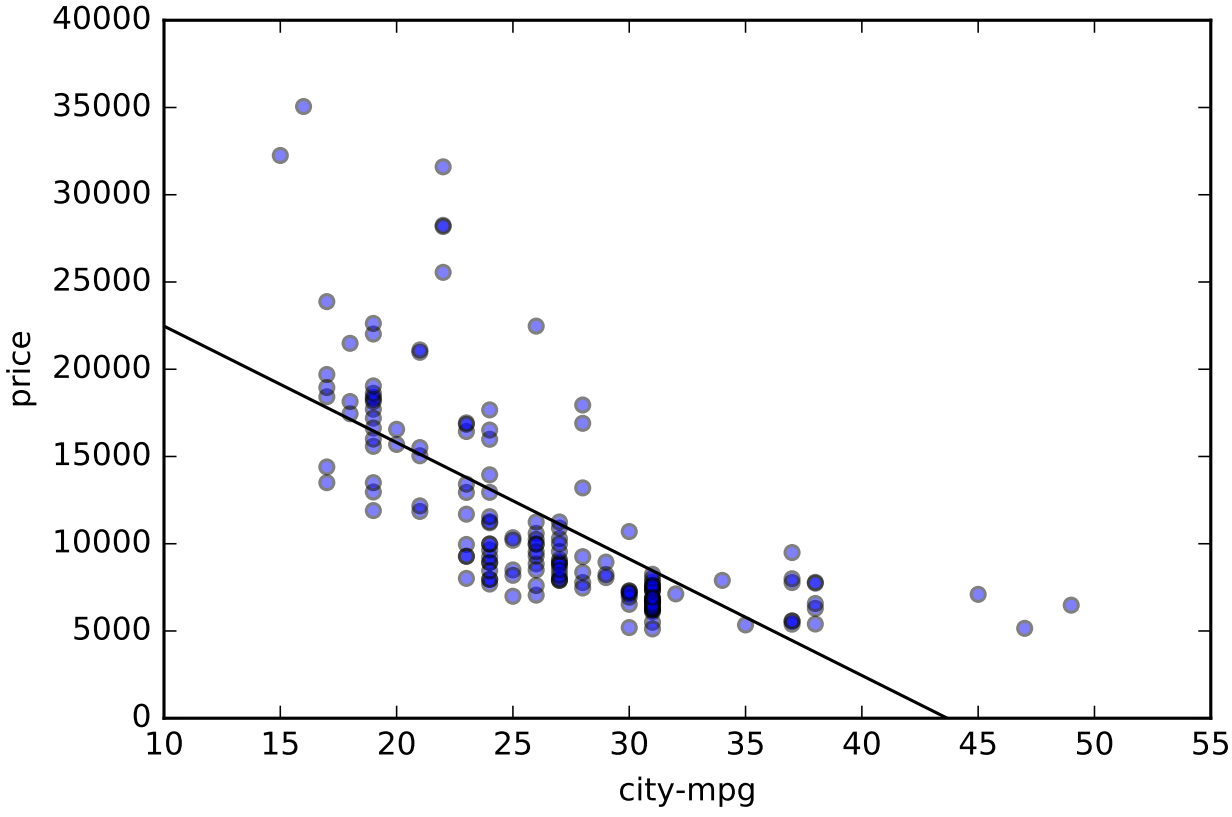
3 Interactions

City MPG vs. Price

City MPG vs. Price

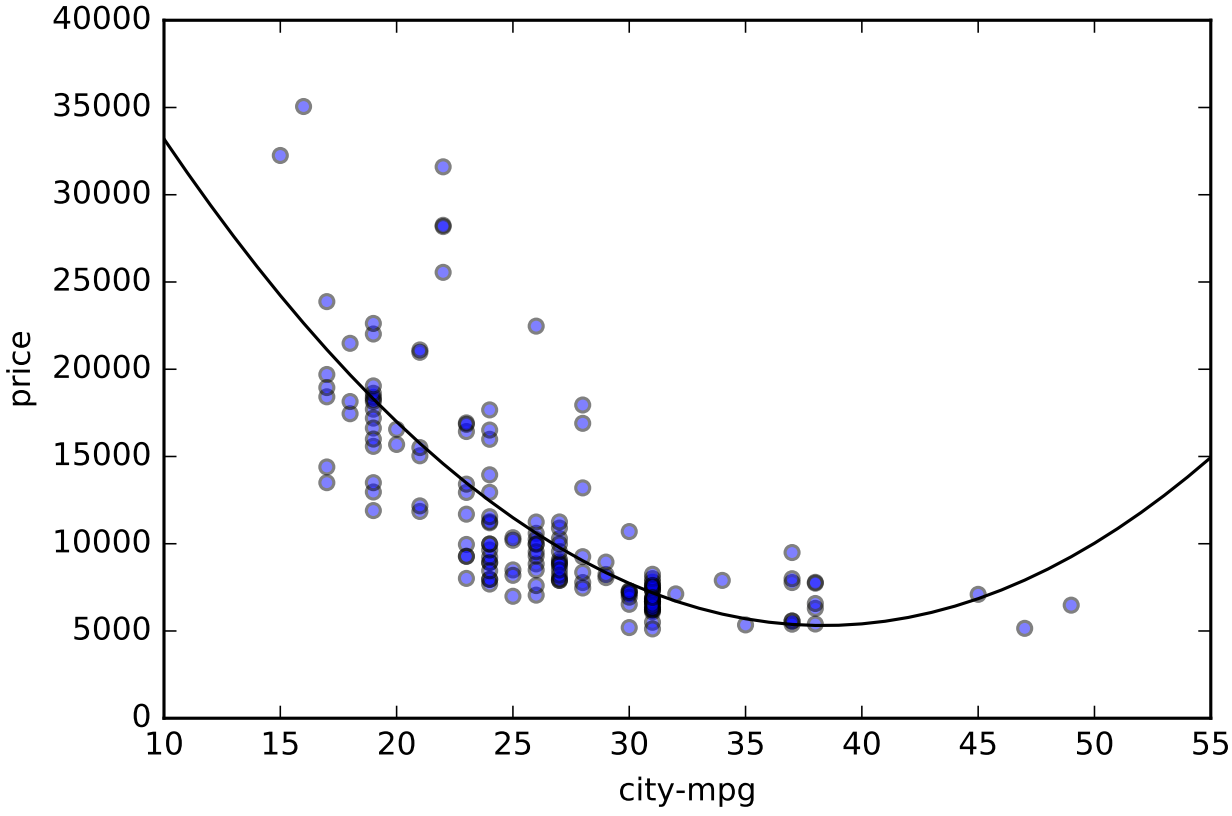


City MPG vs. Price



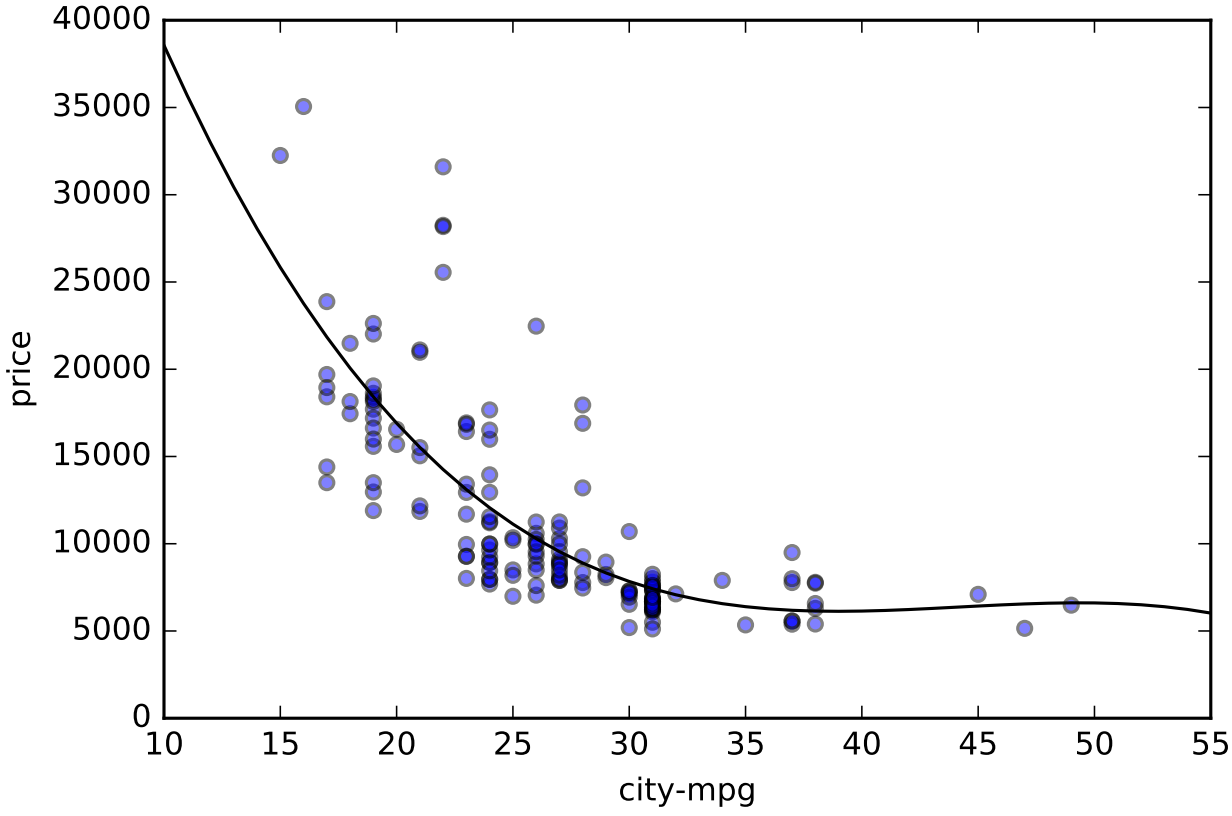
$$y = \beta_0 + \beta_1 x$$

City MPG vs. Price



$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

City MPG vs. Price



