Lecture 7: Splines & GAMs

STAT 1361/2360: Statistical Learning and Data Science

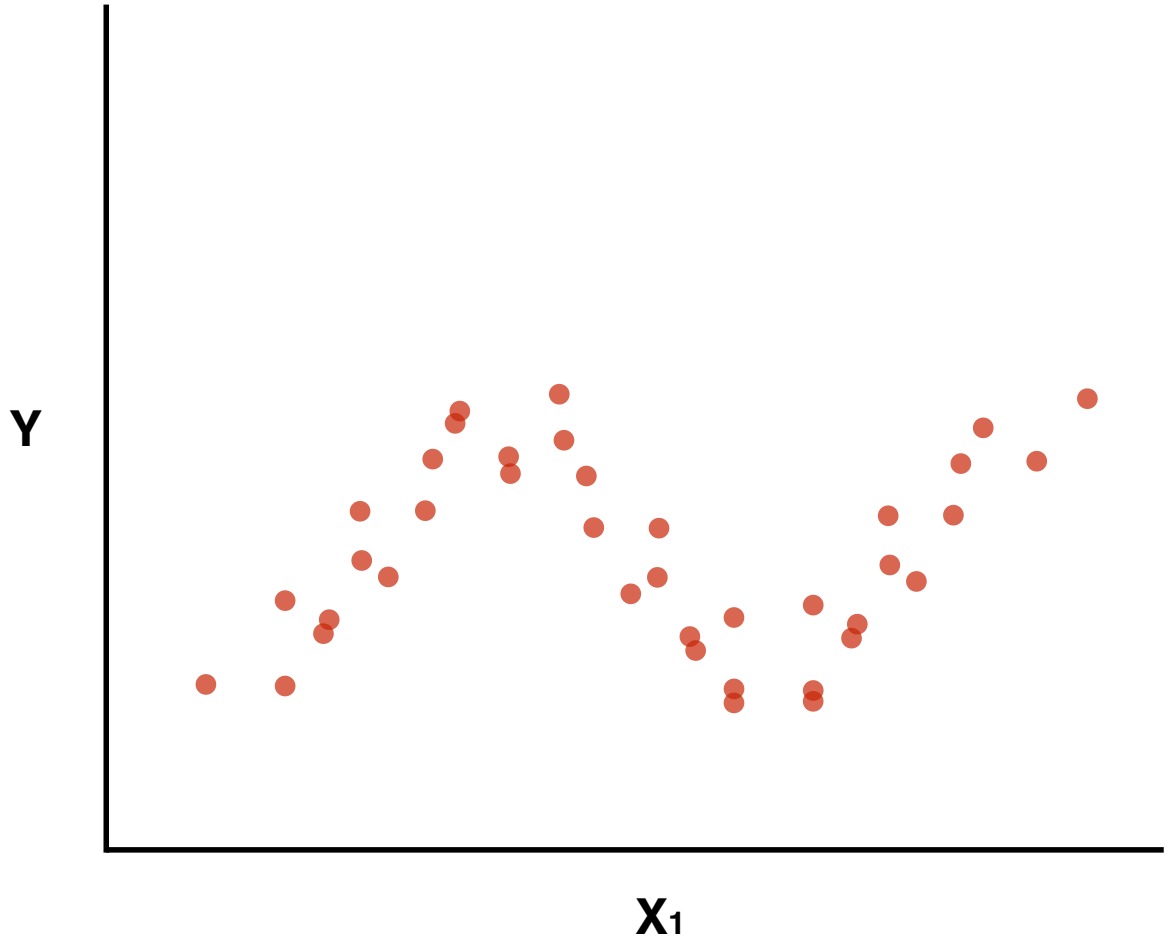
University of Pittsburgh
Prof. Lucas Mentch

Overview

- We've spent a substantial amount of time talking about linear models:
- Model Selection:
  - RSS, R^2, C_p, AIC, BIC, Adjusted R^2, Cross-validation
- Fitting Methods:
  - OLS, Ridge, Lasso
- Dimension Reduction
  - PCA, PLS
- Chapter 7 focuses on more flexible extensions of these models

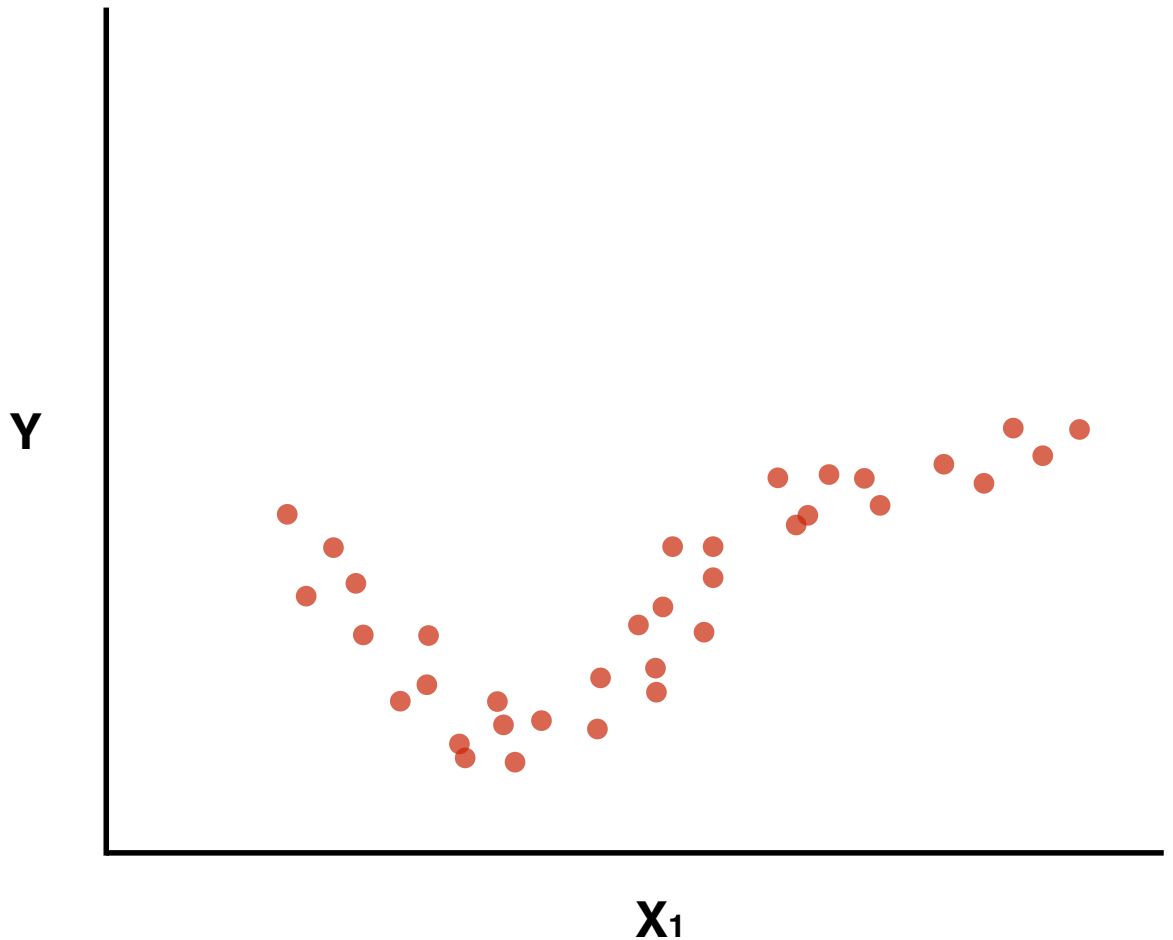
Motivating Examples

As motivation, consider the following data examples:



