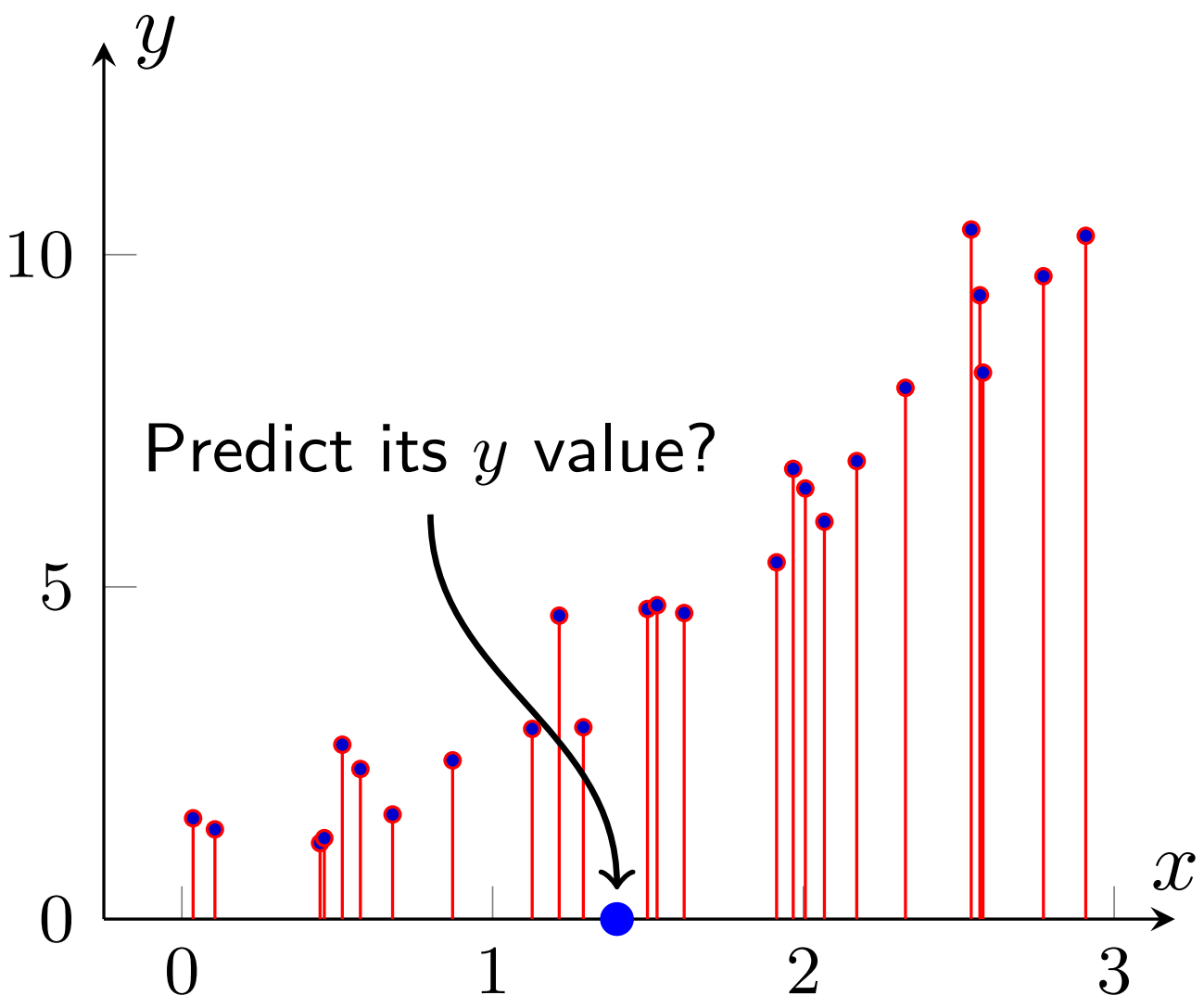
Chapter 2: Overview of Supervised Learning

DD3364

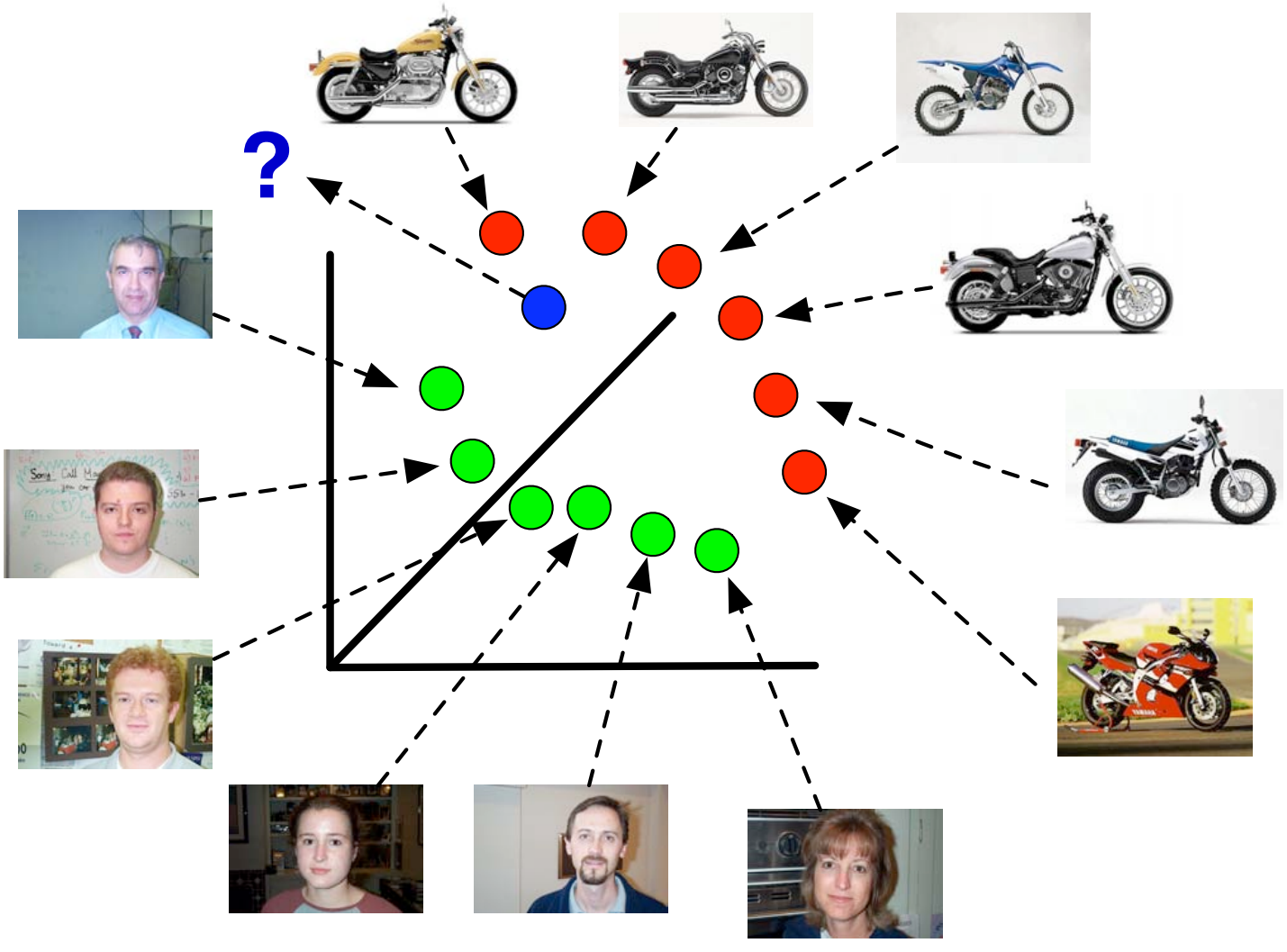
March 9, 2012

Introduction and Notation

Problem 1: Regression



Problem 2: Classification



Some Terminology

- In **Machine Learning** have outputs which are predicted from measured inputs.
- In **Statistical literature** have responses which are predicted from measured predictors.
- In **Pattern Recognition** have responses which are predicted from measured features.

Some Terminology

- In **Machine Learning** have outputs which are predicted from measured inputs.
- In **Statistical literature** have responses which are predicted from measured predictors.
- In **Pattern Recognition** have responses which are predicted from measured features.

The goal of **supervised learning** is to predict the value of the output(s) given an input and lots of labelled training examples

$$\{(\text{input}_1, \text{output}_1), (\text{input}_2, \text{output}_2), \ldots, (\text{input}_n, \text{output}_n)\}$$

Variable types

- **Outputs** can be
  - discrete (categorical, qualitative),
  - continuous (quantitative) or
  - ordered categorical (order is important)
- Predicting a discrete output is referred to as classification.
- Predicting a continuous output is referred to as regression.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

Notation of the book

- Denote an input variable by X.
- If X is a vector, its components are denoted by X_j
- Quantitative (continuous) outputs are denoted by Y
- Qualitative (discrete) outputs are denoted by G
- Observed values are written in lower case.
- x_i is the ith observed value of X. If X is a vector then x_i is a vector of the same length.
- g_i is the ith observed value of G.
- Matrices are represented by bold uppercase letters.

I will try to stick these conventions in the slides.

More Notation

- The prediction of the output for a given value of input vector X is denoted by $\hat{Y}$.
- It is presumed that we have labelled training data for regression problems

$$\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

with each $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$

- It is presumed that we have labelled training data for classification problems

$$\mathcal{T} = \{(x_1, g_1), \ldots, (x_n, g_n)\}$$

with each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, G\}$

More Notation

- The prediction of the output for a given value of input vector X is denoted by $\hat{Y}$.
- It is presumed that we have labelled training data for regression problems

$$\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

with each $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$

- It is presumed that we have labelled training data for classification problems

$$\mathcal{T} = \{(x_1, g_1), \ldots, (x_n, g_n)\}$$

with each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, G\}$

More Notation

- The prediction of the output for a given value of input vector X is denoted by $\hat{Y}$.
- It is presumed that we have labelled training data for regression problems

$$\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

with each $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$

- It is presumed that we have labelled training data for classification problems

$$\mathcal{T} = \{(x_1, g_1), \ldots, (x_n, g_n)\}$$

with each $x_i \in \mathbb{R}^p$ and $g_i \in \{1, \ldots, G\}$

Prediction via Least Squares and Nearest Neighbours

Linear Model

- Have an input vector $X = (X_1, \ldots, X_p)^t$
- A linear model predicts the output Y as

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

  where $\hat{\beta}_0$ is known as the **intercept** and also as the **bias**

- Let $X = (1, X_1, \ldots, X_p)^t$ and $\hat{\beta} = (\hat{\beta}_0, \ldots, \hat{\beta}_p)^t$ then

$$\hat{Y} = X^t \hat{\beta}$$

Linear Model

- Have an input vector $X = (X_1, \ldots, X_p)^t$
- A linear model predicts the output Y as

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j$$

  where $\hat{\beta}_0$ is known as the **intercept** and also as the **bias**

- Let $X = (1, X_1, \ldots, X_p)^t$ and $\hat{\beta} = (\hat{\beta}_0, \ldots, \hat{\beta}_p)^t$ then

$$\hat{Y} = X^t \hat{\beta}$$

Linear Models and Least Squares

- How is a linear model fit to a set of training data?
- Most popular approach is a Least Squares approach
- β is chosen to minimize

$$\text{RSS}(\beta) = \sum_{i=1}^{n} (y_i - x_i^t \beta)^2$$

- As this is quadratic a minimum always exist but it may not be unique.
- In matrix notation can write $\text{RSS}(\beta)$ as

$$\text{RSS}(\beta) = (y - \mathbf{X}\beta)^t (y - \mathbf{X}\beta)$$

where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a matrix with each row being an input vector and $y = (y_1, \ldots, y_n)^t$

Linear Models and Least Squares

- How is a linear model fit to a set of training data?
- Most popular approach is a Least Squares approach
- β is chosen to minimize

$$\text{RSS}(\beta) = \sum_{i=1}^{n} (y_i - x_i^t \beta)^2$$

- As this is quadratic a minimum always exist but it may not be unique.
- In matrix notation can write $\text{RSS}(\beta)$ as

$$\text{RSS}(\beta) = (y - \mathbf{X}\beta)^t (y - \mathbf{X}\beta)$$

where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a matrix with each row being an input vector and $y = (y_1, \ldots, y_n)^t$

- The solution to

$$\hat{\beta} = \arg\min_{\beta} (y - \mathbf{X}\beta)^t (y - \mathbf{X}\beta)$$

  is given by

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

  if $\mathbf{X}^t\mathbf{X}$ is non-singular

- This is easy to show by differentiation of RSS(β)
- This model has $p + 1$ parameters.

Linear Models, Least Squares and Classification

- Assume one has training data $\{(x_i, y_i)\}_{i=1}^n$ with each $y_i \in \{0, 1\}$ (it's really categorical data)
- A linear regression model $\hat{\beta}$ is fit to the data and

$$\hat{G}(x) = \begin{cases} 0 & \text{if } x^t\hat{\beta} \leq .5 \\ 1 & \text{if } x^t\hat{\beta} > .5 \end{cases}$$

- This is not the best way to perform binary classification with a linear discriminant function...

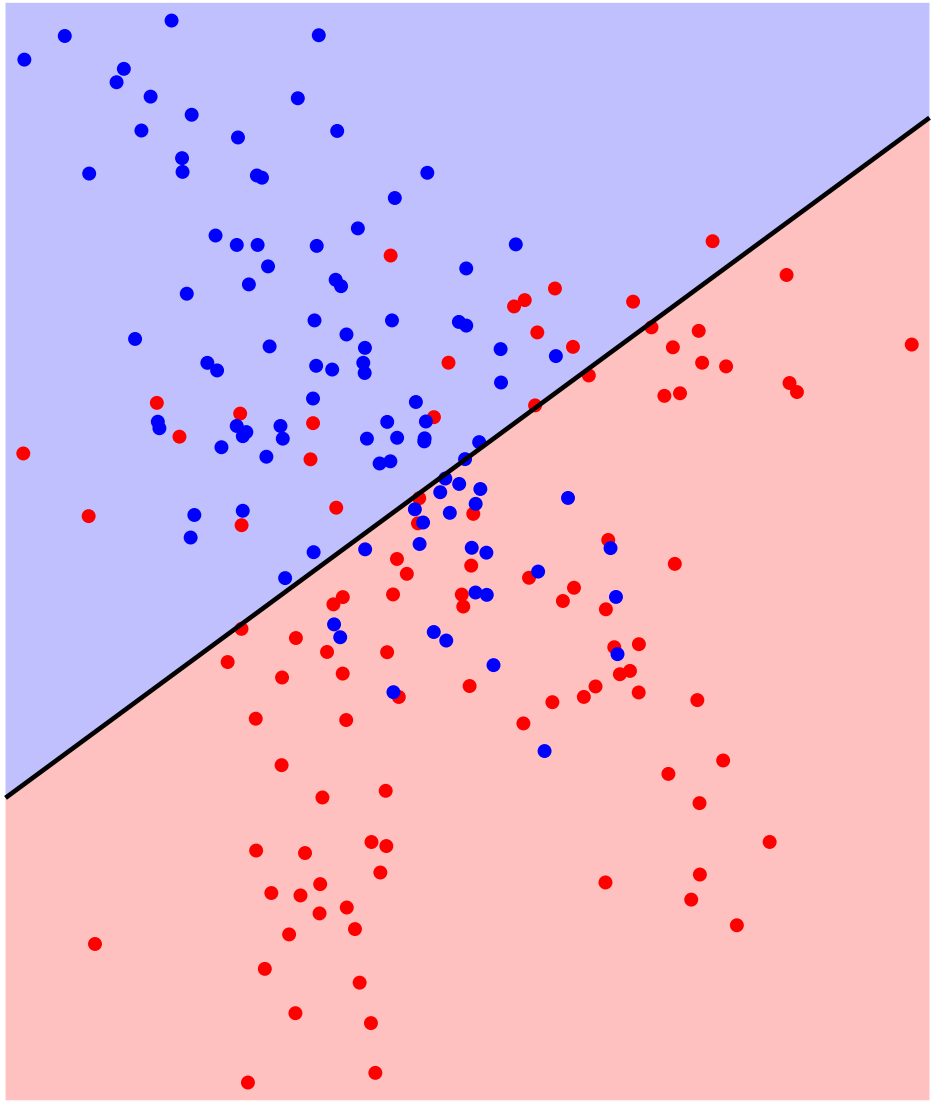
Linear Models, Least Squares and Classification

- Assume one has training data $\{(x_i, y_i)\}_{i=1}^n$ with each $y_i \in \{0, 1\}$ (it's really categorical data)
- A linear regression model $\hat{\beta}$ is fit to the data and

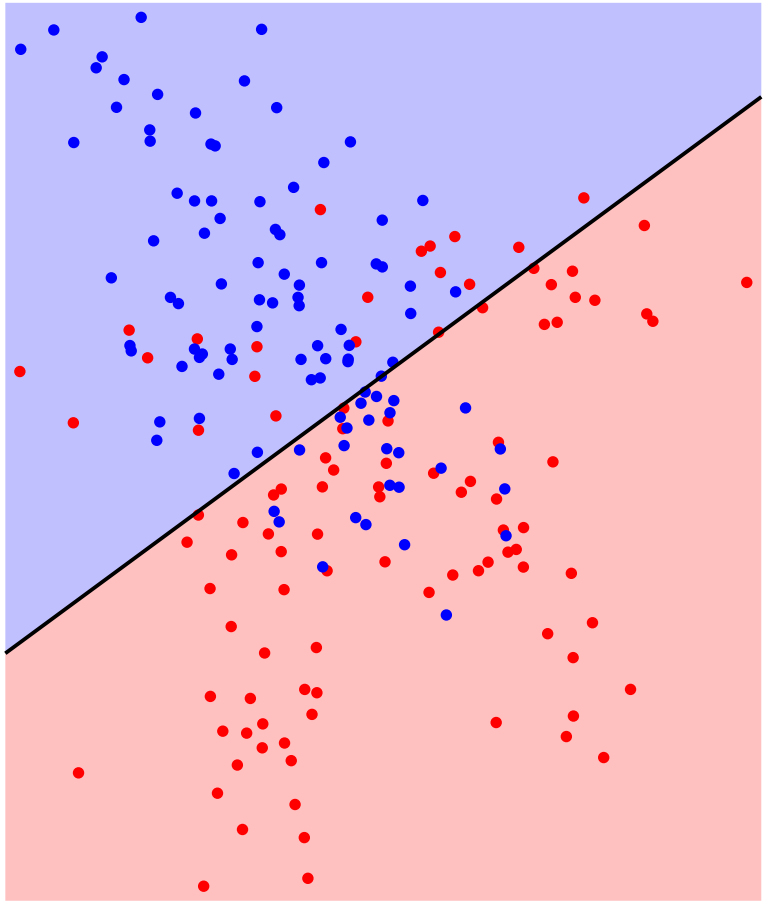
$$\hat{G}(x) = \begin{cases} 0 & \text{if } x^t\hat{\beta} \leq .5 \\ 1 & \text{if } x^t\hat{\beta} > .5 \end{cases}$$

- This is not the best way to perform binary classification with a linear discriminant function...