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

Is More Better?

Each model we looked at was strictly better than the previous model.

Is More Better?

Each model we looked at was strictly better than the previous model.

E.g., you can get any quadratic model using the cubic model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

by simply setting $\beta_3 = 0$. But β_3 gives you additional flexibility.

Is More Better?

Each model we looked at was strictly better than the previous model.

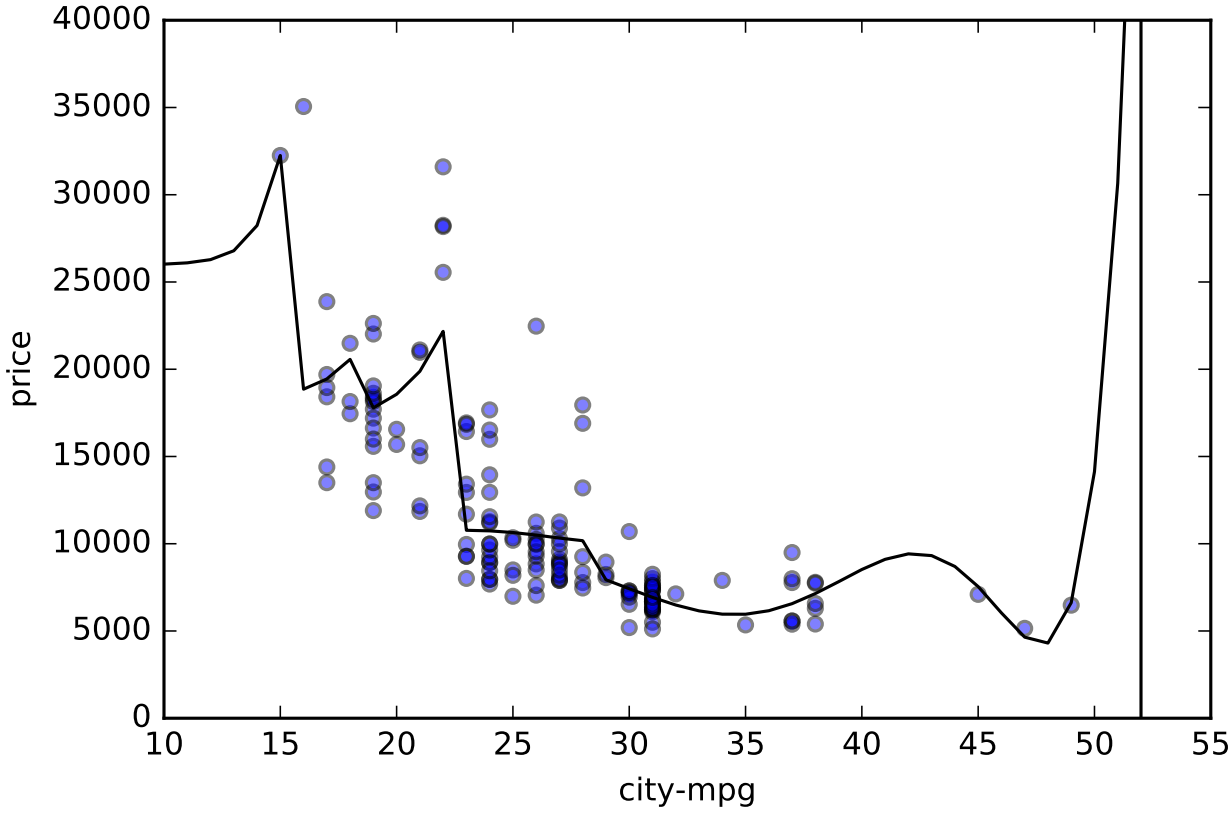
E.g., you can get any quadratic model using the cubic model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

