Lecture 23: Review

Reading: All chapters.

GU4241/GR5241 Statistical Machine Learning

Linxi Liu
April 27, 2018

Optimization Problems

Terminology

An **optimization problem** for a given function $f : \mathbb{R}^d \to \mathbb{R}$ is a problem of the form

$$\min_{\mathbf{x}} \ f(\mathbf{x})$$

which we read as "find $\mathbf{x}_0 = \arg\min_{\mathbf{x}} \ f(\mathbf{x})$".
A **constrained optimization problem** adds additional requirements on $\mathbf{x}$,

$$\begin{aligned} &\min_{\mathbf{x}} \ f(\mathbf{x}) \\ \text{subject to} \quad &\mathbf{x} \in G \ , \end{aligned}$$

where $G \subset \mathbb{R}^d$ is called the **feasible set**. The set G is often defined by equations, e.g.

$$\begin{aligned} &\min_{\mathbf{x}} \ f(\mathbf{x}) \\ \text{subject to} \quad &g(\mathbf{x}) \geq 0 \end{aligned}$$

Convex Optimization

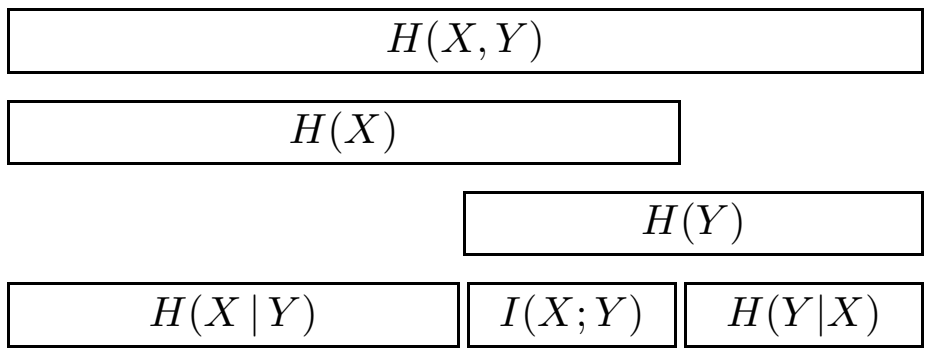
- Unconstrained problems:
  1. Gradient descent
  2. Newton's method
- Constrained problems:
  1. Equality constraints
  2. Inequality constraints

Information Theory

Check the definitions:

- Entropy, conditional entropy
- KL divergence
- Mutual information

The various additive relationships can be summarized as follows:



Unsupervised learning

- In unsupervised learning, all the variables are on equal standing, no such thing as an input and response.
- **Two sets of methods:**
  1. Dimensionality reduction
     - PCA
     - Non-linear dimensionality reduction methods
  2. Clustering: find meaningful groups of samples
     - Hierarchical clustering (single, complete, or average linkage).
     - K-means clustering.
     - EM algorithm
     - Spectral clustering

PCA

1. Find the linear combination of variables

$$\theta_{11}X_1 + \theta_{12}X_2 + \cdots + \theta_{1p}X_p$$

   with $\sum_i \theta_{1i}^2 = 1$, which has the largest variance.

2. Find the linear combination of variables

$$\theta_{21}X_1 + \theta_{22}X_2 + \cdots + \theta_{2p}X_p$$

   with $\sum_i \theta_{2i}^2 = 1$ and $\theta_1 \perp \theta_2$, which has the largest variance.

3. ...

PCA

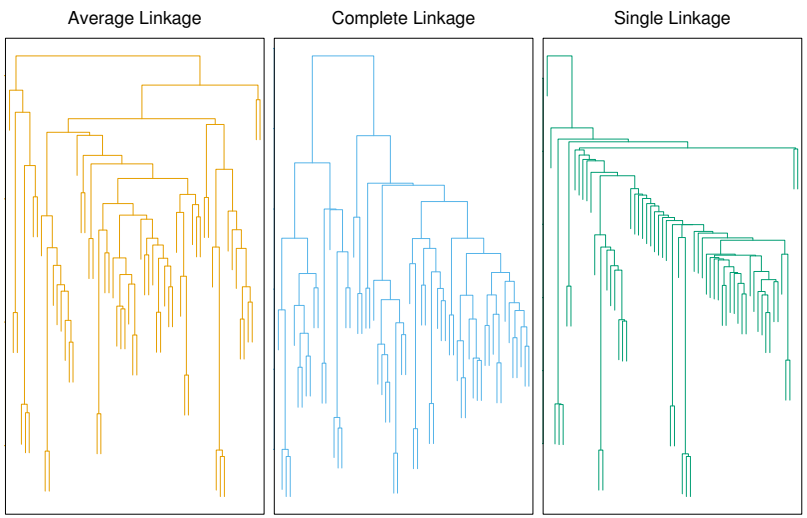
Some questions:

- What are the loadings?
- What are score variables?
- What is a biplot, how is it interpreted?
- What is the proportion of variance explained? A scree plot?
- What is the effect of rescaling variables?

