Regression in high dimensions

STAT 4710

October 5, 2023

Where we are

Unit 1: R for data mining

Unit 2: Prediction fundamentals

Unit 3: Regression-based methods

Unit 4: Tree-based methods

Unit 5: Deep learning

Lecture 1: Linear and logistic regression

Lecture 2: Regression in high dimensions

Lecture 3: Ridge regression

Lecture 4: Lasso regression

Lecture 5: Unit review and quiz in class

High-dimensional data

High-dimensional data

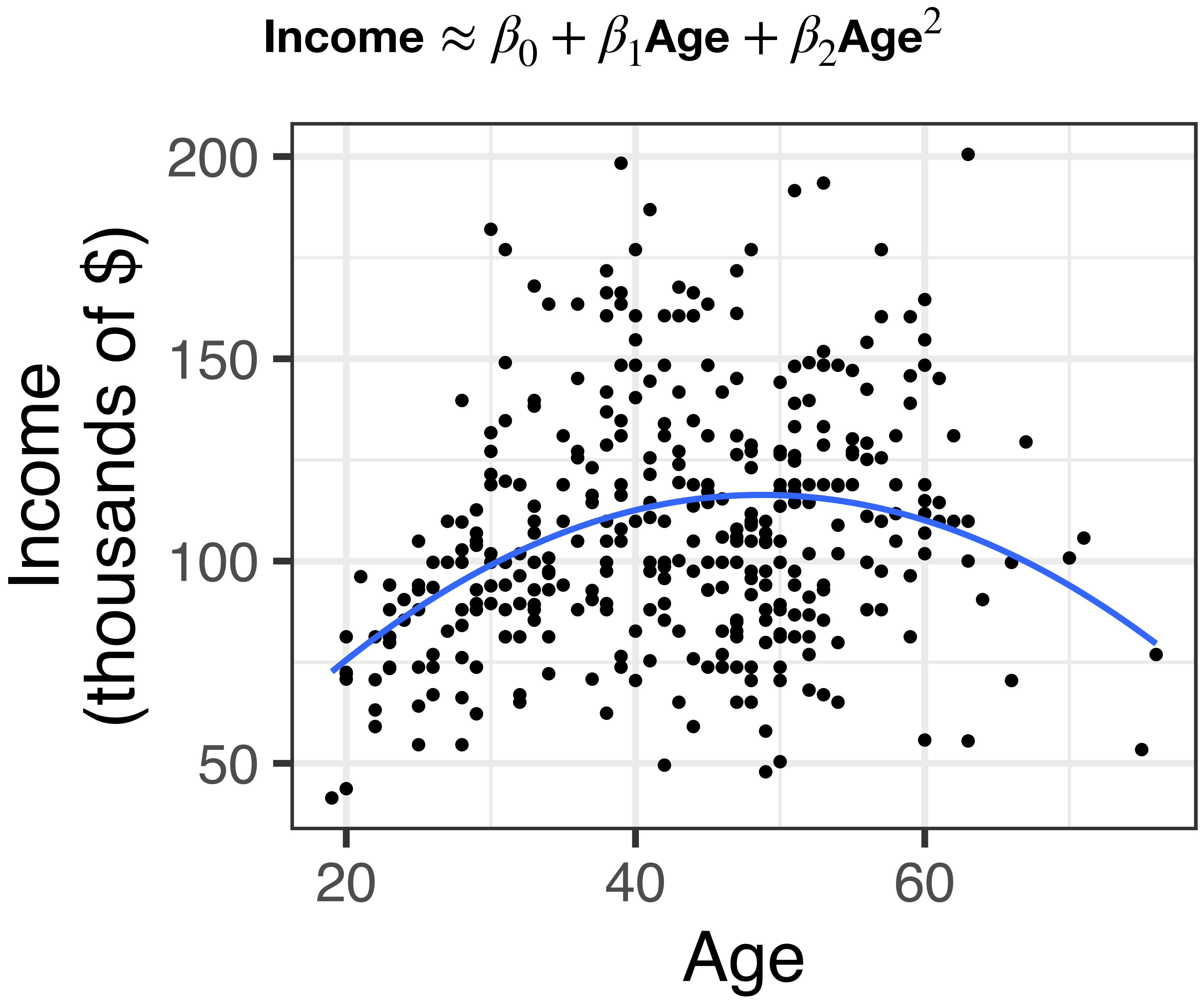
Recall: n is the number of training observations and p is the number of features.

Most datasets we've considered so far have n much larger than p.

High-dimensional data

Recall: n is the number of training observations and p is the number of features.

Most datasets we've considered so far have n much larger than p.



High-dimensional data

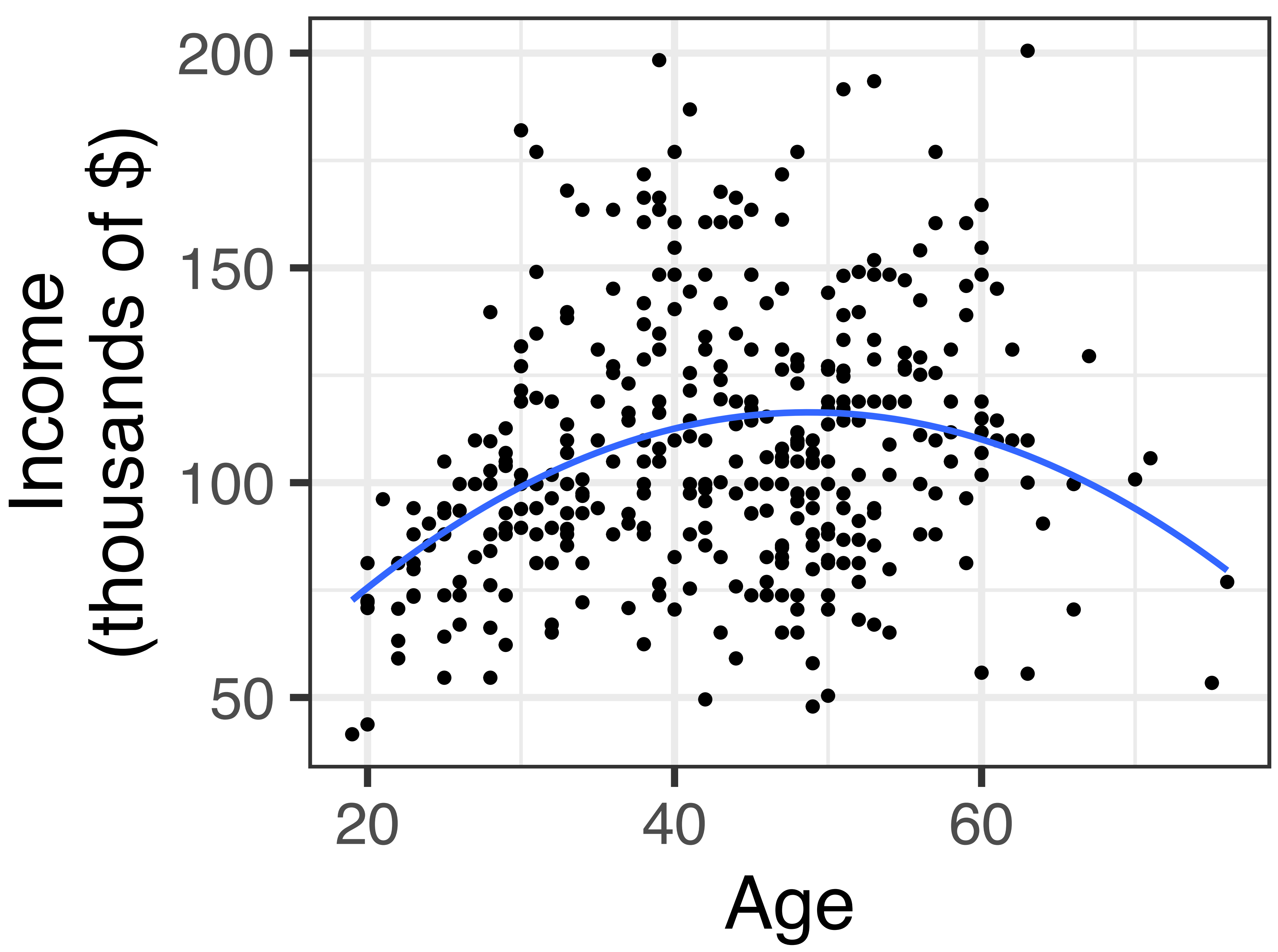
Recall: n is the number of training observations and p is the number of features.

