

# Introduction to Machine Learning (CSCI-UA 473): Fall 2021

## Lecture 3: Linear Models for Regression

**Sumit Chopra**

Courant Institute of Mathematical Sciences  
Department of Radiology - Grossman School of Medicine  
NYU

Slides derived from materials from Benjamin Peherstorfer, Kyunghyun Cho, Andrew Gordon Wilson

# Lecture Outline

Supervised Learning Setting

Linear Parametric Models for Regression

Subset Selection and Coefficient Shrinkage

Comparison of Shrinkage Methods

Bias Variance Tradeoff

# Example Problem: Prostate Cancer

Goal is to predict the value of the log of **Prostate Specific Antigen (Ipsa)**  
(an indicator of presence of prostate cancer) from a number of  
measurements:

*Log cancer volume: lcavol*

*Log prostate weight: lweight*

*Age of patient: age*

*Log of benign prostatic hyperplasia amount: lbph*

*Seminal vesicle invasion: svi*

*Log of capsular penetration: lcp*

*Gleason score: gleason*

*Percent of Gleason scores 4 or 5: pgg45*

*x*

Some of these variables are continuous, some are binary, and some are  
categorical (take only finite number of values)

*Log of Prostate Specific Antigen: Ipsa*

*y*

This is a continuous variable

# Representing Categorical Variables

$x$  can take only a finite number of values

$x \in \{\text{Middle School, High School, Undergraduate, Graduate, Doctoral, Post-doctoral}\}$

## One-Hot Coding

|               |   |   |   |   |   |   |
|---------------|---|---|---|---|---|---|
| Middle School | 1 | 0 | 0 | 0 | 0 | 0 |
| High School   | 0 | 1 | 0 | 0 | 0 | 0 |
| Undergraduate | 0 | 0 | 1 | 0 | 0 | 0 |
| Graduate      | 0 | 0 | 0 | 1 | 0 | 0 |
| Doctoral      | 0 | 0 | 0 | 0 | 1 | 0 |
| Post-doctoral | 0 | 0 | 0 | 0 | 0 | 1 |

# Learning System for Prostate Cancer

Patient measurement (Icavol, Iweight, age etc):  $x$  (the Input)

Log of prostate specific antigen (a real non-negative number):  $y$  (the output)

Unknown target function:  $f : \mathcal{X} \rightarrow \mathcal{Y}$

$\mathcal{X}$ : the entire space of inputs

$\mathcal{Y}$ : the entire space of outputs (all the non-negative real numbers)

Patient data (input-output pairs):  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$  (training data)

The data is such that:  $y_i = f(x_i) : \forall i$

The goal is to use the dataset  $D$  to find a function  $g : \mathcal{X} \rightarrow \mathcal{Y}$  such that:

$g$  approximates the function  $f$

We choose a hypothesis set  $\mathcal{H}$  from where we will choose a function  $g$  that will approximate  $f$  probabilistically

# Learning System for Prostate Cancer

Patient measurement (Icavol, Iweight, age etc):  $x$  (the Input)

Log of prostate specific antigen (a real non-negative number):  $y$  (the output)

Unknown target function:  $f: \mathcal{X} \rightarrow \mathcal{Y}$

$\mathcal{X}$ : the entire space of inputs

$\mathcal{Y}$ : the entire space of outputs (all the non-negative real numbers)

Patient data (input-output pairs):  $D = \{(x^1, y^1), (x^2, y^2), \dots, (x^N, y^N)\}$  (training data)

The data is such that:  $y^i = f(x^i) : \forall i$

The goal is to use the dataset  $D$  to find a function  $g: \mathcal{X} \rightarrow \mathcal{Y}$  such that:

$g$  approximates the function  $f$

We choose a hypothesis set  $\mathcal{H}$  from where we will choose a function  $g$  that will approximate  $f$  probabilistically

All the possible linear parametric functions

# Linear Regression Model

The  $p$  dimensional input vector  $x$  can be explicitly written as  $x = [x_1, x_2, , \dots, x_p]$

The linear regression model takes form

$$g(x) = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

The learnable parameters are  $\beta = (\beta_0, \beta_1, \dots, \beta_p)$

The model assumes that either the unknown function  $f$  is linear or it can be approximated well as a linear function

# Linear Regression Model

The  $p$  dimensional input vector  $x$  can be explicitly written as  $x = [x_1, x_2, \dots, x_p]$

The linear regression model takes form

$$g(x) = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

The learnable parameters are  $\beta = (\beta_0, \beta_1, \dots, \beta_p)$

The model assumes that either the unknown function  $f$  is linear or it can be approximated well as a linear function

Note that while the function  $g$  is a linear function of the parameters  $\beta$  the input variables  $x_i$  could take any form. For instance:

The original inputs:  $x_1, \dots, x_p$

Transformations of inputs:  $\log(x_1), \dots, \log(x_p)$

Basis expansion:  $x_{i+1} = x_1^2; x_{i+2} = x_1^3 \text{ etc}$

Composition of different variables:  $x_{i+1} = x_1 \cdot x_2 \text{ etc}$



# Linear Regression Model

The  $p$  dimensional input vector  $x$  can be explicitly written as  $x = [x_1, x_2, \dots, x_p]$

The linear regression model takes form

$$g(x) = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

The learnable parameters are  $\beta = (\beta_0, \beta_1, \dots, \beta_p)$

Even if the inputs are non-linear the model is still a linear function (in terms of parameters)

Note that while the function  $g$  is a linear function of the parameters  $\beta$  the input variables  $x_i$  could take any form. For instance:

The original inputs:  $x_1, \dots, x_p$

Transformations of inputs:  $\log(x_1), \dots, \log(x_p)$

Basis expansion:  $x_{i+1} = x_1^2; x_{i+2} = x_1^3$  etc

Composition of different variables:  $x_{i+1} = x_1 \cdot x_2$  etc

# Linear Regression Model: Bias

The  $p$  dimensional input vector  $x$  can be explicitly written as  $x = [x_1, x_2, \dots, x_p]$

The linear regression model takes form

$$g(x) = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

The learnable parameters are  $\beta = (\beta_0, \beta_1, \dots, \beta_p)$

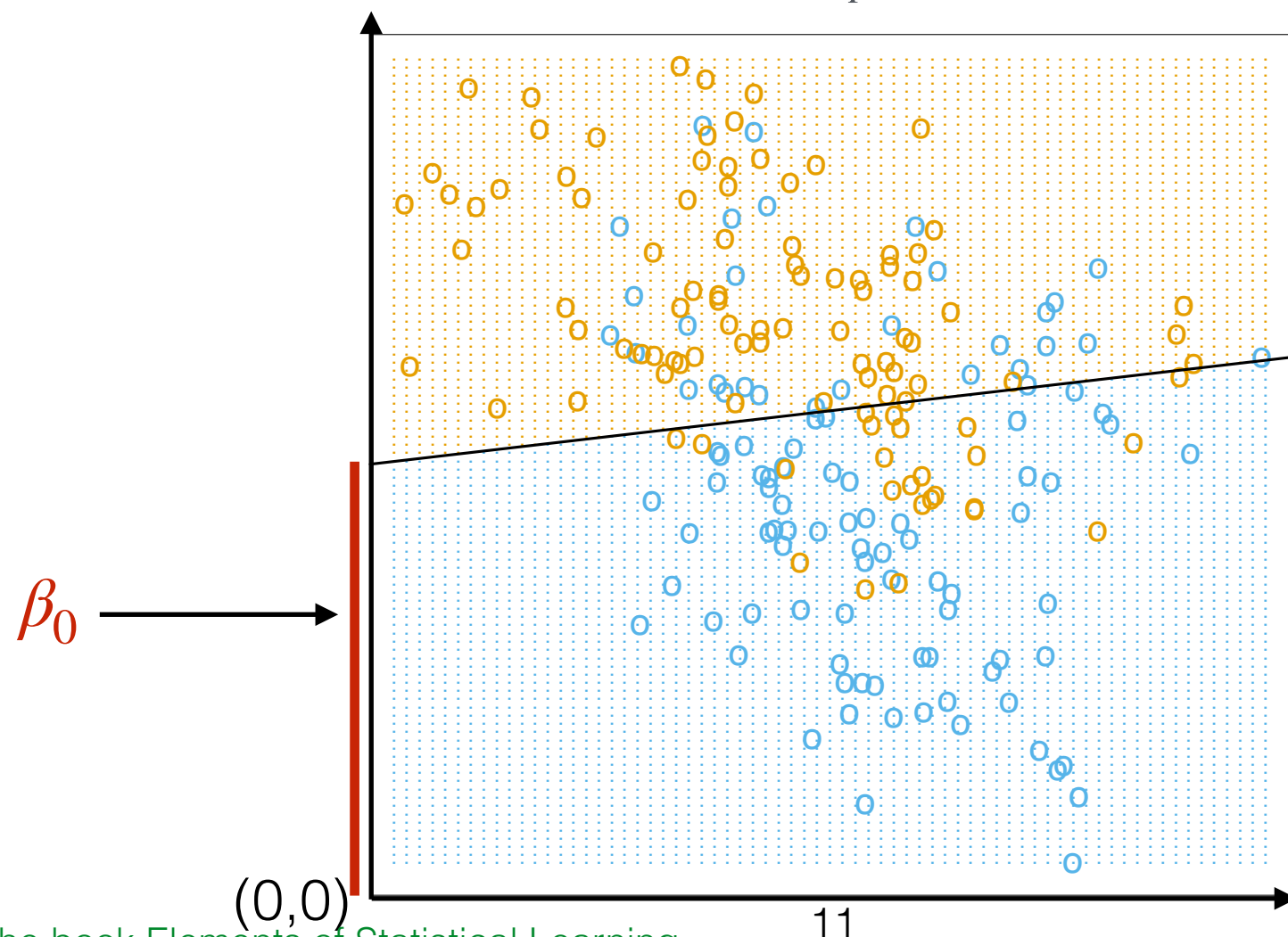
# Linear Regression Model: Bias

The  $p$  dimensional input vector  $x$  can be explicitly written as  $x = [x_1, x_2, \dots, x_p]$

The linear regression model takes form

$$g(x) = \beta_0 + \sum_{i=1}^p \beta_i x_i$$

The learnable parameters are  $\beta = (\beta_0, \beta_1, \dots, \beta_p)$



# Linear Regression Model: Training

We have a set of examples (training data):  $D = \{(x^1, y^1), (x^2, y^2), \dots, (x^N, y^N)\}$

Where  $x^i = [x_1^i, x_2^i, \dots, x_p^i]$

In order to learn we define an error metric (a loss function) which measures the discrepancy between the model prediction and the actual value on the training set

# Linear Regression Model: Training

We have a set of examples (training data):  $D = \{(x^1, y^1), (x^2, y^2), \dots, (x^N, y^N)\}$

Where  $x^i = [x_1^i, x_2^i, \dots, x_p^i]$

In order to learn we define an error metric (a loss function) which measures the discrepancy between the model prediction and the actual value on the training set

A reasonable metric is the squared error between the two quantities

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N (y^i - g_{\beta}(x^i))^2$$

$$= \sum_{i=1}^N \left( y^i - \left( \beta_0 + \sum_{j=1}^p \beta_j x_j^i \right) \right)^2$$

$$= \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$

Residual Sum of Squares

We've made it explicit that  $g$  is a parametric function with parameters  $\beta$