K-means clustering

- The number of clusters is fixed at K.
- Goal is to minimize the average distance of a point to the average of its cluster.
- The algorithm starts from some assignment, and is guaranteed to decrease this average distance.
- This find a local minimum, not necessarily a global minimum, so we typically repeat the algorithm from many different random starting points.

Hierarchical clustering



- Agglomerative algorithm produces a *dendrogram*.
- At each step we join the two clusters that are "closest":
  - **Complete:** distance between clusters is maximal distance between any pair of points.
  - **Single:** distance between clusters is minimal distance.
  - **Average:** distance between clusters is the average distance.
- Height of a branching point = distance between clusters joined.

EM algorithm

Latent variables as auxiliary information

If we knew the correct assignments m_i, we could:

- Estimate each component distribution $p(x|\theta_k)$ separately, using only the data assigned to cluster k.
- Estimate the cluster proportions c_k as $\hat{c}_k := \frac{\#\text{points in cluster } k}{n}$.

EM algorithm: Idea

The EM algorithm estimates values of the latent variables to simplify the estimation problem. EM altnernates between two steps:

1. Estimate assignments m_i given current estimates of the parameters c_k and θ_k ("E-step").
2. Estimate parameters c_i and θ_k given current estimates of the assignments ("M-step").

These two steps are iterated repeatedly.

Representation of Assignments

We re-write the assignments as vectors of length K:

$$\mathbf{x}_i \text{ in cluster } k \qquad \text{as} \qquad M_i := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow k\text{th entry}$$

so $M_{ik} = 1$ if x_i in cluster k, and $M_{ik} = 0$ otherwise.
We collect the vectors into a matrix

$$\mathbf{M} = \begin{pmatrix} M_{11} & \dots & M_{1K} \\ \vdots & & \vdots \\ M_{n1} & \dots & M_{nK} \end{pmatrix}$$

Note: Rows = observations, columns = clusters
Row sums = 1, column sums = cluster sizes.

E-Step

Hard vs soft assignments

- The vectors M_i are "hard assignments" with values in $\{0, 1\}$.
- EM computes "soft assignments" a_{ik} with values in $[0, 1]$.
- Once the algorithm terminates, each point is assigned to a cluster by setting

$$m_i := \operatorname*{argmax}_k a_{ik}$$

The vectors M_i are the latent variables in the EM algorithm. The a_{ik} are there current estimates.

Assignment probabilities

The soft assignments are computed as

$$a_{ik} := \frac{c_k p(x_i|\theta_k)}{\sum_{l=1}^{K} c_l p(x_i|\theta_l)} .$$

They can be interpreted as

$$a_{ik} := \mathbb{E}[M_{ik}|x_i, \mathbf{c}, \boldsymbol{\theta}] = \Pr\{x_i \text{ generated by component } k \,|\, \mathbf{c}, \boldsymbol{\theta}\}$$

M-Step (1)

Objective

The M-Step re-estimates $\mathbf{c}$ and $\boldsymbol{\theta}$. In principle, we use maximum likelihood within each cluster, but we have to combine it with the use of weights a_{ik} instead "switch variables" M_{ik}.

Cluster sizes

If we know which points belong to which cluster, we can estimate the cluster proportions c_k by counting point:

$$\hat{c}_k = \frac{\#\text{ points in cluster } k}{n} = \frac{\sum_{i=1}^{n} M_{ik}}{n}$$

Since we do not know M_{ik}, we substitute our current best guess, which are the expectations a_{ik}:

$$\hat{c}_k := \frac{\sum_{i=1}^{n} a_{ik}}{n}$$

M-Step (2)

Gaussian special case

The estimation of the component parameters θ depends on which distribution we choose for p. For now, we assume a Gaussian.