Most datasets we've considered so far have n much larger than p.

In modern applications, can collect very many features for each observation, e.g.:

- Natural language processing
- Image processing
- Genetics/Genomics
- E-commerce

Income $\approx \beta_0 + \beta_1$**Age** $+ \beta_2$**Age**^{2}



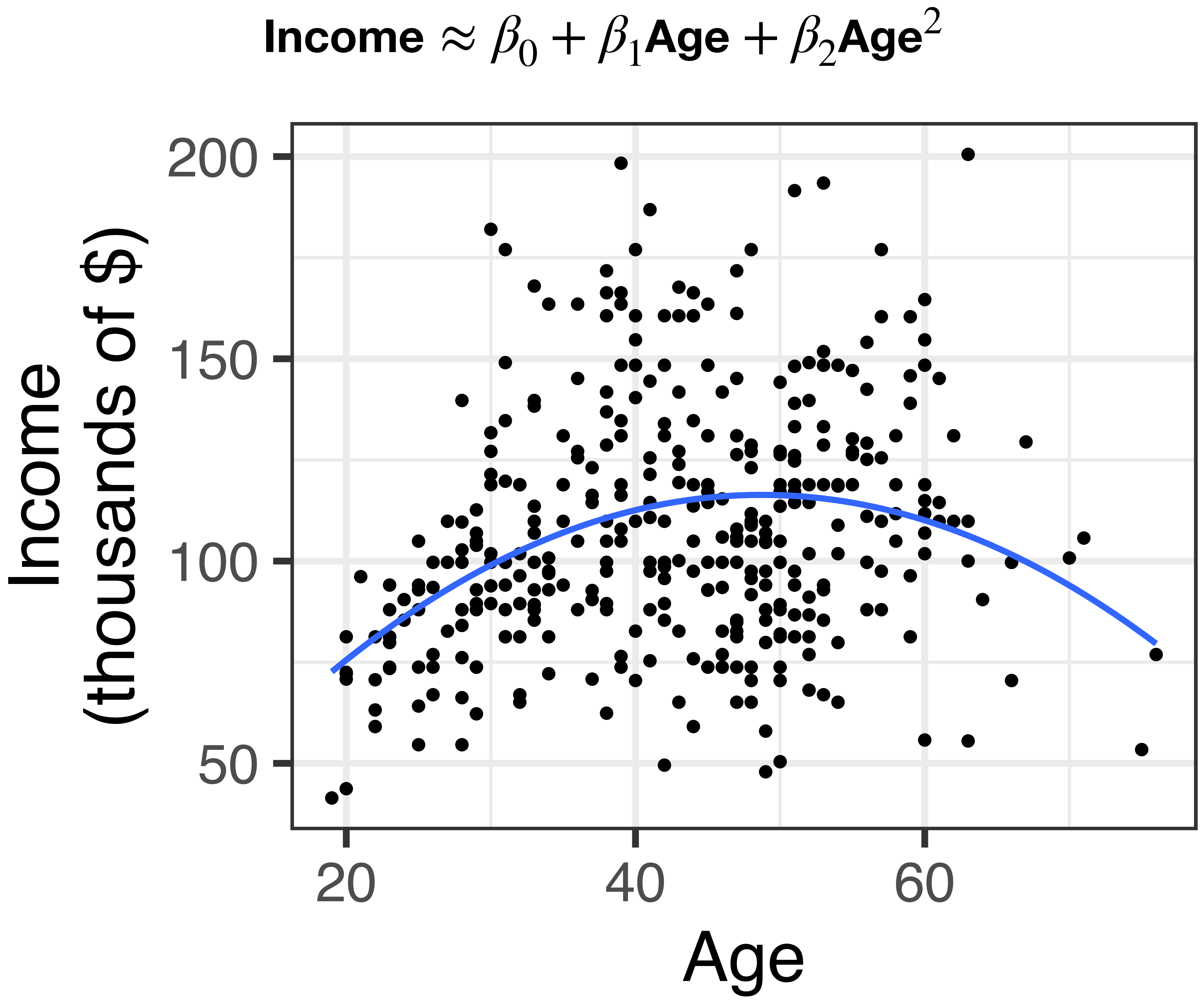
High-dimensional data

Recall: n is the number of training observations and p is the number of features.

Most datasets we've considered so far have n much larger than p.

In modern applications, can collect very many features for each observation, e.g.:

- Natural language processing
- Image processing
- Genetics/Genomics
- E-commerce



High-dimensional data: Data with $p > n$ or $p \approx n$

Challenges in high dimensions

Let’s consider fitting a linear regression with n observations and p features.

If $p > n$, the columns of the feature matrix X guaranteed to be multi-collinear, so the least squares linear regression estimate is not even defined.

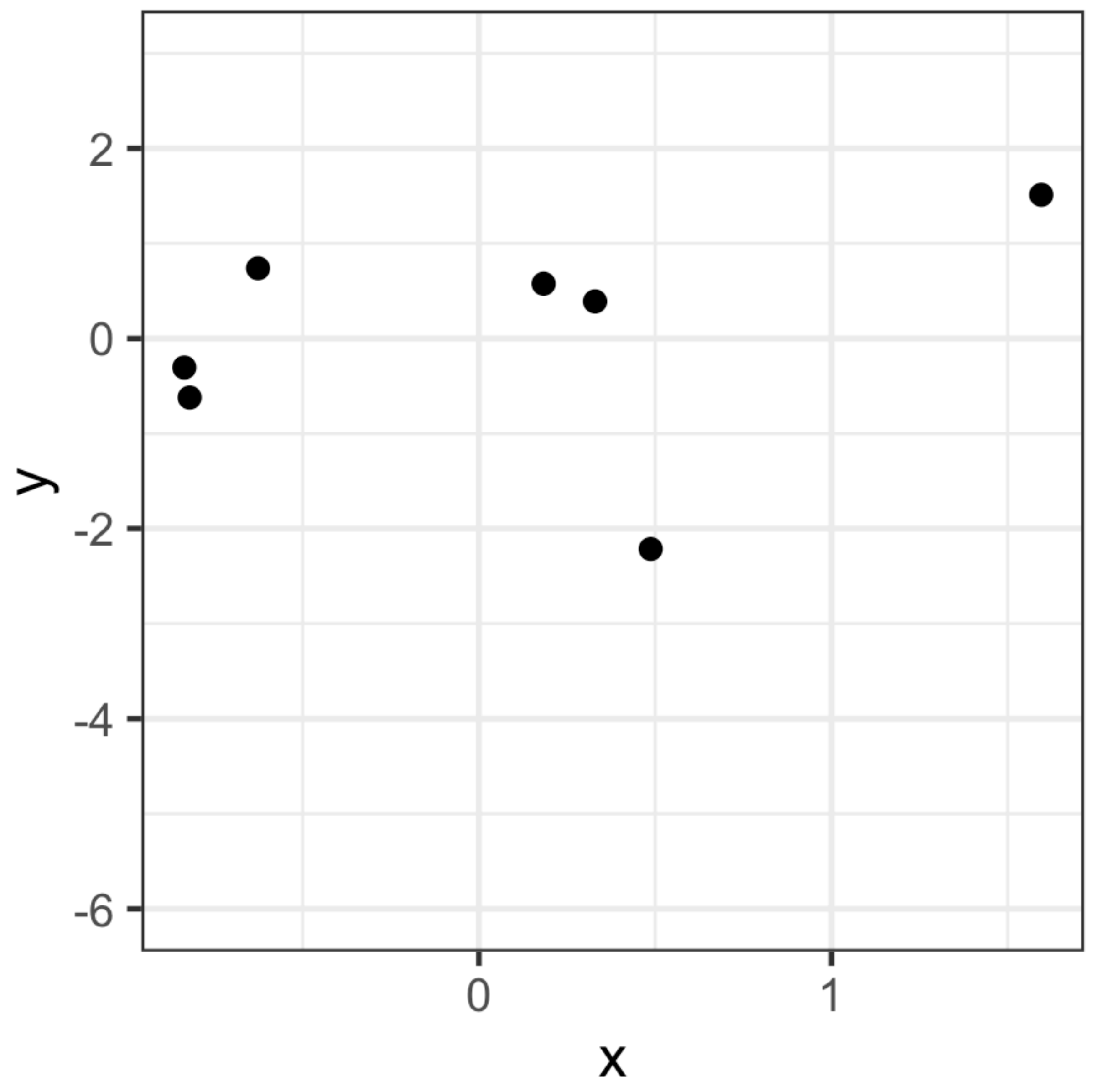
If $p = n$, linear regression will perfectly fit training set, even with “junk” features.