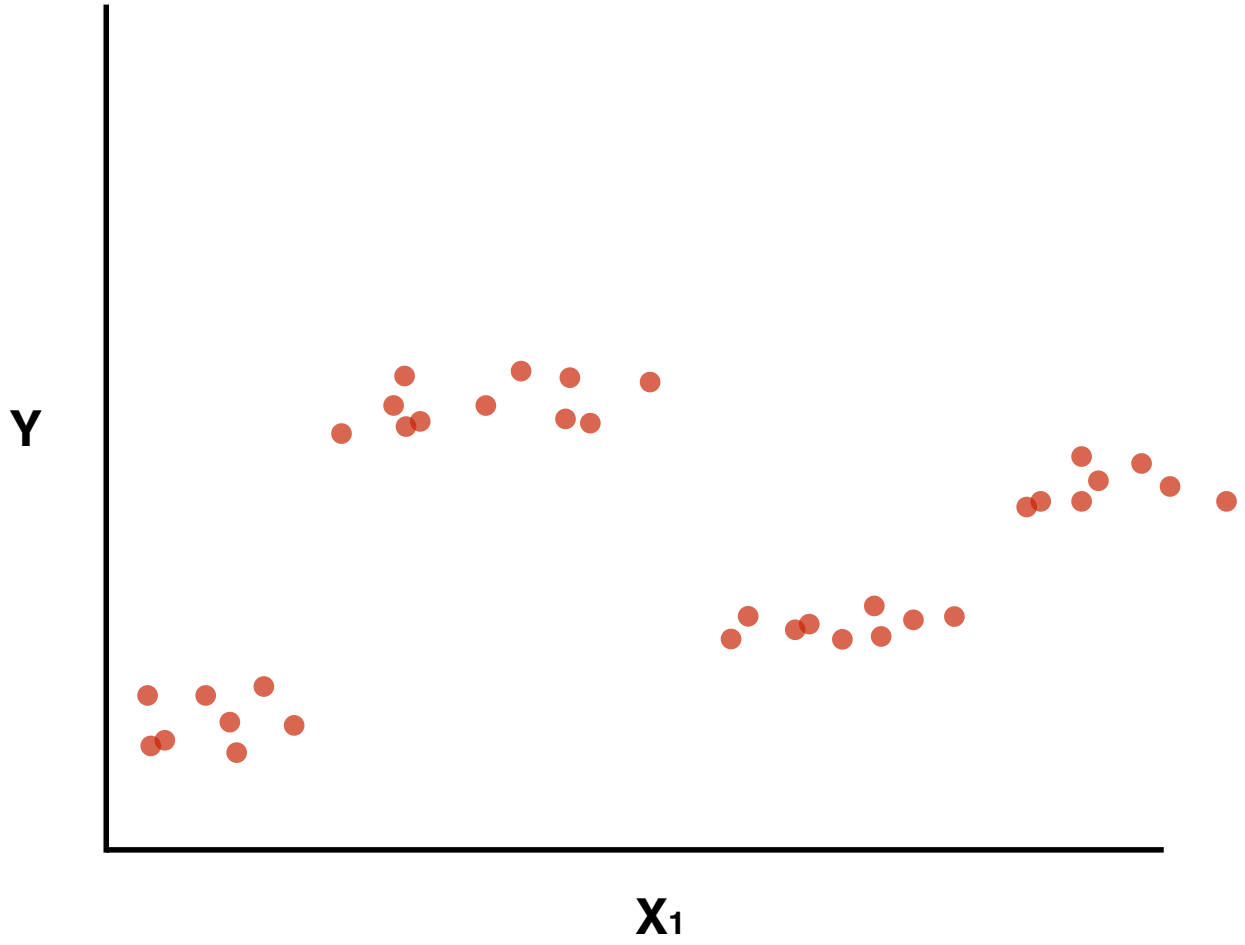
Motivating Examples

As motivation, consider the following data examples:



Motivating Examples

As motivation, consider the following data examples:



Motivating Examples

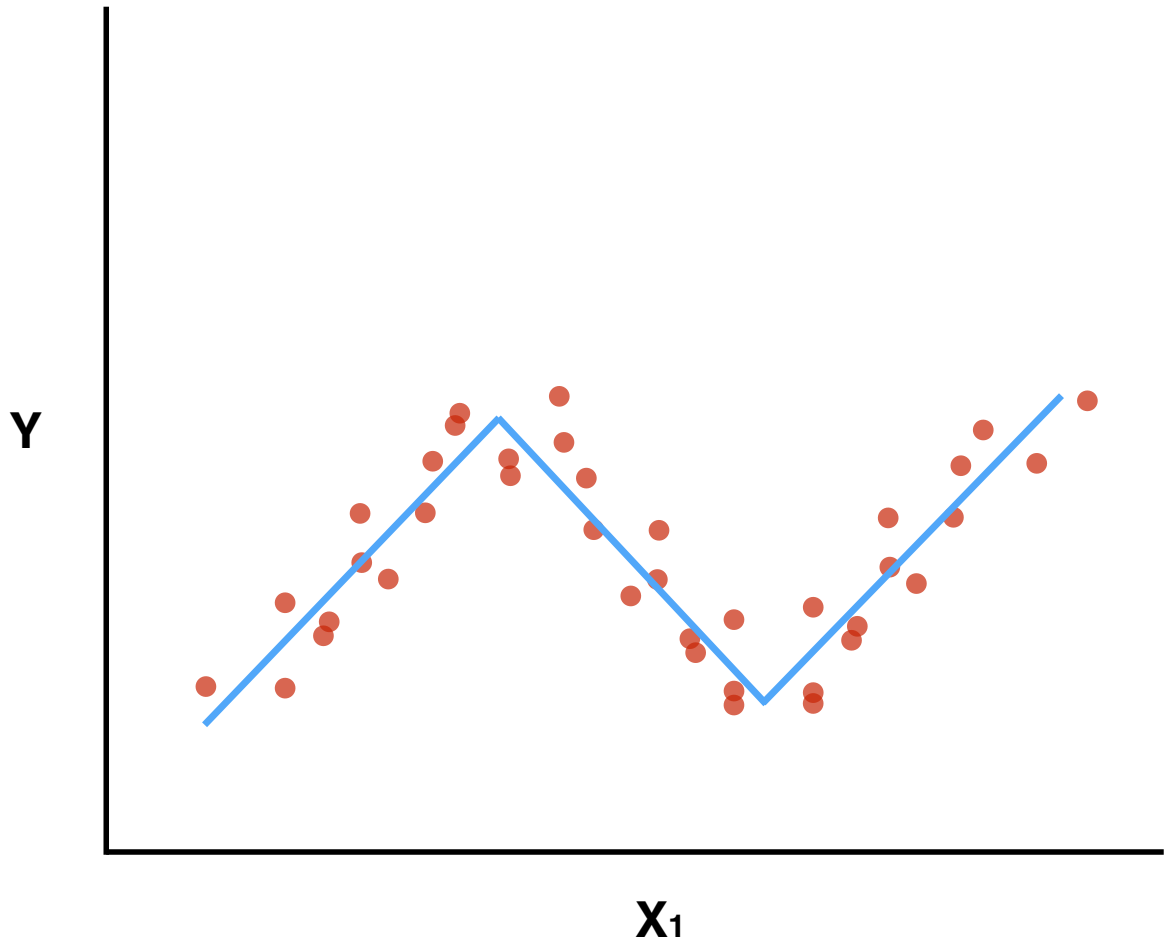
What do these examples have in common? Intuitively, what does it feel like you want to do in order to fit a model that captures these relationships?

Motivating Examples

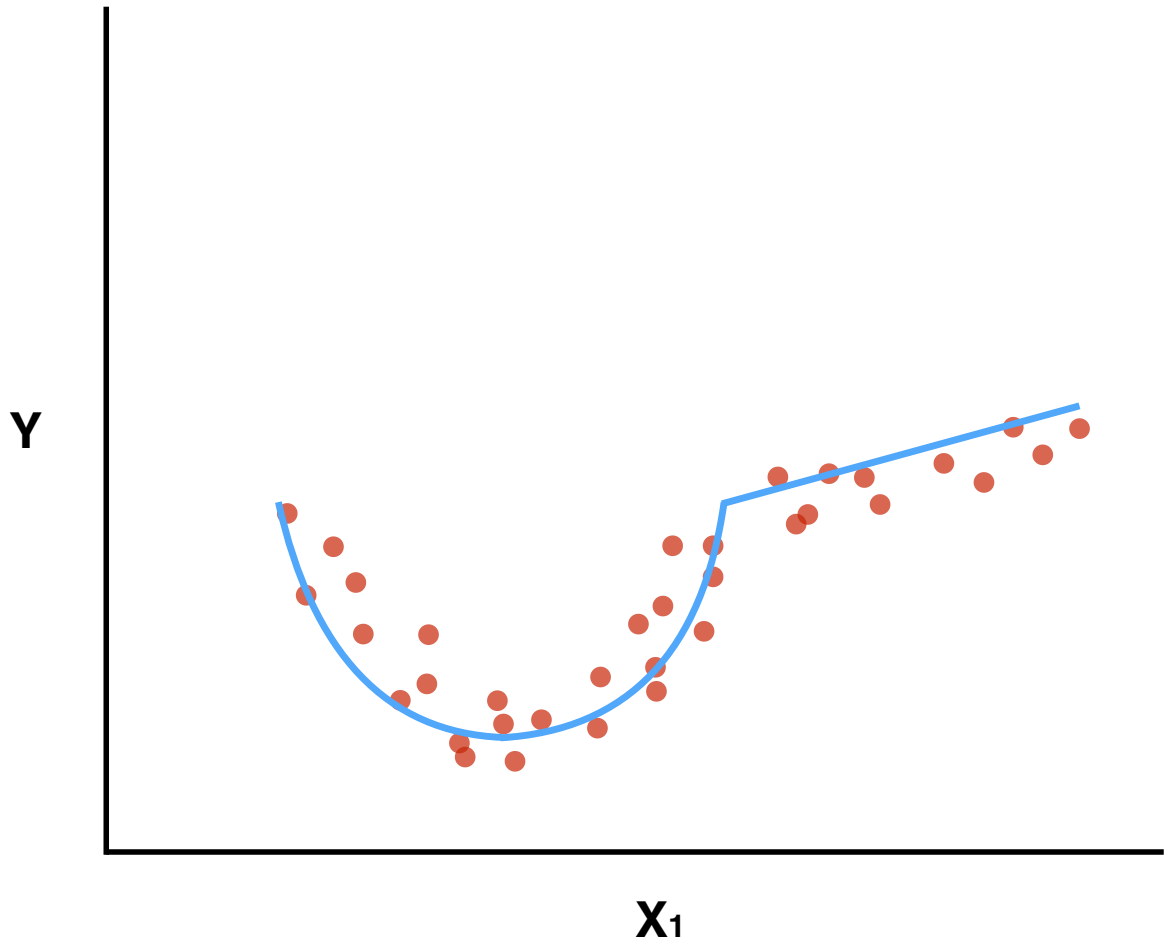
What do these examples have in common? Intuitively, what does it feel like you want to do in order to fit a model that captures these relationships?