Component parameters

We use maximum likelihood to estimate $\theta = (\mu, \Sigma)$. We can write the MLE of μ_k as

$$\hat{\mu}_k := \frac{1}{\#\text{ points in cluster } k} \sum_{i:\ x_i \text{ in } k} x_i = \frac{\sum_{i=1}^n M_{ik} x_i}{\sum_{i=1}^n M_{ik}}$$

By substituting current best guesses ($=a_{ik}$) again, we get:

$$\hat{\mu}_k := \frac{\sum_{i=1}^n a_{ik} x_i}{\sum_{i=1}^n a_{ik}}$$

For the covariance matrices:

$$\hat{\Sigma}_k := \frac{\sum_{i=1}^n a_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^t}{\sum_{i=1}^n a_{ik}}$$

Supervised learning

Now, we have a response variable y_i associated to each vector of predictors x_i.

Two classes of problem:

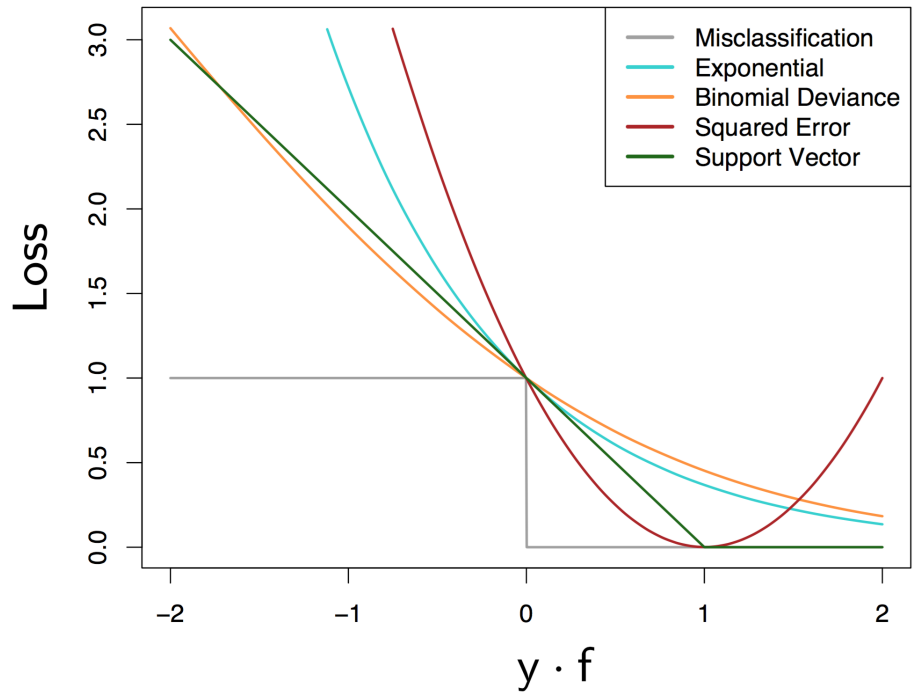
- Regression: y_i is numerical

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Classification: y_i is categorical

$$0-1 \text{ loss} = \sum_{i=1}^{n}\mathbf{1}(y_i \neq \hat{y}_i).$$

Loss Functions



Loss functions for two-class classification. Each function has been scaled so that it passes through the point $(0, 1)$.

The response is $y = \pm 1$; the prediction is f, with class prediction $\text{sign}(f)$.

misclassification: $I(\text{sign}(f) \neq y)$
exponential: $\exp(-yf)$
squared error: $(y - f)^2$

binomial deviance: $\log(1 + \exp(-2yf))$
support vector: $(1 - yf)_+$

Training vs. test error

Both the MSE for regression, and the 0-1 loss for classification can be computed:

1. On the training data.
2. On an independent test set.

Training vs. test error

Both the MSE for regression, and the 0-1 loss for classification can be computed:

1. On the training data.
2. On an independent test set.

We want to minimize the error on a very large test set which is sampled from the same process as the training data. This is called the *test error*.

Bias-variance decomposition

Consider a regression method, which given some data $(x_1, y_1), \ldots, (x_n, y_n)$ outputs a prediction $\hat{f}(x)$ for the regression function.