Example binary classification with



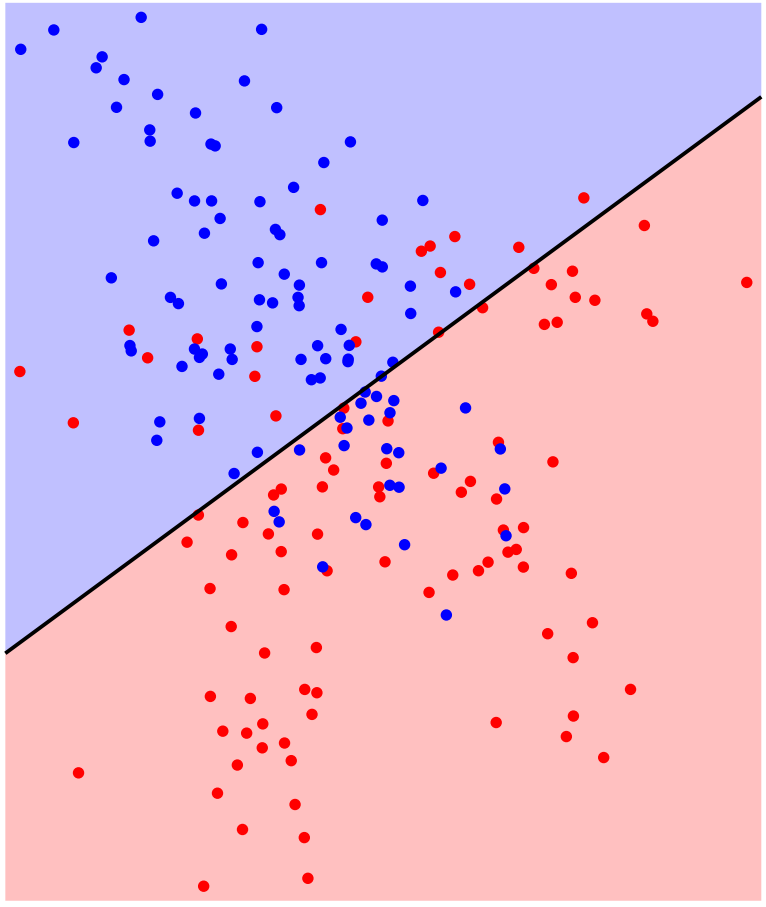
Example binary classification with



$k = 1$

- The linear classifier mis-classifies quite a few of the training examples
- The linear model may be too rigid
- By inspection it seem the two classes cannot be separated by a line
- Points from each class are generated from a GMM with 10 mixtures

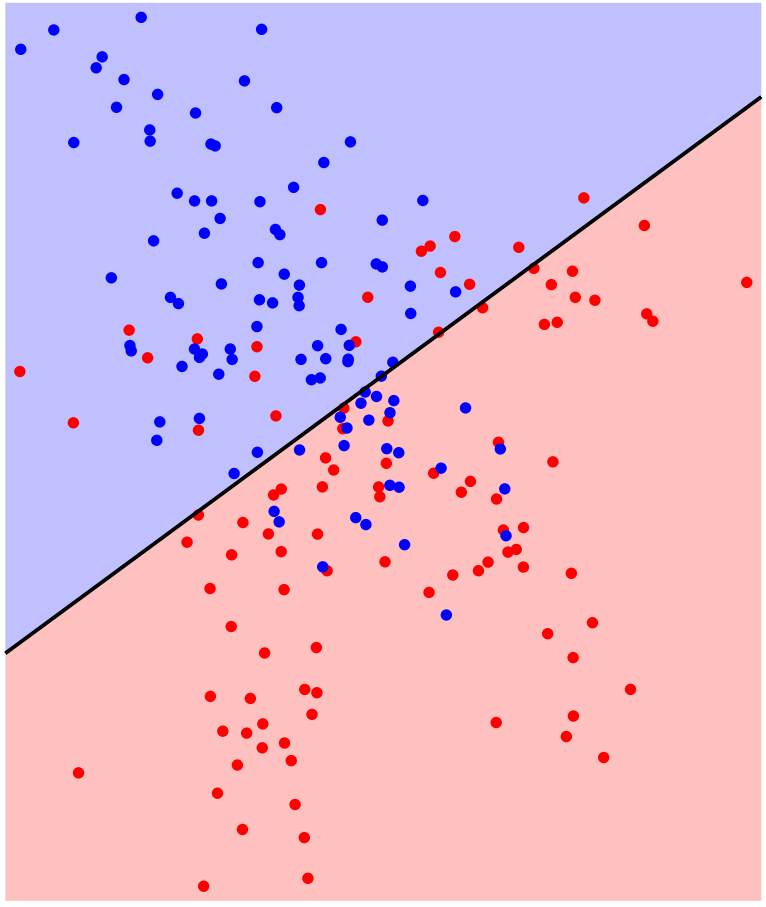
Example binary classification with



$k = 1$

- The linear classifier mis-classifies quite a few of the training examples
- The linear model may be too rigid
- By inspection it seem the two classes cannot be separated by a line
- Points from each class are generated from a GMM with 10 mixtures

Example binary classification with



$k = 1$

- The linear classifier mis-classifies quite a few of the training examples
- The linear model may be too rigid
- By inspection it seem the two classes cannot be separated by a line
- Points from each class are generated from a GMM with 10 mixtures

k-Nearest Neighbour regression fitting

- the k-nearest neighbour fit for $\hat{Y}$ is

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  where $N_k(x)$ is the neighbourhood of x defined by the k closest points x_i in the training data.

- Closeness if defined by some metric.

- For this lecture assume it is the Euclidean distance.

- k-nearest neighbours in words:
  Find the k observations x_i closest to x and average their responses.

k-Nearest Neighbour binary classification

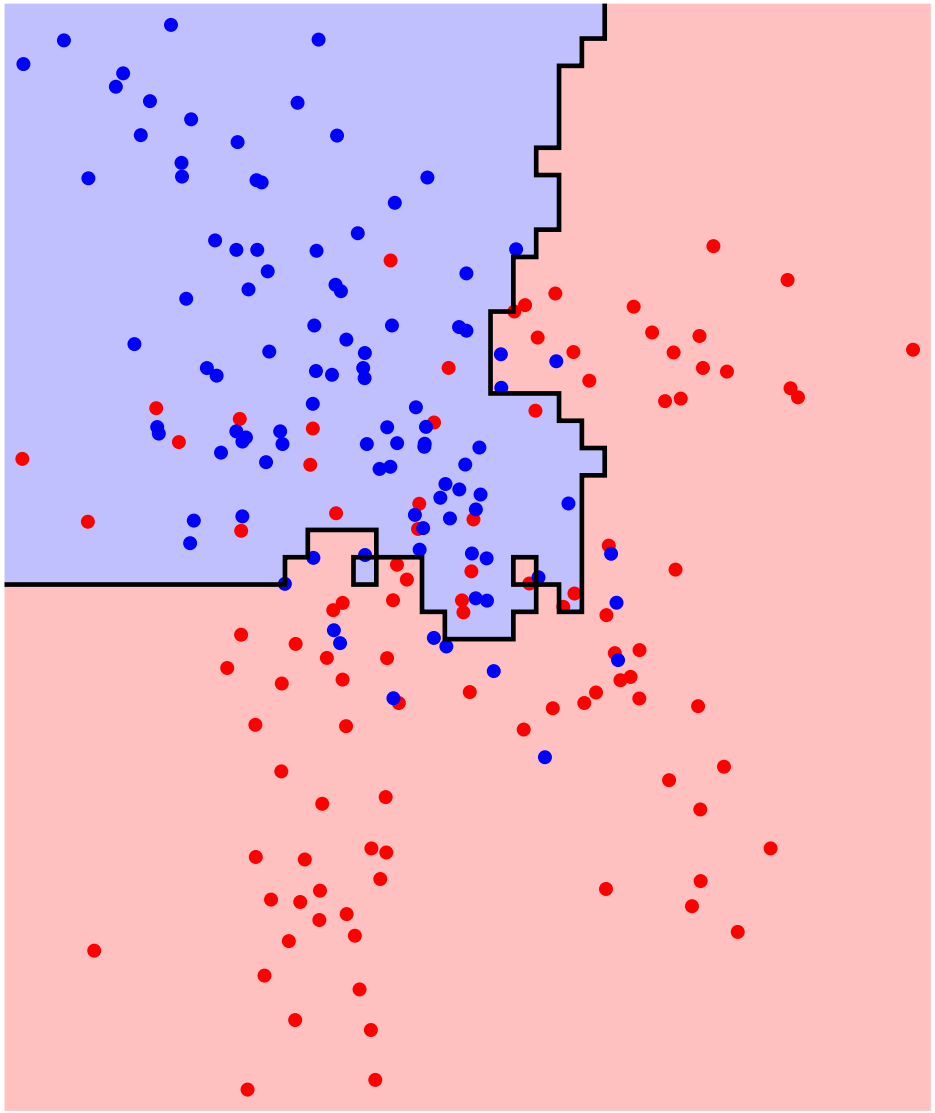
- Training data: $\{(x_i, g_i)\}$ with each $g_i \in \{0, 1\}$
- the k-nearest neighbour estimate for $\hat{G}$ is

$$\hat{G}(x) = \begin{cases} 0 & \text{if } \left(\frac{1}{k}\sum_{x_i \in N_k(x)} g_i\right) \leq .5 \\ 1 & \text{otherwise} \end{cases}$$

  where $N_k(x)$ is the neighbourhood of x defined by the k closest points x_i in the training data.

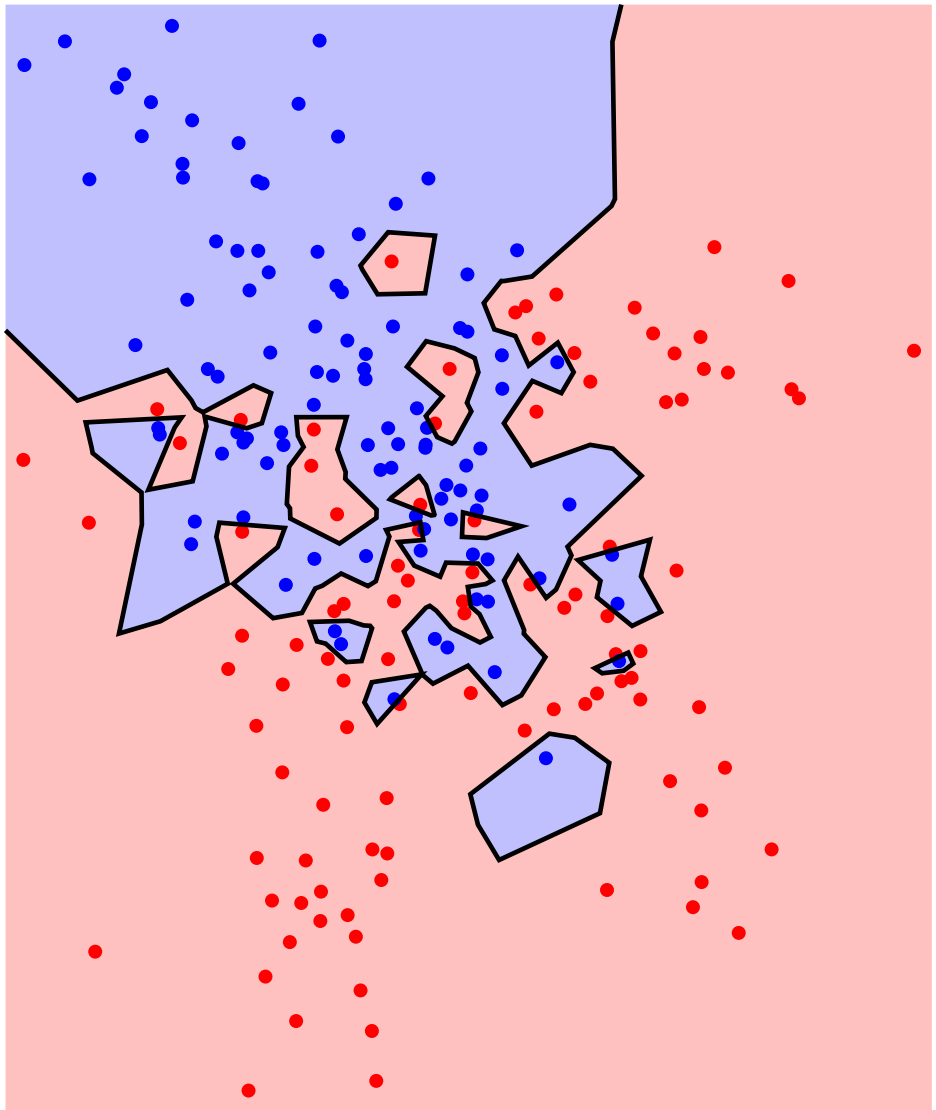
- k-nearest neighbours in words:
  Find the k observations x_i closest to x and estimate the class of x as the majority class amongst the neighbours.

Example: k-Nearest Neighbour classification



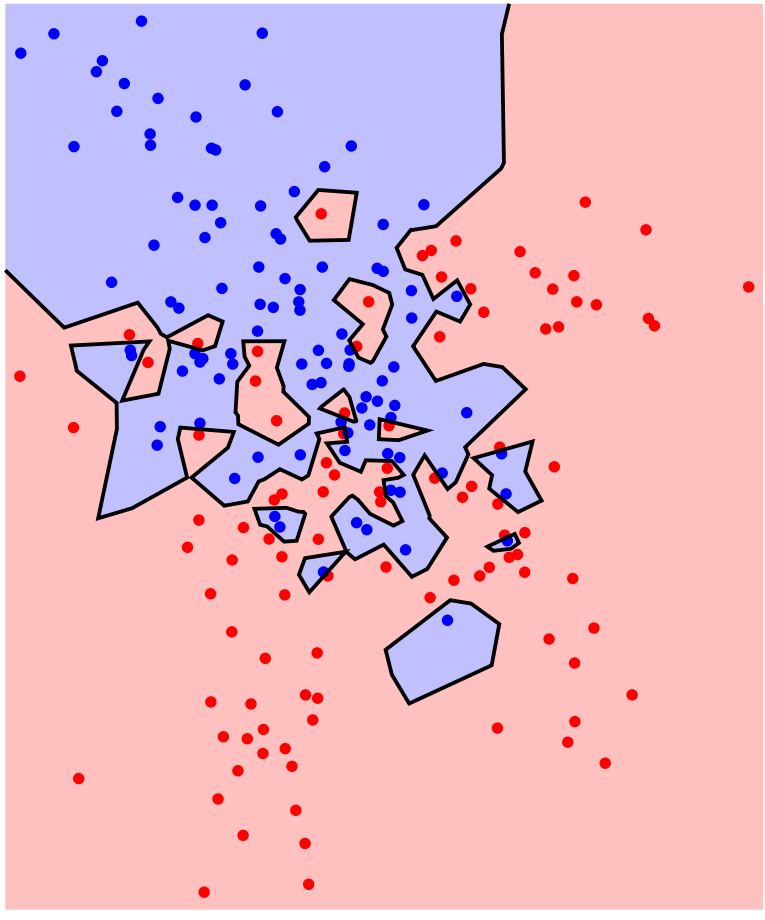
$k = 15$

Example: k-Nearest Neighbour classification



$k = 1$

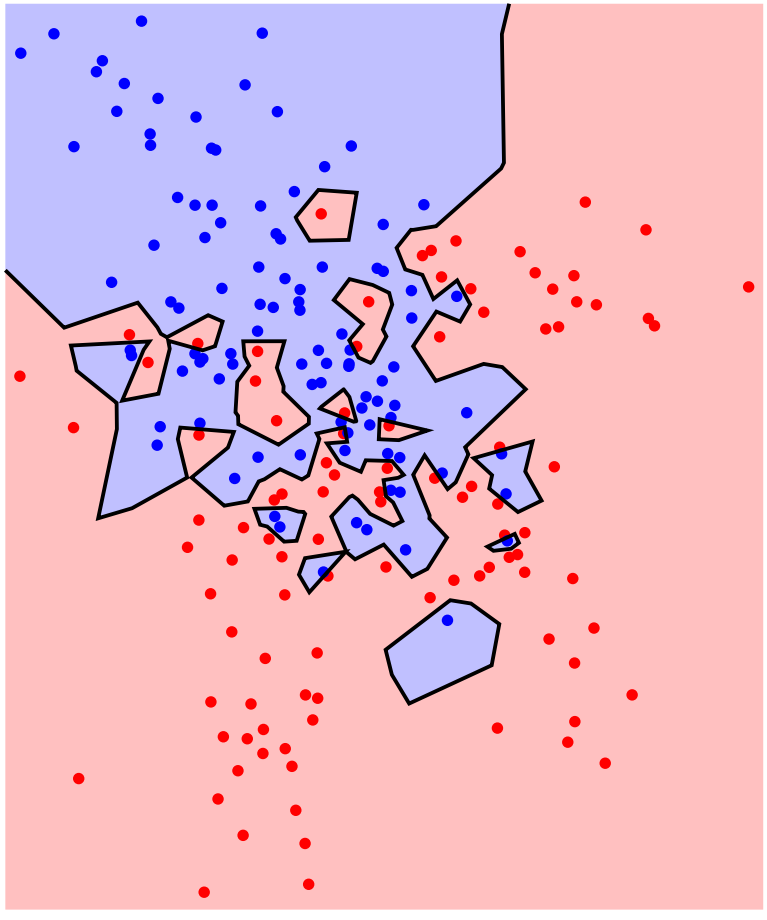
Example: k-Nearest Neighbour classification



$k = 1$

- For $k = 1$ all the training examples are correctly classified.
- This is always the case !
- But how well will it perform on test data drawn from the same distribution?