Fit different (linear) models in different regions

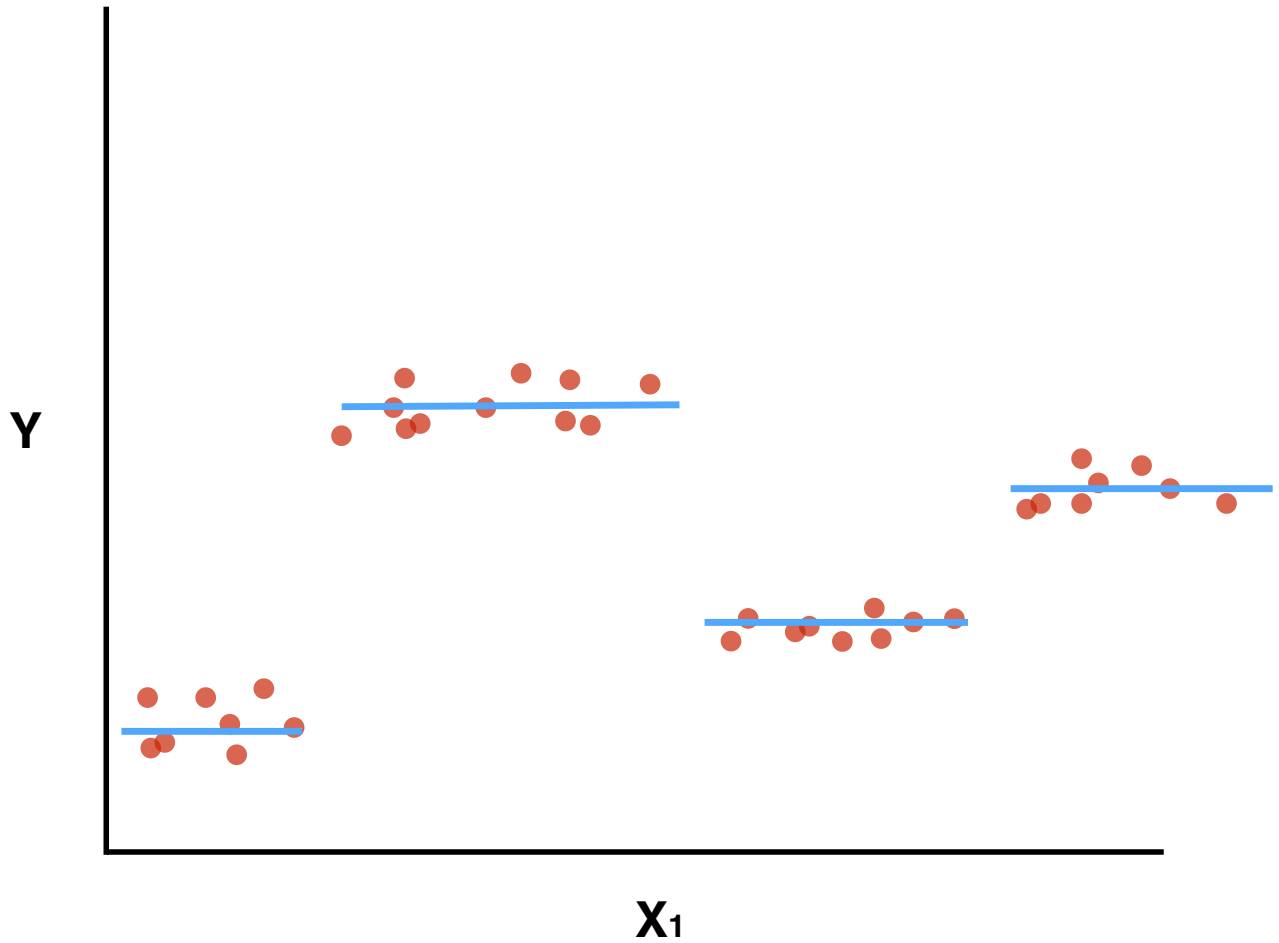
Motivating Examples



Motivating Examples



Motivating Examples



Splines

First Idea

How do we go about doing this? Let's first consider a very basic, straightforward approach,

First Idea

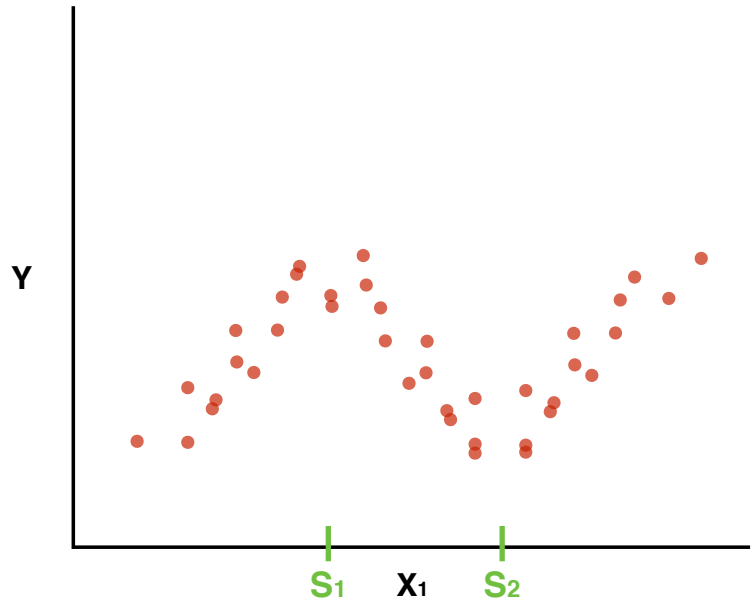
How do we go about doing this? Let's first consider a very basic, straightforward approach,

Create dummy/indicator variables to include in the model that identify each region:

First Idea

How do we go about doing this? Let's first consider a very basic, straightforward approach,

Create dummy/indicator variables to include in the model that identify each region:



$$C_1(X_1) = I\{X_1 \leq S_1\}$$
$$C_2(X_1) = I\{S_1 < X_1 \leq S_2\}$$
$$C_3(X_1) = I\{X_1 \geq S_2\}$$

First Idea

Fitting the following model via OLS would produce a piecewise constant model (one constant per region)

$$Y = \beta_0 + \beta_1 C_1(X_1) + \beta_2 C_2(X_1) + \beta_3 C_3(X_1) + \epsilon$$

First Idea

Fitting the following model via OLS would produce a piecewise constant model (one constant per region)

$$Y = \beta_0 + \beta_1 C_1(X_1) + \beta_2 C_2(X_1) + \beta_3 C_3(X_1) + \epsilon$$

Downsides:

First Idea

Fitting the following model via OLS would produce a piecewise constant model (one constant per region)

$$Y = \beta_0 + \beta_1 C_1(X_1) + \beta_2 C_2(X_1) + \beta_3 C_3(X_1) + \epsilon$$

Downsides:

- Probably want to fit something more complex than just a constant
- Even if we stick with constants, we would want a lot of them and want to be able to choose them in some intelligent, automated way (difficult in high dimensions – more on this in Chapter 8)

Basis Functions

Let's consider the more general idea we've seen before of fitting a model of the form

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_k b_k(X) + \epsilon$$

Basis Functions

Let's consider the more general idea we've seen before of fitting a model of the form

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_k b_k(X) + \epsilon$$

Where have we seen this?

Basis Functions

Let's consider the more general idea we've seen before of fitting a model of the form

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdots + \beta_k b_k(X) + \epsilon$$

Where have we seen this?

- Linear regression: $b_i(X) = X_i$
- Polynomial regression: $b_i(X) = X_i^m$
- PCA: $b_i(X) = Z_i$; Z_i are some linear combination of the X_i
- Step Functions: $b_i(X) = C_i(X) = I\{S_i < X < S_j\}$

Importantly: We can fit any functions of the X_i that we like and still fit via OLS.

Piecewise Linear Models

Think of constants as polynomials of degree 0. How do we go about fitting higher order polynomials in different regions instead of only constants?

Piecewise Linear Models

Think of constants as polynomials of degree 0. How do we go about fitting higher order polynomials in different regions instead of only constants?