Challenges in high dimensions

Let's consider fitting a linear regression with n observations and p features.

If $p > n$, the columns of the feature matrix X guaranteed to be multi-collinear, so the least squares linear regression estimate is not even defined.

If $p = n$, linear regression will perfectly fit training set, even with "junk" features.

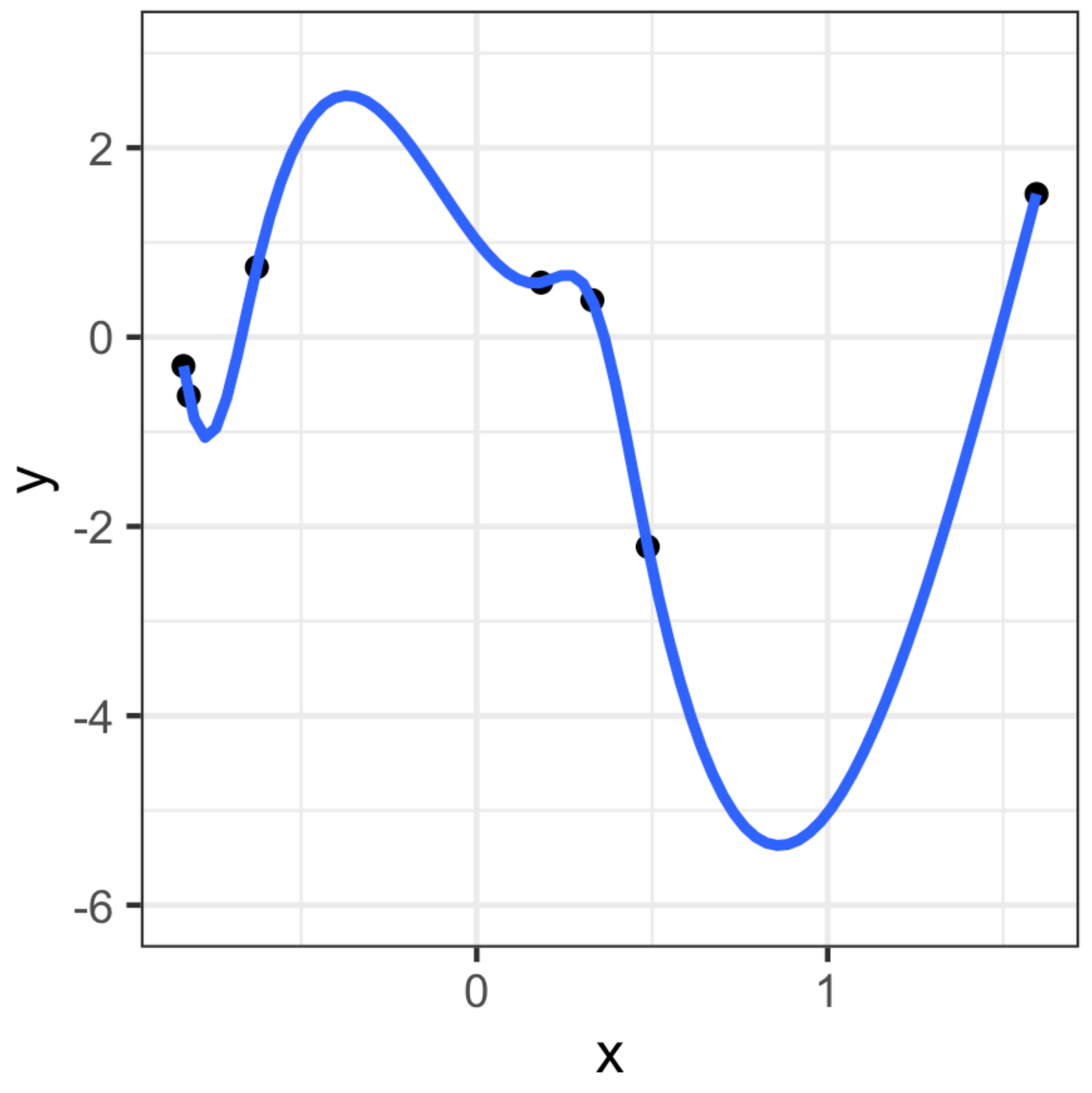


Challenges in high dimensions

Let's consider fitting a linear regression with n observations and p features.

If $p > n$, the columns of the feature matrix X guaranteed to be multi-collinear, so the least squares linear regression estimate is not even defined.

If $p = n$, linear regression will perfectly fit training set, even with "junk" features.