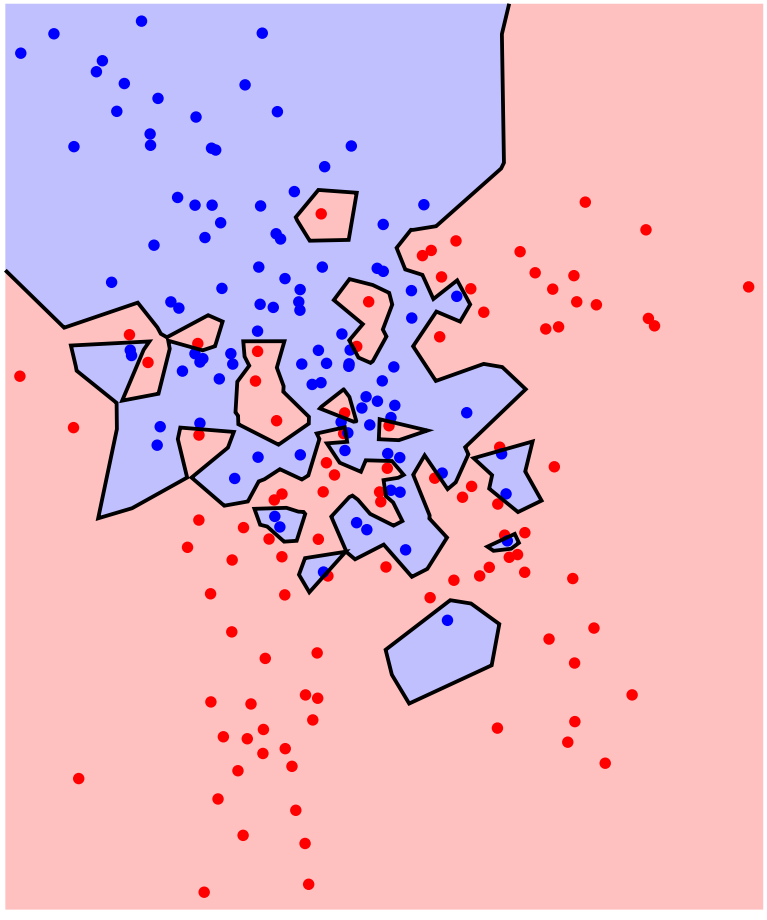
Example: k-Nearest Neighbour classification



$k = 1$

- For $k = 1$ all the training examples are correctly classified.
- This is always the case !
- But how well will it perform on test data drawn from the same distribution?

Example: k-Nearest Neighbour classification



$k = 1$

- For $k = 1$ all the training examples are correctly classified.
- This is always the case !
- But how well will it perform on test data drawn from the same distribution?

Effective number of parameters of k-nn

- There are **two** parameters that control the behaviour of k-nn.
- These are k and n the number of training samples
- The effective number of parameters of k-nn is n/k
- Intuitively
  - say the nbds were non-overlapping
  - Would have n/k neighbourhoods
  - Need to fit one parameter (a mean) to each neighbourhood

Effective number of parameters of k-nn

- There are **two** parameters that control the behaviour of k-nn.
- These are k and n the number of training samples
- The effective number of parameters of k-nn is n/k
- Intuitively
  - say the nbds were non-overlapping
  - Would have n/k neighbourhoods
  - Need to fit one parameter (a mean) to each neighbourhood

k-nn Vs Linear decision boundaries

- **Linear** decision boundary is
  - smooth,
  - stable to fit
  - assumes a linear decision boundary is suitable

  In statistical learning lingo: it has **low variance** and **high bias**
- k-**nn** decision boundary is
  - can adapt to any shape of the data,
  - unstable to fit (for small k)
  - not smooth, wiggly (for small k)

  In statistical learning lingo: it has **high variance** and **low bias**

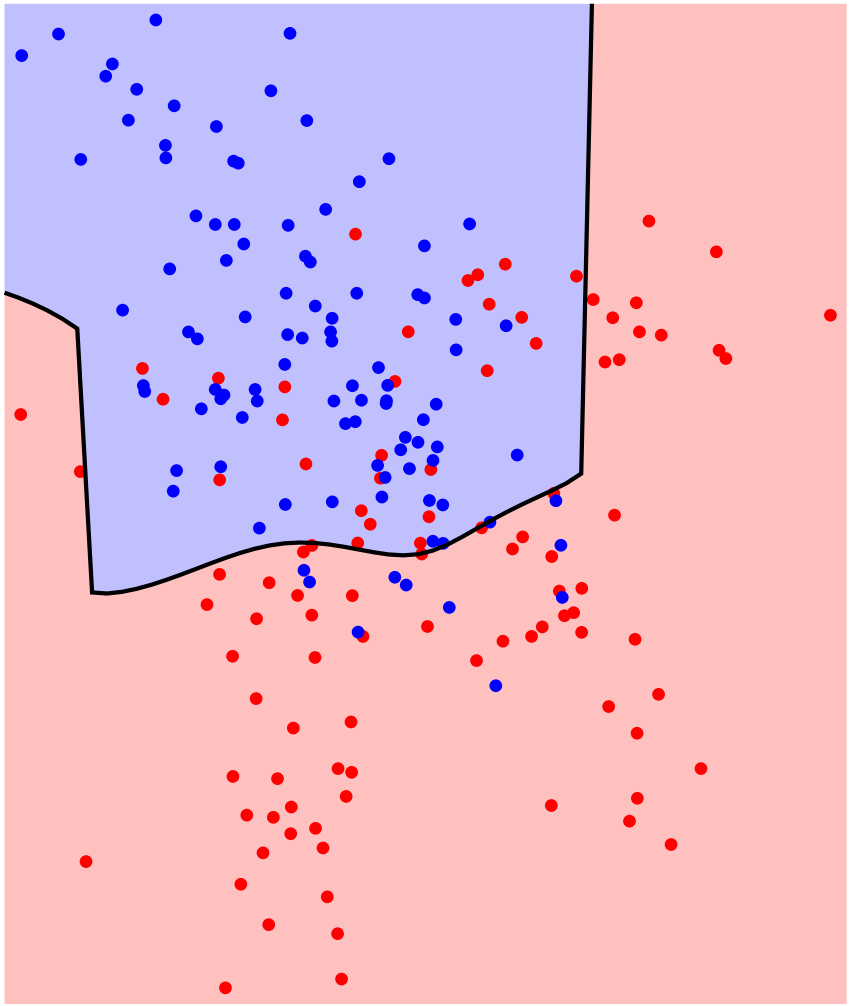
k-**nn Vs Linear** decision boundaries

- **Linear** decision boundary is
  - smooth,
  - stable to fit
  - assumes a linear decision boundary is suitable

  In statistical learning lingo: it has **low variance** and **high bias**
- k-**nn** decision boundary is
  - can adapt to any shape of the data,
  - unstable to fit (for small k)
  - not smooth, wiggly (for small k)

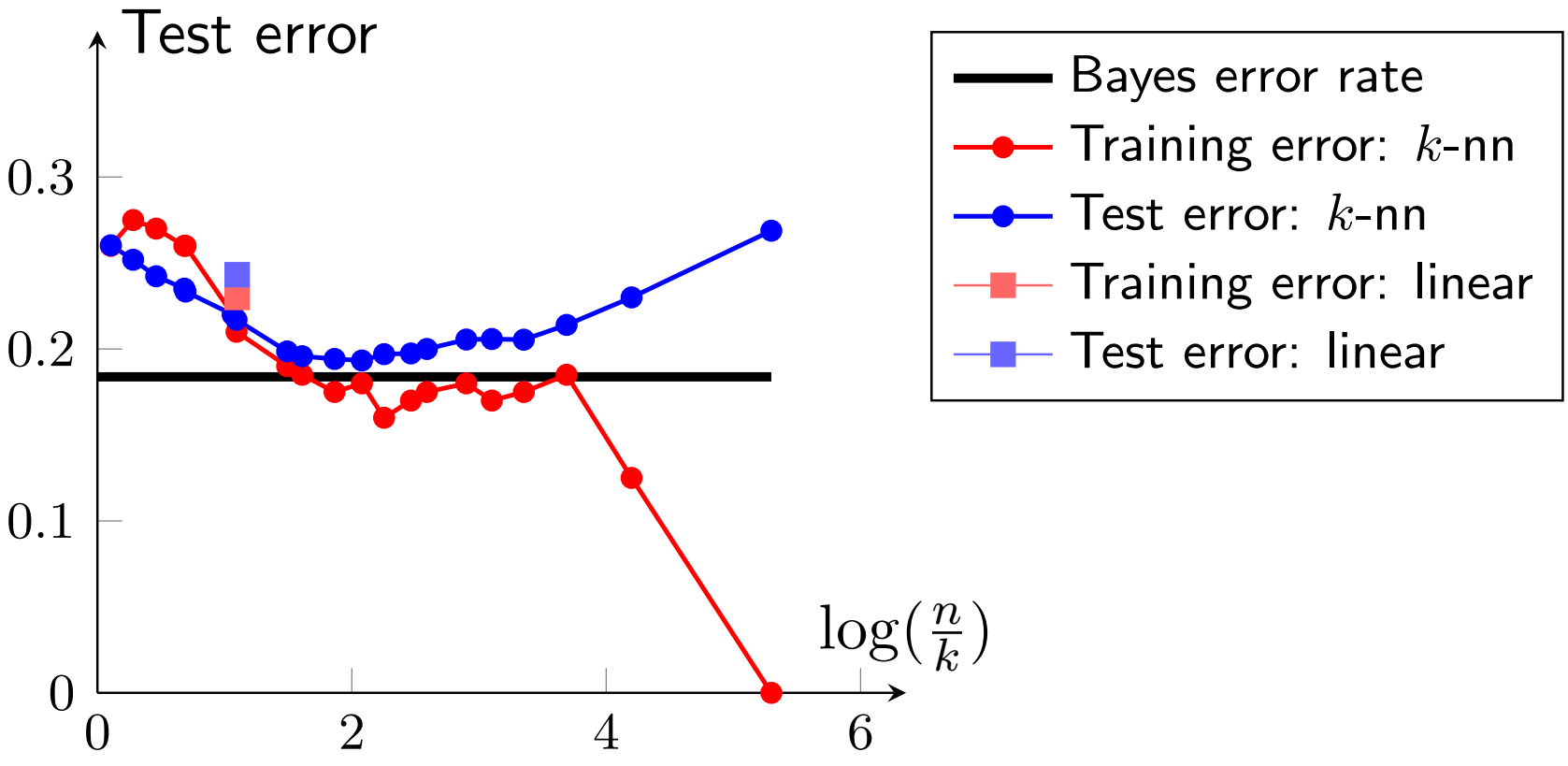
  In statistical learning lingo: it has **high variance** and **low bias**

Optimal Bayes decision boundary



This is the optimal decision boundary computed from the known pdfs for the two classes.

Mis-classification rate for the simulation experiment



$n_{\text{train}} = 200$ and $n_{\text{test}} = 10,000$

Statistical Decision Theory

- How do we measure how well $f(X)$ predicts Y?
- Statisticians would compute the **Expected Prediction Error** w.r.t. some loss function

$$\text{EPE}(f) = E_{X,Y}[L(Y, f(X))] = \int \int L(y, f(x))\, p(x, y)\, dx\, dy$$

- A common loss function is the squared error loss

$$L(y, f(x)) = (y - f(x))^2$$

- By conditioning on X can write

$$\text{EPE}(f) = E_{X,Y}[(Y - f(X))^2] = E_X[\, E_{Y|X}[(Y - f(X))^2|X]\,]$$

- How do we measure how well $f(X)$ predicts Y?
- Statisticians would compute the **Expected Prediction Error** w.r.t. some loss function

$$\text{EPE}(f) = E_{X,Y}[L(Y, f(X))] = \int \int L(y, f(x))\, p(x, y)\, dx\, dy$$

- A common loss function is the squared error loss

$$L(y, f(x)) = (y - f(x))^2$$

- By conditioning on X can write

$$\text{EPE}(f) = E_{X,Y}[(Y - f(X))^2] = E_X[\, E_{Y|X}[(Y - f(X))^2|X]\,]$$

- At a point x can minimize EPE to get the best prediction of y

$$f(x) = \arg\min_c E_{Y|X}[(Y-c)^2|X=x]$$

- The solution is

$$f(x) = E[Y|X=x]$$

This is known as the regression function.

- At a point x can minimize EPE to get the best prediction of y

$$f(x) = \arg\min_c E_{Y|X}[(Y-c)^2|X=x]$$

- The solution is

$$f(x) = E[Y|X=x]$$

  This is known as the regression function.

- **Only one problem with this**: one rarely knows the pdf $p(Y|X)$.

- The regression methods we encounter can be viewed as ways to approximate $E[Y|X=x]$.

Local Methods in High Dimensions

Intuition and $k-$nearest neighbour averaging

Example:

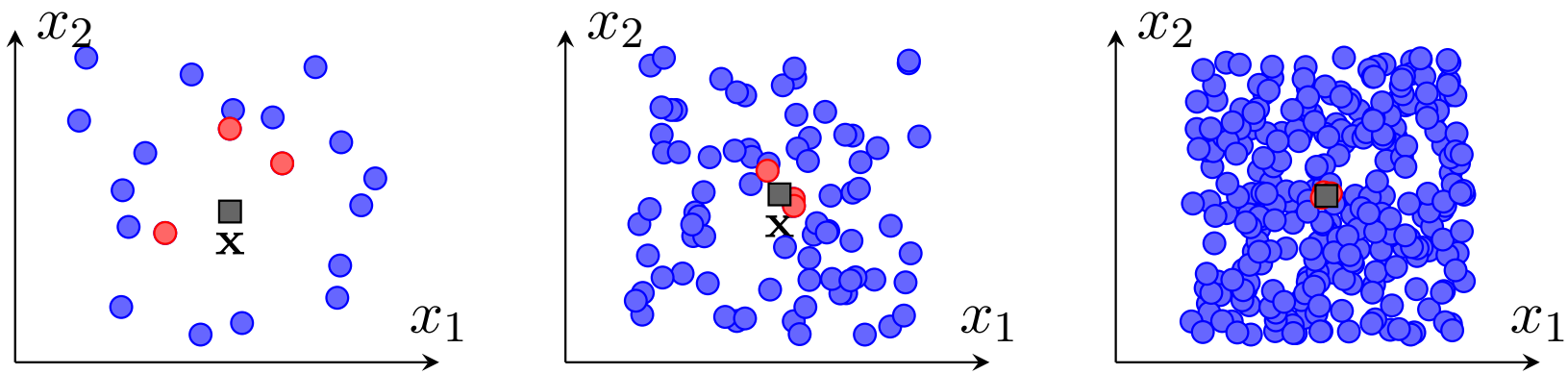
- Training data $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{X} \subset \mathbb{R}^p$ and $y_i \in \mathbb{R}$
- Predict response at $x \in \mathcal{X}$ using the *training data* and 3-*nn averaging*.

Intuition and $k-$nearest neighbour averaging

Let

- $\mathcal{X} = [-1, 1]^2$ and
- the training x_i's be uniformly sampled from $\mathcal{X}$.

x_i's from training sets of different size



- As n increases the expected area of the nbd containing the 3 nearest neighbours decreases
- $\Longrightarrow$ accuracy of $\hat{y}$ increases.

Intuition and $k-$nearest neighbour averaging

Therefore intuition says:

Lots of training data

$\Downarrow$

k-nearest neighbour produces accurate stable prediction.

More formally:

As n increases then

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \longrightarrow E[y \,|\, x]$$

Intuition and $k-$nearest neighbour averaging

Therefore intuition says:

Lots of training data

$\Downarrow$

k-nearest neighbour produces accurate stable prediction.

More formally:

As n increases then

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \longrightarrow E[y \,|\, x]$$

However for large p...

The *Curse of Dimensionality* (Bellman, 1961)

- k-nearest neighbour averaging approach and our intuition breaks down in high dimensions.

However for large p...

The *Curse of Dimensionality* (Bellman, 1961)

- k-nearest neighbour averaging approach and our intuition breaks down in high dimensions.

Manifestations of this problem

For large p

- Nearest neighbours are not so close !
- The k-nn of x are closer to the boundary of $\mathcal{X}$.
- Need a prohibitive number of training samples to densely sample $\mathcal{X} \subset R^p$

However for large p...

The *Curse of Dimensionality* (Bellman, 1961)

- k-nearest neighbour averaging approach and our intuition breaks down in high dimensions.

Manifestations of this problem

For large p

- Nearest neighbours are not so close !
- The k-nn of x are closer to the boundary of $\mathcal{X}$.
- Need a prohibitive number of training samples to densely sample $\mathcal{X} \subset R^p$

However for large p...

The *Curse of Dimensionality* (Bellman, 1961)

- k-nearest neighbour averaging approach and our intuition breaks down in high dimensions.

Manifestations of this problem

For large p

- Nearest neighbours are not so close !
- The k-nn of x are closer to the boundary of $\mathcal{X}$.
- Need a prohibitive number of training samples to densely sample $\mathcal{X} \subset R^p$

Curse of Dimensionality: Problem 1

For large p nearest neighbours are not so close

Scenario:
Estimate a regression function, $f : \mathcal{X} \to \mathbb{R}$, using a k-nn regressor.
Have

- $\mathcal{X} = [0, 1]^p$ (the *unit hyper-cube*)
- training inputs are uniformly sampled from $\mathcal{X}$.

For large p nearest neighbours are not so close

Scenario:
Estimate a regression function, $f : \mathcal{X} \to \mathbb{R}$, using a k-nn regressor.
Have

- $\mathcal{X} = [0, 1]^p$ (the *unit hyper-cube*)
- training inputs are uniformly sampled from $\mathcal{X}$.

Question:
Let $k = r\,n$ where $r \in [0, 1]$ and $x = \mathbf{0}$.

What is the expected length of the side of the minimal hyper-cube containing the k-nearest neighbours of x?

For large p nearest neighbours are not so close

Scenario:
Estimate a regression function, $f : \mathcal{X} \to \mathbb{R}$, using a k-nn regressor. Have

- $\mathcal{X} = [0, 1]^p$ (the *unit hyper-cube*)
- training inputs are uniformly sampled from $\mathcal{X}$.

Question:
Let $k = r\, n$ where $r \in [0, 1]$ and $x = \mathbf{0}$.

What is the expected length of the side of the minimal hyper-cube containing the k-nearest neighbours of x?