# Linear Regression Model: Training

We have a set of examples (training data):  $D = \{(x^1, y^1), (x^2, y^2), \dots, (x^N, y^N)\}$

Where  $x^i = [x_1^i, x_2^i, \dots, x_p^i]$

In order to learn we define an error metric (a loss function) which measures the discrepancy between the model prediction and the actual value on the training set

A reasonable metric is the squared error between the two quantities

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N (y^i - g_{\beta}(x^i))^2$$

Residual Sum of Squares

$$= \sum_{i=1}^N \left( y^i - \left( \beta_0 + \sum_{j=1}^p \beta_j x_j^i \right) \right)^2$$

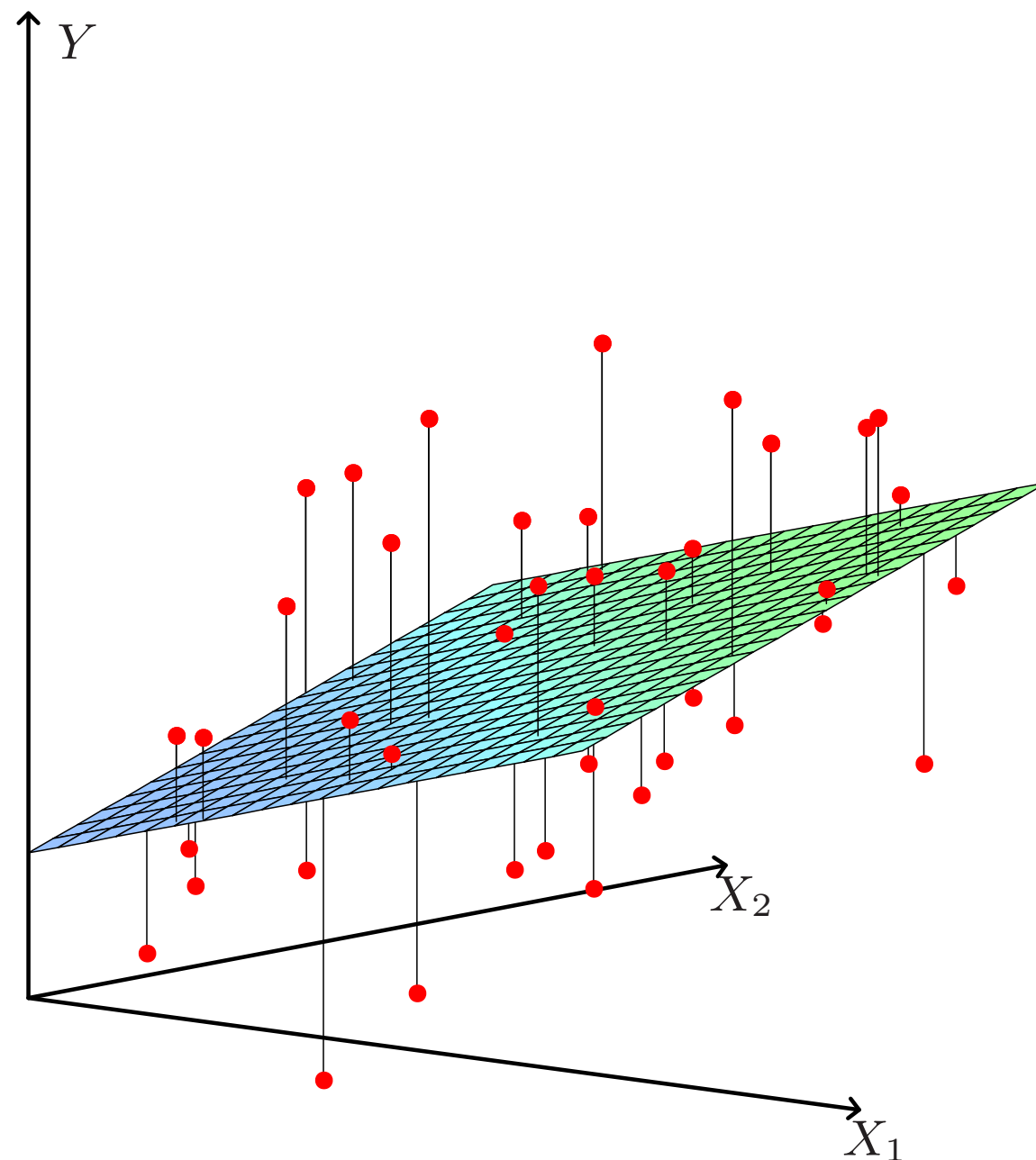
$$= \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$

We've made it explicit that  $g$  is a parametric function with parameters  $\beta$

Learning: adjust the parameters  $\beta = [\beta_0, \beta_1, \dots, \beta_p]$  in a way that it minimizes the *loss*

# Linear Regression Model: Training

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$



# Linear Regression Model: Training

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$

Simplify the equation by writing it in matrix form and augmenting the vector of inputs  $x$  by 1 and subsuming the intercept/bias  $\beta_0$  into the parameter vector  $\beta$

$$\beta^T = [\beta_0, \beta_1, \dots, \beta_p]$$

$$X = [1, x_1, x_2, \dots, x_p]$$

$$y = g_{\beta}(X) = X \cdot \beta$$



# Linear Regression Model: Training

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$

Simplify the equation by writing it in matrix form and augmenting the vector of inputs  $x$  by 1 and subsuming the intercept/bias  $\beta_0$  into the parameter vector  $\beta$

$$\beta^T = [\beta_0, \beta_1, \dots, \beta_p]$$

$$X = [1, x_1, x_2, \dots, x_p]$$

$$y = g_{\beta}(X) = X \cdot \beta$$

Denote by  $\mathbf{X}$  the  $N \times (p + 1)$  matrix with each row as one input sample

$$Loss(\beta) = RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

# Linear Regression Model: Training

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$

Simplify the equation by writing it in matrix form and augmenting the vector of inputs  $x$  by 1 and subsuming the intercept/bias  $\beta_0$  into the parameter vector  $\beta$

$$\beta^T = [\beta_0, \beta_1, \dots, \beta_p]$$

$$X = [1, x_1, x_2, \dots, x_p]$$

$$y = g_{\beta}(X) = X \cdot \beta$$

Denote by  $\mathbf{X}$  the  $N \times (p + 1)$  matrix with each row as one input sample

$$Loss(\beta) = RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

Compute the  
derivatives of  
 $RSS(\beta)$  and set it to  
zero

$$\frac{\delta RSS}{\delta \beta} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\beta) = 0$$

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

Closed form solution!

# Linear Regression Model: Prediction

$$Loss(\beta) = RSS(\beta) = \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p \beta_j x_j^i \right)^2$$

Simplify the equation by writing it in matrix form and augmenting the vector of inputs  $x$  by 1 and subsuming the intercept/bias  $\beta_0$  into the parameter vector  $\beta$

$$\beta^T = [\beta_0, \beta_1, \dots, \beta_p]$$

$$X = [1, x_1, x_2, \dots, x_p]$$

$$y = g_{\beta}(X) = X \cdot \beta$$

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

Now for a new observation  $x_0$  its prediction can be computed as

$$\begin{aligned} \hat{y}_0 &= [1 \ x_0] \cdot \hat{\beta} \\ &= [1 \ x_0] \cdot (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \end{aligned}$$

# What is the Model Doing?

In addition to checking the accuracy of model predictions it is equally important to understand which variables are most informative

Define  $\sigma^2$  as the variance of the observations  $y^i$  which are assumed to be uncorrelated and have constant variance

We can estimate  $\sigma^2$  using the training examples

$$\hat{\sigma}^2 = \frac{1}{N - P - 1} \sum_{i=1}^N (y^i - \hat{y}^i)^2$$

For any variable  $\beta_j$  define the Z-score as

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma} \sqrt{v_j}}$$

$v_j$  is the  $j^{th}$  diagonal element of  $(X^T X)^{-1}$

Informally, smaller the value of Z-score the less important the variable  $j$

# What is the Model Doing?

Many a times one needs to assess the significance of a group of coefficients simultaneously

We can do that by defining the F-statistics

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N - p_1 - 1)}$$

$RSS_1$  is the residual sum of squares of the bigger model with  $p_1 + 1$  parameters

$RSS_0$  is the residual sum of squares for the nested smaller model  $p_0 + 1$

The F-statistics measures the change in residual

# Shrinkage Methods (Regularization)

Linear models as-is, particularly those with a large number of variables (and their compositions), trained using least squares estimates have two flaws

1. They have the ability to tune well on the provided snapshot of the training data but may not have a low prediction error. **This is the Bias-Variance tradeoff which we will talk about in detail later**
2. A model with large number of variables is less suited for model interpretation. Typically we would like to know which (handful) variables have the most predictive power

# Shrinkage Methods (Regularization)

Linear models as-is, particularly those with a large number of variables (and their compositions), trained using least squares estimates have two flaws

1. They have the ability to tune well on the provided snapshot of the training data but may not have a low prediction error. **This is the Bias-Variance tradeoff which we will talk about in detail later**
2. A model with large number of variables is less suited for model interpretation. Typically we would like to know which (handful) variables have the most predictive power

Accomplished by augmenting the training of the models in a way so that minimizing the loss function leads to “shrinking”/“deleting” the effect of unimportant variables

One way to reduce the capacity of the hypothesis space and to ensure better prediction error on unseen


**This process is also called Regularization and is used to prevent Overfitting**

# Ridge Regression

Impose a penalty on the size of the coefficients being learned

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p x_j^i \cdot \beta_j \right)^2$$

subject to  $\sum_{j=1}^p \beta_j^2 \leq s$



Ensures that sum  
of magnitude of  
all coefficients is  
within limits

Also called  $L_2$  penalty

Or

Which can be re-written as

Weight Decay

$$\hat{\beta}^{ridge} = \underset{\beta}{\operatorname{argmin}} \left[ \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p x_j^i \cdot \beta_j \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right]$$

When there are correlated variables their coefficients can become poorly determined