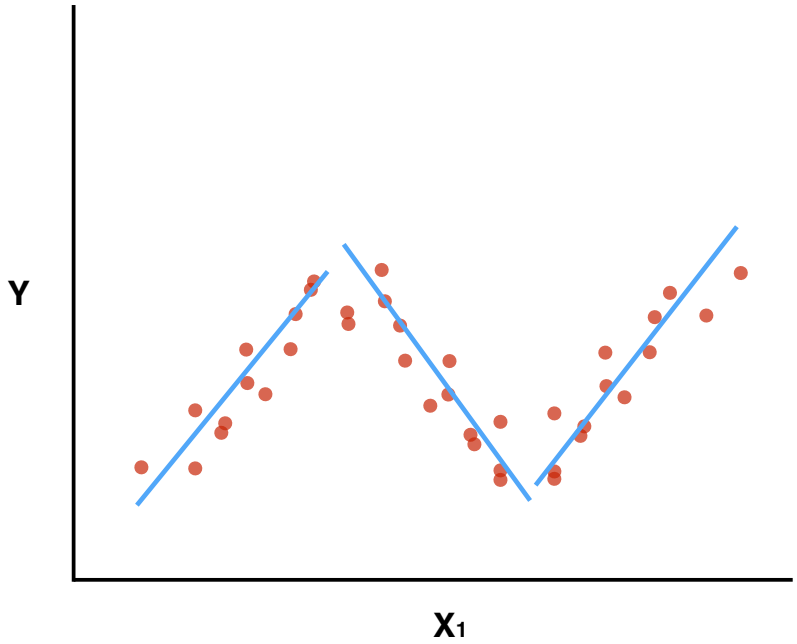
We could just enforce this directly by splitting up the dataset. Suppose for example we wanted to fit two different cubic regression models:

$$Y = \begin{cases} \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_1^3 + \epsilon & \text{if } X_1 \leq S_1 \\ \beta_0 + \beta_1' X_1 + \beta_2' X_1^2 + \beta_3' X_1^3 + \epsilon' & \text{if } X_1 > S_1 \end{cases}$$

Piecewise Linear Models

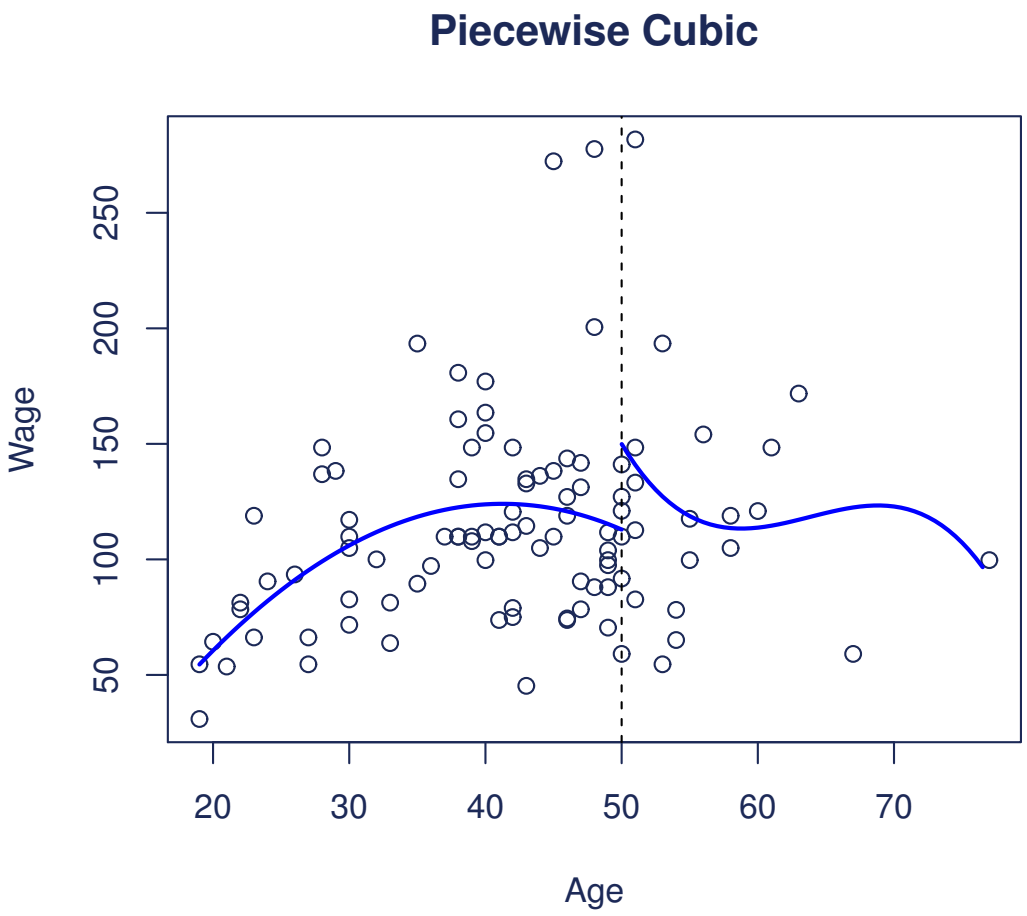
Downsides:

- Now we've got a lot more parameters that need fit (and thus we need a lot more data)
- Just doesn't really seem right – too flexible. Why? What would you like to change about the example below where we fit first order polynomials?



Piecewise Linear Models

Can see the problem more clearly here:



ISLR Fig. 7.3 (top-left)

Piecewise Linear Models

So what's the problem?

Piecewise Linear Models

So what's the problem?

We would like for the functions to be continuous at the break/split points.

Piecewise Linear Models

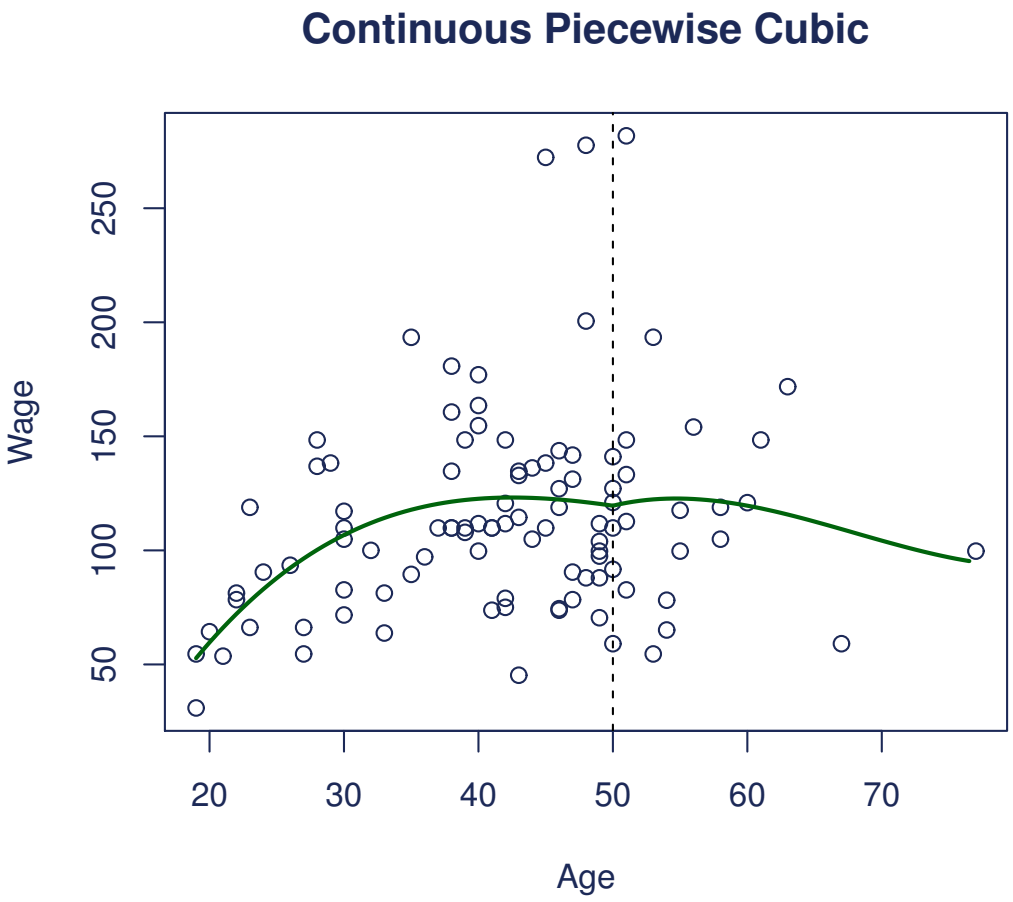
So what's the problem?

We would like for the functions to be continuous at the break/split points.

But is that the whole problem?

Piecewise Linear Models + Continuity

Suppose we enforce continuity on the previous example:



ISLR Fig. 7.3 (top-right)

Splines

Still doesn't seem quite right. It looks better, but ideally we like something that's continuous *and* somewhat smooth at the break/split points.

Splines

Still doesn't seem quite right. It looks better, but ideally we like something that's continuous *and* somewhat smooth at the break/split points.

- This is the big idea behind **splines**.
- The break/split points are called *knots* (get it?)
- We get smoothness across the entire domain by enforcing continuity on not only the function, but also on the first and second derivatives

How do we accomplish this? We'll discuss two different ways ...