Solution:
Volume of hyper-cube of side a is a^p. Looking for a s.t. a^p equals a fraction r of the unit hyper-cube volume. Therefore

$$a^p = r \implies a = r^{1/p}$$

For large p nearest neighbours are not close

To recap the expected edge length of the hyper-cube containing a fraction r of the training data is

$$e_p(r) = r^{1/p}$$

For large p nearest neighbours are not close

To recap the expected edge length of the hyper-cube containing a fraction r of the training data is

$$e_p(r) = r^{1/p}$$

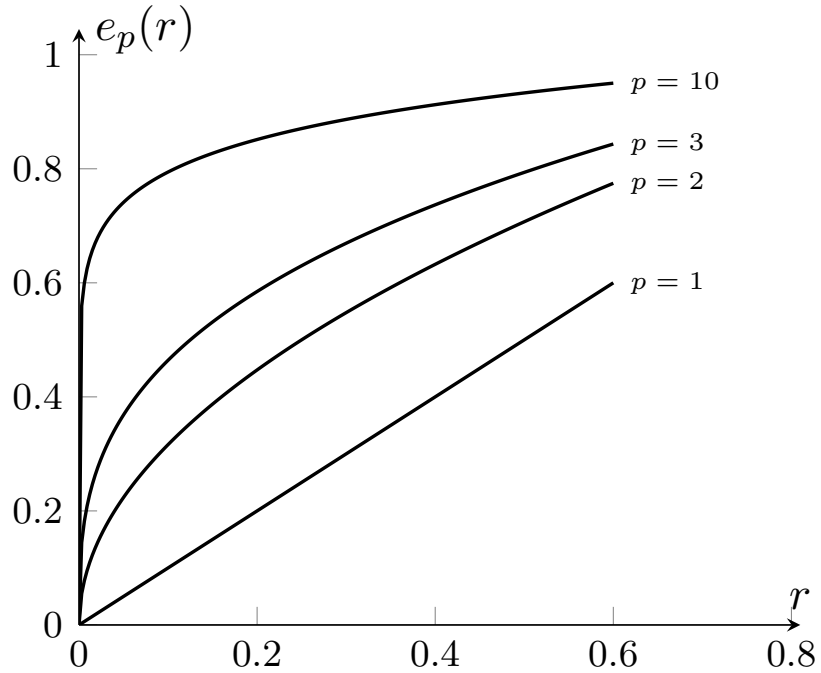
Plug in some numbers

Let $p = 10$ then

$$e_p(.01) = .63, \quad e_p(.1) = .80$$

Entire range for each input is 1.

Therefore in this case 1% and 10% nearest neighbour estimate are not *local* estimates.



Curse of Dimensionality: Problem 2

For large p nearest neighbours are not close II

Scenario:
Estimate a regression function, $f : \mathcal{X} \to \mathbb{R}$, using a k-nearest neighbour regressor. Have

- $\mathcal{X}$ is the *unit hyper-sphere*(ball) in $\mathbb{R}^p$ centred at the origin.
- n training inputs are uniformly sampled from $\mathcal{X}$.

For large p nearest neighbours are not close II

Scenario:
Estimate a regression function, $f : \mathcal{X} \to \mathbb{R}$, using a k-nearest neighbour regressor. Have

- $\mathcal{X}$ is the *unit hyper-sphere*(ball) in $\mathbb{R}^p$ centred at the origin.
- n training inputs are uniformly sampled from $\mathcal{X}$.

Question:
Let $k = 1$ and $x = \mathbf{0}$.

What is the median distance of the nearest neighbour to x?

For large p nearest neighbours are not close II

Scenario:
Estimate a regression function, $f : \mathcal{X} \to \mathbb{R}$, using a k-nearest neighbour regressor. Have

- $\mathcal{X}$ is the *unit hyper-sphere*(ball) in $\mathbb{R}^p$ centred at the origin.
- n training inputs are uniformly sampled from $\mathcal{X}$.

Question:
Let $k = 1$ and $x = \mathbf{0}$.

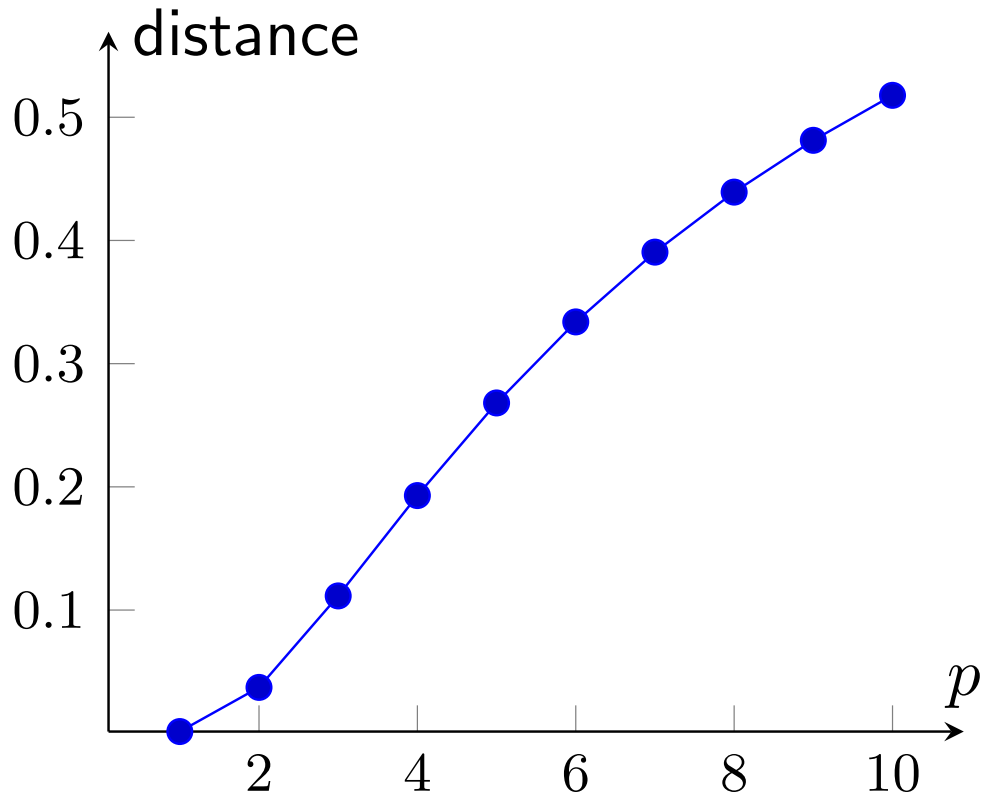
What is the median distance of the nearest neighbour to x?

Solution:
This median distance is given by the expression

$$d(p, n) = (1 - .5^{\frac{1}{n}})^{\frac{1}{p}}$$

Median distance of nearest neighbour to the origin

Plot of $d(p, n)$ for $n = 500$



Note: For $p = 10$ the closest training point is closer to the boundary of $\mathcal{X}$ than to x

Consequence of this expression

Consequence

For large p most of the training data points are closer to the boundary of $\mathcal{X}$ than to x.

This is bad because

- To make a prediction at x, you will use training samples near the edge of the training data
- Therefore perform extrapolation as opposed to interpolation between neighbouring samples.

Curse of Dimensionality: Problem 3

Dense sampling in high dimensions is prohibitive

Explanation:

- Say $n_1 = 100$ samples represents a dense sampling for a single input problem
- Then $n_{10} = 100^{10}$ is required to densely sample with 10 such inputs.

Therefore in high dimensions all feasible training sets **sparsely** sample the input space.

Dense sampling in high dimensions is prohibitive

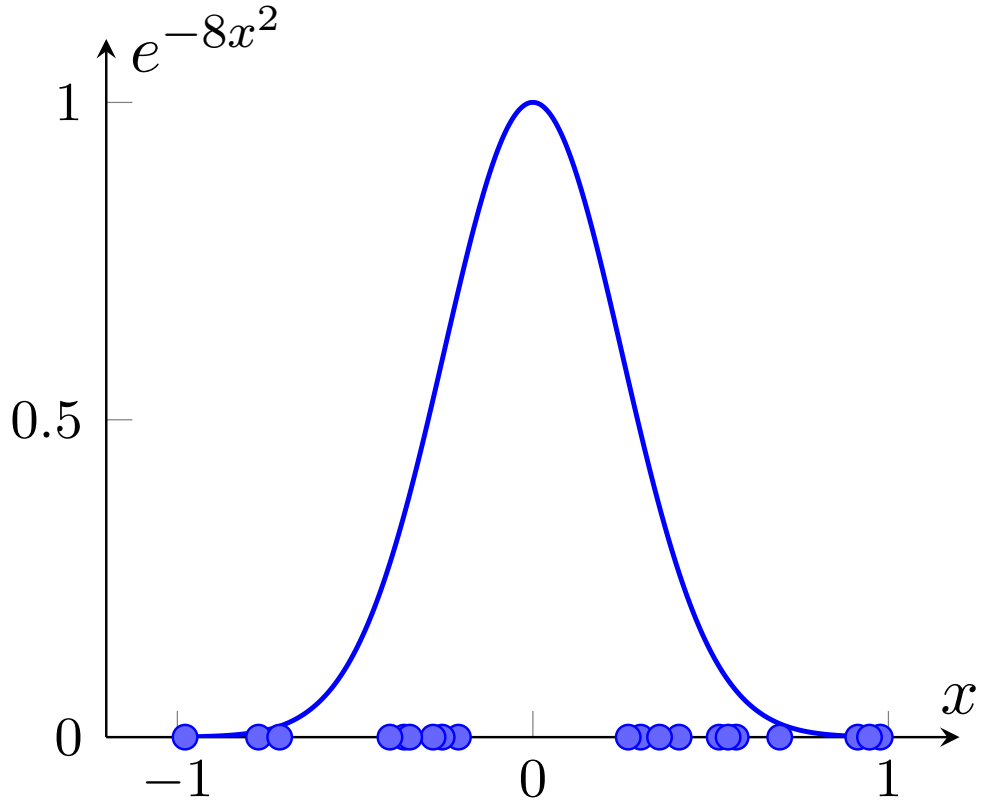
Explanation:

- Say $n_1 = 100$ samples represents a dense sampling for a single input problem
- Then $n_{10} = 100^{10}$ is required to densely sample with 10 such inputs.

Therefore in high dimensions all feasible training sets **sparsely** sample the input space.

Simulated Example

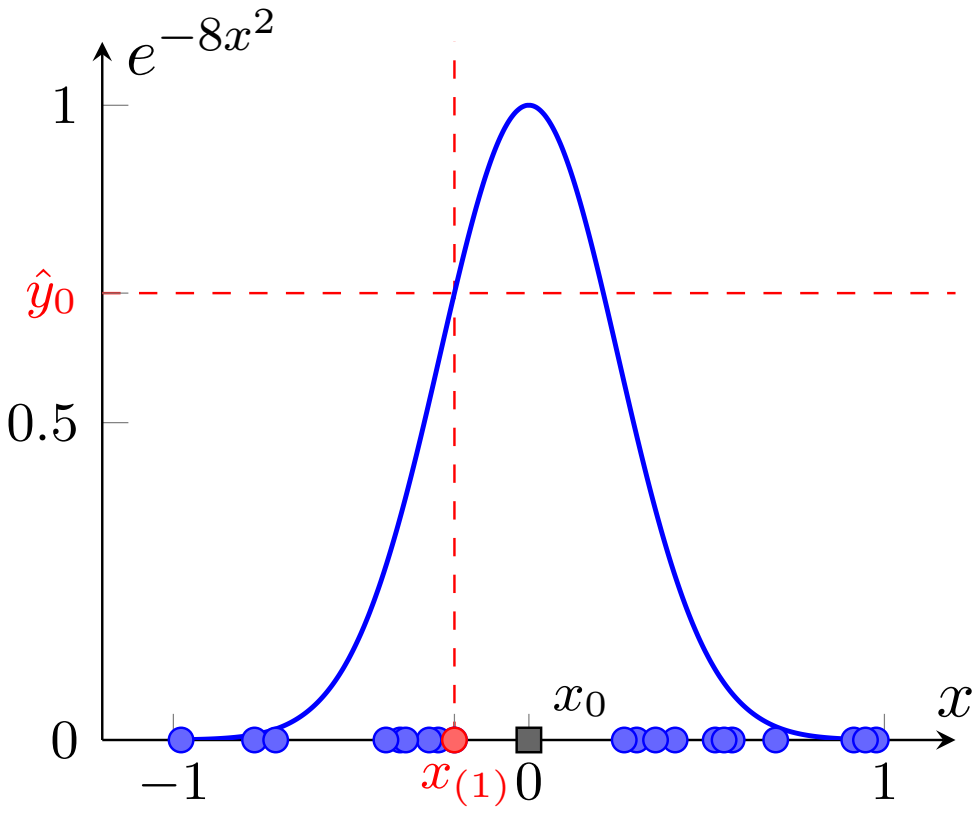
The Set-up



- Let $\mathcal{X} = [-1, 1]^p$ and have $n = 1000$ training examples x_i uniformly sampled from $\mathcal{X}$.
- The relationship between the inputs and output is defined by

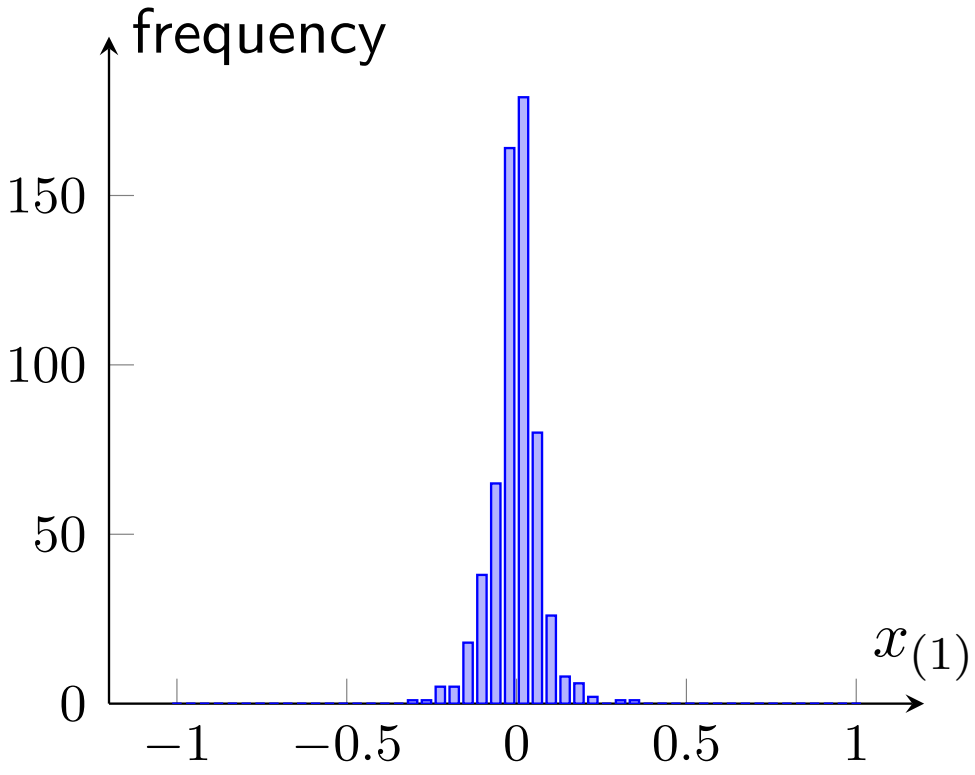
$$Y = f(X) = e^{-8\|X\|^2}$$

The regression method



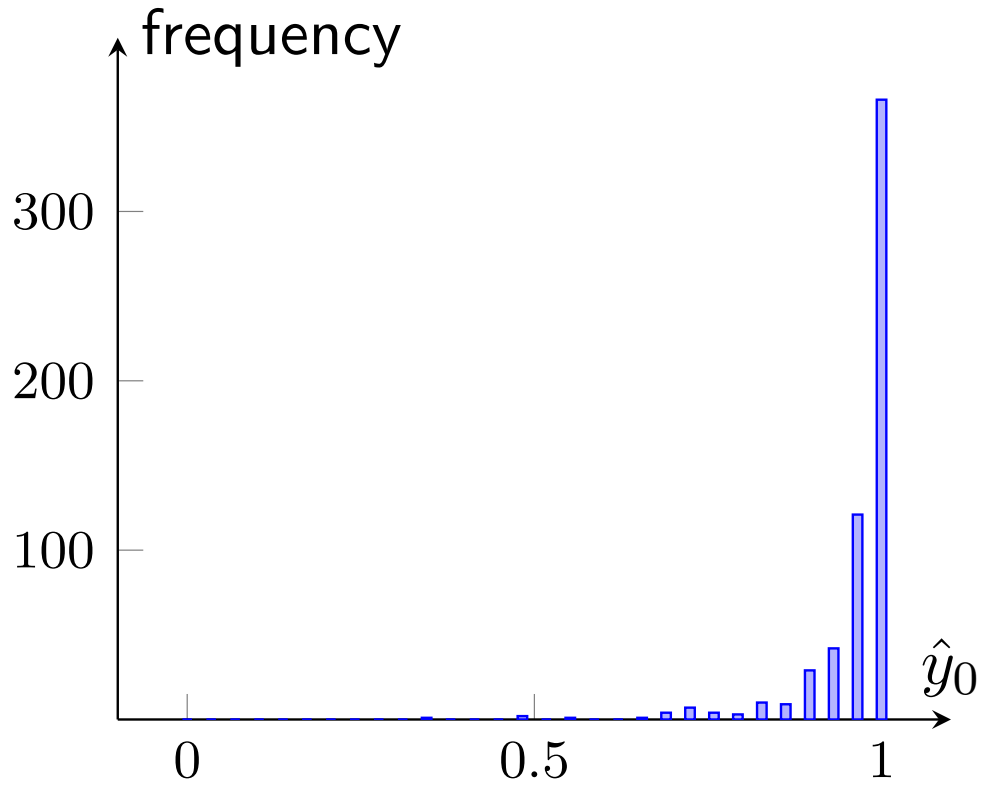
Use 1-nearest neighbour rule to predict y_0 at a test point x_0