Bounding the magnitudes of the coefficients can prevent this phenomena



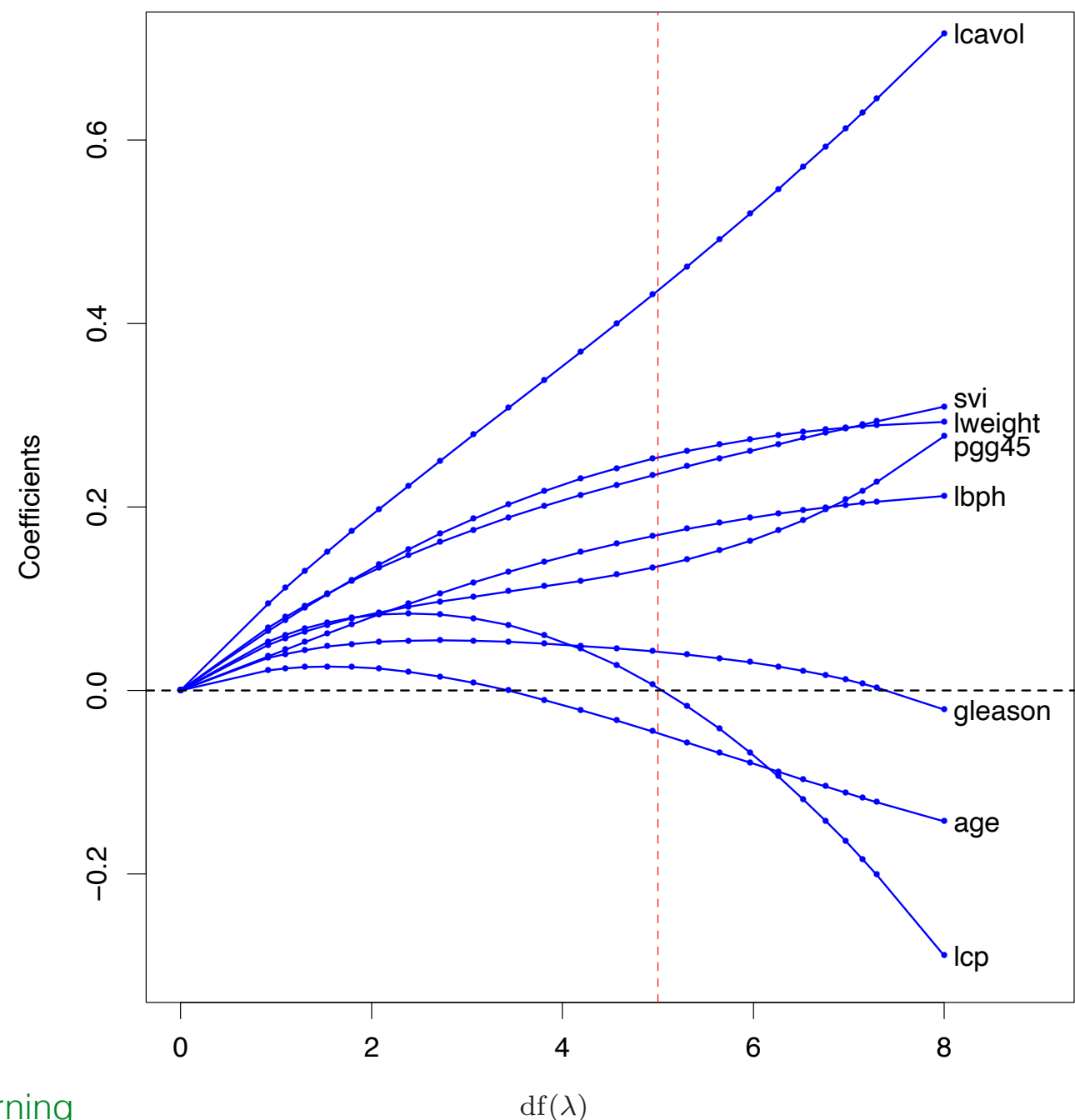
# Ridge Regression

In matrix notation the loss function can be written as

$$RSS(\lambda) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

And the ridge regression solution can easily be computed in closed form

$$\hat{\boldsymbol{\beta}}^{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$



# Lasso

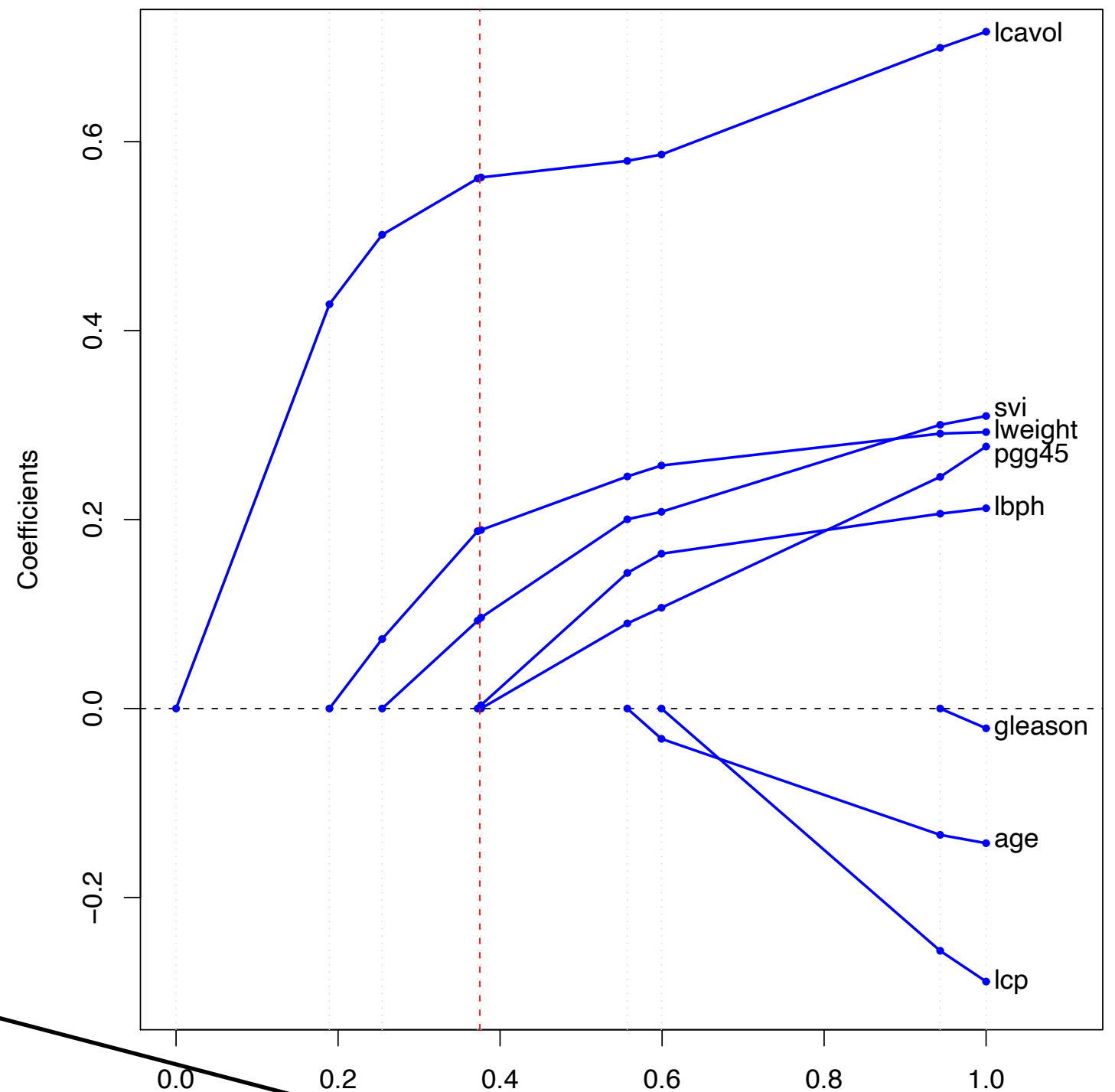
Similar to the Ridge Regression but with a small difference

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^N \left( y^i - \beta_0 - \sum_{j=1}^p x_j^i \cdot \beta_j \right)^2$$

$$\text{subject to } \sum_{j=1}^p |\beta_j| \leq t$$

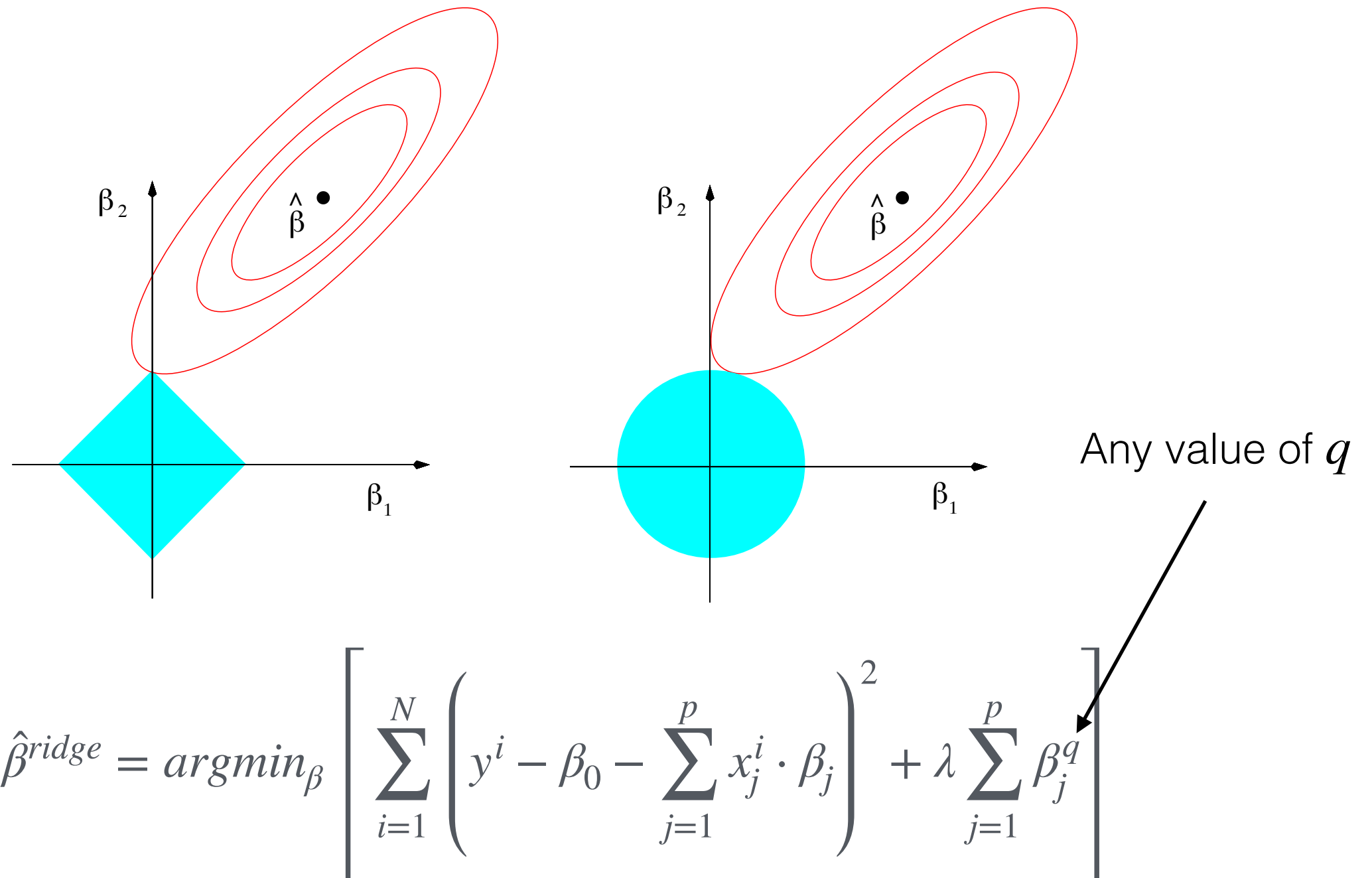
Acts like a sparsity  
penalty

$$s = \frac{t}{\sum_{j=1}^p |\beta_j|}$$



# Ridge vs Lasso

When to choose which type of regularization?