Regression Splines

1. Regression Splines: To fit a cubic regression spline with k knots $s_1, \ldots, s_k$, we fit the model with $k+3$ predictors

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \beta_3 X_1^3 + \beta_4 h(X_1, s_1) + \cdots + \beta_{k+3} h(X_1, s_k) + \epsilon$$

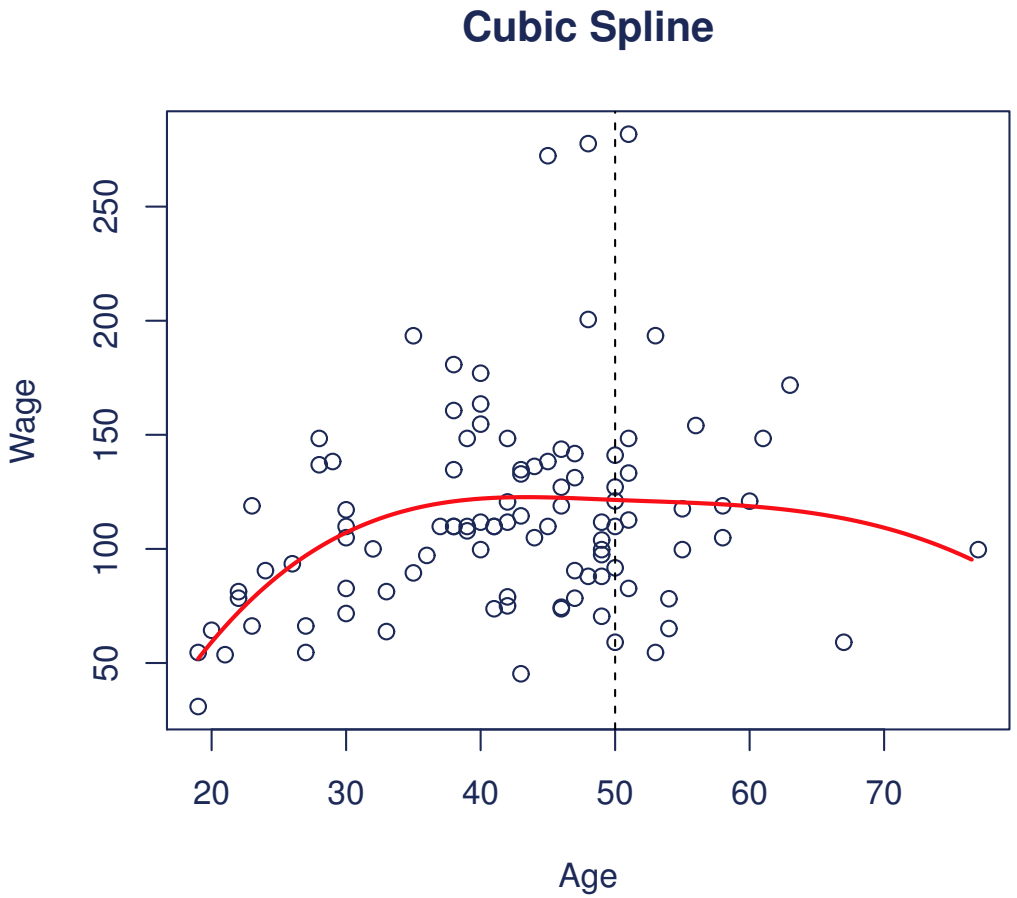
where

$$h(x_1, s_i) = (x_1 - s_i)_+^3 = \begin{cases} (x_1 - s_i)^3 & \text{if } x_1 > s_i \\ 0 & \text{Otherwise} \end{cases}$$

- h is called the truncated power basis function
- Results in “smooth" (continuous) model with continuous first and second derivatives
- Can still be fit via OLS

Cubic Splines

Cubic (regression) splines $\implies$ cubic polynomial fit between each knot:



ISLR Fig. 7.3 (bottom-left)

Cubic Splines

Cubic splines are by far the most popular – why?

Cubic Splines

Cubic splines are by far the most popular – why?

- Seems to be the lowest-order polynomial for which we can't easily see where the knots occur
- Cubic models are fairly robust and tend to cover most relationships we see in practice.
- If the "true model" is actually a lower-degree polynomial, coefficients for higher orders will be small. But what if truth is that it's really a higher order?

Cubic Splines

Cubic splines are by far the most popular – why?

- Seems to be the lowest-order polynomial for which we can't easily see where the knots occur
- Cubic models are fairly robust and tend to cover most relationships we see in practice.
- If the "true model" is actually a lower-degree polynomial, coefficients for higher orders will be small. But what if truth is that it's really a higher order?

We can add more knots!

Regression Spline Notes

Lingering Issues:

1. Where do we put the knots?

Regression Spline Notes

Lingering Issues:

1. Where do we put the knots?

   Intuitively we should put more where the function seems to change the most. In practice, they're usually spaced uniformly.

Regression Spline Notes

Lingering Issues:

1. Where do we put the knots?

   Intuitively we should put more where the function seems to change the most. In practice, they're usually spaced uniformly.

2. How many knots should we use?

Regression Spline Notes

Lingering Issues:

1. Where do we put the knots?

   Intuitively we should put more where the function seems to change the most. In practice, they're usually spaced uniformly.

2. How many knots should we use?

   Equivalent to asking how complex a model we should choose. Can think of this as a tuning parameter $\Longrightarrow$ can choose via cross-validation.

Regression Spline Notes

- Regression Splines tend to be very high-variance in the outermost (first and last) regions
- Solution is to enforce boundary constraints
  - Usually we make the function be linear in the first and last (boundary) regions
  - Enforcing this particular constraint results in what we refer to as a **natural spline**
- Why the term **natural spline**? We'll find out later ...

Smoothing Splines

Smoothing Splines

Let's back up for a second and consider the fact that all we're trying to do with any of this is find some function/model $g(X)$ that fits the data well. Why can't we just minimize some general residual sum of squares

$$RSS = \sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2 \qquad ?$$

Smoothing Splines

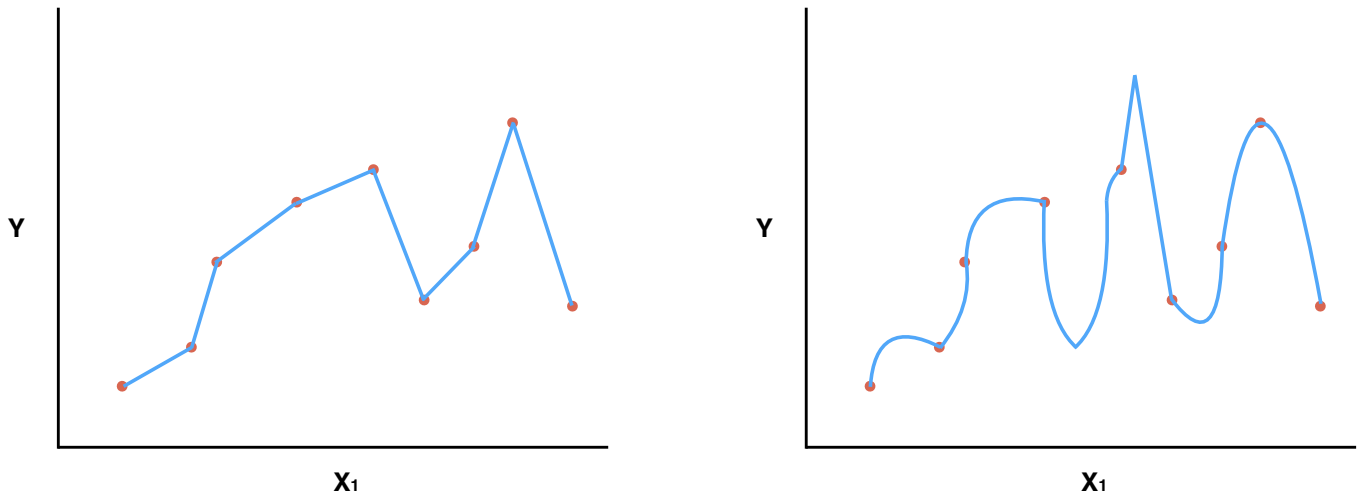
Let's back up for a second and consider the fact that all we're trying to do with any of this is find some function/model $g(X)$ that fits the data well. Why can't we just minimize some general residual sum of squares

$$RSS = \sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2 \qquad ?$$

Because lots and lots (and lots – infinitely many) such functions g would fit perfectly, so we need additional constraints.

Smoothing Splines

Consider the following two models, both with perfect RSS:



Smoothing Splines

- With linear regression, we minimized this subject to the additional constraint that the model g be linear
- Suppose instead that we impose a *smoothness constraint* by finding the model g that minimizes the following loss:

$$(*) \qquad \sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2 \quad + \quad \lambda \int g''(t)^2 dt$$

- Look familiar?

Smoothing Splines

- With linear regression, we minimized this subject to the additional constraint that the model g be linear
- Suppose instead that we impose a *smoothness constraint* by finding the model g that minimizes the following loss:

$$(*) \qquad \sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2 \quad + \quad \lambda \int g''(t)^2 dt$$

- Look familiar? This is the same kind of "loss + penalty" we saw in the regularization methods from Chapter 6. Here again, λ is a tuning parameter.
  - What are we penalizing here?