Histogram of the position of nearest neighbour



$p = 1, n = 20$

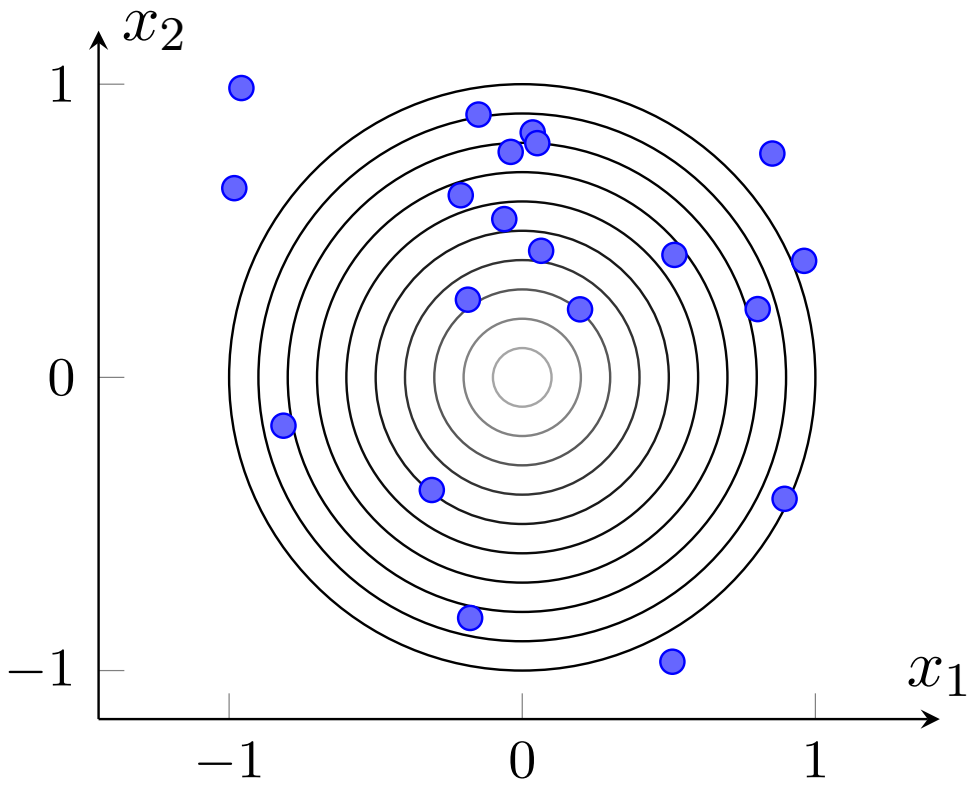
Average estimate of $\hat{y}_0$



$p = 1, n = 20, n_{\text{trial}} = 400$

Note: True value is $y = 1$

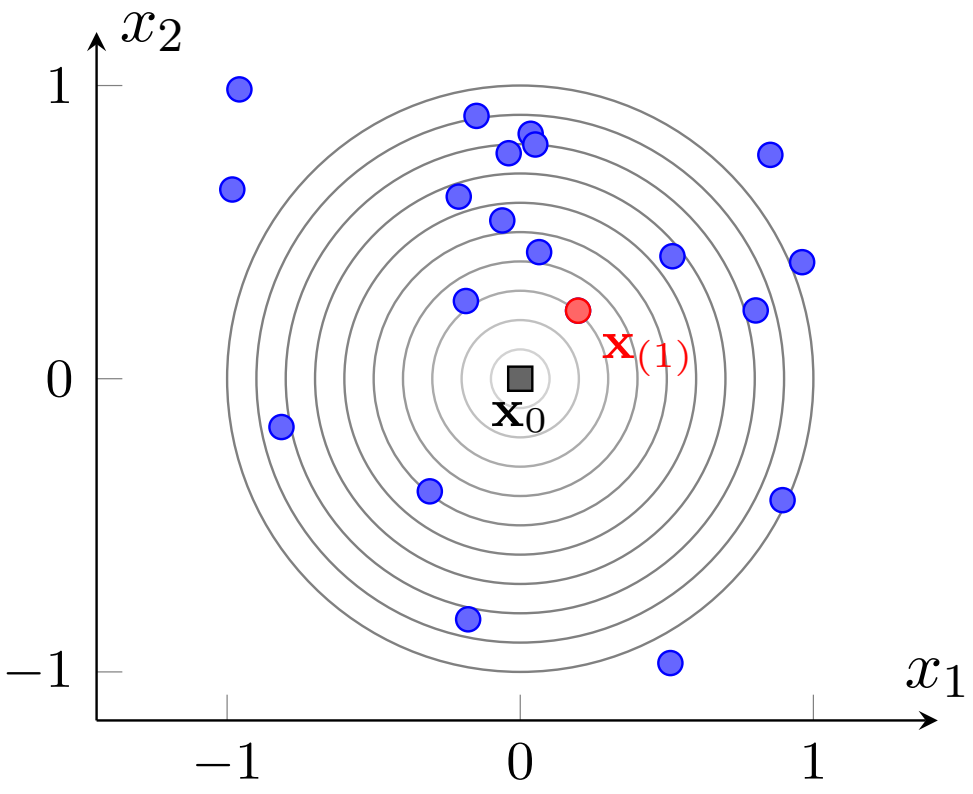
$p = 2$



- Let $\mathcal{X} = [-1, 1]^p$ and have $n = 1000$ training examples x_i uniformly sampled from $\mathcal{X}$.
- The relationship between the inputs and output is defined by

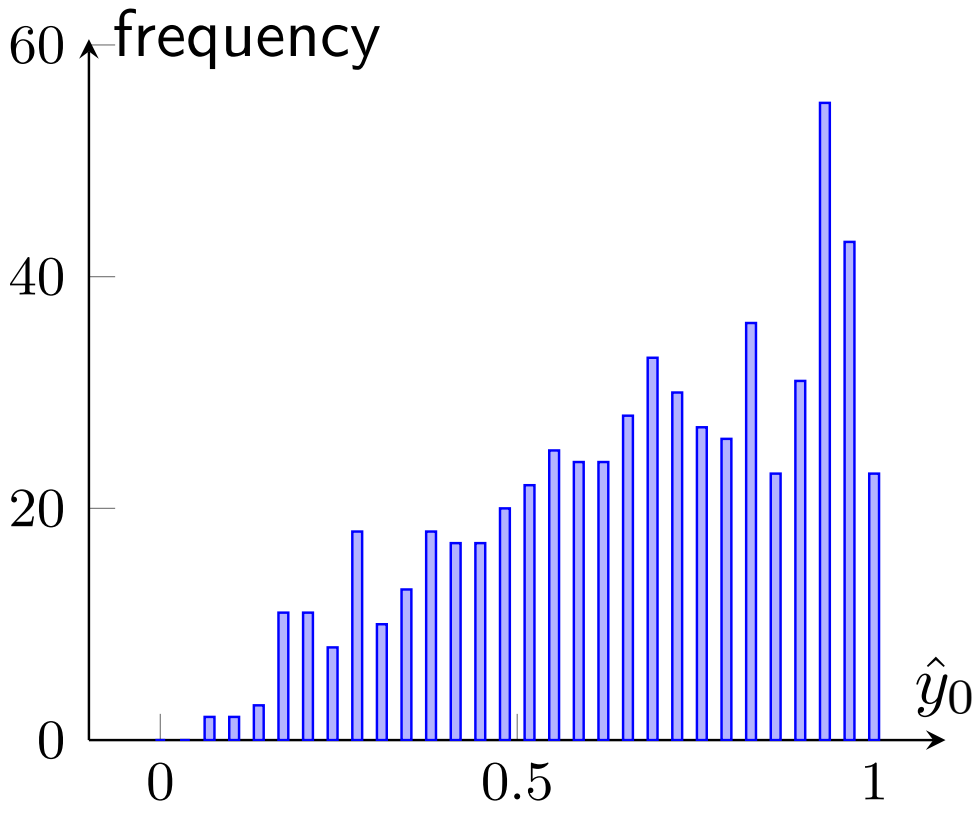
$$Y = f(X) = e^{-8\|X\|^2}$$

$p = 2$



Use 1-nearest neighbour rule to predict y_0 at a test point x_0

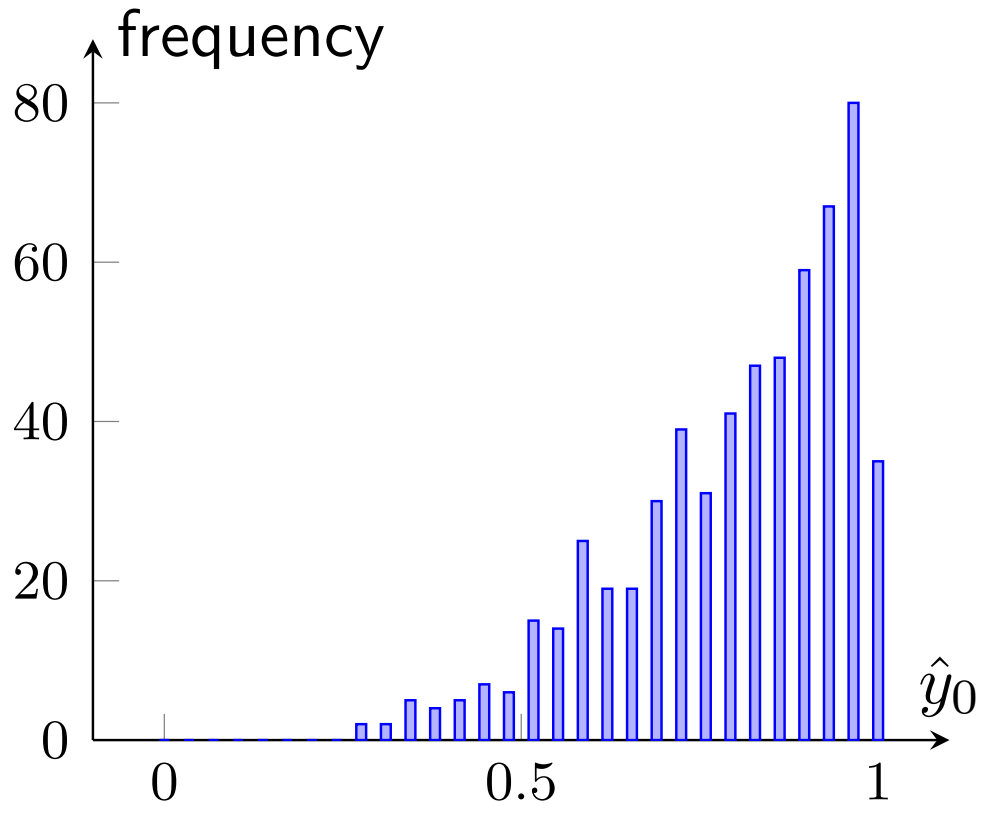
1-nn estimate of y_0



$p = 2, n = 20, n_{\text{trial}} = 600$

Note: True value is $y = 1$

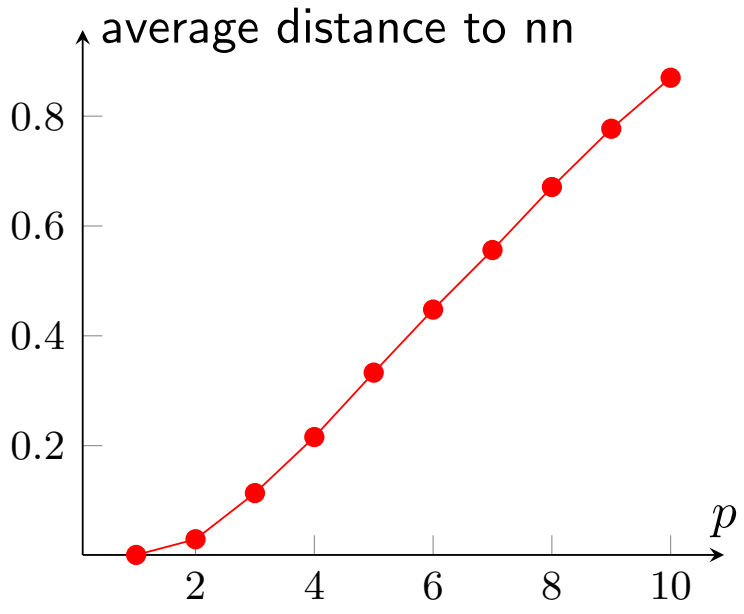
1-nn estimate of y_0

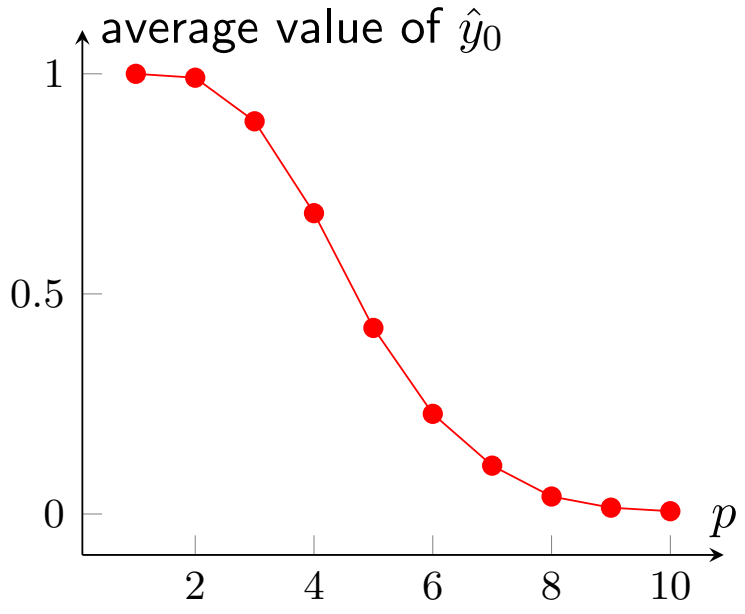


$p = 2, n = 40, n_{\text{trial}} = 600$

Note: True value is $y = 1$

As p increases





$n_{\text{train}} = 1000, n_{\text{trial}} = 400$

- average distance to nearest neighbour increases rapidly with p
- thus average estimate of $\hat{y}_0$ also rapidly degrades

Bias-Variance Decomposition

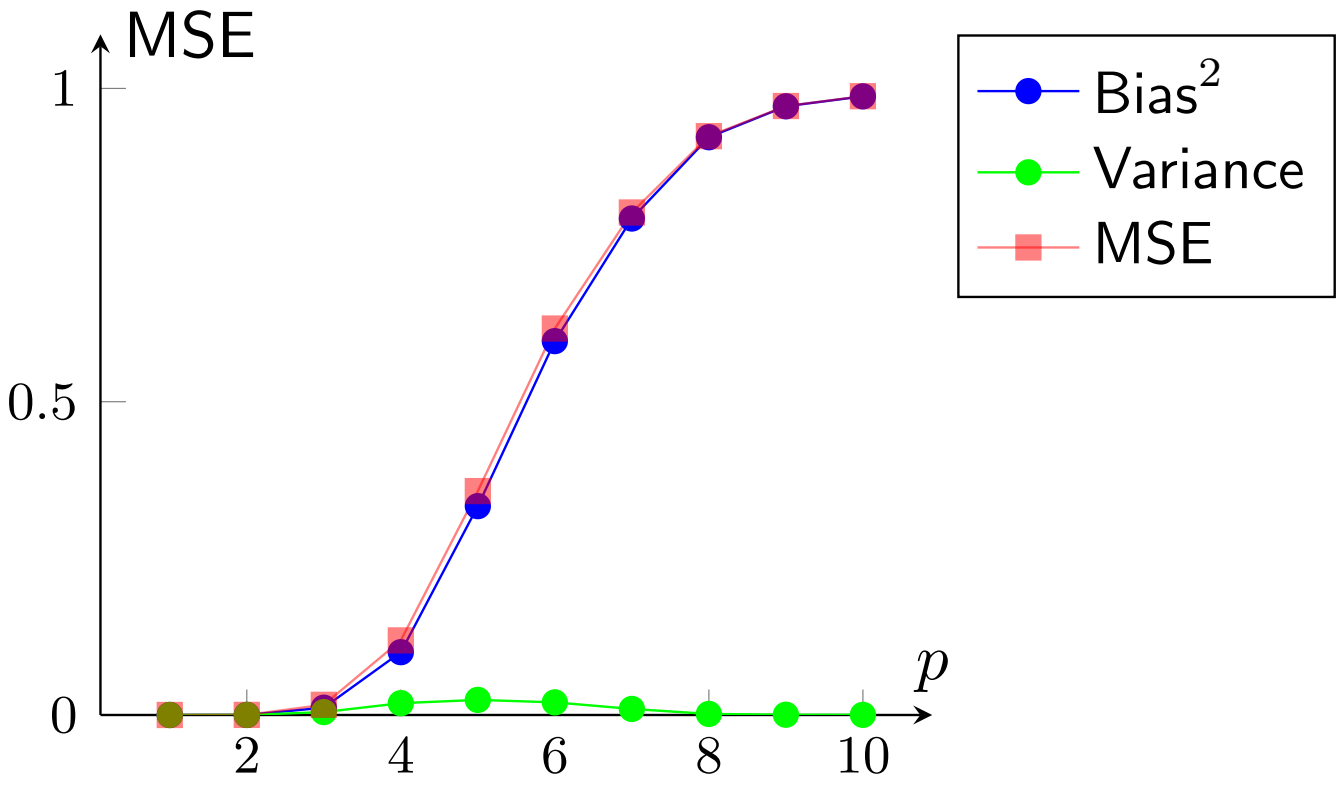
- For the simulation experiment have a completely deterministic relationship:

$$Y = f(X) = e^{-8\|X\|^2}$$

- Mean Squared Error for estimating $f(0)$ is

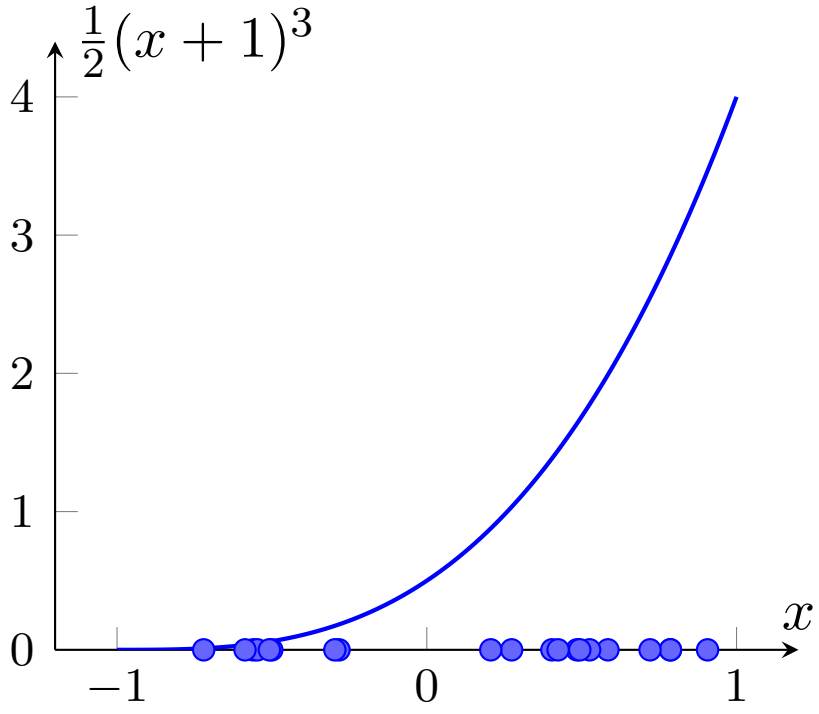
$$\begin{aligned}
\mathsf{MSE}(x_0) &= E_{\mathcal{T}}[(f(x_0) - \hat{y}_0)^2] \\
&= E_{\mathcal{T}}[(\hat{y}_0 - E_{\mathcal{T}}[\hat{y}_0])^2] + (E_{\mathcal{T}}[\hat{y}_0] - f(x_0))^2 \\
&= \mathsf{Var}_{\mathcal{T}}(\hat{y}_0) + \mathsf{Bias}^2(\hat{y}_0)
\end{aligned}$$