by simply setting $\beta_3 = 0$. But β_3 gives you additional flexibility.

So is the optimal strategy to use as many terms as possible?

City MPG vs. Price



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_{19} x^{19}$$

Overfitting

A more complex model would be better if we had enough data.

Overfitting

A more complex model would be better if we had enough data.

But we often don't have enough data to estimate the parameters in the more complex model.

Overfitting

A more complex model would be better if we had enough data.

But we often don't have enough data to estimate the parameters in the more complex model.

If you increase the number of parameters, you can get closer to the true underlying function (lower **bias**).

Overfitting

A more complex model would be better if we had enough data.

But we often don't have enough data to estimate the parameters in the more complex model.

If you increase the number of parameters, you can get closer to the true underlying function (lower **bias**).

But if you try to estimate more parameters, each individual parameter estimate will be noisier (higher **variance**).

Overfitting

A more complex model would be better if we had enough data.

But we often don't have enough data to estimate the parameters in the more complex model.

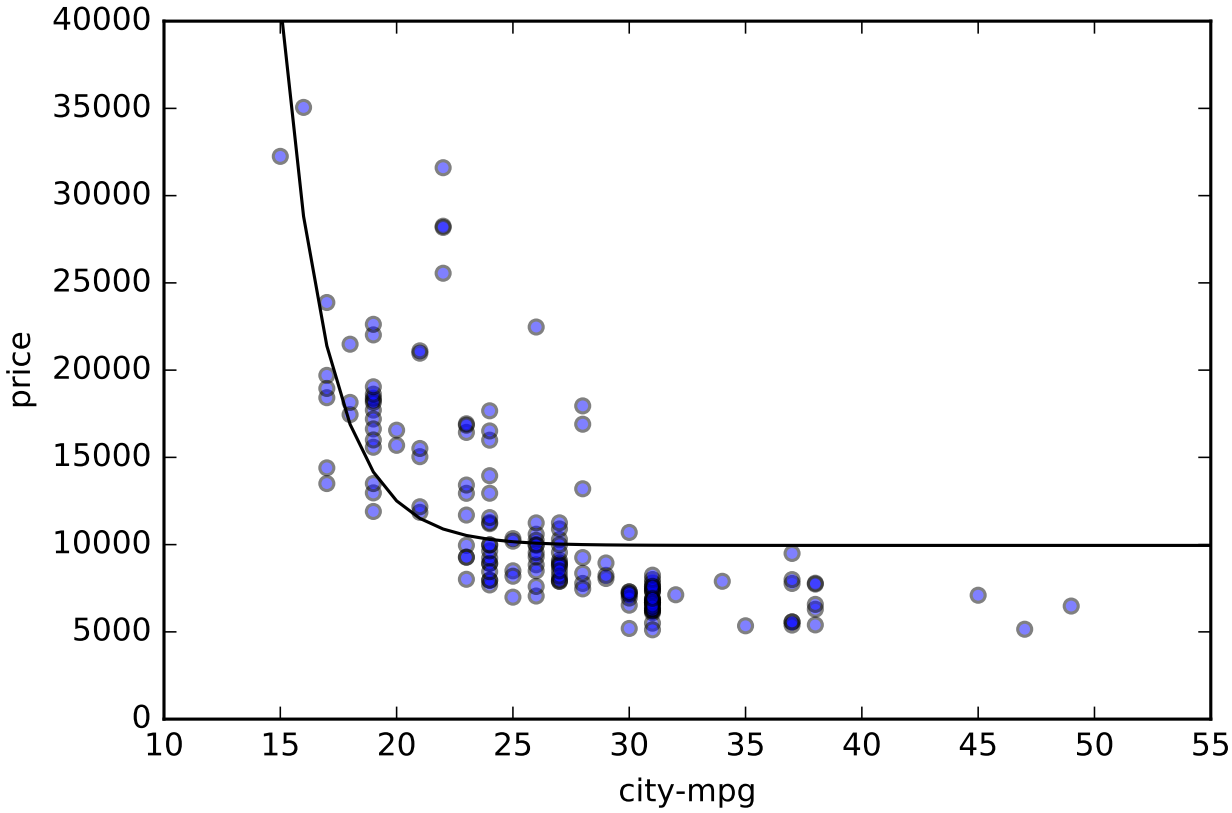
If you increase the number of parameters, you can get closer to the true underlying function (lower **bias**).