Smoothing Splines

- With linear regression, we minimized this subject to the additional constraint that the model g be linear
- Suppose instead that we impose a *smoothness constraint* by finding the model g that minimizes the following loss:

$$(*) \qquad \sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2 \quad + \quad \lambda \int g''(t)^2 dt$$

- Look familiar? This is the same kind of "loss + penalty" we saw in the regularization methods from Chapter 6. Here again, λ is a tuning parameter.
  - What are we penalizing here? **The size of the second derivative of the model g.**

Smoothing Splines

$\lambda \int g''(t)^2 dt$ measures the total change in $g'(t)$

$\iff$ Controls the "jumpiness"/smoothness of g

- Larger $\lambda \implies$ smoother g
- $\lambda = 0 \implies$ no penalty $\implies$ g very jumpy (hits all points)
- $\lambda = \infty \implies$ g perfectly smooth (reduces to linear regression OLS line)
- As with every model we've seen with a tuning parameter, λ controls the bias-variance trade-off

Smoothing Splines

- Now, in general, the function that minimizes $(*)$ is called a **smoothing spline**
- However, somewhat amazingly, the function (model) g that minimizes $(*)$ is a cubic spline that is linear in the outermost (first and last) regions, with knots at each data point $x_1, \ldots, x_n$
  - We call this the **natural cubic spline**
  - This is *not* the same as the natural spline we defined before – this one has shrinkage and knots at specific locations – but it explains the name origin

Smoothing Splines

- How do we choose the value of λ?

Smoothing Splines

- How do we choose the value of λ?
  CROSS-VALIDATION!

Smoothing Splines

- How do we choose the value of λ?
  CROSS-VALIDATION!
- BUT, because of this very cool property of smoothing splines, LOOCV is very efficient to compute (essentially n models can be fit at the same computational cost as one single model)
- Argument involves discussion of *effective degrees of freedom* – see book for details

Splines Recap

- Splines allow us to fit different (local) linear models, but in such a way that the end result is a smooth, continuous function
- We looked at two ways of fitting splines:
  - Directly and explicitly via basis functions $\implies$ regression splines
  - By minimizing RSS with a smoothness penalty. In this case it just so happens that the optimal model is a spline (cubic with linear boundaries and knots at data points) $\implies$ smoothing (natural cubic) splines
    - λ controls the amount of shrinkage on the coefficient estimates in the (cubic) spline

Splines Recap

One question that should be in the back of your minds:

If we fit, for example, a cubic regression spline with several knots, we're necessarily introducing *a lot* of extra parameters. Would we do better to fit a single linear model with many more higher-order polynomial terms? E.g. Would a linear model with a 10-degree polynomial do better than a cubic spline with 2 knots?

Splines Recap

One question that should be in the back of your minds:

If we fit, for example, a cubic regression spline with several knots, we're necessarily introducing *a lot* of extra parameters. Would we do better to fit a single linear model with many more higher-order polynomial terms? E.g. Would a linear model with a 10-degree polynomial do better than a cubic spline with 2 knots?

No – It turns out that the higher order polynomial terms introduce too much extra variance, especially at the boundaries

Local Regression

Local Regression

We now move away from splines and consider an idea similar to the one we used earlier as motivation:

Instead of trying to fit polynomials in multiple regions and trying to creatively blend them together, why not just fit a polynomial (linear) model *local* to where we want to predict?

Local Regression

More specifically, suppose we have some point x^* where we want to make a prediction:

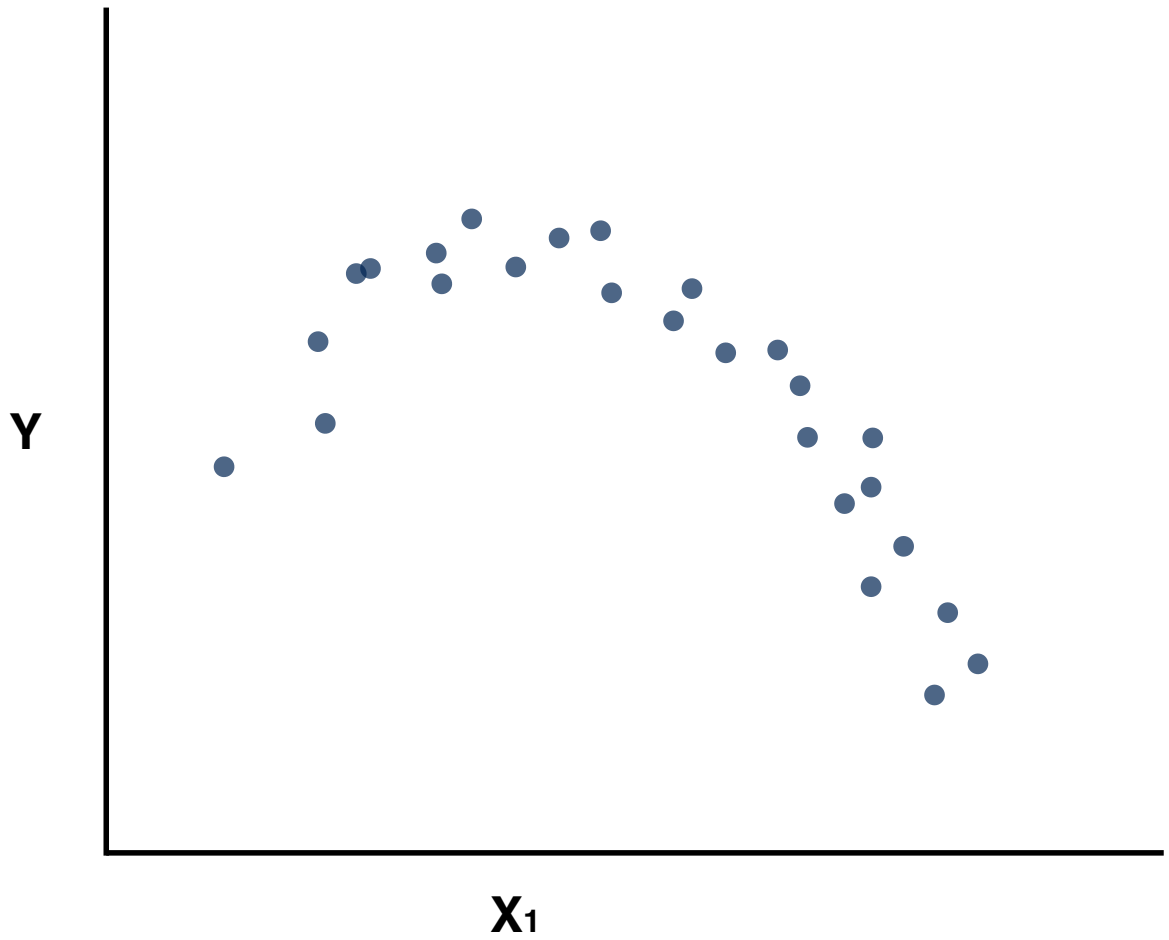
- Define some local neighborhood around x^*
- Assign weights $w_1, ..., w_n$ to all points $x_1, ..., x_n$ in the original dataset
  - Points closer to x^* get more weight; points that fall outside the neighborhood get weights equal to 0
- To fit, for example, a simple linear regression, choose $\hat{\beta}_0, \hat{\beta}_1$ to minimize