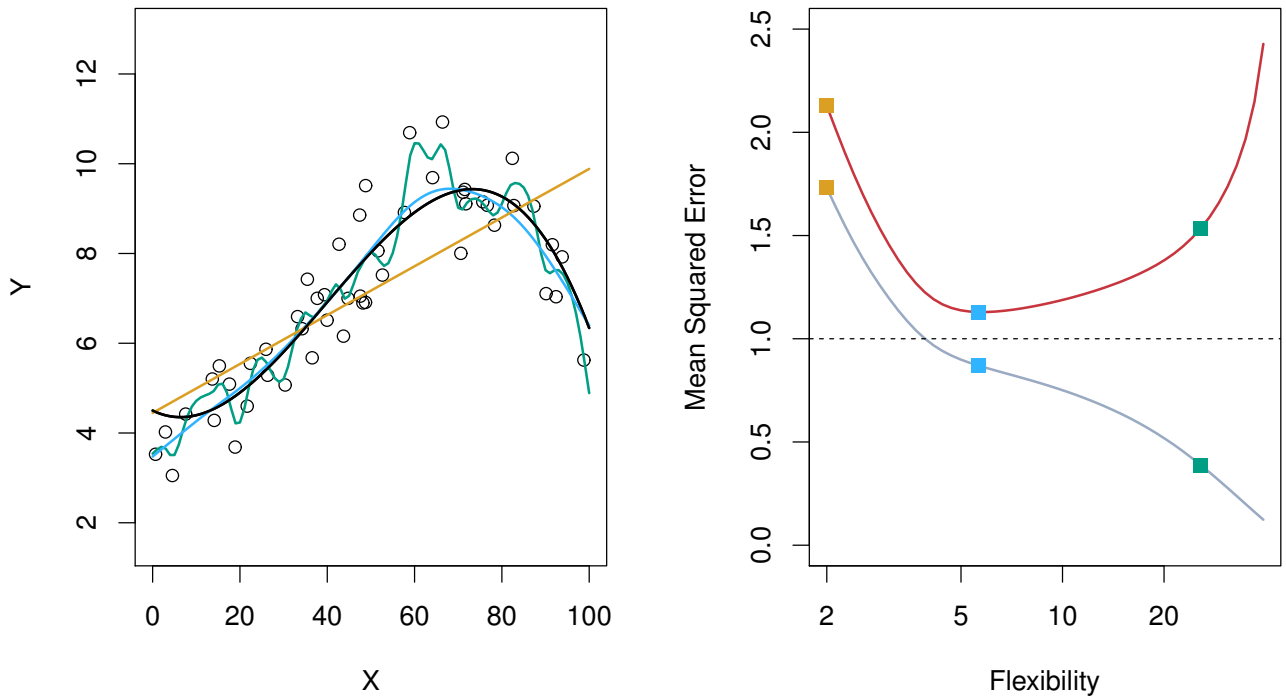
If we think of the training data as coming from some distribution, then the function $\hat{f}$ can be considered a random variable as well.

The expected test MSE of $\hat{f}$ has the following decomposition for any fixed x:

$$E([\hat{f}(x) - f(x)]^2) = \underbrace{E([\hat{f}(x) - E\hat{f}(x)]^2]}_{\text{Var}(\hat{f}(x)) > 0} + \underbrace{[E(\hat{f}(x) - f(x))]^2}_{\text{Square bias of } \hat{f}(x). > 0} + \text{Var}(\epsilon)$$

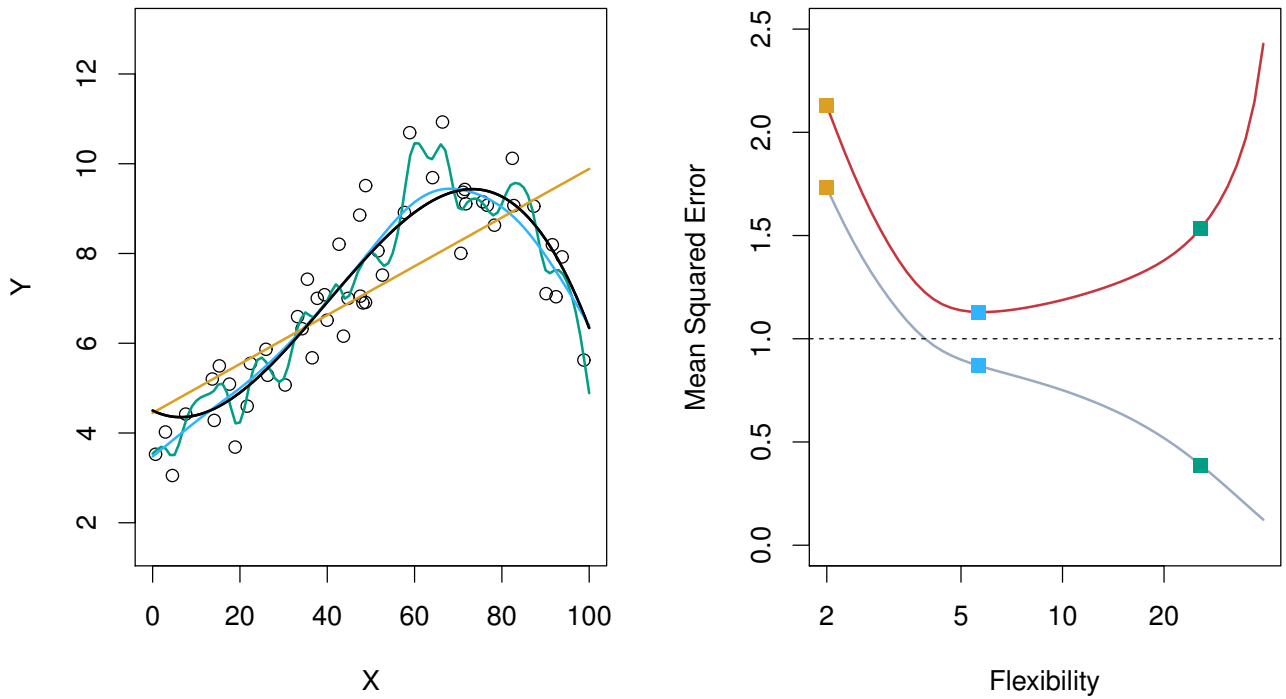
Variance: Increases with the flexibility of the model
Bias: Decreases as the flexibility of the model increases