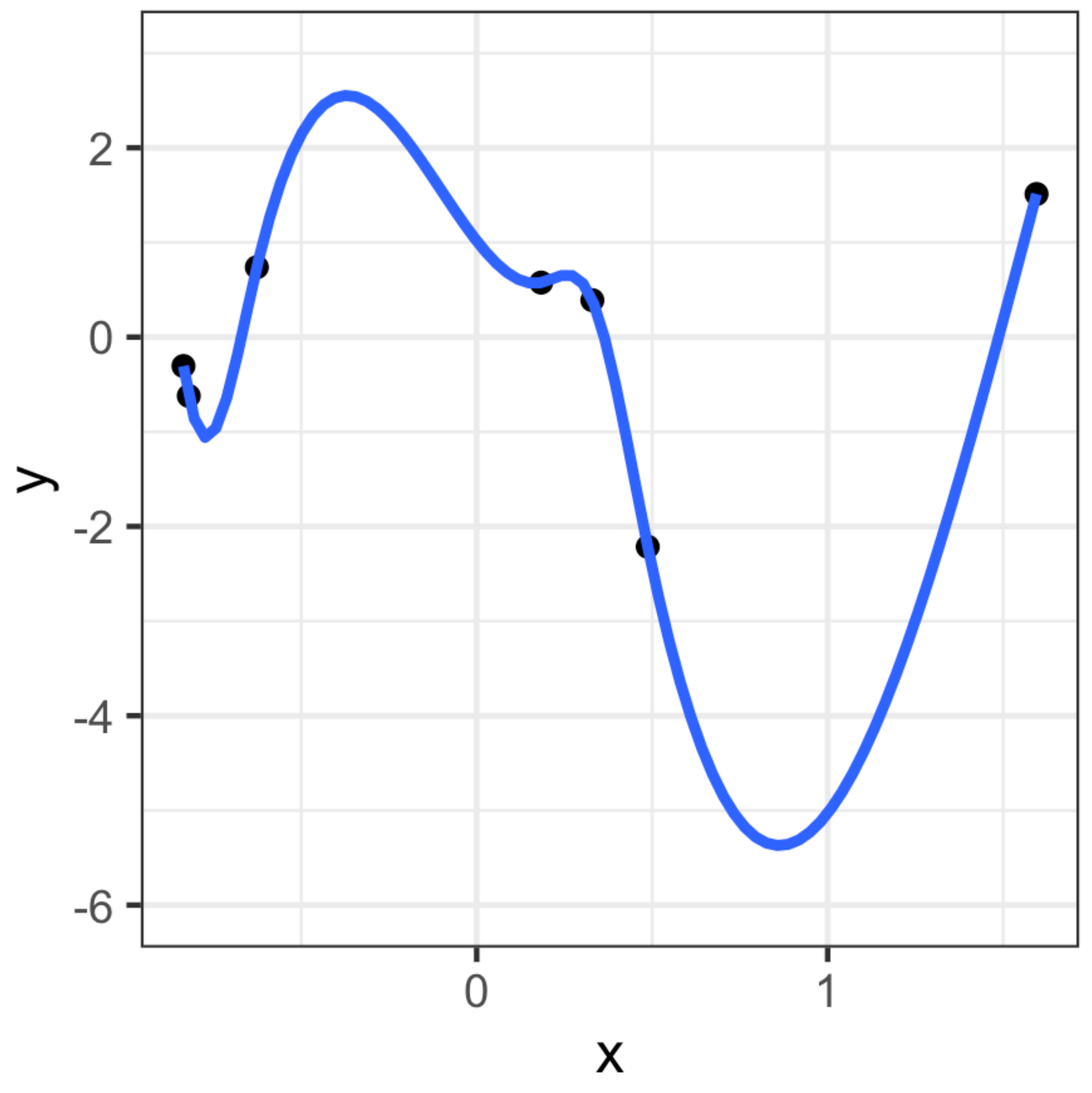
Challenges in high dimensions

Let's consider fitting a linear regression with n observations and p features.

If $p > n$, the columns of the feature matrix X guaranteed to be multi-collinear, so the least squares linear regression estimate is not even defined.

If $p = n$, linear regression will perfectly fit training set, even with "junk" features.

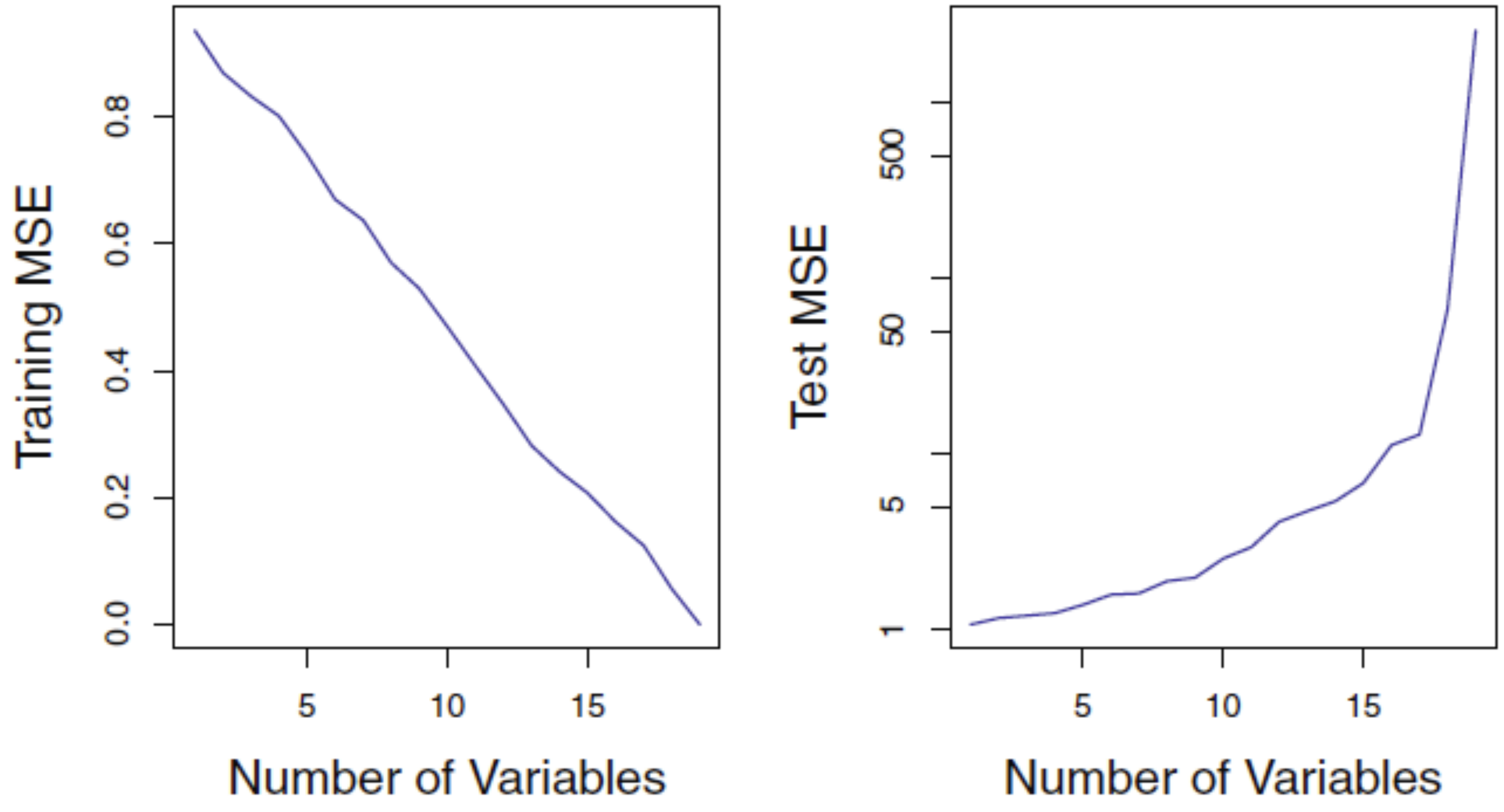
If $p < n$, recall that linear regression variance is $\sigma^2 p/n$.
Therefore, if $p \approx n$ then variance will be very high.



Linear models fit using too many features (i.e. too many degrees of freedom) perform poorly due to high variance.

Challenges in high dimensions (illustration)

Linear regression for $n = 20$; p features unrelated to response



The solution

The solution is to constrain the fitted coefficients in some way, e.g.:

1. Make sure fitted coefficients are not too large (ridge regression).

2. Make sure fitted coefficients are mostly equal to zero (lasso regression).

These constraints reduce the degrees of freedom of the fit, reducing variance.

We are still fitting p coefficients, but using fewer than p degrees of freedom.

Penalization: A way of constraining the fit

Recall least squares solution:

$$\widehat{\beta} = \underset{\beta_0,\beta_1,\ldots,\beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2.$$

Here we let $\widehat{\beta}$ fit the data as close as possible, putting no constraints.

Penalization: A way of constraining the fit

Recall least squares solution:

$$\widehat{\beta} = \underset{\beta_0,\beta_1,\ldots,\beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2.$$

Here we let $\widehat{\beta}$ fit the data as close as possible, putting no constraints.

Penalization: Add a term $P(\beta)$ that measures how “wild” β is, to incentivize β not to be too wild:

$$\widehat{\beta}' = \underset{\beta_0,\beta_1,\ldots,\beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2 + \lambda \cdot P(\beta).$$

Penalization: A way of constraining the fit

Recall least squares solution:

$$\widehat{\beta} = \underset{\beta_0, \beta_1, \ldots, \beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2.$$

Here we let $\widehat{\beta}$ fit the data as close as possible, putting no constraints.