But if you try to estimate more parameters, each individual parameter estimate will be noisier (higher **variance**).

This fundamental tradeoff is called the **bias-variance tradeoff**. We want to use a sufficiently complex model to reduce bias, but we don't want a model so complex that we incur a variance penalty.

Other Basis Expansions

We don't have to use polynomials!



$$y = \beta_0 + \beta_1 \exp(-.5x)$$

Other Basis Expansions

In general, we can choose any functions f_1, f_2, ..., f_k:

$$y = \beta_0 + f_1(x) + f_2(x) + ... + f_k(x).$$

These are known as **basis expansions**.

Other Basis Expansions

In general, we can choose any functions f_1, f_2, ..., f_k:

$$y = \beta_0 + f_1(x) + f_2(x) + ... + f_k(x).$$

These are known as **basis expansions**.

Common choices are polynomials and spline basis functions.

Linear Algebra Details

Last time we talked about the linear algebra representation of linear regression.

Linear Algebra Details

Last time we talked about the linear algebra representation of linear regression.

How do we come up with the X matrix for

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3?$$

Linear Algebra Details

Last time we talked about the linear algebra representation of linear regression.

How do we come up with the X matrix for

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 ?$$

$$X = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{pmatrix}$$

Basis Expansions in Scikit-Learn

```
from sklearn.linear_model import LinearRegression
for i in range(1, p+1):
  data["city-mpg^%d" % i] = data["city-mpg"] ** i

model = LinearRegression()
model.fit(data[["city-mpg^%d" % i for i in range(1, p+1)]], data["price

# plot the line
x_test = np.arange(10, 60)
y_test = model.intercept_
for i in range(1, p+1):
  y_test += model.coef_[i-1] * x_test ** i
plt.plot(x_test, y_test)
```

Basis Expansions in Code

In-Class Exercise

Open the notebook `Implementing Linear Regression.ipynb`. *Pick a different quantitative predictor in the autos data set and fit a curve to it. You should try a line first, then using basis expansion if that is unsatisfactory.*

1 Review of Last Time

2 Basis Expansions

3 Interactions

Interactions

We can also combine two or more features. These are known as **interactions**.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2.$$

Interactions

We can also combine two or more features. These are known as **interactions**.