# Bias and Variance

Regularization provides you with a knob which you can adjust to control the complexity of hypothesis space

Use this knob to strike a balance between:

Lowering the in-sample error (training)

Bounding the difference between in-sample and out-of-sample error, thereby lowering the out-of-sample error (generalization)

But which hypothesis space to start with? Can we say anything about the space even before we start learning?

VC Dimension is one way of measuring the complexity of the hypothesis space and modeling the tradeoff between training vs generalization

Bias-Variance is another way

# Bias and Variance

## Recap

$E_{out}$ : out-of-sample error (unknown)

$E_{in}$ : in-sample error (training error - known)

Complex  $\mathcal{H}$ : better chance of approximating  $f$  in-sample (training error)

Simple  $\mathcal{H}$ : better chance of generalizing out-of-sample (generalization error)

Goal of Machine Learning is to lower  $E_{out}$  as much as possible given  $D$

Bias-Variance analysis allows us to express  $E_{out}$  in a way that provides some cues to what hypothesis space to start with. It tells you

How well  $\mathcal{H}$  can approximate  $f$

How well we can zoom in on a good hypothesis  $h \in \mathcal{H}$

Applies only to real-valued targets and uses squared error metric

# Bias and Variance

We've made  
explicit the  
dependence  
of  $g$  on  $D$



$$E_{out}(g^D) = \mathbb{E}_x [(g^D(x) - f(x))^2]$$

$\mathbb{E}_x$  denotes the expectation over  $x$  based on the probability distribution on  $\mathcal{X}$

# Bias and Variance

We've made  
explicit the  
dependence  
of  $g$  on  $D$


$$E_{out}(g^D) = \mathbb{E}_x [(g^D(x) - f(x))^2]$$

$\mathbb{E}_x$  denotes the expectation over  $x$  based on the probability distribution on  $\mathcal{X}$

$$\mathbb{E}_D[E_{out}(g^D)] = \mathbb{E}_D \left[ \mathbb{E}_x \left[ (g^D(x) - f(x))^2 \right] \right]$$

# Bias and Variance

We've made  
explicit the  
dependence  
of  $g$  on  $D$


$$E_{out}(g^D) = \mathbb{E}_x [(g^D(x) - f(x))^2]$$

$\mathbb{E}_x$  denotes the expectation over  $x$  based on the probability distribution on  $\mathcal{X}$

$$\begin{aligned}\mathbb{E}_D[E_{out}(g^D)] &= \mathbb{E}_D \left[ \mathbb{E}_x \left[ (g^D(x) - f(x))^2 \right] \right] \\ &= \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x) - f(x))^2 \right] \right]\end{aligned}$$



# Bias and Variance

We've made  
explicit the  
dependence  
of  $g$  on  $D$


$$E_{out}(g^D) = \mathbb{E}_x [(g^D(x) - f(x))^2]$$

$\mathbb{E}_x$  denotes the expectation over  $x$  based on the probability distribution on  $\mathcal{X}$

$$\begin{aligned}\mathbb{E}_D[E_{out}(g^D)] &= \mathbb{E}_D \left[ \mathbb{E}_x \left[ (g^D(x) - f(x))^2 \right] \right] \\ &= \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x) - f(x))^2 \right] \right] \\ &= \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x))^2 \right] - 2 \mathbb{E}_D [g^D(x)] f(x) + f(x)^2 \right]\end{aligned}$$



Average hypothesis where the average is taken over multiple draws of  $D$

# Bias and Variance

$$\mathbb{E}_D[E_{out}(g^D)] = \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x))^2 \right] - 2\mathbb{E}_D [g^D(x)] f(x) + f(x)^2 \right]$$

Assume you generate many training datasets:  $\{D_1, D_2, \dots, D_K\}$

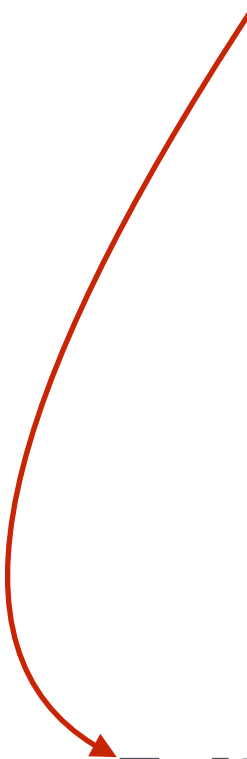
$$\mathbb{E}_D [g^D(x)] \approx \frac{1}{K} \sum_{i=1}^K g^{D_i}(x) = \bar{g}(x)$$

# Bias and Variance

$$\mathbb{E}_D[E_{out}(g^D)] = \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x))^2 \right] - 2\mathbb{E}_D [g^D(x)] f(x) + f(x)^2 \right]$$

Assume you generate many training datasets:  $\{D_1, D_2, \dots, D_K\}$

$$\mathbb{E}_D [g^D(x)] \approx \frac{1}{K} \sum_{i=1}^K g^{D_i}(x) = \bar{g}(x)$$


$$\mathbb{E}_D[E_{out}(g^D)] = \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x))^2 \right] - 2\bar{g}(x)f(x) + f(x)^2 \right]$$

$$= \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x))^2 \right] - \bar{g}(x)^2 + \bar{g}(x)^2 - 2\bar{g}(x)f(x) + f(x)^2 \right]$$

$$= \mathbb{E}_x \left[ \mathbb{E}_D \left[ (g^D(x))^2 - \bar{g}(x)^2 \right] + [\bar{g}(x) - f(x)]^2 \right]$$

Variance

Bias

# Bias and Variance

$$\mathbb{E}_D[E_{out}(g^D)] = \mathbb{E}_x \left[ \underbrace{\mathbb{E}_D \left[ (g^D(x))^2 - \bar{g}(x)^2 \right]}_{\text{Variance}} + \underbrace{[\bar{g}(x) - f(x)]^2}_{\text{Bias}} \right]$$

Variance

Bias

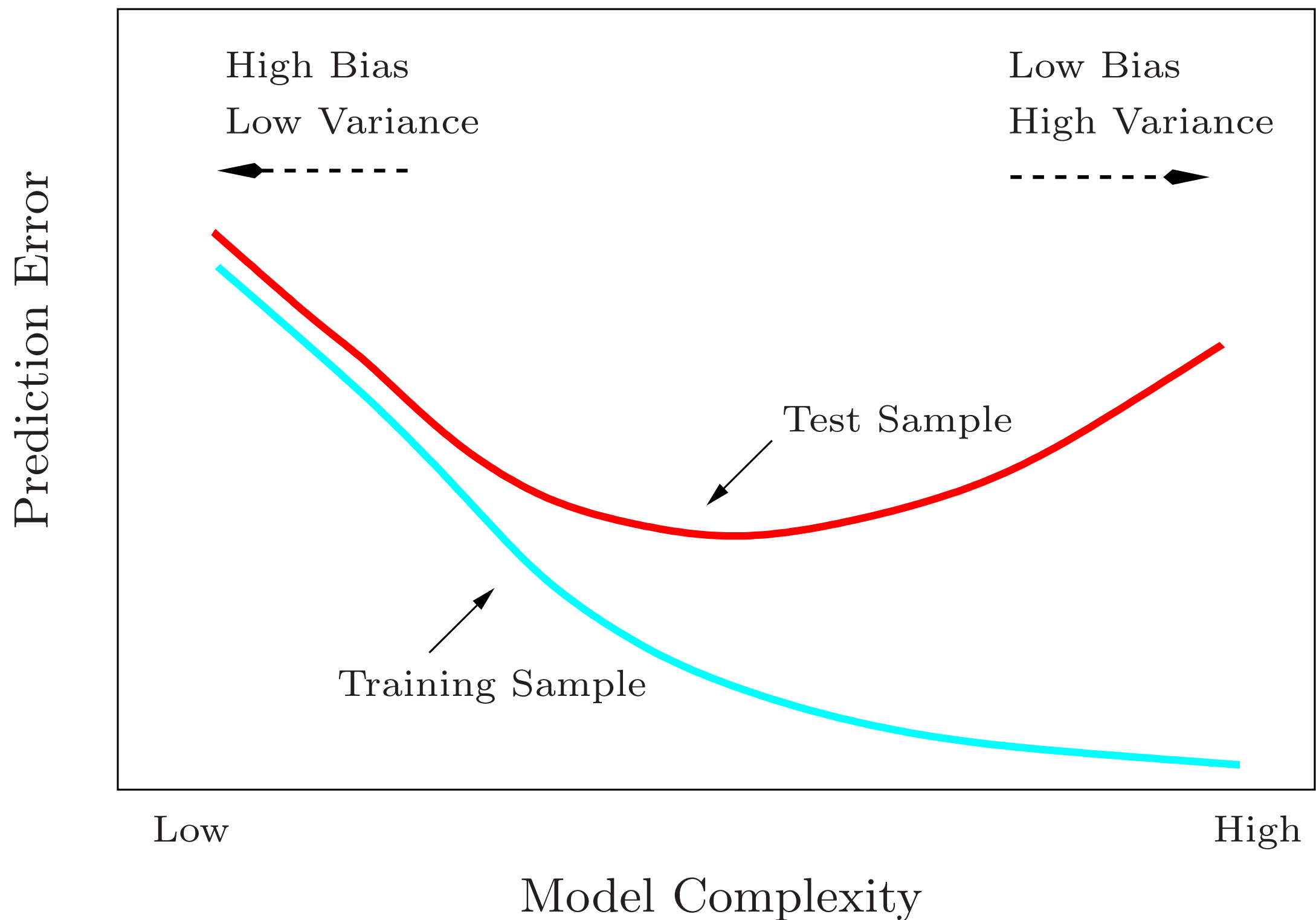
Measures the variation of the final hypothesis depending on the dataset  $D$  used for training

By how much the average function that we will learn using different instances of the dataset will deviate from the true function

This in turn is a measure of how much our learning model is biased away from the true function

$$\begin{aligned} \mathbb{E}_D[E_{out}(g^D)] &= \mathbb{E}_x [bias(x) + var(x)] \\ &= \mathbf{bias + var} \end{aligned}$$