Bias-Variance Decomposition for this example



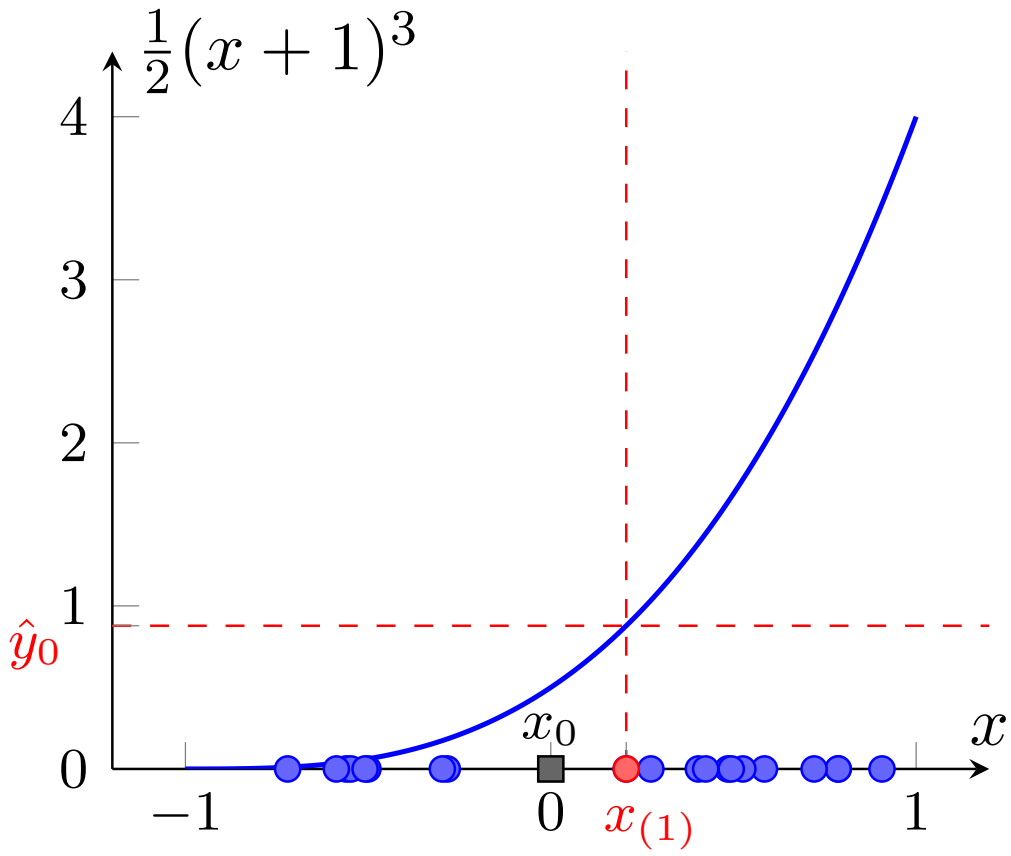
- The Bias dominates the MSE as p increases.
- **Why?**
  - As p increases the nearest neighbour is never close to $x_0 = 0$
  - Hence the estimate $\hat{y}_0$ tends to 0.

Another Simulated Example where variance dominates the MSE



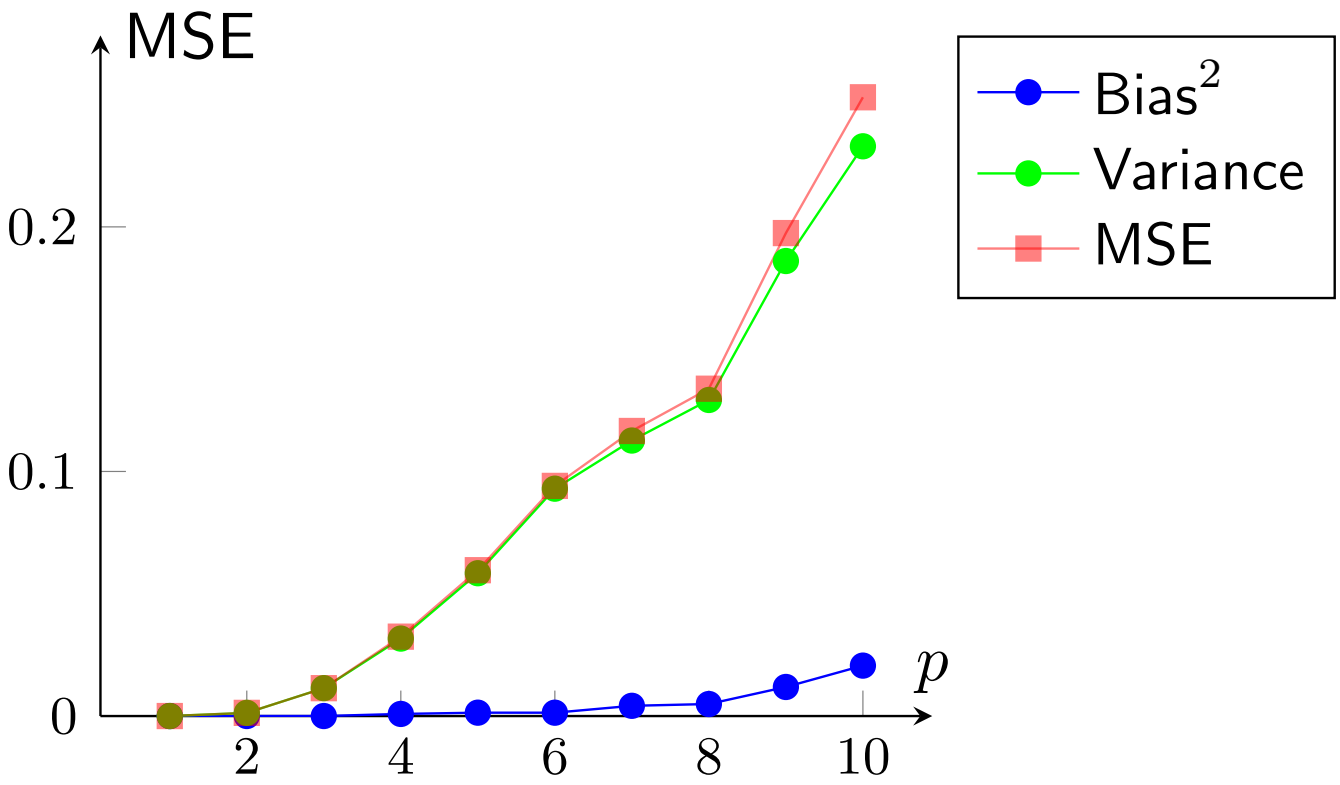
- Let $\mathcal{X} = [-1, 1]^p$ and have $n = 1000$ training examples x_i uniformly sampled from $\mathcal{X}$.
- The relationship between the inputs and output is defined by

$$Y = f(X) = \frac{1}{2}(X_1 + 1)^3$$



Use a 1-nn to estimate $f(x_0)$ where $x_0 = 0$.

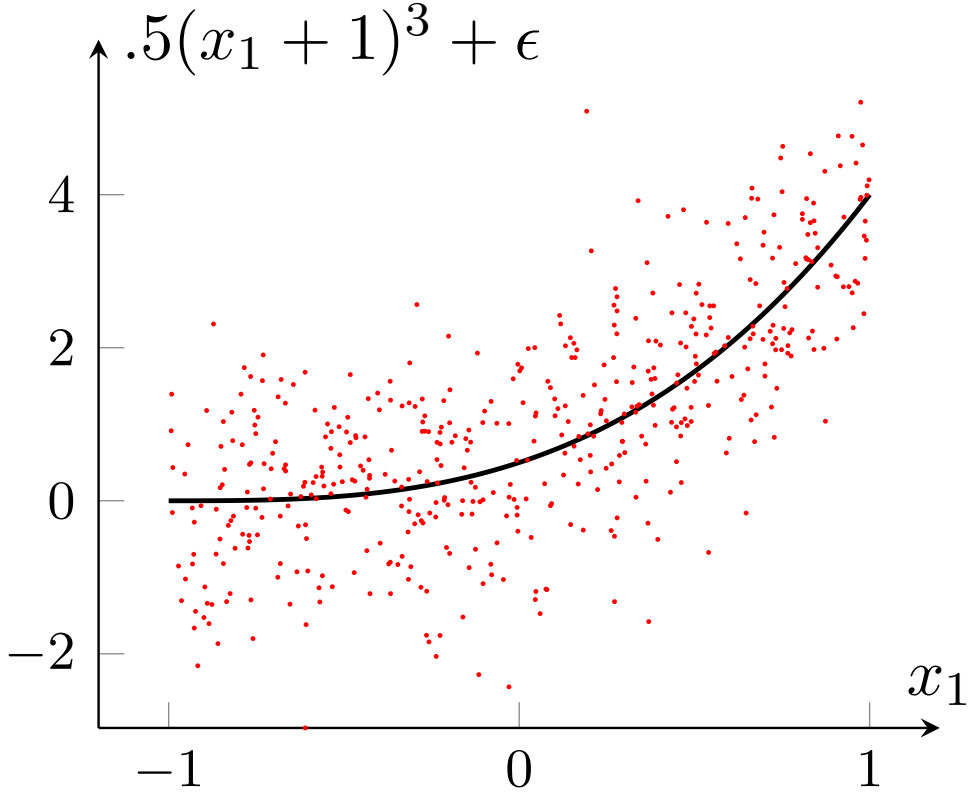
Variance dominates the MSE as p increases



- The variance dominates the MSE as p increases.
- **Why?**
  - as the deterministic function only involves one dimension the bias doesn't explode as p increases!

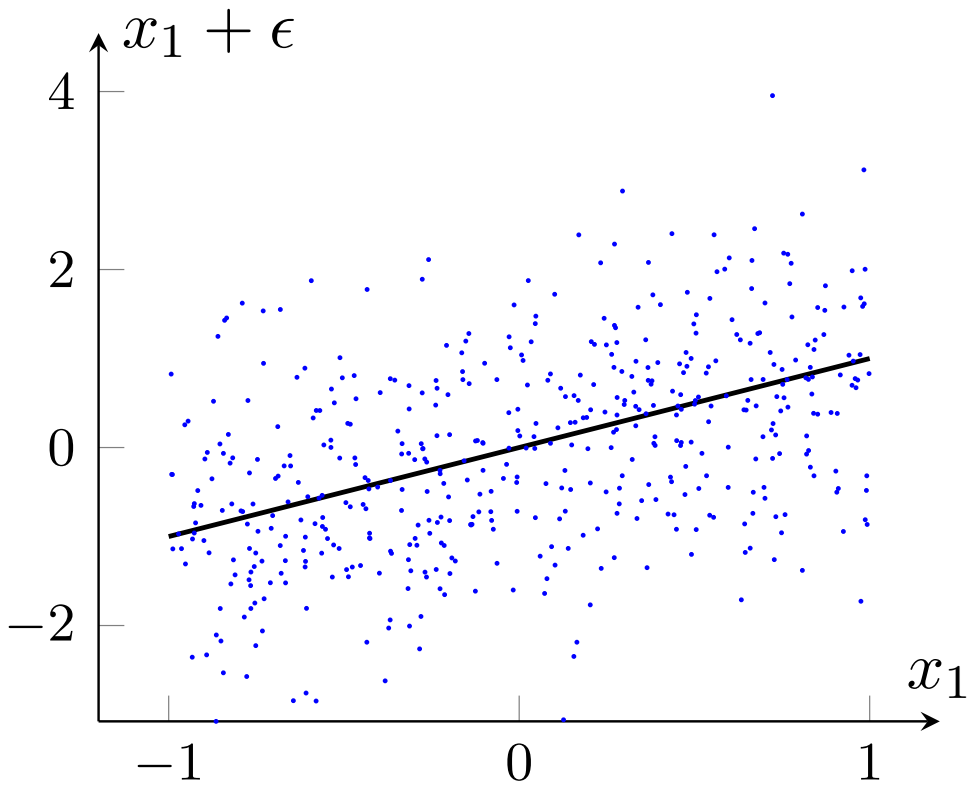
Comparison of Linear and NN predictors

Case 1



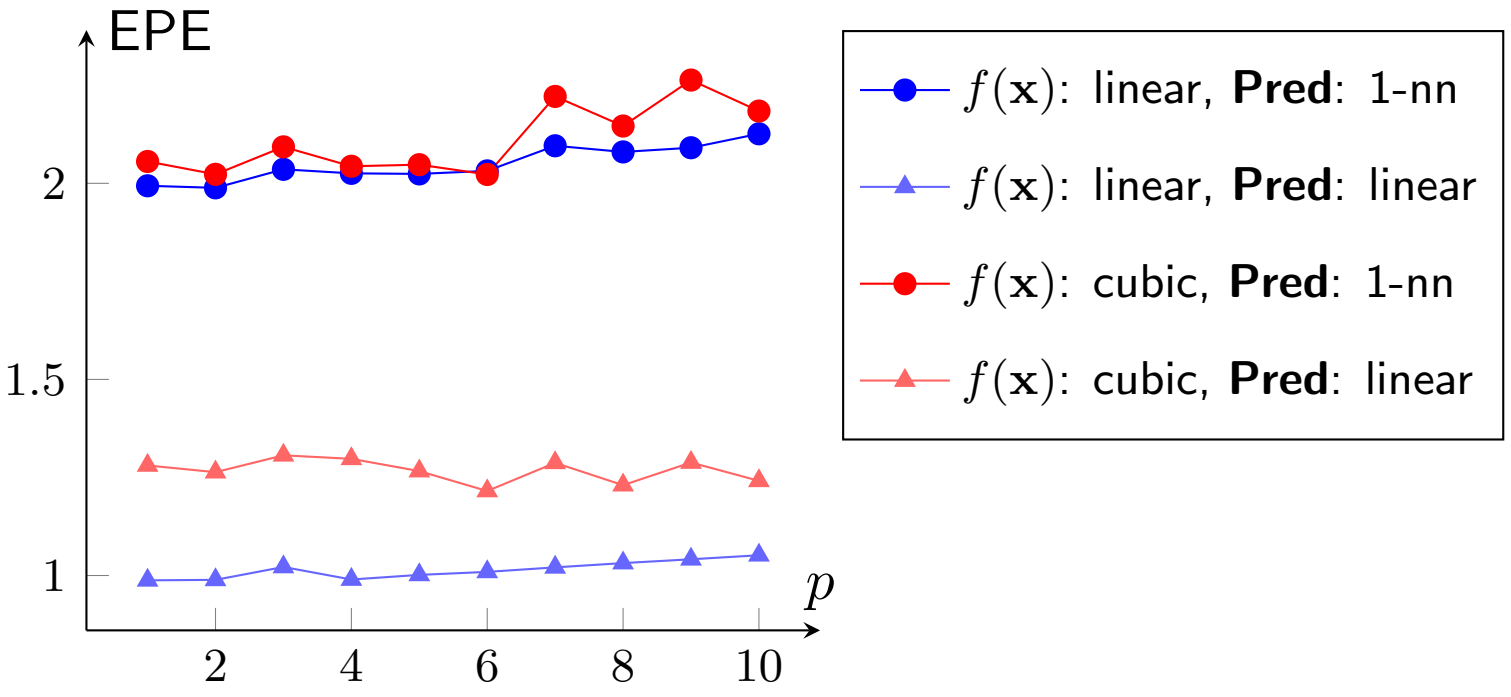
$$Y = .5(X_1 + 1)^3 + \epsilon, \quad \epsilon \sim N(0, 1)$$

Case 2



$$Y = X_1 + \epsilon, \quad \epsilon \sim N(0, 1)$$

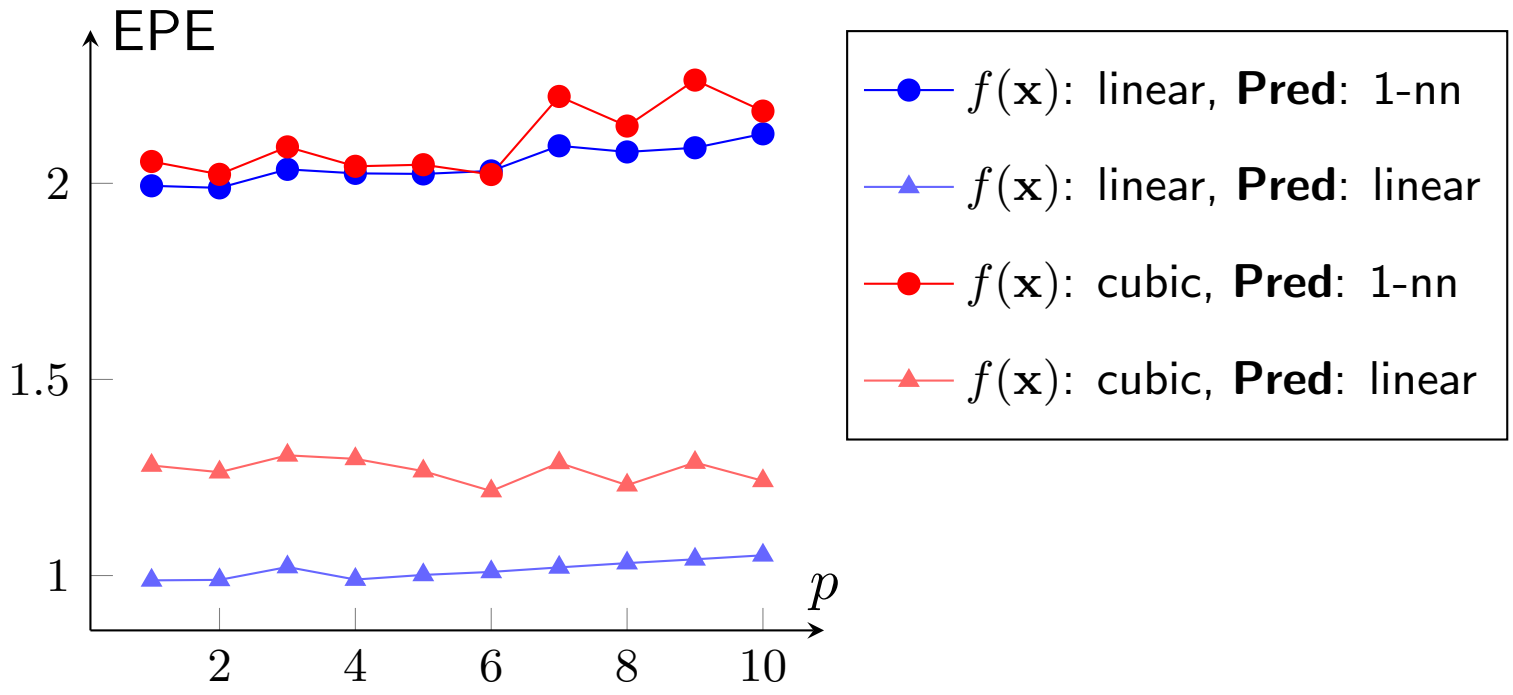
Linear predictor Vs 1-NN predictor



- EPE refers to the **expected prediction error** at point $x_0 = 0$

$$\text{EPE}(x_0) = E_{y_0|x_0}[\, E_{\mathcal{T}}[(y_0 - \hat{y}_0)^2]\,]$$

Linear predictor Vs 1-NN predictor



- The noise level destroys the 1-nn predictor
- linear predictor has a biased estimate of the cubic function
- linear predictor fits well even in the presence of noise and high dimension for the linear f
- linear model beats curse of dimensionality

Words of Caution

Case of horses for courses

- In previous example linear predictor out-performed the 1-nn regression function as

  bias of linear predictor $\ll$ variance of the 1-nn predictor

- But could easily manufacture and example where

  bias of linear predictor $\gg$ variance of the 1-nn predictor

More predictors than linear and NN

- There are a whole hosts of models in between the rigid linear model and the extremely flexible 1-nn method
- Each one has it own assumptions and biases
- Many are specifically designed to avoid the exponential growth in complexity of functions in high dimensions.