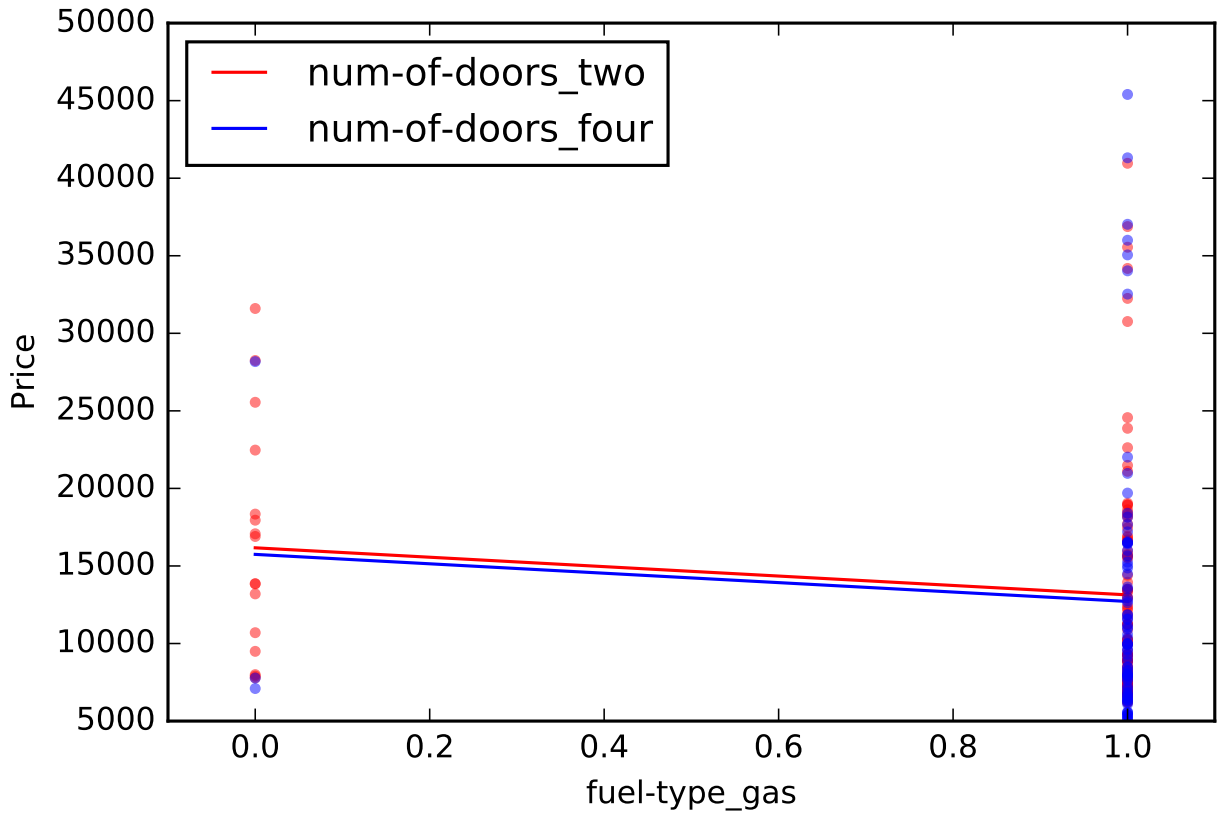
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2.$$

Let's consider the case where x_1 and x_2 are both binary variables, e.g., fuel-type and engine-location.

No Interaction Model

No interaction:

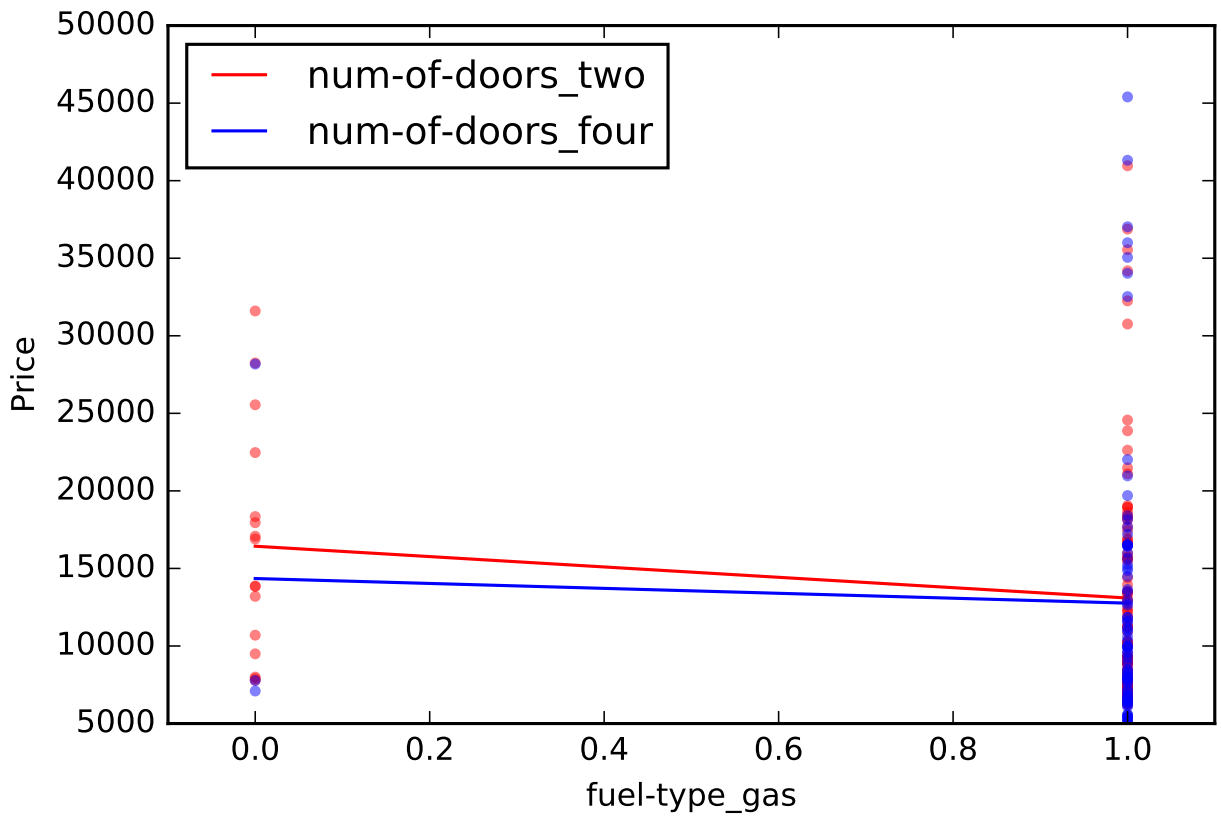
$$y = \beta_0 + \beta_1 x_{\text{fuel-type\_gas}} + \beta_2 x_{\text{num-of-doors\_four}}$$