Penalization: Add a term $P(\beta)$ that measures how "wild" β is, to incentivize β not to be too wild:

$$\widehat{\beta}' = \underset{\beta_0, \beta_1, \ldots, \beta_{p-1}}{\arg\min} \underbrace{\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2}_{\text{how well } \beta \text{ fits the data}} + \lambda \cdot P(\beta).$$

Penalization: A way of constraining the fit

Recall least squares solution:

$$\widehat{\beta} = \underset{\beta_0, \beta_1, \ldots, \beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2.$$

Here we let $\widehat{\beta}$ fit the data as close as possible, putting no constraints.

Penalization: Add a term $P(\beta)$ that measures how "wild" β is, to incentivize β not to be too wild:

$$\widehat{\beta}' = \underset{\beta_0, \beta_1, \ldots, \beta_{p-1}}{\arg\min} \underbrace{\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2}_{\text{how well } \beta \text{ fits the data}} + \underbrace{\lambda \cdot P(\beta)}_{\text{how wild } \beta \text{ is}}.$$

Penalization: A way of constraining the fit

Recall least squares solution:

$$\widehat{\beta} = \underset{\beta_0,\beta_1,\ldots,\beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2.$$

Here we let $\widehat{\beta}$ fit the data as close as possible, putting no constraints.

Penalization: Add a term $P(\beta)$ that measures how "wild" β is, to incentivize β not to be too wild:

$$\widehat{\beta}' = \underset{\beta_0,\beta_1,\ldots,\beta_{p-1}}{\arg\min} \underbrace{\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2}_{\text{how well } \beta \text{ fits the data}} + \underbrace{\lambda \cdot P(\beta)}_{\text{how wild } \beta \text{ is}}.$$

how well β fits the data ⟵ compromise ⟶ how wild β is

Example: L0-penalized regression

Consider the penalized regression

$$\widehat{\beta}' = \underset{\beta_0, \beta_1, \ldots, \beta_{p-1}}{\arg\min} \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 X_{i1} + \cdots + \beta_{p-1} X_{i,p-1}))^2 + \lambda \cdot P(\beta),$$

with $P(\beta) = |\{j : \beta_j \neq 0\}|$.

The L0 penalty P counts the number of nonzero entries in β, and creates sparse solutions $\widehat{\beta}$.

The optimization above is computationally infeasible, so in practice we use a different penalty (called the lasso) to achieve sparsity (stay tuned for Lecture 4).

How and when penalization works

Penalization reduces the variance, but increases the bias of the predictions.

$\Rightarrow$ Reduces test error when reduction in variance outweighs increase in bias.

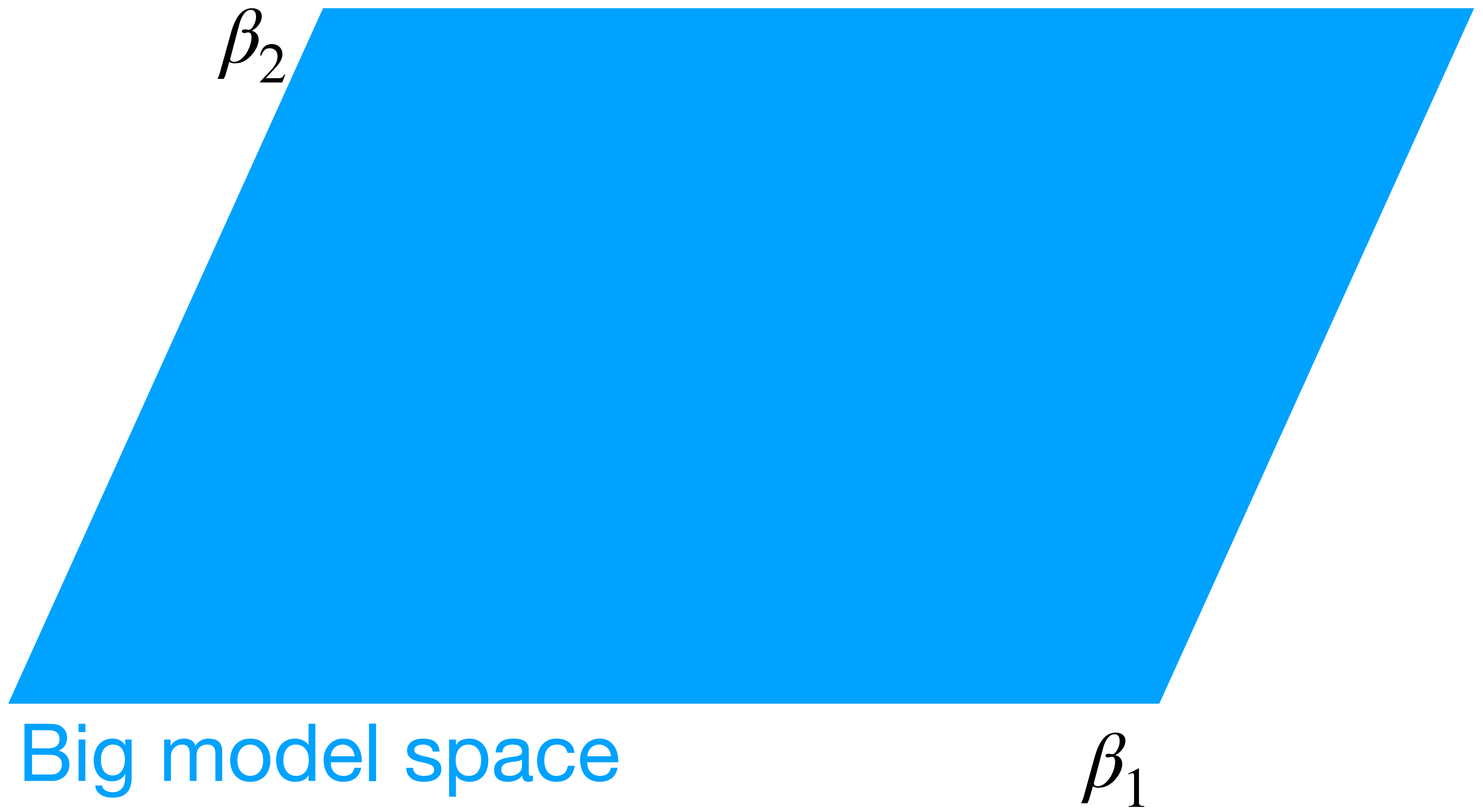
The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