ISL Figure 2.9.



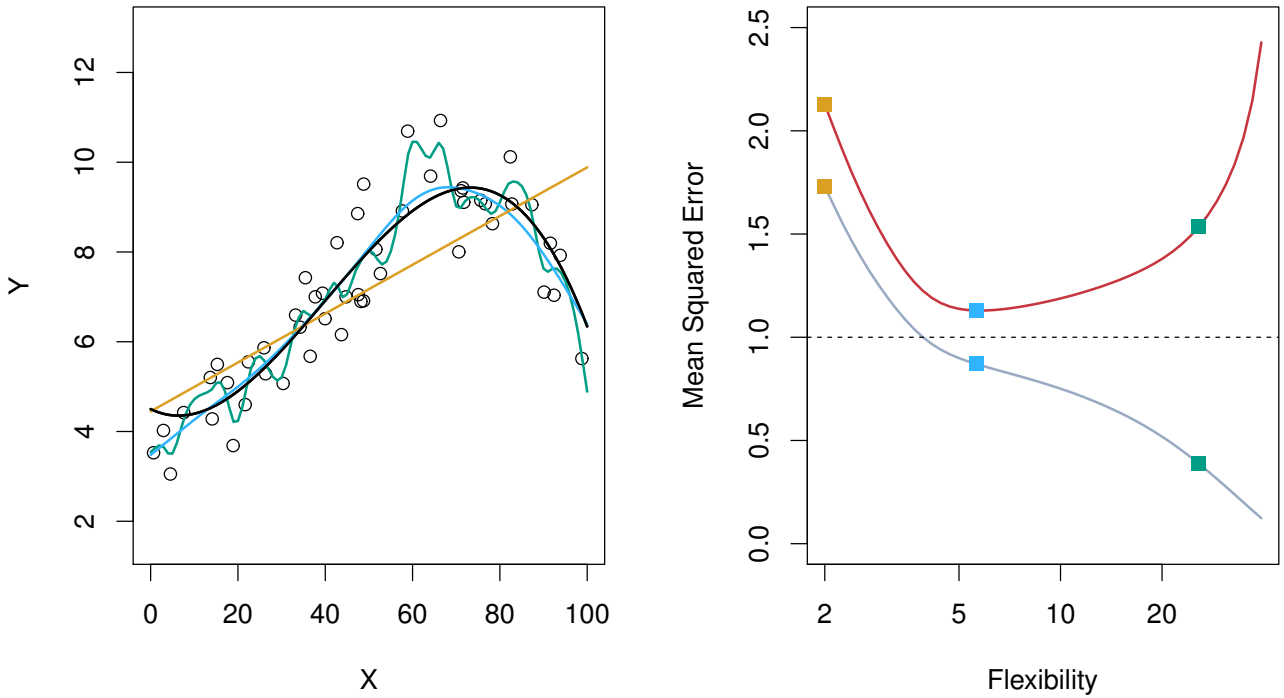
The circles are simulated data from the black curve.

ISL Figure 2.9.



The circles are simulated data from the black curve. In this artificial example, we *know* what f is.