Statistical models,

Supervised learning and

Function approximation

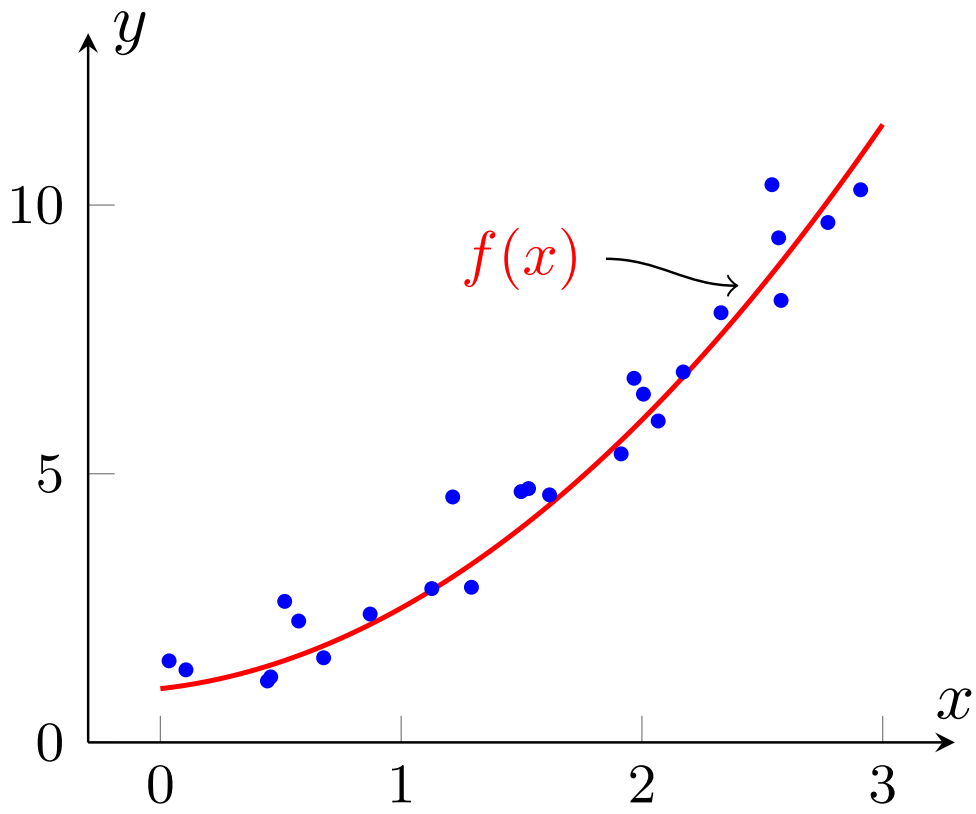
Goal

- Know there is a function $f(x)$ relating inputs to outputs:

$$Y \approx f(X)$$

- Want to find an estimate $\hat{f}(x)$ of $f(x)$ from labelled training data.
- This is difficult when X is high dimensional
- In this case need to incorporate special structure
  - reduce the bias and variance of the estimates
  - help combat the **curse of dimensionality**

A Statistical Model for Regression



$$Y = f(X) + \epsilon$$

output

deterministic relationship

random variable indept of input X

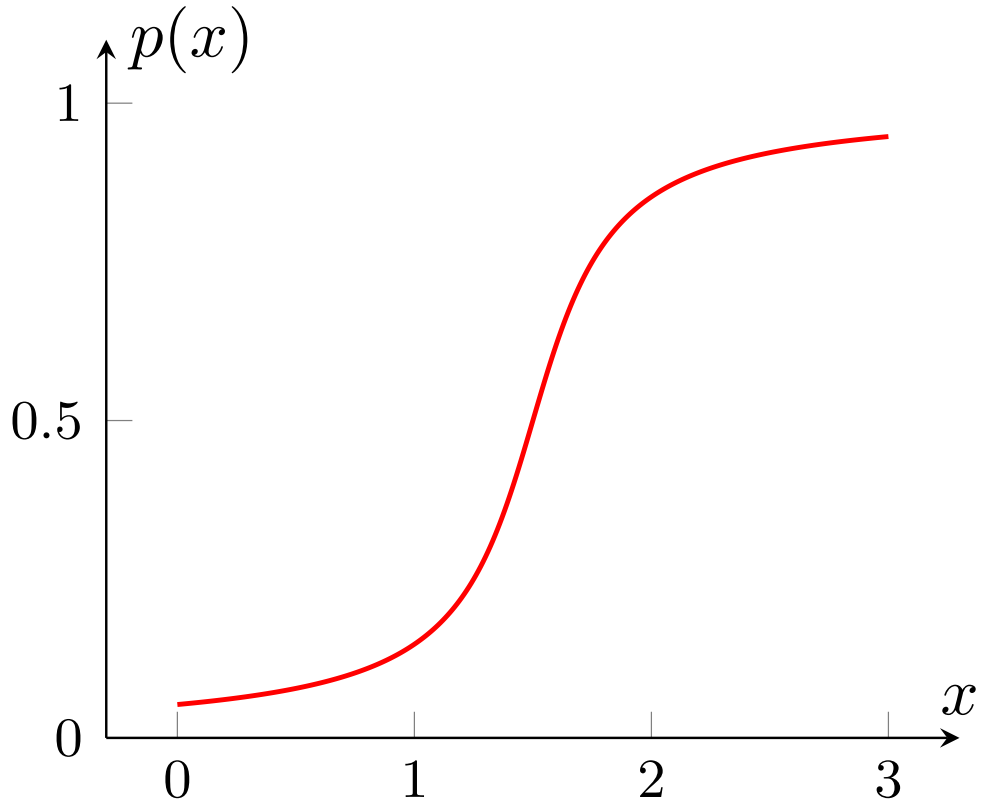
Additive Error Model

$$Y = f(X) + \epsilon$$

where

- the random variable ϵ has $E[\epsilon] = 0$
- ϵ is independent of X
- $f(x) = E[Y|X = x]$
- any departures from the deterministic relationship are mopped up by ϵ

Statistical model for binary classification



$p(G|X = x)$ is modelled as a Bernoulli distribution with

$$p(x) = p(G = 1|X = x)$$

Therefore

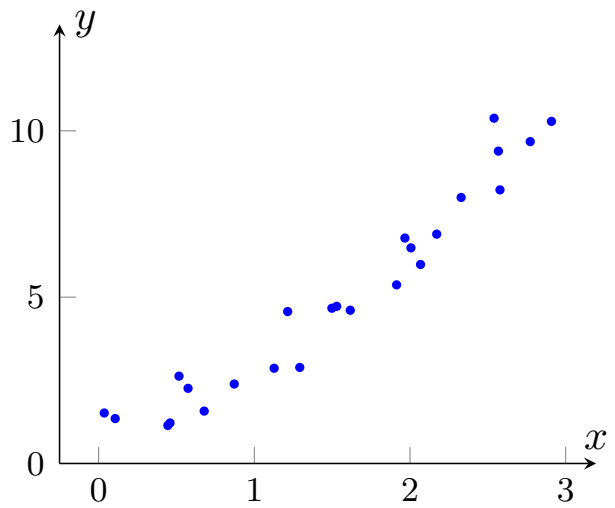
$$E[G|X = x] = p(x) \quad \textbf{and} \quad \text{Var}[G|X = x] = p(x)(1 - p(x))$$

Supervised Learning - Function Approximation

- Have training data

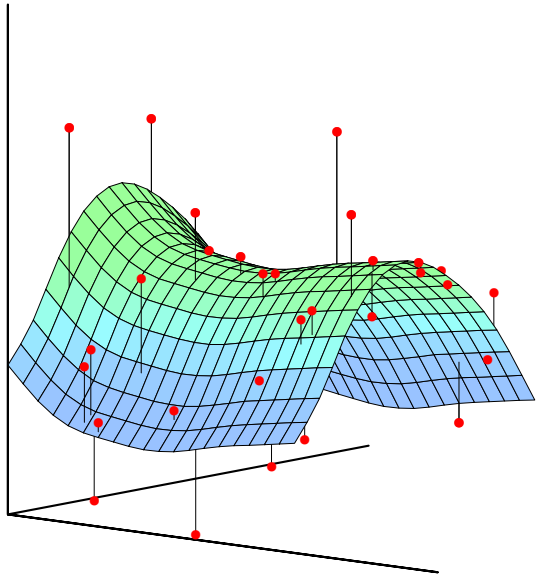
$$\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

where each $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$.



- Learn deterministic relationship f between X and Y from $\mathcal{T}$.
- In book **Supervised Learning** is viewed as a problem in function approximation.

Common approach



- Decide on parametric form of f_θ, i.e. *linear basis expansion*

$$f_\theta(x) = \sum_{m=1}^{M} h_m(x)\,\theta_m$$

- Use least squares to estimate θ in by minimizing

$$\mathsf{RSS}(\theta) = \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2$$

Don't have to always use least squares

- Can find θ by optimizing other criteria.
- Another option is **Maximum Likelihood Estimation**
- For the additive model, $Y = f_\theta(X) + \epsilon$ have

$$P(Y|X, \theta) = N(f_\theta(X), \sigma^2)$$

- Log-likelihood of the training data is

$$\begin{aligned} L(\theta) &= \sum_{i=1}^{n} \log P(Y = y_i|X = x_i, \theta) \\ &= \sum_{i=1}^{n} \log \left(N(y_i; f_\theta(x_i), \sigma^2) \right) \end{aligned}$$

- Find the θ that minimizes $L(\theta)$

Structured Regression Models

Why do we need structure?

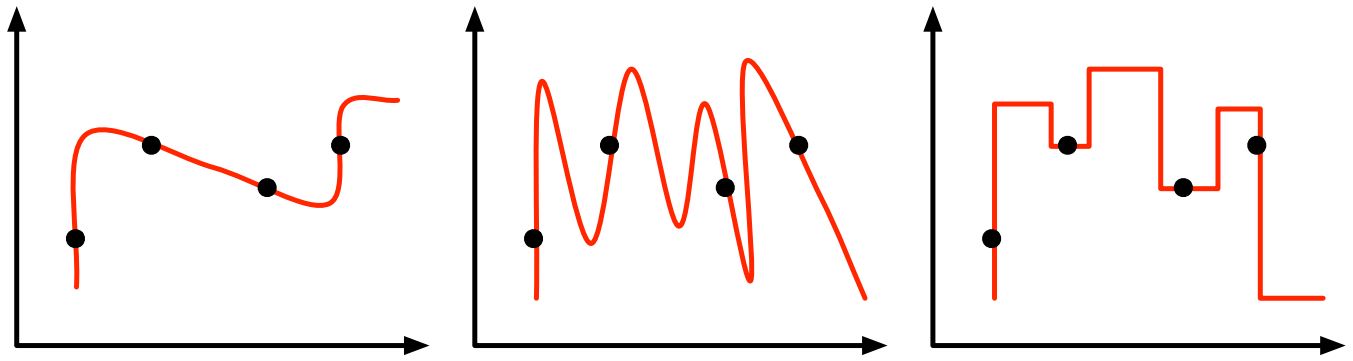
- Consider the *Residual Sum of Squares* for a function f

$$\mathsf{RSS}(f) = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- There are infinitely many $\hat{f}$ with

$$\hat{f} = \arg\min_f \mathsf{RSS}(f) \quad \textbf{and} \quad \mathsf{RSS}(\hat{f}) = 0$$

Why do we need structure?



- Any function $\hat{f}$ passing through the training points (x_i, y_i) is a solution.
- Obviously not all the $\hat{f}$ will be equally good at predicting the value of unseen test points...

Must restrict the class of f considered

- Don't consider and arbitrary function $\hat{f}$,
- Instead restrict ourselves to $\hat{f} \in \mathcal{F}$

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \text{RSS}(f)$$

- But what restrictions should be used....
- Initial ambiguity in choosing $\hat{f}$ has just been transferred to choice of constraint.

Must restrict the class of f considered

- Don't consider and arbitrary function $\hat{f}$,
- Instead restrict ourselves to $\hat{f} \in \mathcal{F}$

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \text{RSS}(f)$$

- But what restrictions should be used....
- Initial ambiguity in choosing $\hat{f}$ has just been transferred to choice of constraint.

Options to restrict the class of f

- Have a parametric representation of f_θ
  - Linear model: $f_\theta(x) = \theta_1^t\, x + \theta_0$
  - Quadratic: $f_\theta(x) = x^t \Theta\, x + \theta_1^t\, x + \theta_0$
- Impose **complexity** restrictions on the function.
  - i.e. $\hat{f}$ must have some regular behaviour in small neighbourhoods of the input space, but then
  - What size should the neighbourhood be?
  - What form should f have in the neighbourhood?
- No unique way to impose complexity constraints

Complexity and Neighbourhood size

- Large neighbourhood $\Longrightarrow$ strong constraint
- Small neighbourhood $\Longrightarrow$ weak constraint

Classes of Restricted Estimators

How to restrict the predictor $\hat{f}$

- The techniques used to restrict the regression or classification function learned loosely fall into several classes.
- Each class has parameter(s) termed **smoothing** parameters which control the effective size of the local neighbourhood.
- Some examples from each class follow.

How to restrict the predictor $\hat{f}$

- The techniques used to restrict the regression or classification function learned loosely fall into several classes.
- Each class has parameter(s) termed **smoothing** parameters which control the effective size of the local neighbourhood.
- Some examples from each class follow.

Note:

- It is assumed we have training examples $\{(x_i, y_i)\}_{i=1}^{n}$ **and**
- We present the energy functions or functionals which are minimised in order to find $\hat{f}$

Class 1: Roughness Penalty

ensure f predicts the training values

penalty parameter

$$\text{PRSS}(f, \lambda) = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \; J(f)$$

functional measuring smoothness of f

- One such penalty functional is

$$J(f) = \int [f''(x)]^2 dx$$

  For wiggly f's this functional will have a large value while for linear f's it is zero.

- **Regularization** methods express our belief that the f we're trying to approximate has a certain smoothness properties.

Class 2: Kernel Methods and Local Regression

- Estimate the regression or classification function in a local neighbourhood.
- Need to specify
  - the nature of local neighbourhood
  - the class of functions used in local fit

Kernel Methods and Local Regression

- Can define a local regression estimate of $f(x_0)$, from training data $\{(x_i, y_i)\}$, as $f_{\hat{\theta}}(x_0)$ where $\hat{\theta}$ minimizes

$$\text{RSS}(f_\theta, x_0) = \sum_{i=1}^{n} K_\lambda(x_0, x_i)(y_i - f_\theta(x_i))^2$$

  where
  - **Kernel function:** $K_\lambda(x_0, x_i)$ assign weights to x_i depending on its closeness to x_0.
  - **Base regression function:** f_θ is a parameterized function such as a low order polynomial.

- A common kernel is the Gaussian kernel

$$K_\lambda(x_0, x) = \frac{1}{\lambda} \exp\left[-\frac{\|x_0 - x\|^2}{2\lambda}\right]$$

Class 3: Basis functions and Dictionary methods

- f is modelled as a linear expansion of basis functions

$$f_\theta(x) = \sum_{m=1}^{M} \theta_m \, h_m(x)$$

- Each h_m is a function of the input x.
- Linear refers to the actions of the θ parameters.

Example 1: Radial Basis Functions

$$f_\theta(x) = \sum_{m=1}^{M} K_\lambda(\mu_m, x)\, \theta_m$$

where

- $K_{\lambda_m}(\mu_m, x)$ is a symmetric kernel centred at location μ_m.
- the Gaussian kernel is a popular kernel to use

$$K_\lambda(\mu_m, x) = \exp(-\|\mu_m - x\|^2/(2\lambda))$$

- If μ_m's and λ_m's pre-defined $\implies$ estimating θ a linear problem.
- However, if μ_m's and λ_m's **not pre-defined** $\implies$ estimating θ, λ_m's and μ_m's is a hard non-linear problem.

Example 2: Adaptive basis function method

$$f_\theta(x) = \sum_{m=1}^{M} \beta_m \, \sigma(\alpha_m^t \, x + b_m)$$

where

- $\theta = (\beta_1, \ldots, \beta_M, \alpha_1, \ldots, \alpha_M, b_1, \ldots, b_m)^t$
- $\sigma(z) = 1/(1 + e^{-z})$ is the **activation** function.
- The directions α_m and bias terms b_m have to be determined and estimating them is the core of the estimation.