How and when penalization works

Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

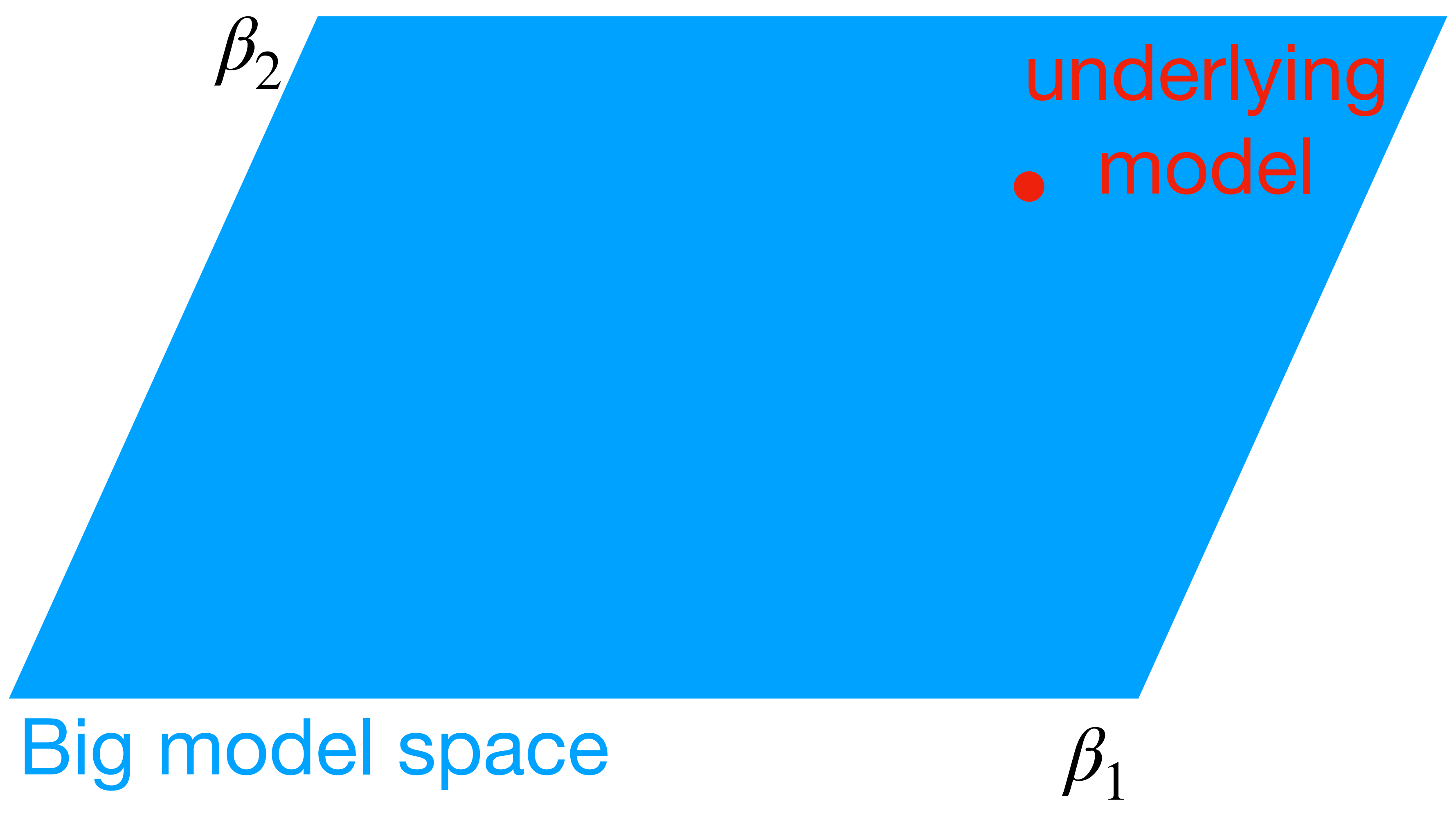


How and when penalization works

Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.



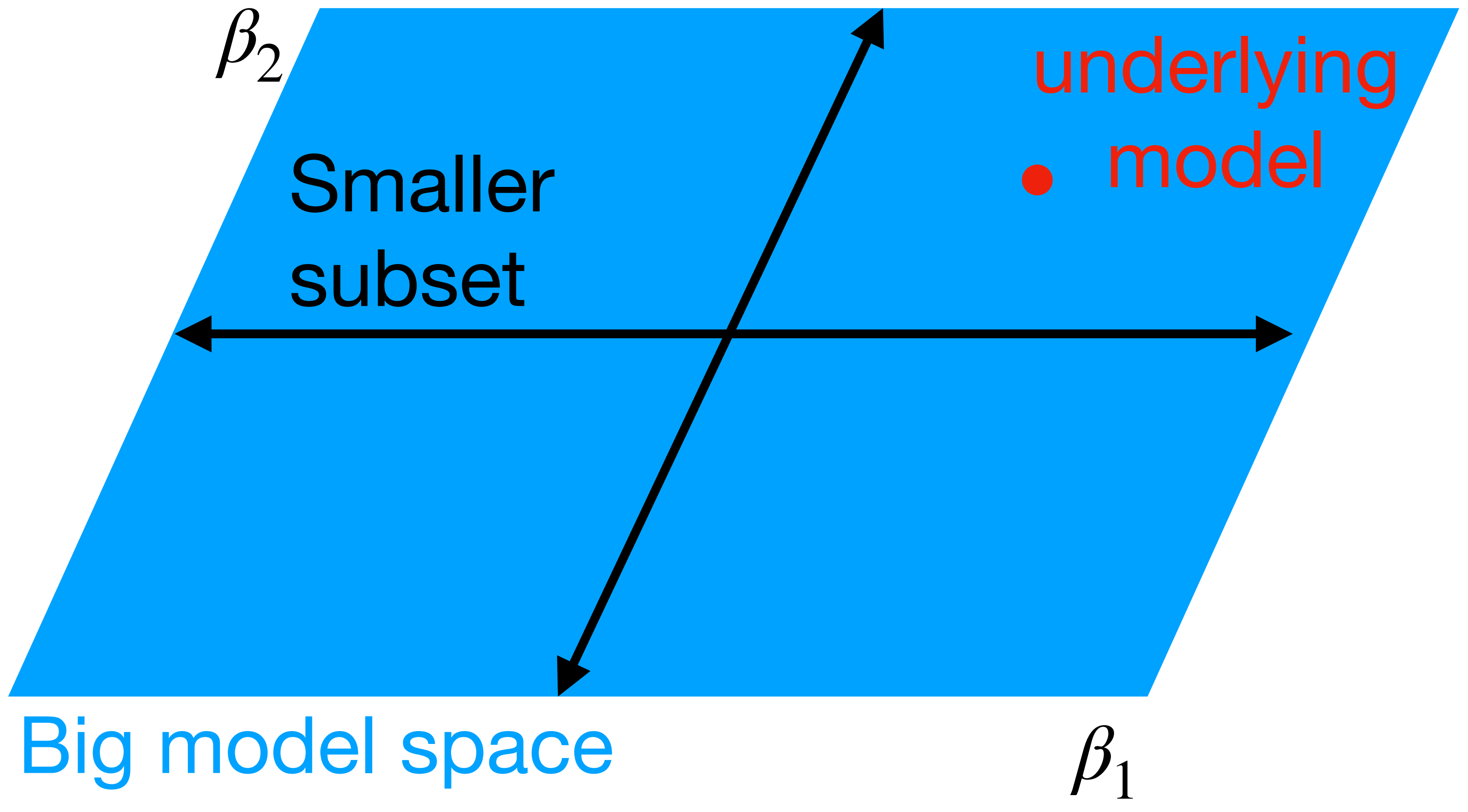
How and when penalization works

Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

Penalization: a bet on the model being close to a smaller subset of the big model space:



How and when penalization works

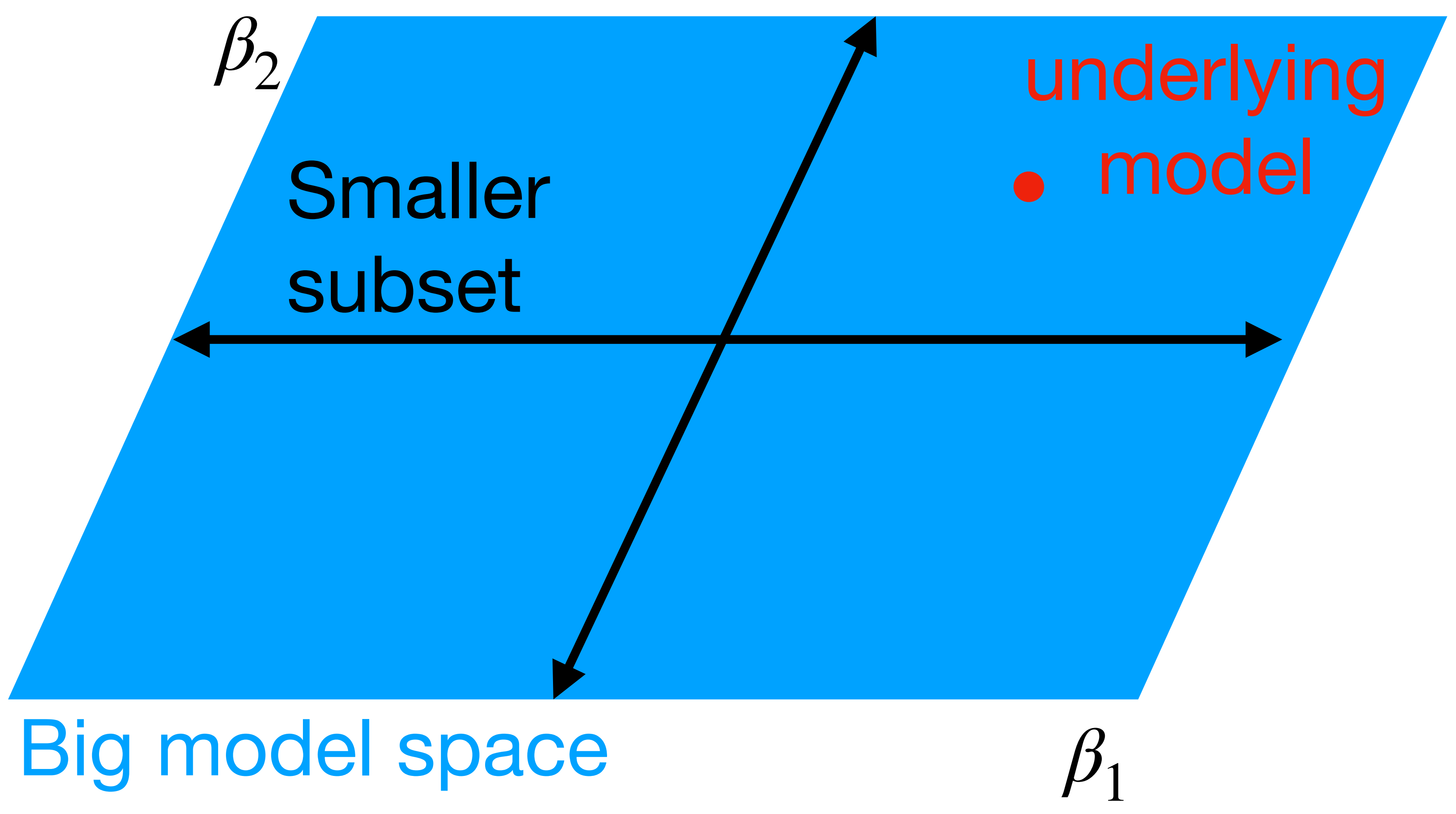
Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

Penalization: a bet on the model being close to a smaller subset of the big model space:

- If so, overall win (the bias is not too big);



How and when penalization works

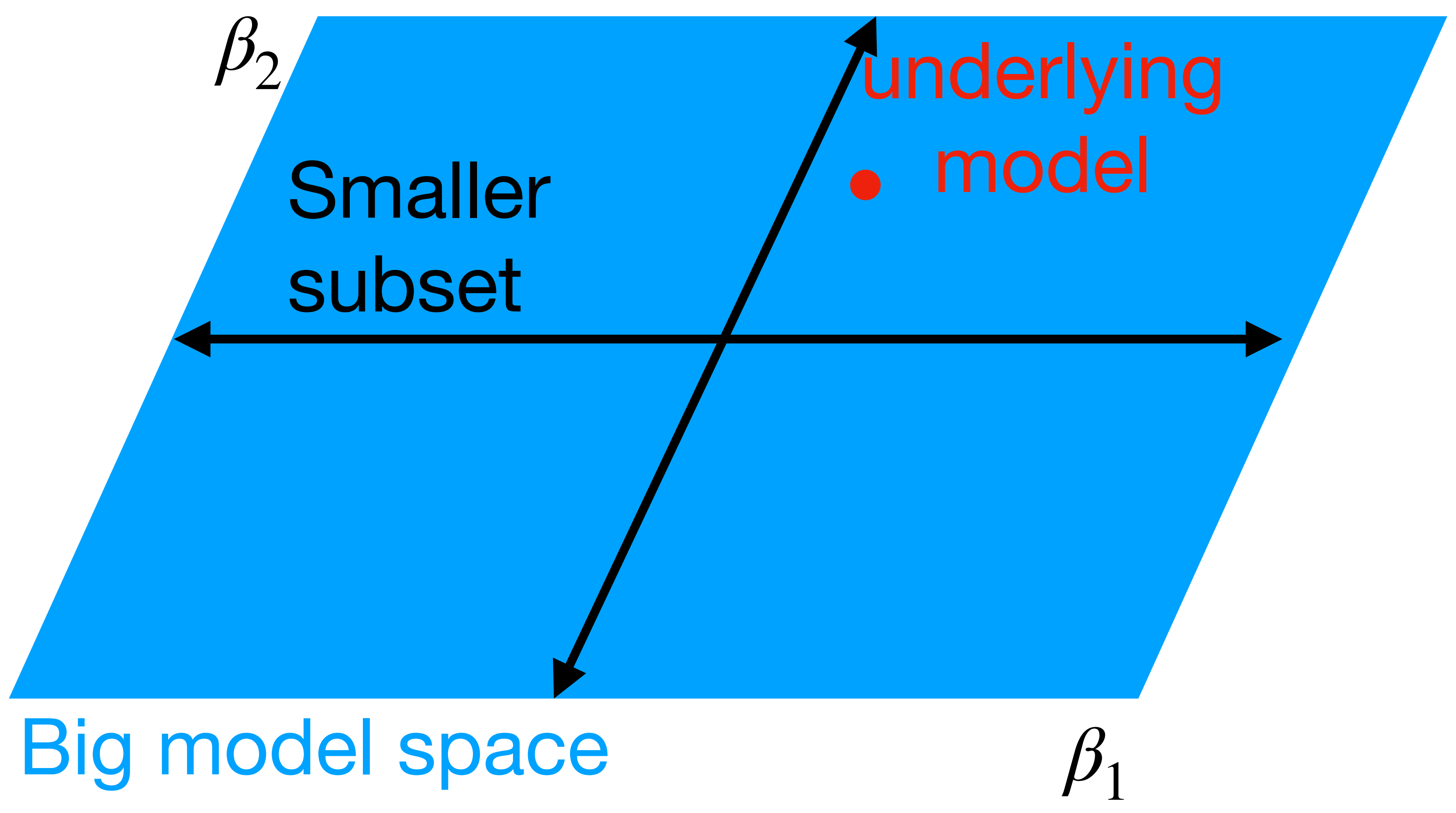
Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

Penalization: a bet on the model being close to a smaller subset of the big model space:

- If so, overall win (the bias is not too big);