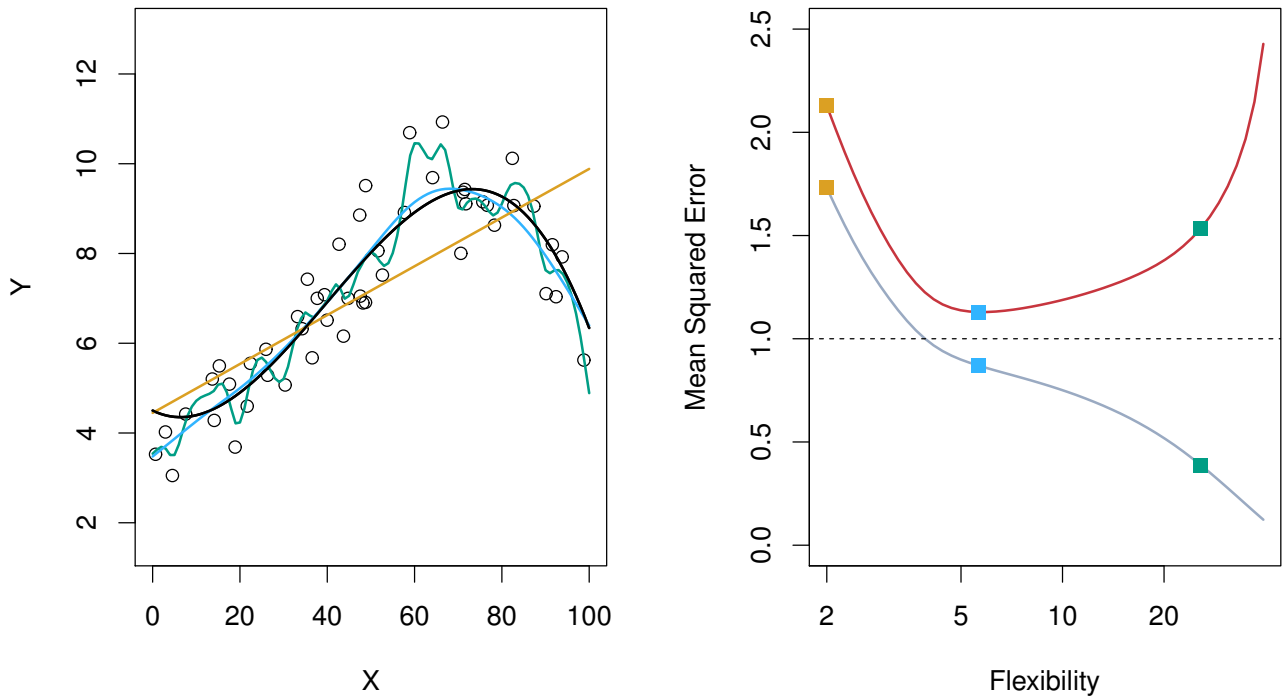
ISL Figure 2.9.



Three estimates $\hat{f}$ are shown:

1. Linear regression.
2. Splines (very smooth).
3. Splines (quite rough).

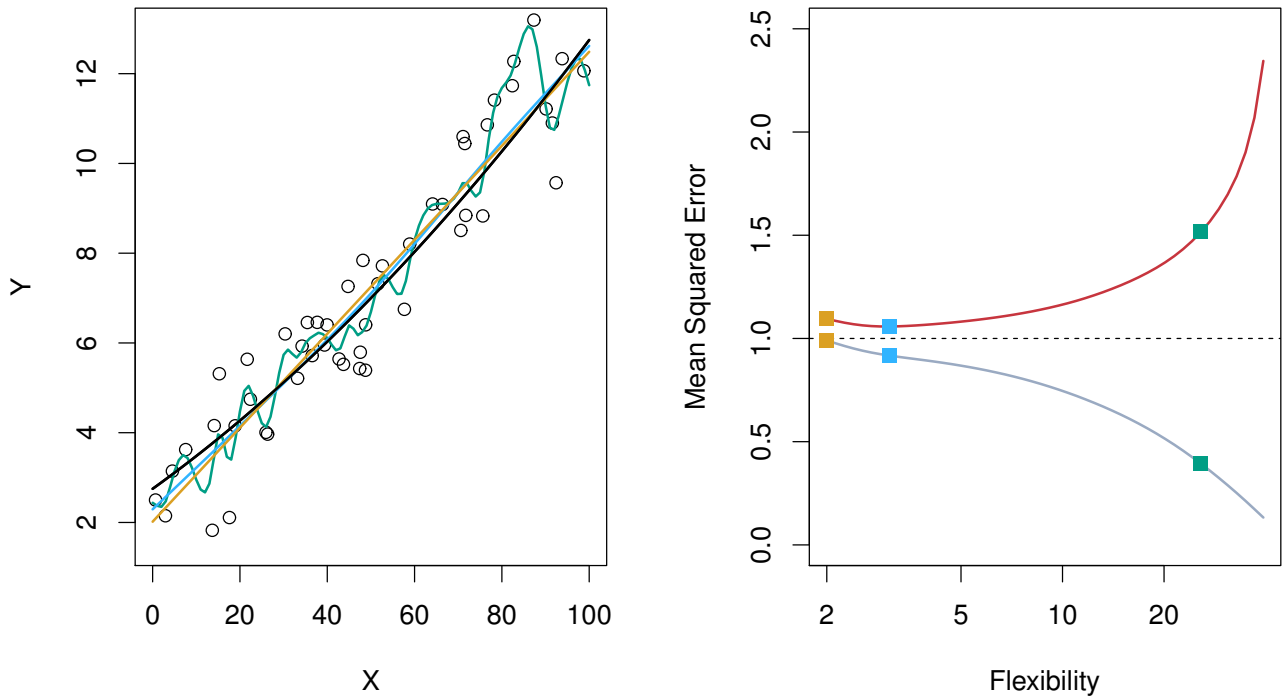
ISL Figure 2.9.



Red line: Test MSE.

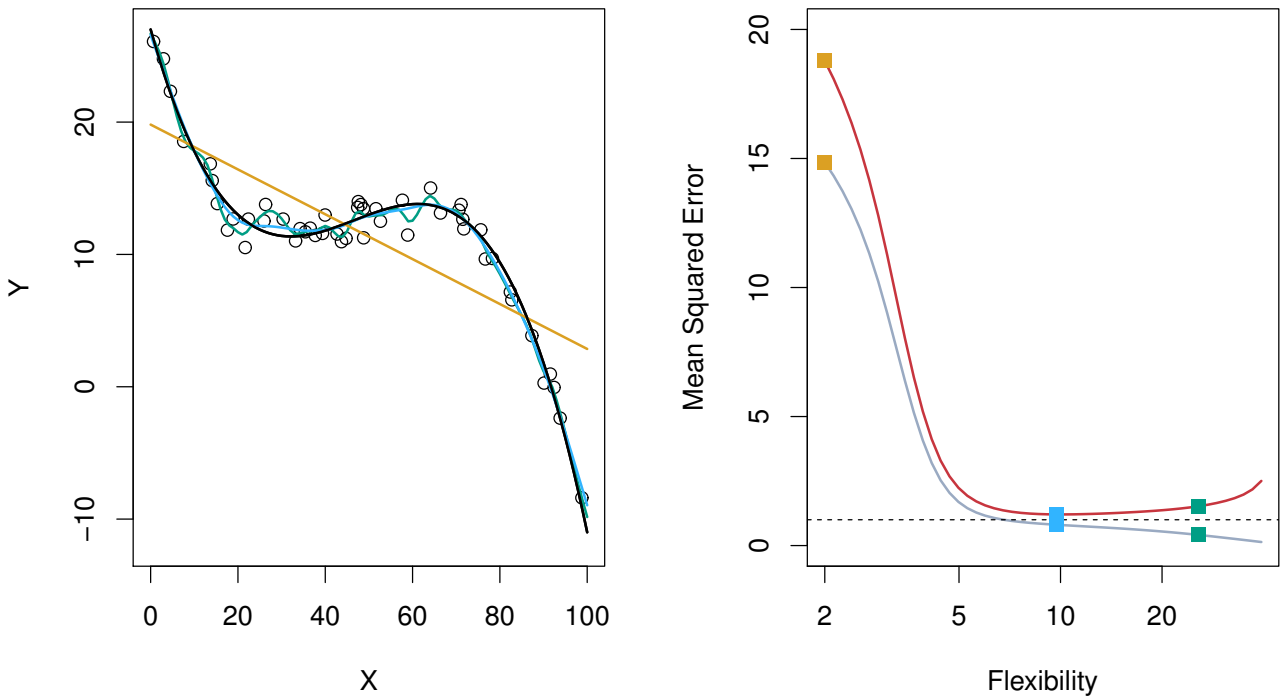
Gray line: Training MSE.