$$\sum_{i=1}^{n} w_i \left(y_i - (\beta_0 + \beta_1 x)\right)^2$$

- Predict using $\hat{f}(x^*) = \hat{\beta}_0 + \hat{\beta}_1 x^*$

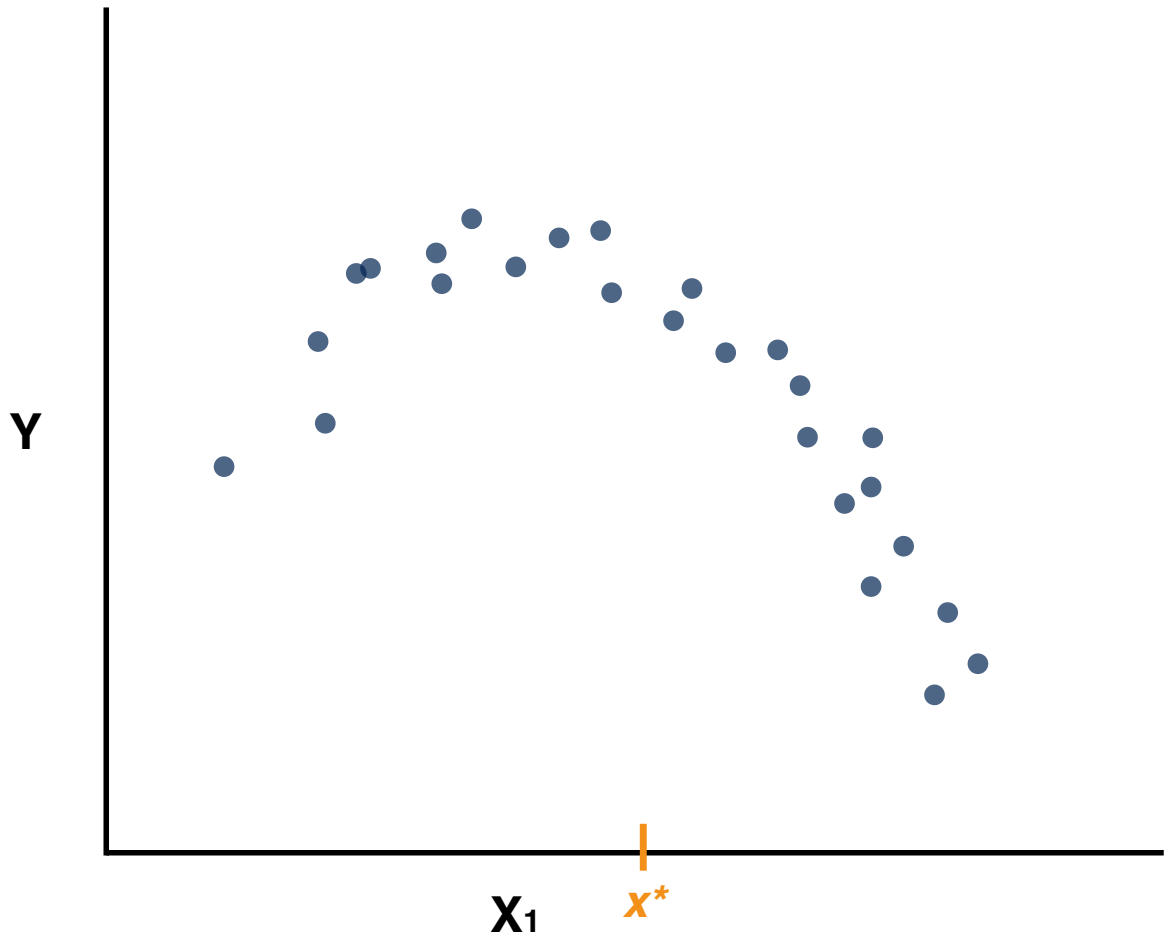
Local Regression: In Pictures

Given the following data ...



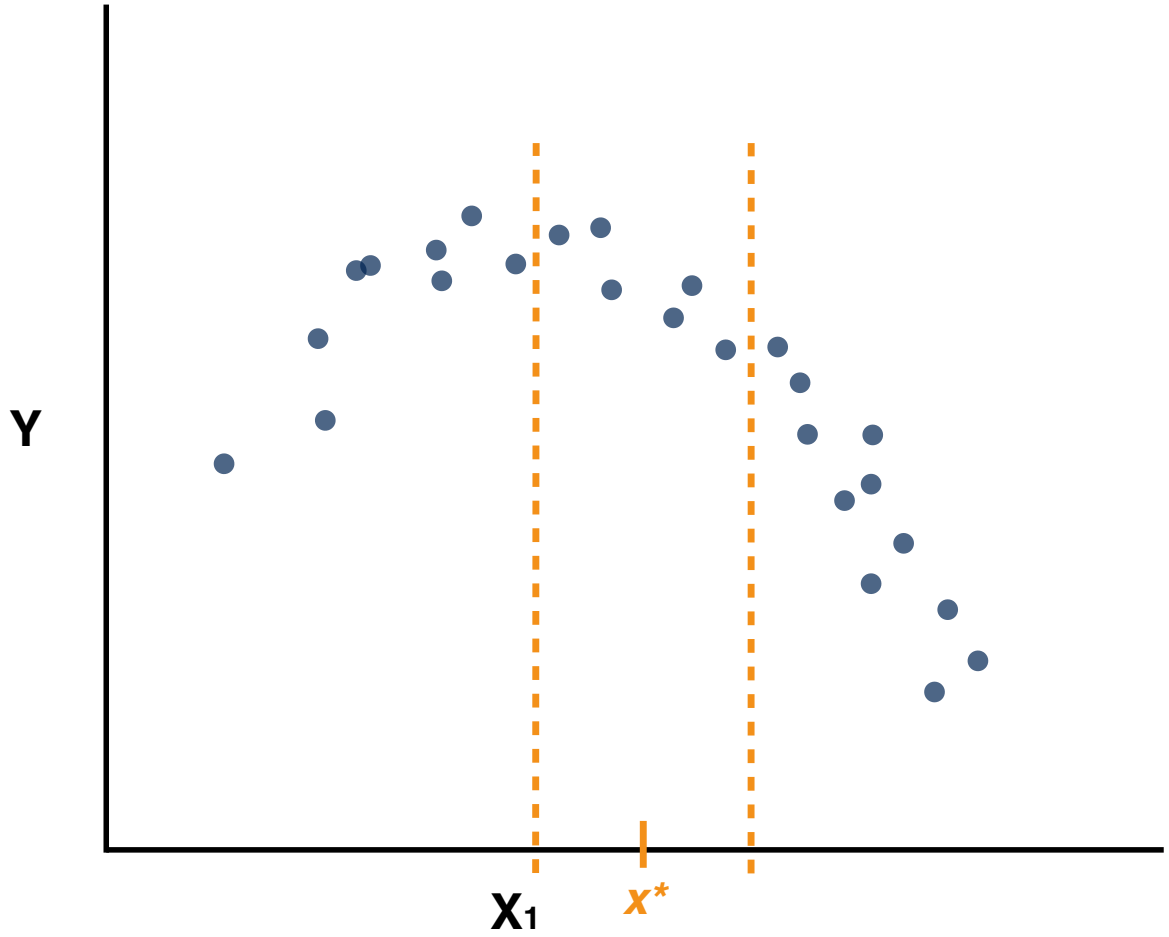
Local Regression: In Pictures

and a place where we want to predict x^* ...



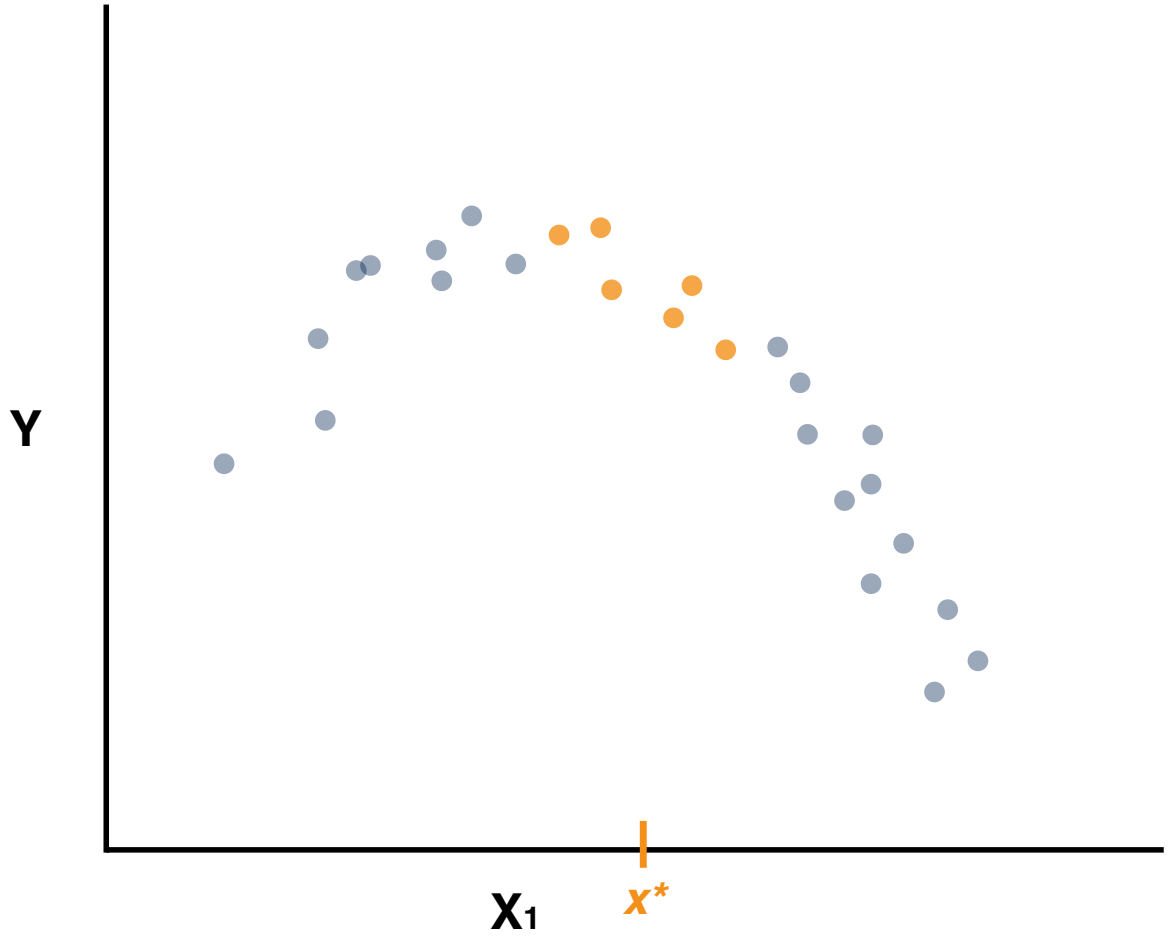
Local Regression: In Pictures

define some neighborhood around x^* ...