# Bias and Variance



# Example

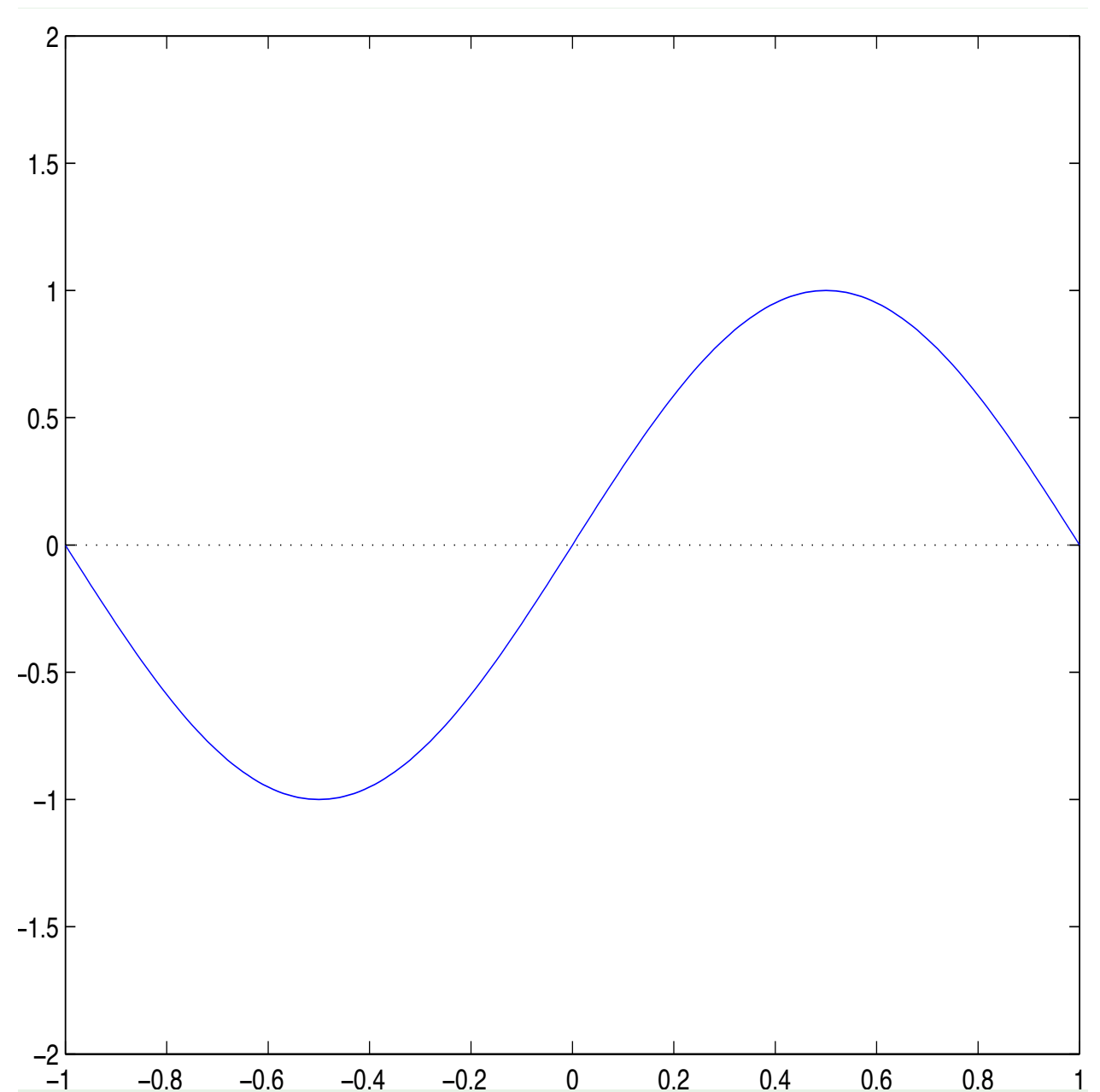
$$f: [-1, +1] \rightarrow \mathfrak{R} \quad f = \text{Sin}(\pi x)$$

As part of your training set you are only given two (2) examples

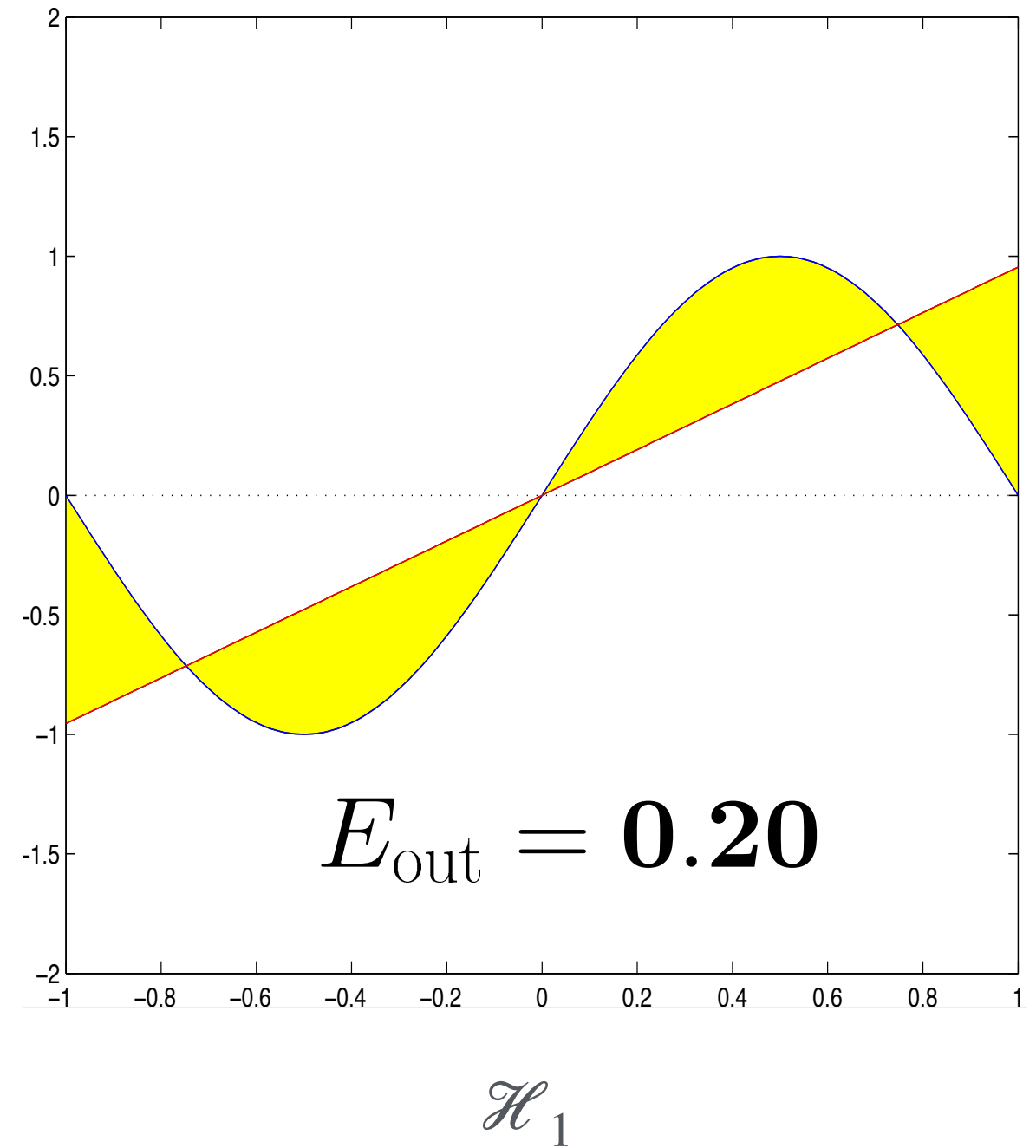
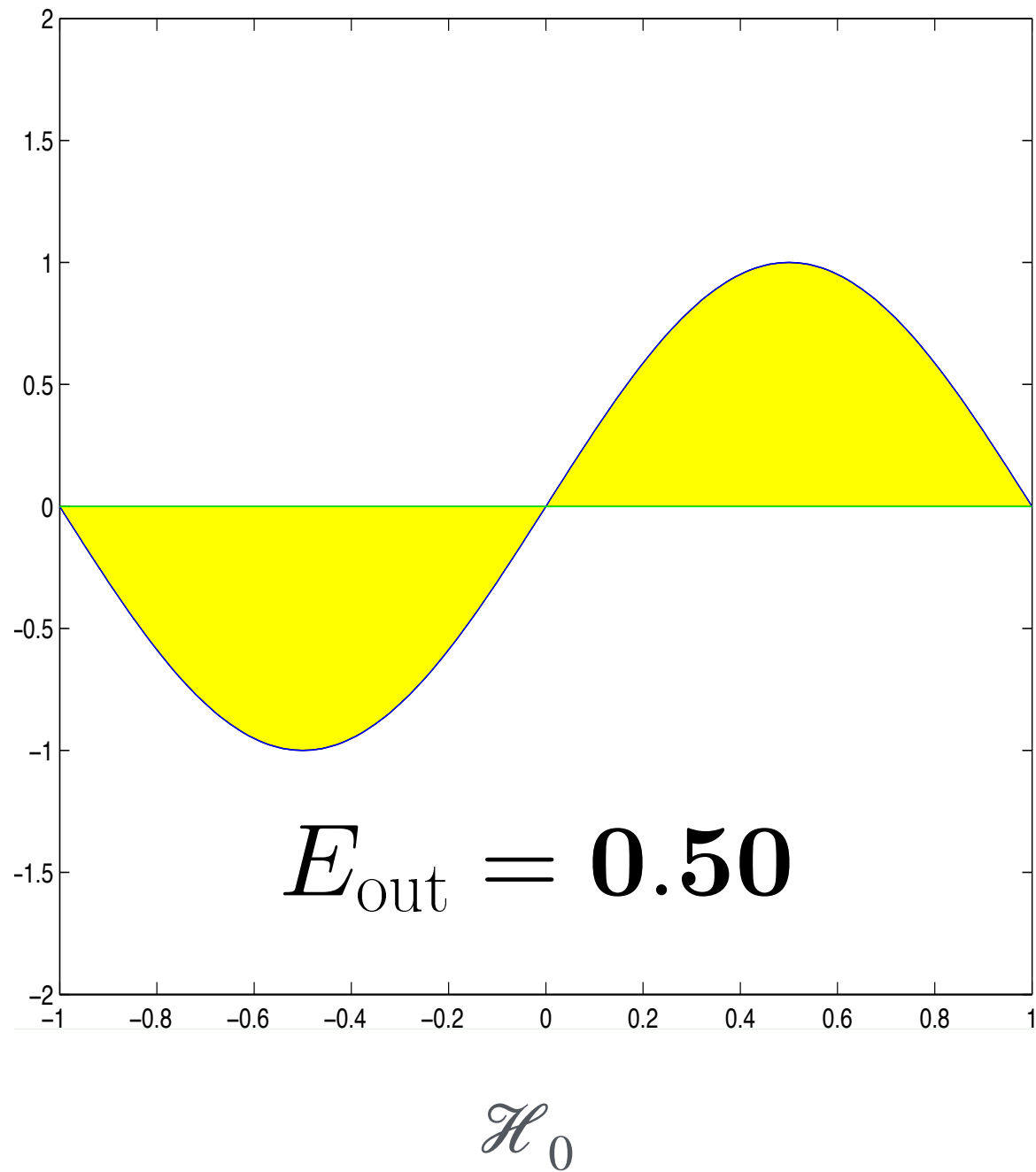
You want to compare the following two hypothesis and pick the one best one