With Interaction Model

With interaction:

$$y = \beta_0 + \beta_1 x_{\text{fuel-type\_gas}} + \beta_2 x_{\text{num-of-doors\_four}} + \beta_{12} x_{\text{fuel-type\_gas, num-of-doors\_four}}$$



Interactions with Categorical Variables

If you interact a categorical variable with k levels with another categorical variable with ℓ levels, then you end up with $(k-1)(\ell-1)$ binary variables.

Interactions with Categorical Variables

If you interact a categorical variable with k levels with another categorical variable with ℓ levels, then you end up with $(k-1)(\ell-1)$ binary variables.

For example, make (audi, bmw, etc.) and body-style (wagon, sedan, etc.):

$$\begin{aligned} y = \beta_0 + \beta_1 x_{bmw} + ... + \beta_{k-1} x_{volvo} + \\ \alpha_1 x_{sedan} + ... + \alpha_{\ell-1} x_{convertible} + \\ (\alpha\beta)_{11} x_{bmw} x_{sedan} + ... + (\alpha\beta)_{\ell-1,k-1} x_{volvo} x_{convertible} \end{aligned}$$

Interactions with Categorical Variables

If you interact a categorical variable with k levels with another categorical variable with ℓ levels, then you end up with $(k-1)(\ell-1)$ binary variables.

For example, make (audi, bmw, etc.) and body-style (wagon, sedan, etc.):

$$\begin{aligned} y = \beta_0 + \beta_1 x_{bmw} + ... + \beta_{k-1} x_{volvo} + \\ \alpha_1 x_{sedan} + ... + \alpha_{\ell-1} x_{convertible} + \\ (\alpha\beta)_{11} x_{bmw} x_{sedan} + ... + (\alpha\beta)_{\ell-1,k-1} x_{volvo} x_{convertible} \end{aligned}$$

For this model, we have one parameter for each make, body-style combo. So this model is equivalent to calculating the mean price for each make, body-style combo.

Interactions with Categorical Variables

If you interact a categorical variable with k levels with another categorical variable with ℓ levels, then you end up with $(k-1)(\ell-1)$ binary variables.

For example, make (audi, bmw, etc.) and body-style (wagon, sedan, etc.):

$$\begin{aligned} y = \beta_0 + \beta_1 x_{bmw} + ... + \beta_{k-1} x_{volvo} + \\ \alpha_1 x_{sedan} + ... + \alpha_{\ell-1} x_{convertible} + \\ (\alpha\beta)_{11} x_{bmw} x_{sedan} + ... + (\alpha\beta)_{\ell-1,k-1} x_{volvo} x_{convertible} \end{aligned}$$

For this model, we have one parameter for each make, body-style combo. So this model is equivalent to calculating the mean price for each make, body-style combo.

What are some advantages and disadvantages of fitting an interaction model?