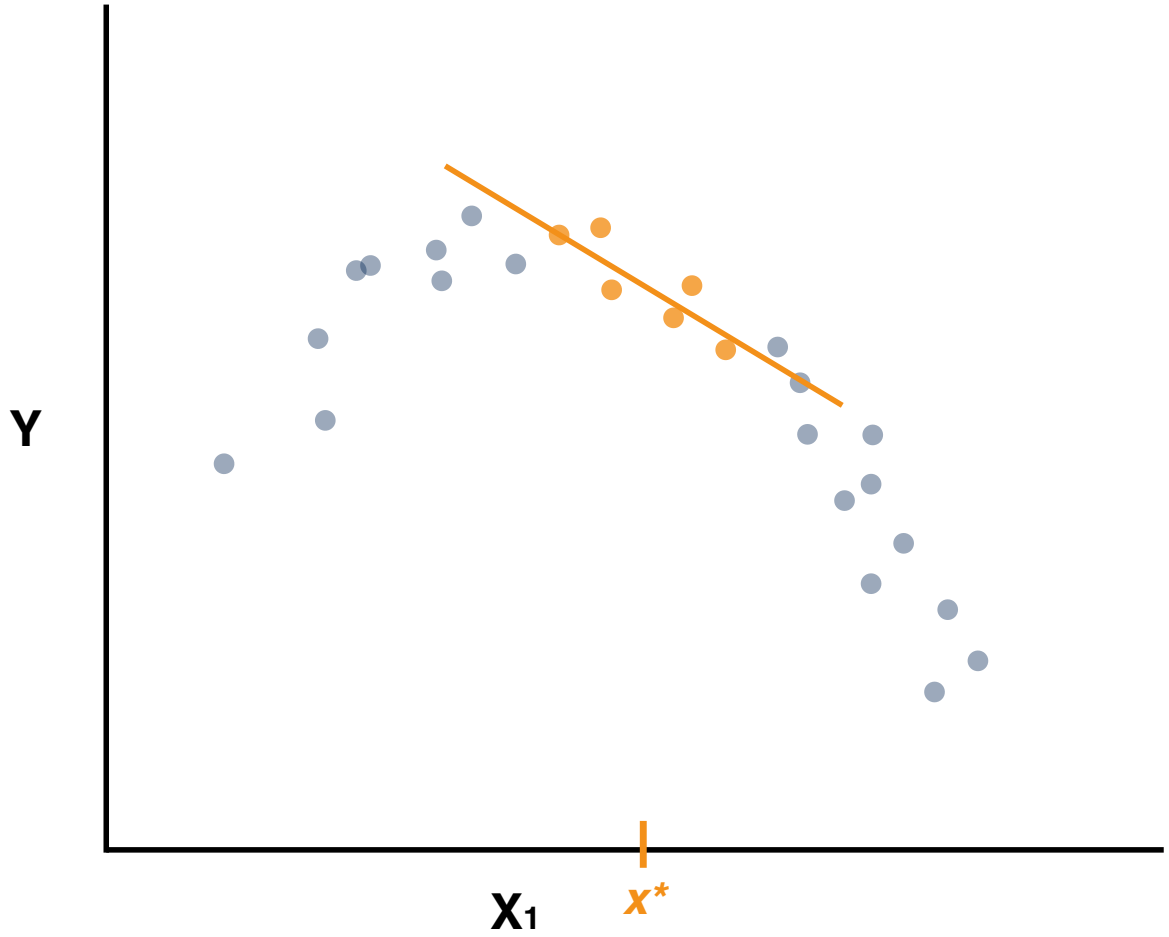
Local Regression: In Pictures

give points closer to x^* more weight and no weight to others ...



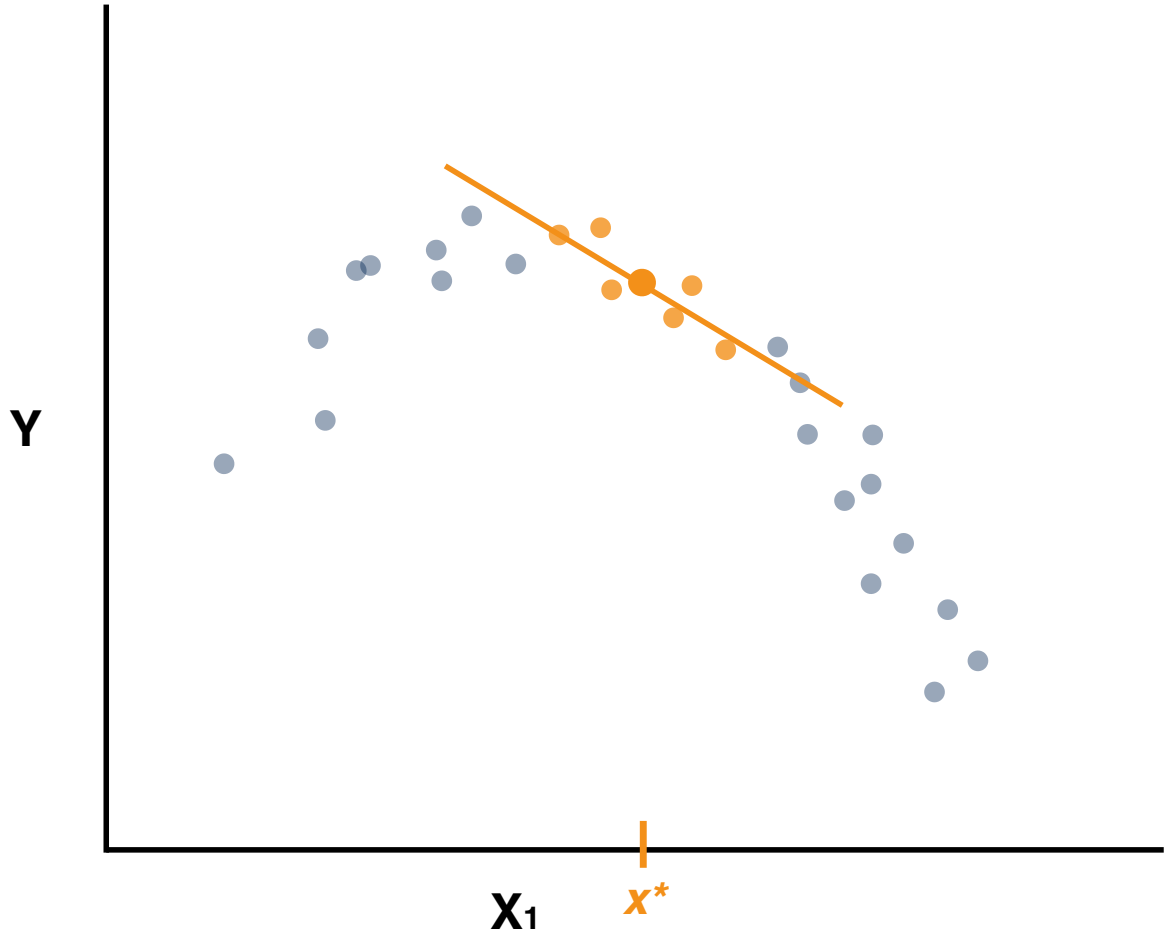
Local Regression: In Pictures

fit some linear model using those data and weights ...



Local Regression: In Pictures

and predict using that model.



Local Regression Notes

- Can fit any kind of local linear model you like, provided there is enough data in the neighborhood
- Some issues with this being too flexible (low bias; high variance)
- Here we only need this basic idea – we'll look at far more intricate and developed versions of this idea in Chapter 8

- Now onto GAMs – our final extension of linear models that we'll see in this class

Generalized Additive Models

Linear Models

We've seen **linear models**:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

and **generalized linear models** (GLM, more flexible):

$$g(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

Generalized Additive Models

and now **generalized *additive* models** (GAM, even more flexible):

$$Y = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + \epsilon$$

Note importantly:

$$(*) \qquad Y = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + \epsilon$$

Is *not* the same as

$$(**) \qquad Y = \beta_0 + \beta_1 f_1(X_1) + \beta_2 f_2(X_2) + \cdots + \beta_p f_p(X_p) + \epsilon$$

Note importantly:

$$(*) \qquad Y = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + \epsilon$$

Is *not* the same as

$$(**) \qquad Y = \beta_0 + \beta_1 f_1(X_1) + \beta_2 f_2(X_2) + \cdots + \beta_p f_p(X_p) + \epsilon$$

(*) is a GAM; (**) is a linear model

- **Both are additive models**

Generalized Additive Models

- The idea of GAMs is similar to the idea we saw in fitting splines with basis functions but with GAMs, the functions $f_i(X_i)$ can be whatever you like

- The functions $f_i(X_i)$ are usually chosen to be some smooth, non-linear function (in fact, splines themselves are a popular choice)

How do we fit GAMs?

- Each function/predictor might be fit according to a different method
  - Means that we can't do something simple like OLS or minimizing RSS
- Most popular methods are based on a method called **backfitting** – hold all predictors fixed except one, update that one, move on to next, repeat

Generalized Additive Models

Pros of GAMs:

- We can fit/involve non-linear functions while maintaining a nice overall model form
  - Can potentially lead to more accurate predictions
  - Additive form means we can still examine individual influence of each predictor on the response
  - Nice attempt at balancing predictive accuracy with flexibility and interpretability (bias/variance)
- Can be used for different kinds of data and for regression or classification (book gives examples of both)

Generalized Additive Models

Cons of GAMs:

- Can be substantially more difficult to fit depending on which functions are chosen
- Still have to choose those functions in the first place (makes balancing bias and variance a bit more tricky – many more knobs to turn)
- Restricted to additive forms, **BUT** interaction terms can still be added manually by adding predictors of the form, e.g. $X_3 = X_1 X_2$, just as we did with standard linear models