$$\mathcal{H}_0 : \quad h(x) = b$$

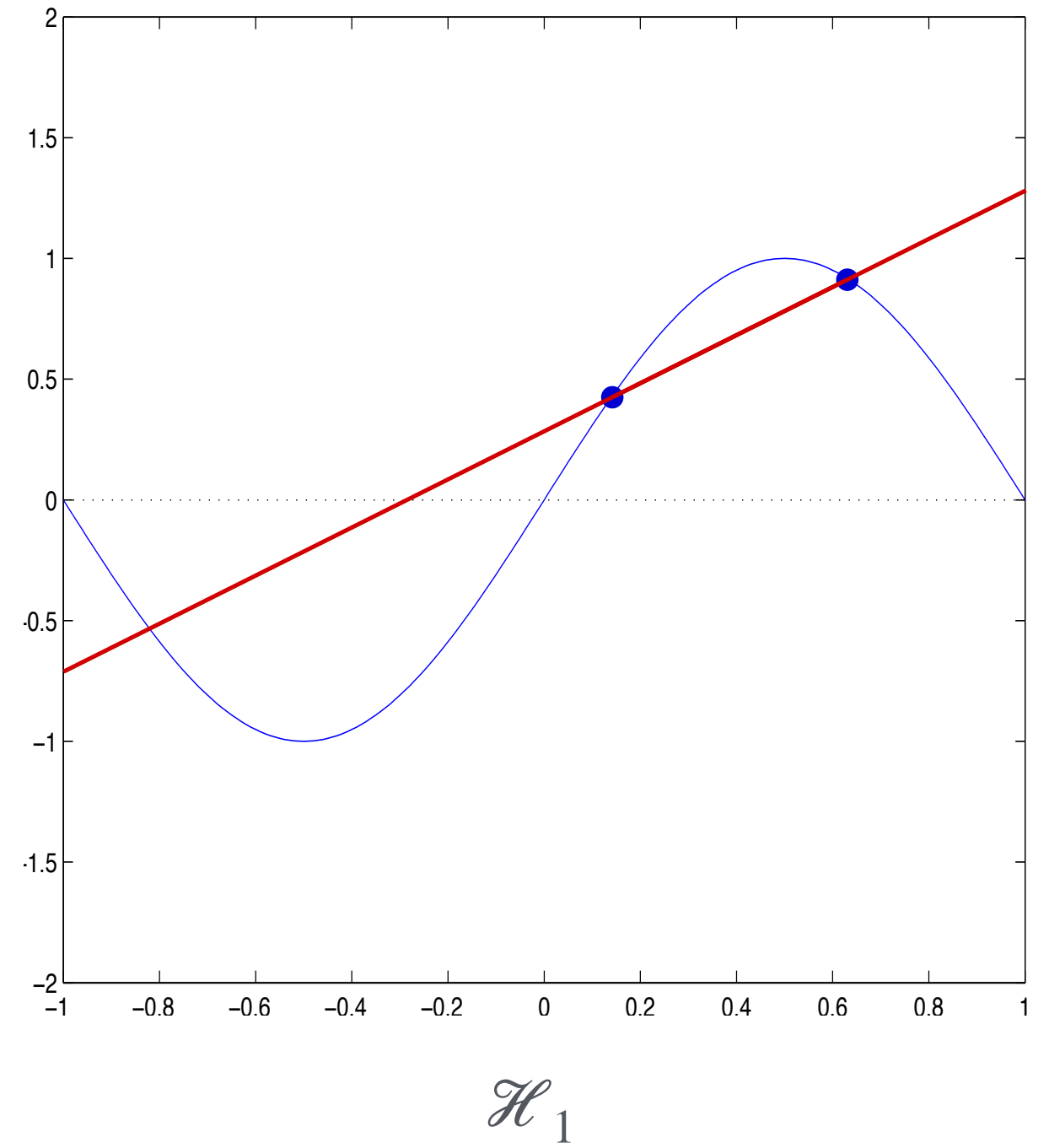
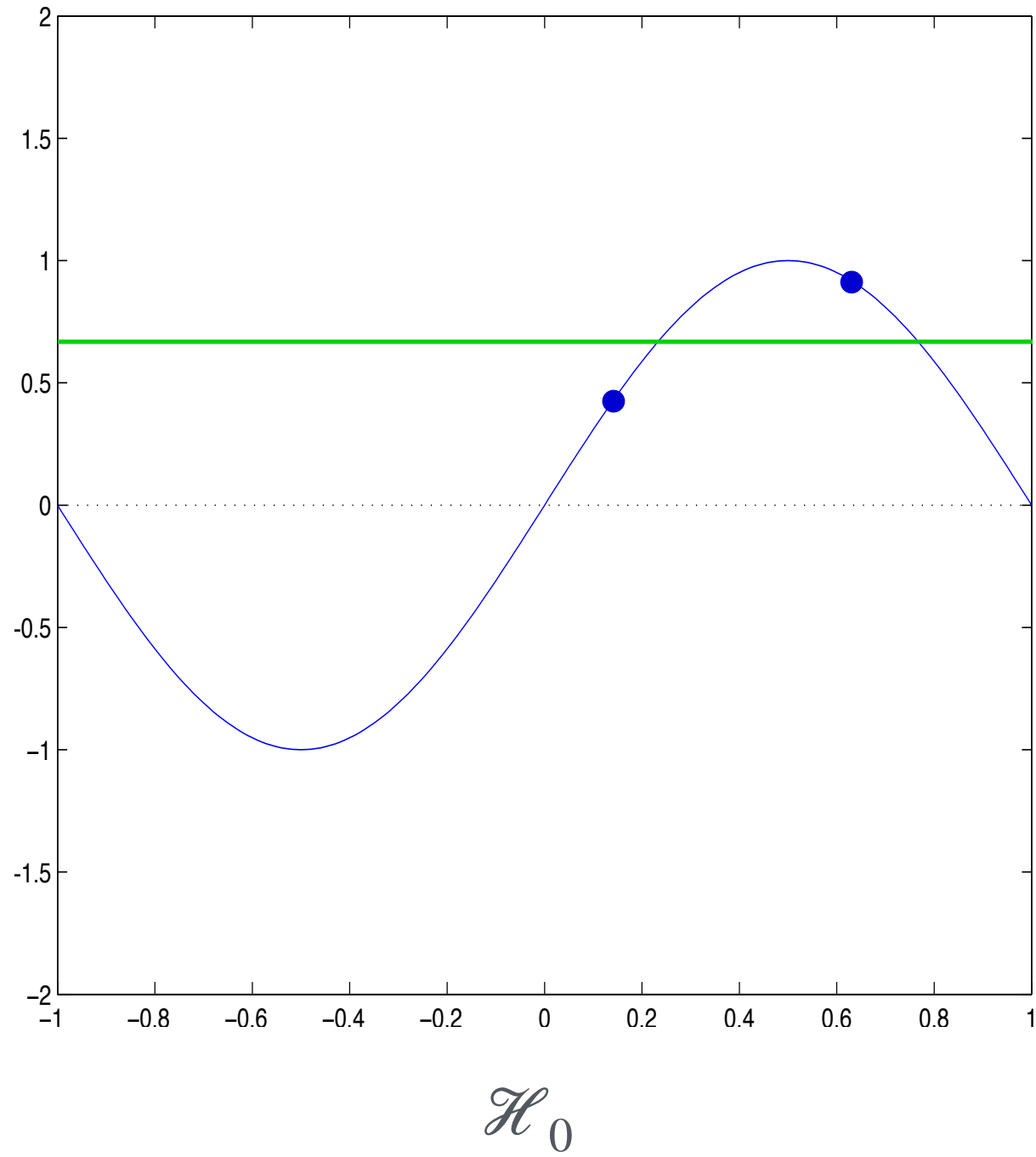
$$\mathcal{H}_1 : \quad h(x) = ax + b$$



# Example



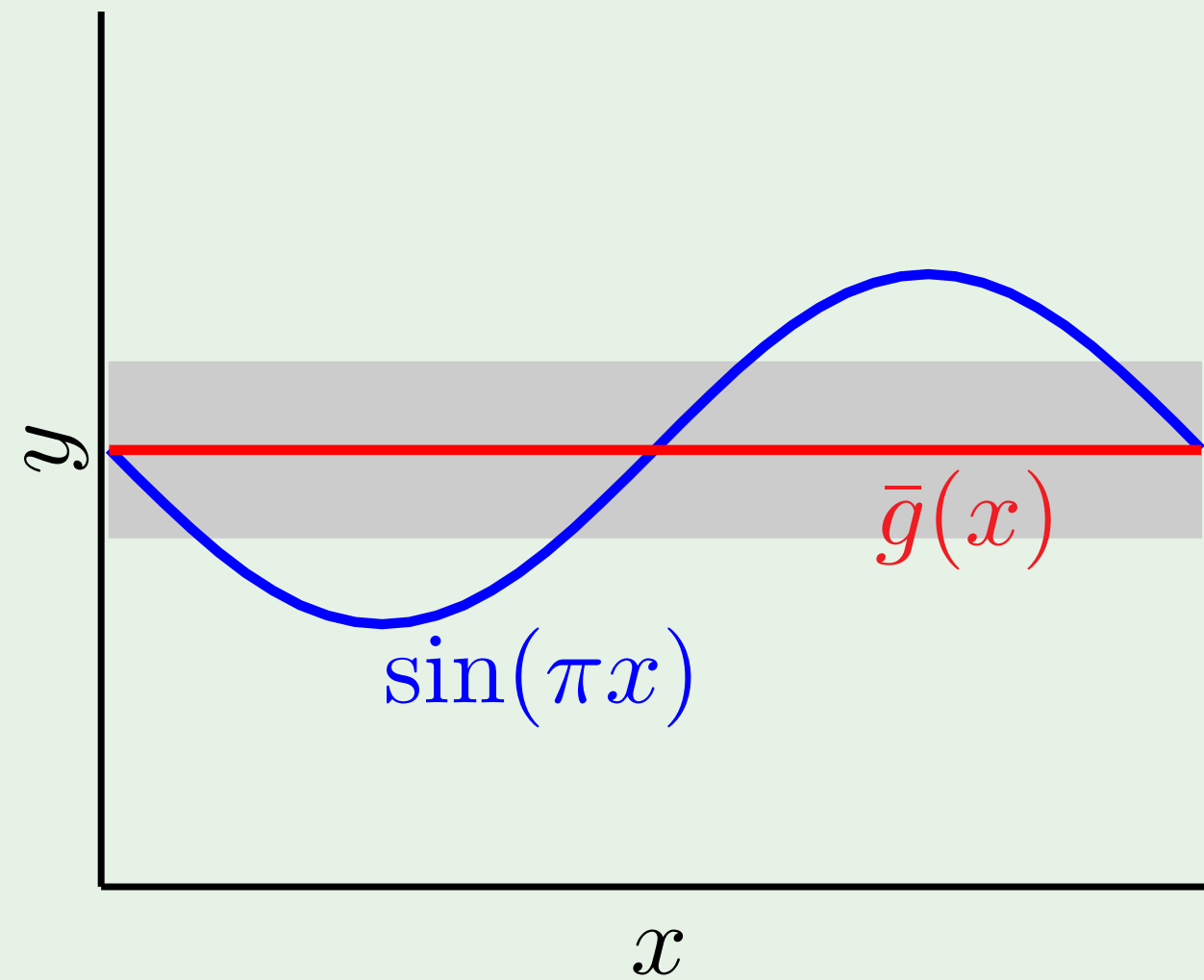
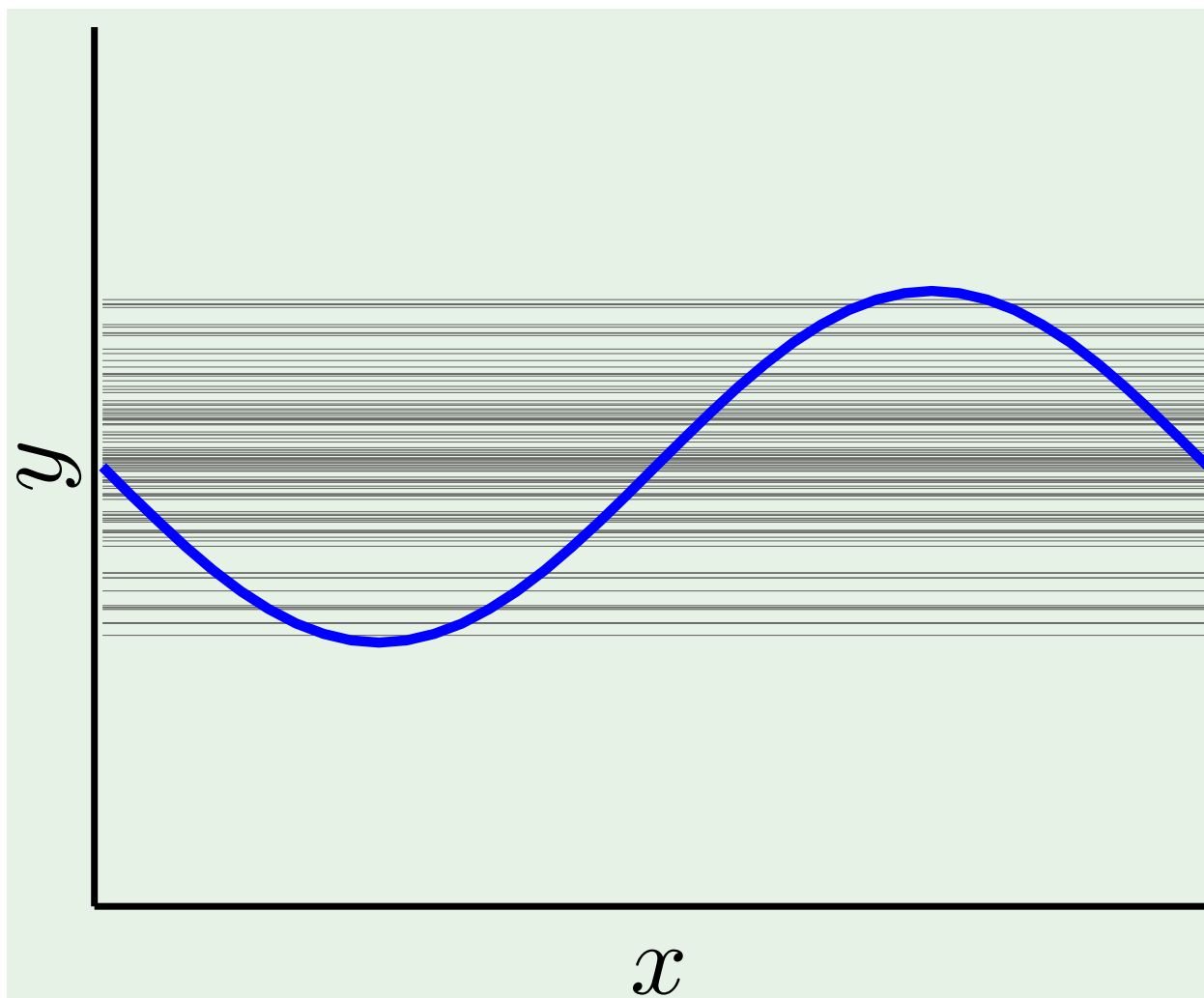
# Example