How and when penalization works

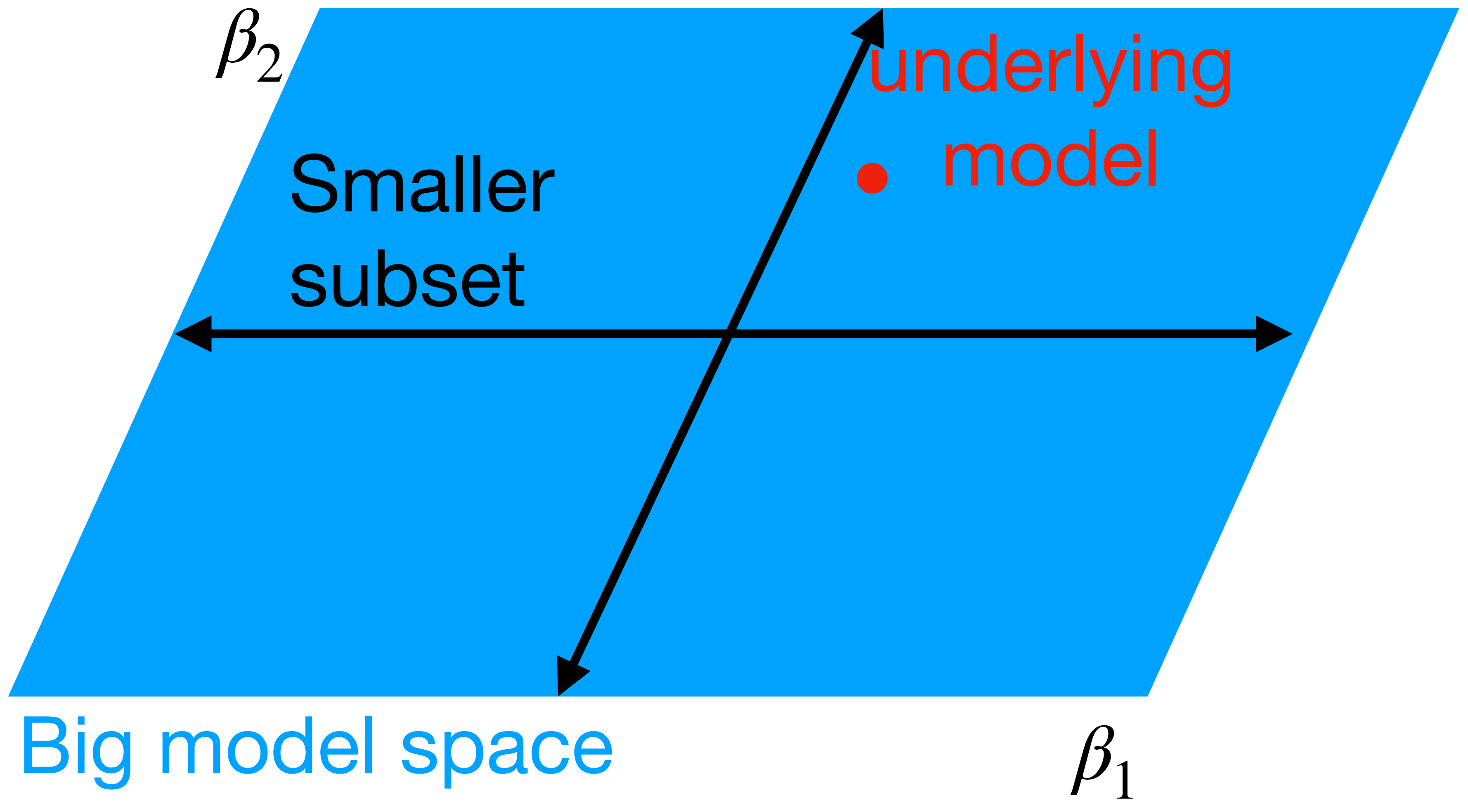
Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

Penalization: a bet on the model being close to a smaller subset of the big model space:

- If so, overall win (the bias is not too big);
- If not, out of luck (the bias is too big).



How and when penalization works

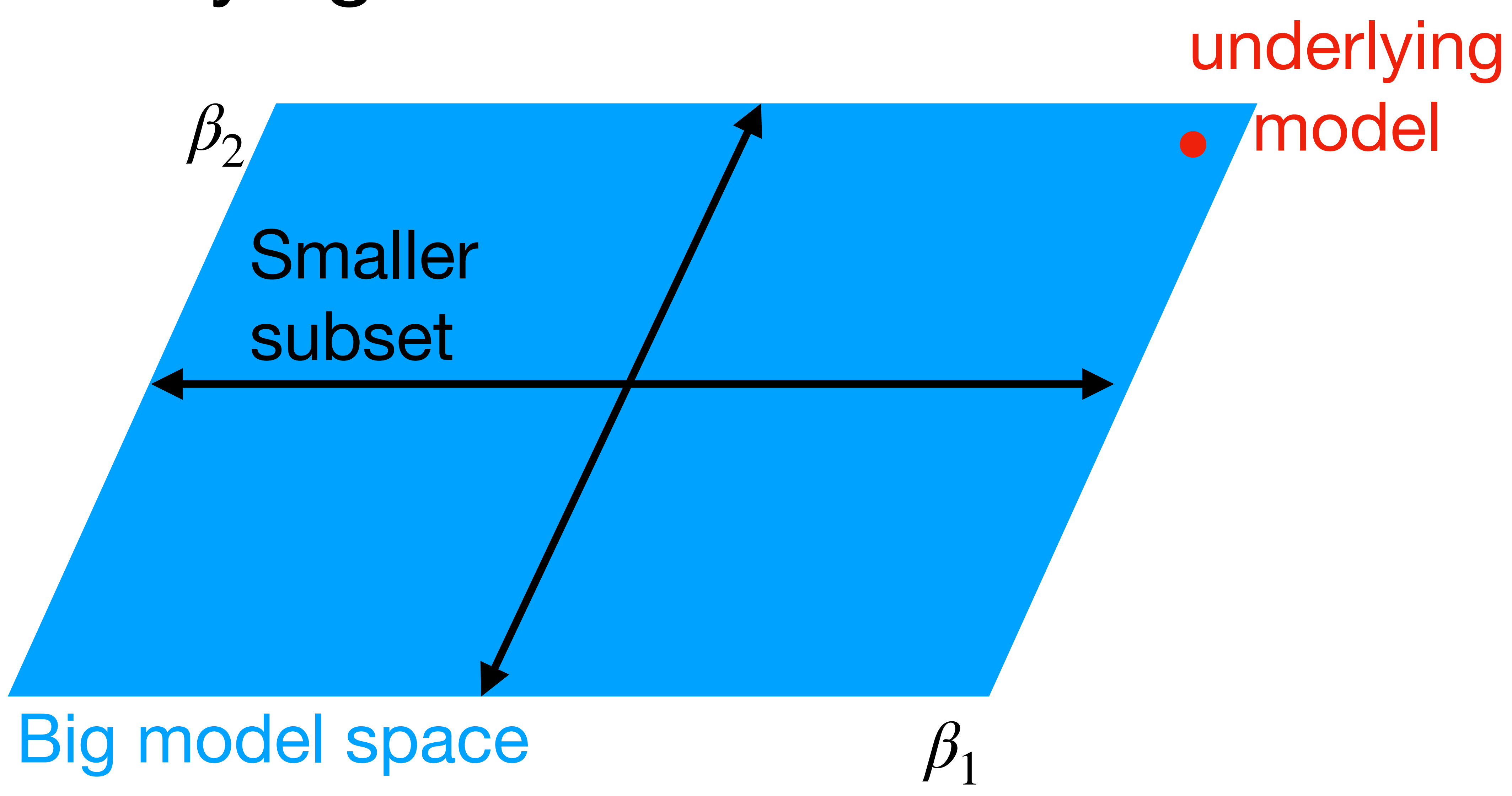
Penalization reduces the variance, but increases the bias of the predictions.

⇒ Reduces test error when reduction in variance outweighs increase in bias.

The bias is a function of the complexity of the underlying model, and in high dimensions, we can have some very complex underlying models.

Penalization: a bet on the model being close to a smaller subset of the big model space:

- If so, overall win (the bias is not too big);
- If not, out of luck (the bias is too big).



Looking ahead to lectures 3 and 4

Lecture 3: Ridge regression (constraining coefficients not to be too large)

Lecture 4: Lasso regression (constraining coefficients to be sparse)

We'll learn about the theory and practice of these penalized regression methods.

Looking ahead to lectures 3 and 4

Lecture 3: Ridge regression (constraining coefficients not to be too large)

Lecture 4: Lasso regression (constraining coefficients to be sparse)

We'll learn about the theory and practice of these penalized regression methods.

Quiz Practice