Dictionary methods

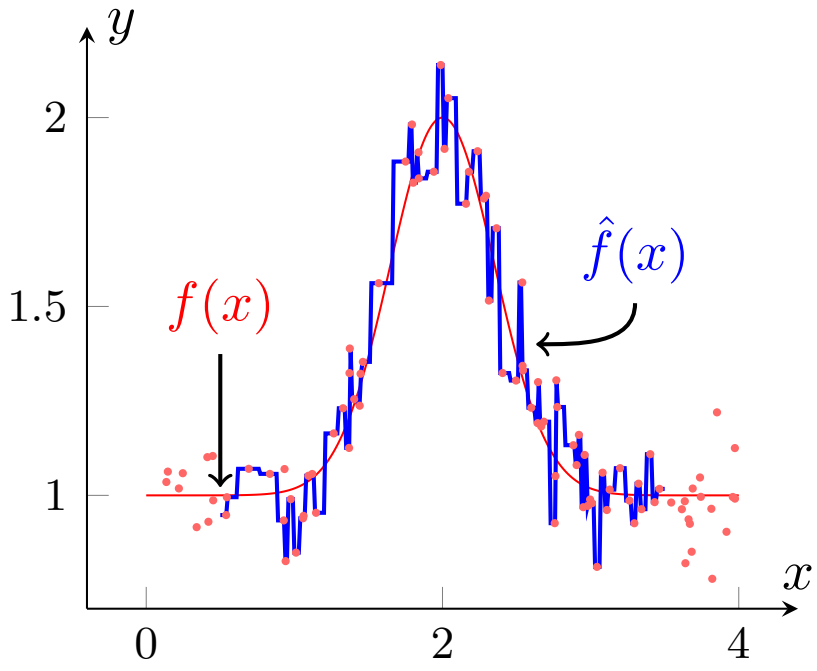
- Adaptively chosen basis function methods aka **dictionary methods**
- Challenge is to choose a number of basis functions from a dictionary set $\mathcal{D}$ of candidate basis functions (possibly infinite).
- Models are built up by employing some kind of search mechanism

Model Selection and, the Bias-Variance Trade-off

The complexity of learnt function

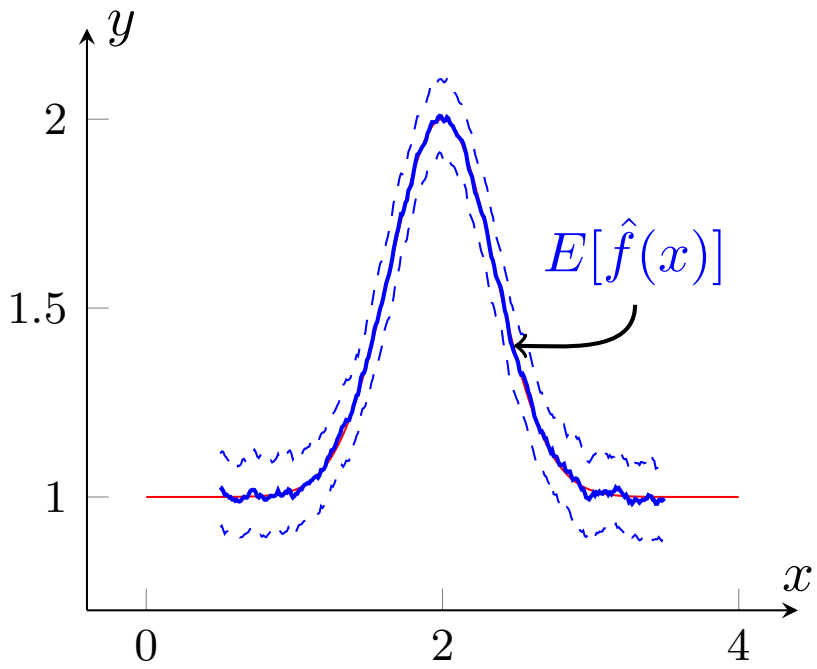
- Many models have a parameter which control its complexity.
- We have seen examples of this
  - k - number of nearest neighbours (*nearest neighbour classifier*)
  - σ - width of the kernel (*radial basis functions*)
  - M - number of basis functions (*dictionary methods*)
  - λ - weight of the penalty term (*spline fitting*)
- How does *increasing* or *decreasing* the **complexity** of the model affect their predictive behaviour?

Consider the nearest neighbour regression fit



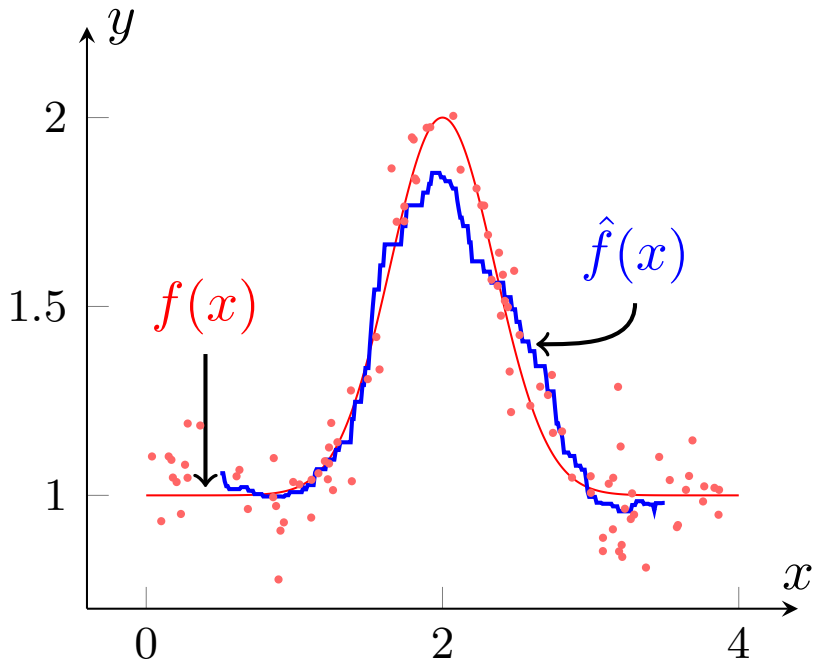
- Approximate $f(x)$ with 1-nn regression fit $\hat{f}_1(x)$ given $\{(x_i, y_i)\}_{i=1}^n$ and $n = 100$.
- Each training example is $y_i = f(x_i) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$ and $\sigma = .1$

Expected predictor when $k = 1$



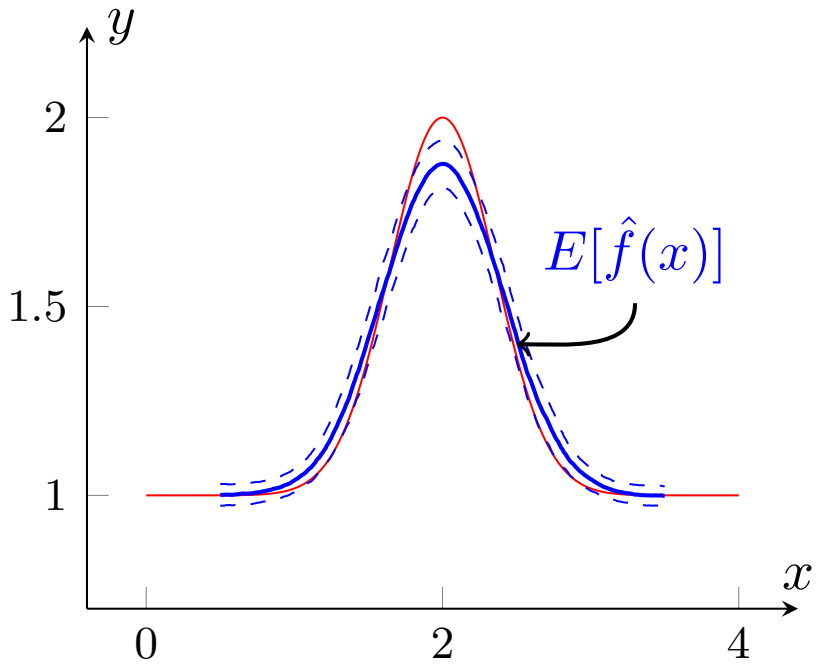
- Shown above is the expected prediction of the 1-nn regression fit given $n = 100$ and $\sigma = .1$
- $E[\hat{f}_1(x)]$ is a good approximation to $f(x)$. **There is no bias!**
- At each x one std of the estimate is shown. Note its magnitude.

15-nn regression fit



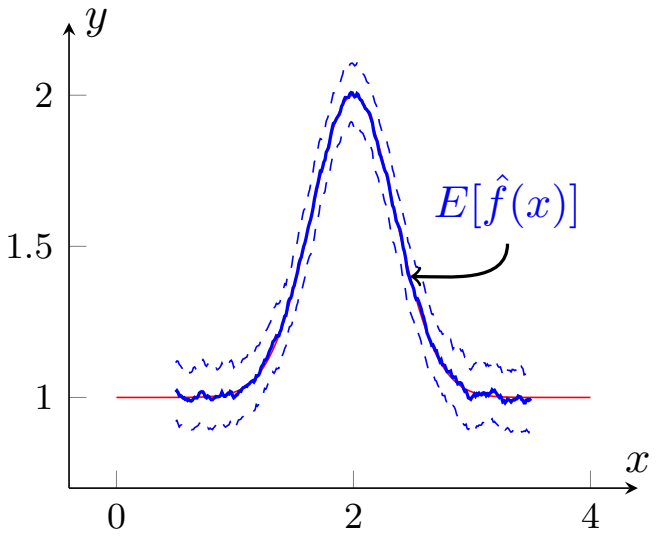
- Approximate $f(x)$ with 15-nn regression fit $\hat{f}_{15}(x)$ given $\{(x_i, y_i)\}_{i=1}^n$ and $n = 100$.
- Each training example is $y_i = f(x_i) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$ and $\sigma = .1$

Expected predictor when $k = 15$

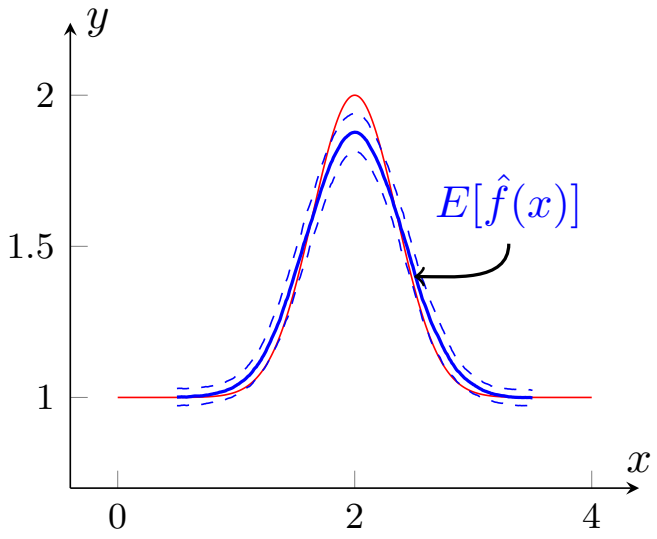


- $E[\hat{f}_{15}(x)]$ is smooth but **biased**.
- Compare the peak of $f(x)$ and $E[\hat{f}_{15}(x)]$!
- **Note** the variance of estimate is much smaller than when $k = 1$.

Have illustrated the **Bias-Variance** trade-off



High complexity: $k = 1$



Lower complexity: $k = 15$

- Model **complexity increased**, the variance tends to increase and the squared bias tends to decrease.
- Model **complexity is decreased**, the variance tends to decrease, but the squared bias tends to increase.

What not to do:

- Want to choose model complexity which minimizes test error.
- Training error is one estimate of the test error.
- Could choose the model complexity that produces the predictor which minimizes the training error.
- **Not a good idea!**

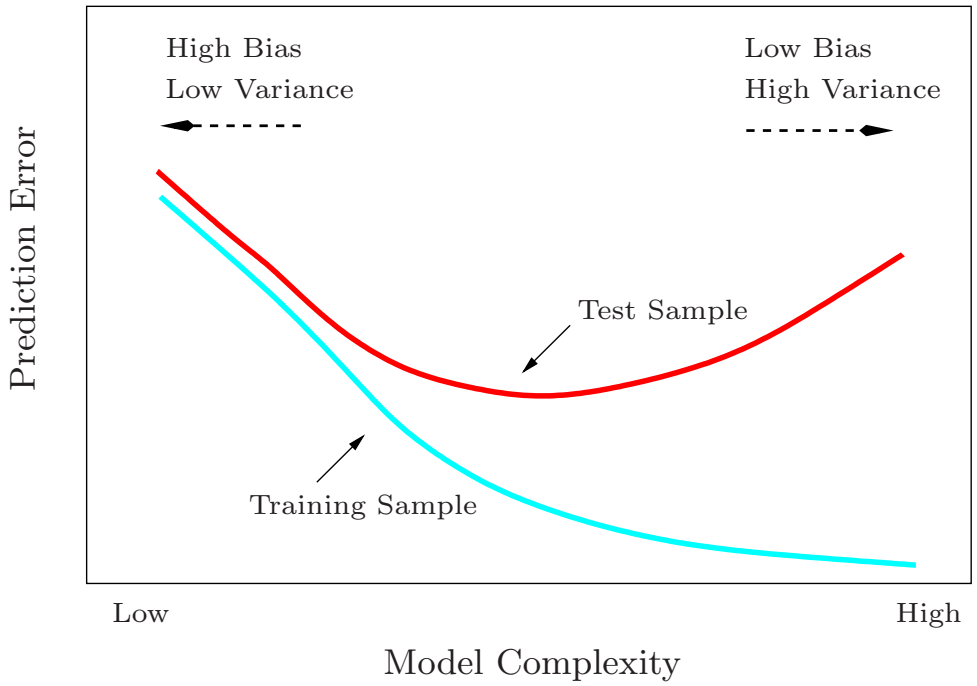
What not to do:

- Want to choose model complexity which minimizes test error.
- Training error is one estimate of the test error.
- Could choose the model complexity that produces the predictor which minimizes the training error.
- **Not a good idea!**

Why??

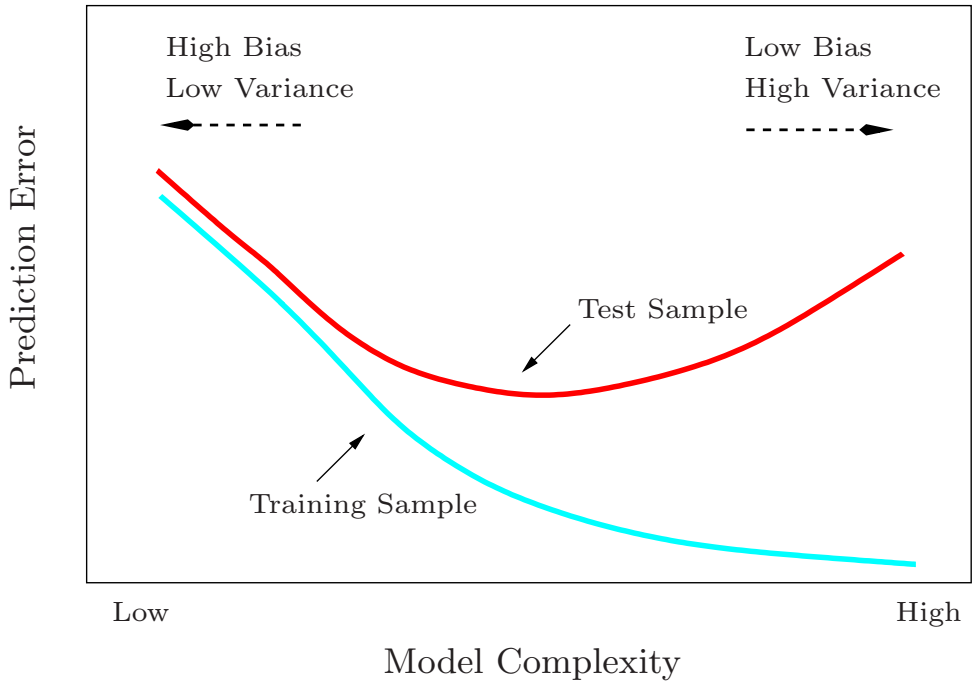
Training error decreases when model complexity increases

Overfitting



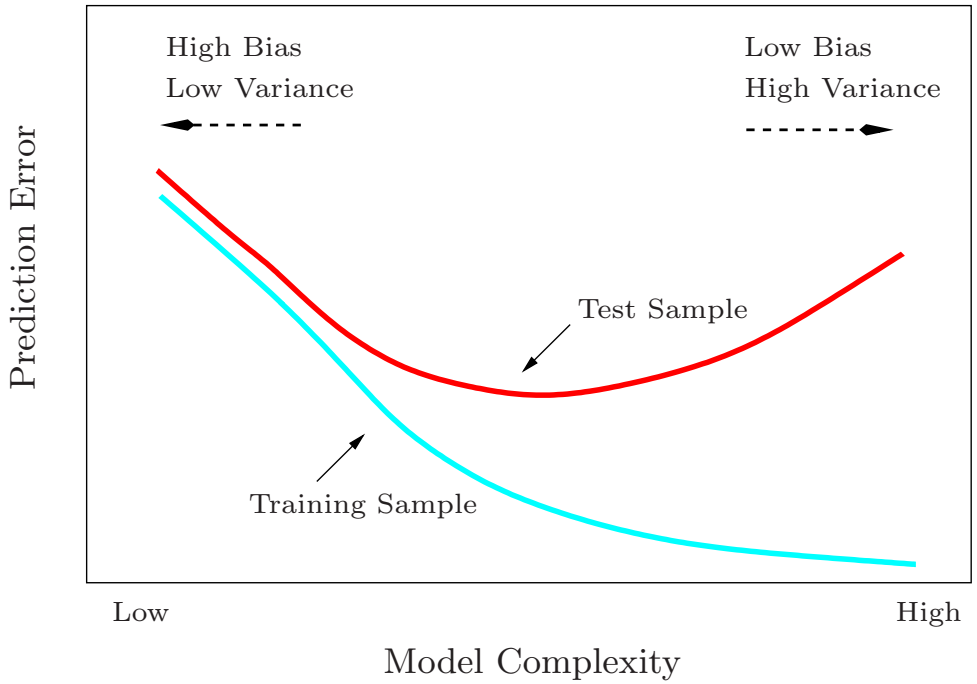
- Too much fitting $\Longrightarrow$ adapt too closely to the training data
- Have a high variance predictor
- This scenario is termed **overfitting**
- In such cases predictor loses the ability to generalize

Underfitting



- Low complexity model $\Longrightarrow$ predictor may have large bias
- Therefore predictor has poor generalization
- Latter on in the course will discuss how to overcome these problems.

Underfitting



- Low complexity model $\Longrightarrow$ predictor may have large bias
- Therefore predictor has poor generalization
- Latter on in the course will discuss how to overcome these problems.