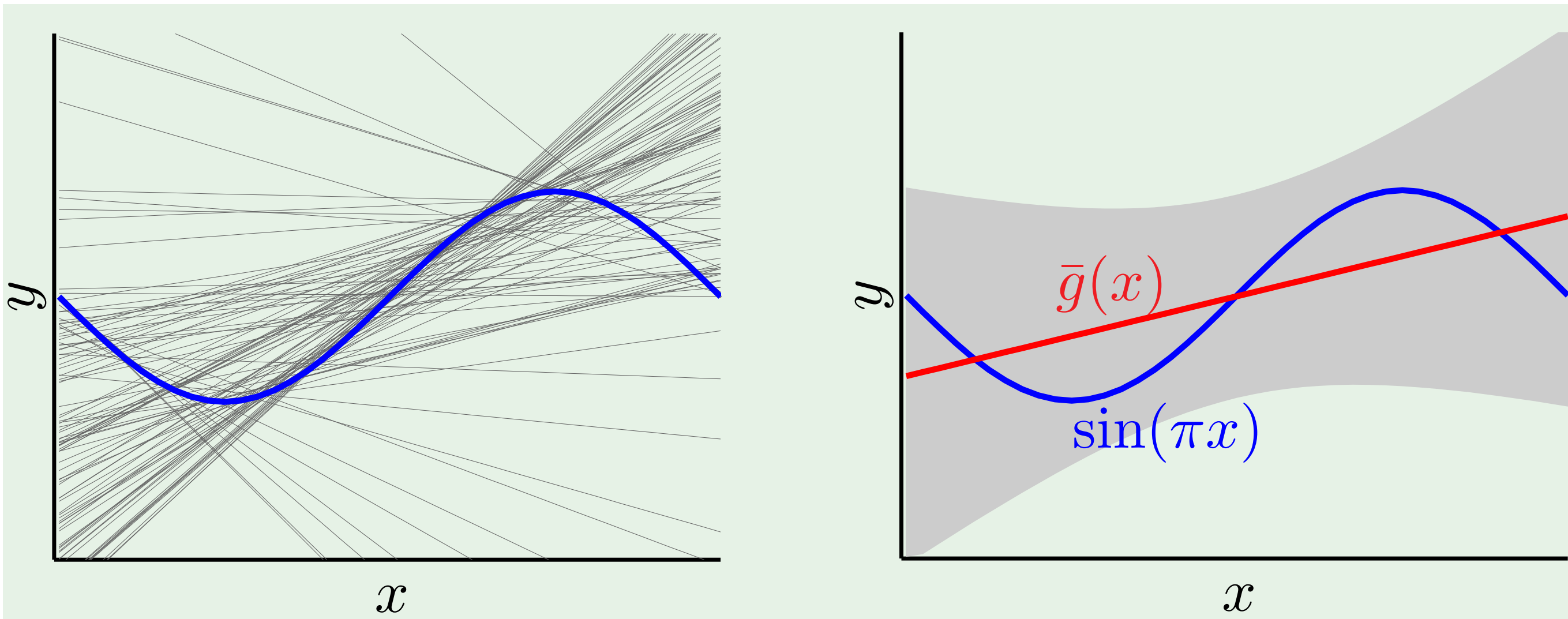
# Example

Bias and Variance of  $\mathcal{H}_0$



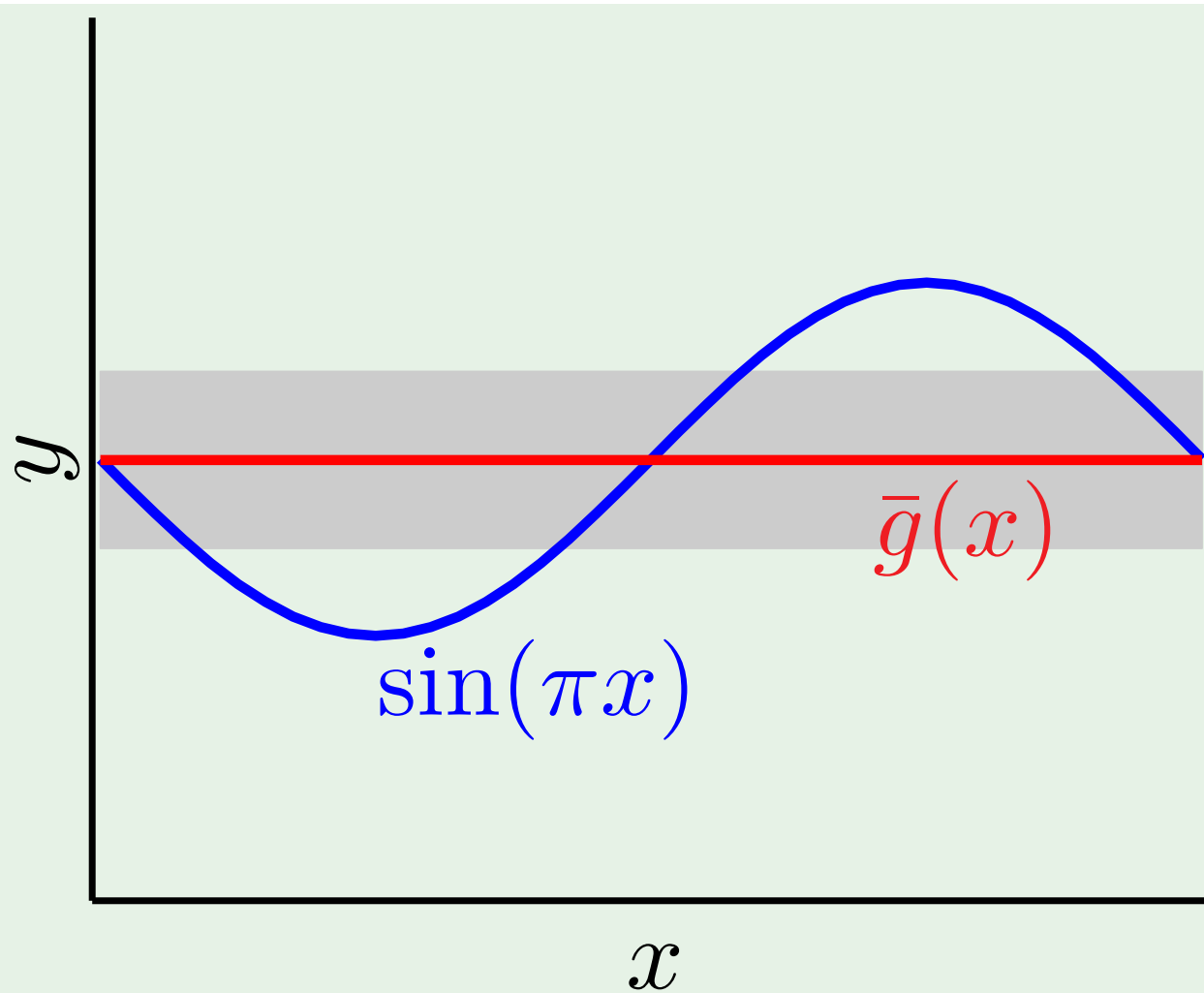
# Example

Bias and Variance of  $\mathcal{H}_1$



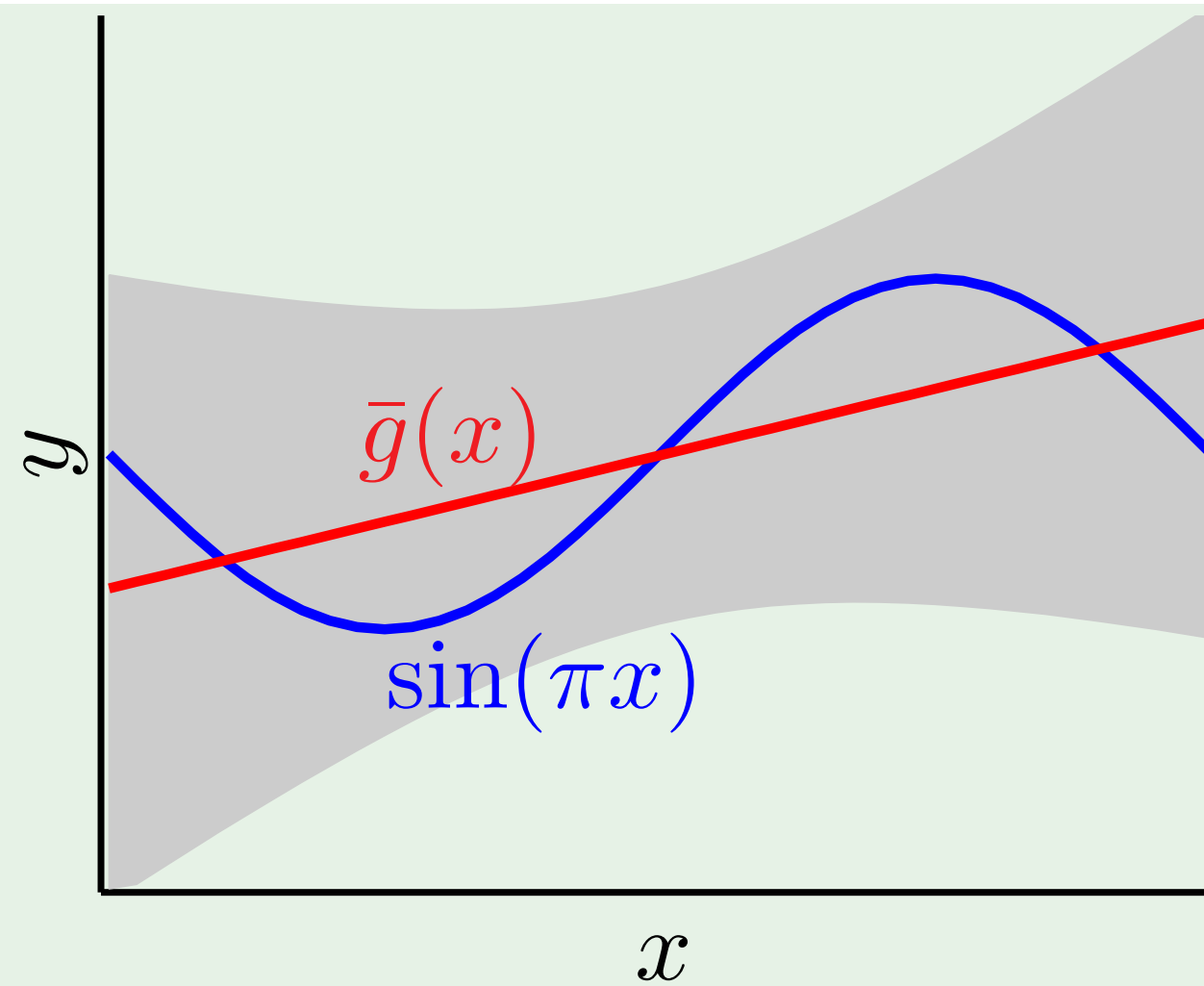
# Example

$\mathcal{H}_0$



Bias = 0.50 and Variance = 0.25

$\mathcal{H}_1$

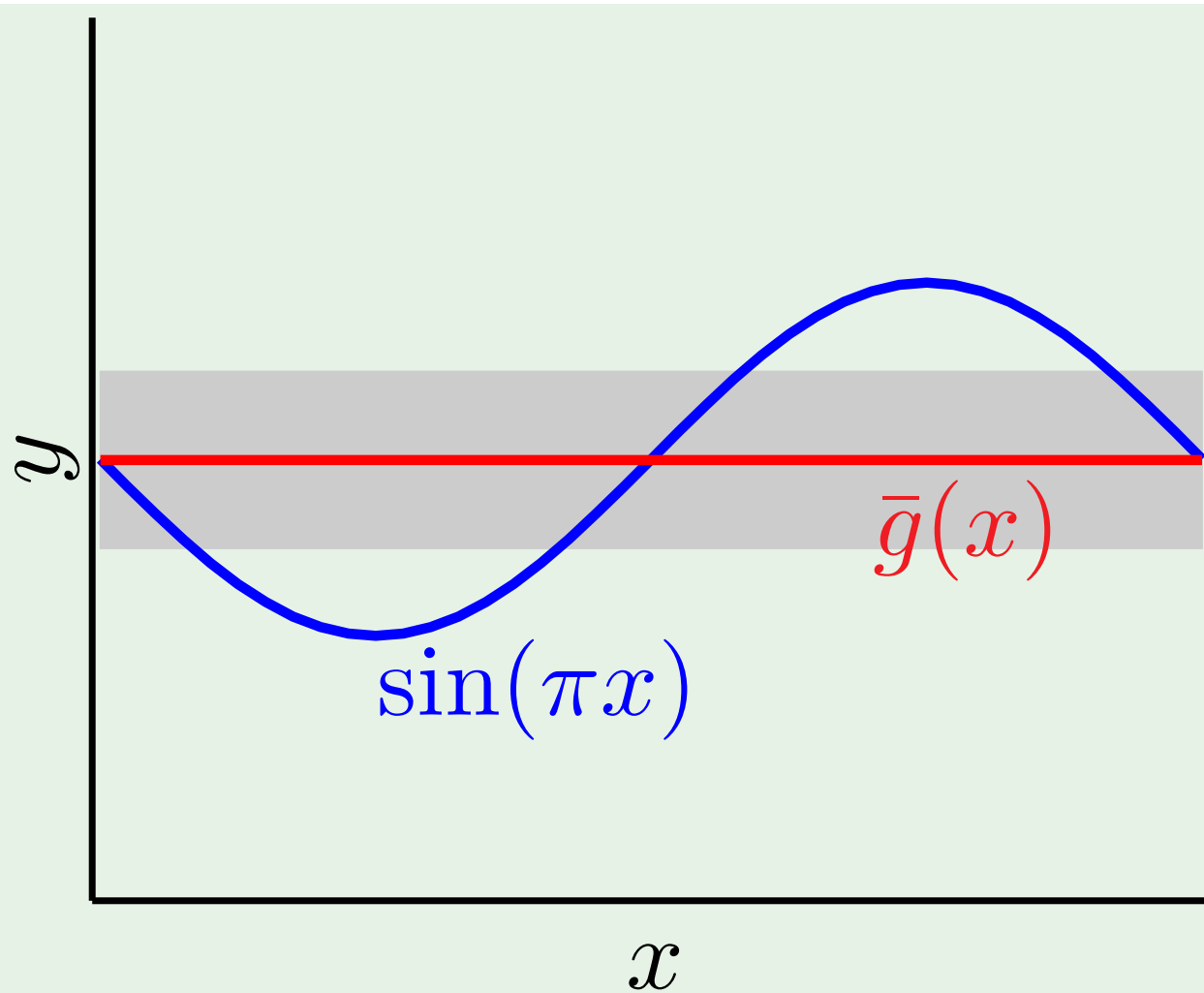


Bias = 0.21 and Variance = 1.69

# Example

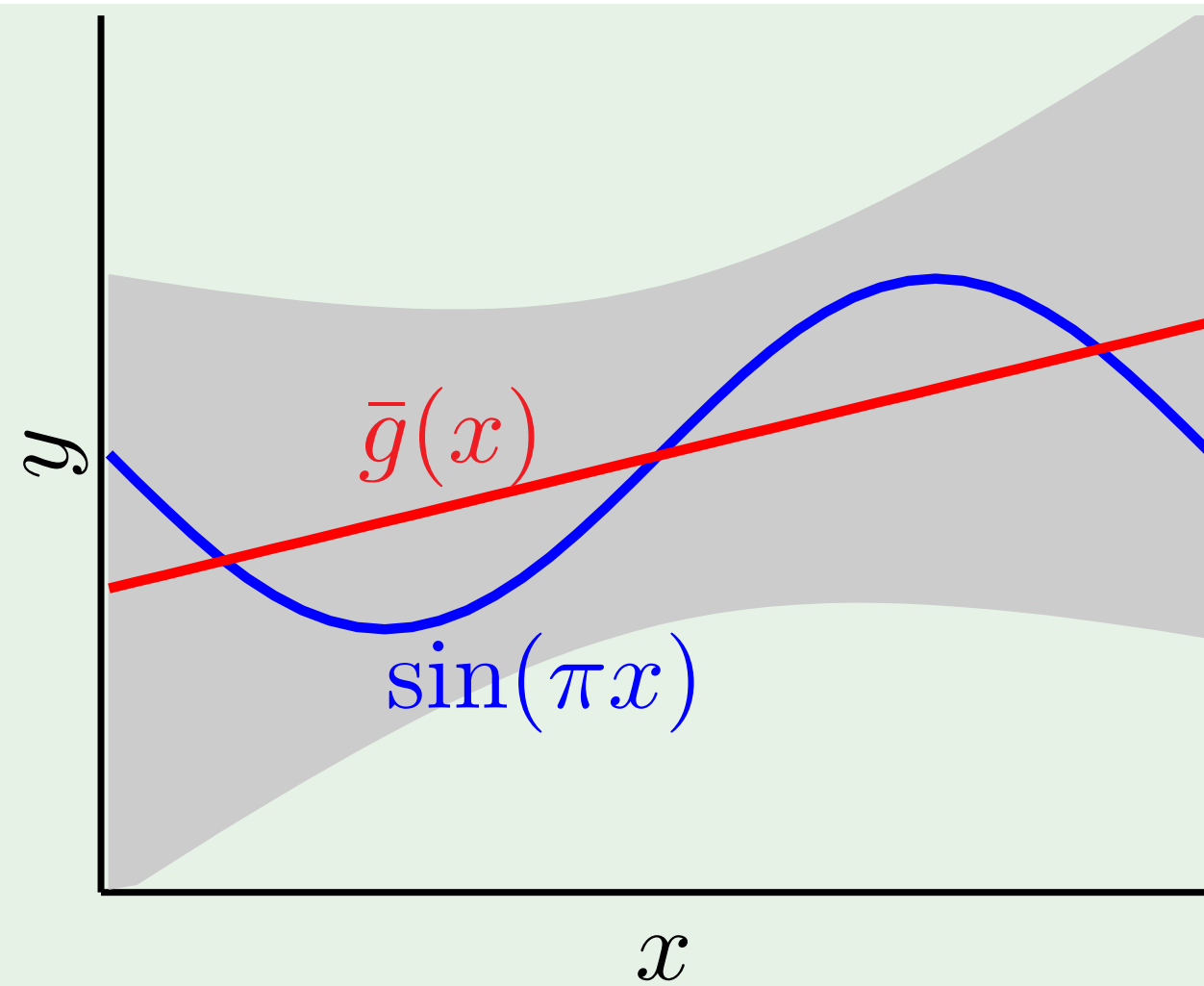
Winner!

$\mathcal{H}_0$



Bias = 0.50 and Variance = 0.25

$\mathcal{H}_1$



Bias = 0.21 and Variance = 1.69

# Moral of the Story

Match the model complexity **to the data** resources and **not the target** complexity

# Lecture Summary

Supervised Learning Setting

Linear Parametric Models for Regression

Subset Selection and Coefficient Shrinkage

Comparison of Shrinkage Methods

Bias Variance Tradeoff