ISL Figure 2.10



The function f is now almost linear.

ISL Figure 2.11



When the noise ε has small variance, the third method does well.

How do we estimate the test error?

- Our main technique is cross-validation.
- Different approaches:
  1. **Validation set:** Split the data in two parts, train the model on one subset, and compute the test error on the other.
  2. k-**fold:** Split the data into k subsets. Average the test errors computed using each subset as a validation set.
  3. **LOOCV:** k-fold cross validation with $k = n$.
- No approach is superior to all others.
- What are the main differences? How do the bias and variance of the test error estimates compare? Which methods depend on the random seed?

The Bootstrap

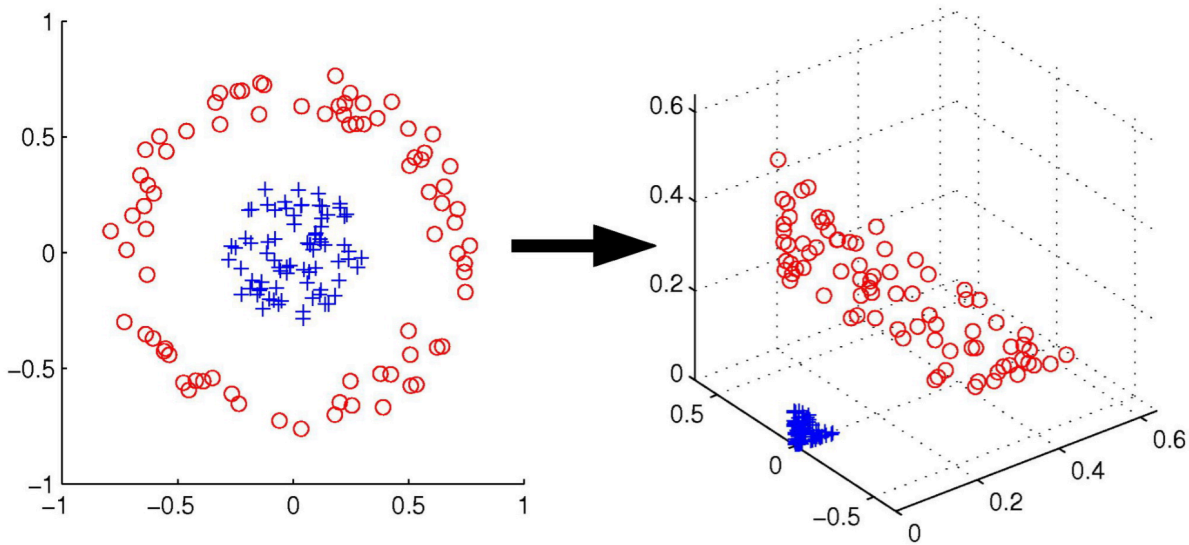
- **Main idea:** If we have enough data, the empirical distribution is similar to the actual distribution of the data.
- Resampling with replacement allows us to obtain pseudo-independent datasets.
- They can be used to:
  1. Approximate the standard error of a parameter (say, β in linear regression), which is just the standard deviation of the estimate when we repeat the procedure with many independent training sets.
  2. **Bagging**: By averaging the *predictions* $\hat{y}$ made with many independent data sets, we eliminate the variance of the predictor.

Kernel Trick: Mapping into Higher Dimensions

Example

How can a map into higher dimensions make class boundary (more) linear?
Consider

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3 \qquad \text{where} \qquad \phi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} x_1^2 \\ 2x_1x_2 \\ x_2^2 \end{pmatrix}$$



Regression methods

- Nearest neighbors regression
- Multiple linear regression
- Stepwise selection methods
- Ridge regression and the Lasso
- Principal Components Regression
- Non-linear methods:
  - Polynomial regression
  - Cubic splines
  - Smoothing splines
  - Local regression
  - GAMs: Combining the above methods with multiple predictors
  - Neural networks
- Decision trees, Bagging, Random Forests, and Boosting

Classification methods

- Nearest neighbors classification
- Logistic regression
- LDA and QDA
- Stepwise selection methods
- Decision trees, Bagging, Random Forests, and Boosting
- Neural networks
- Support vector classifier and support vector machines

Self testing questions

For each of the regression and classification methods:

1. What are we trying to optimize?
2. What does the fitting algorithm consist of, roughly?
3. What are the tuning parameters, if any?
4. How is the method related to other methods, mathematically and in terms of bias, variance?
5. How does rescaling or transforming the variables affect the method?
6. In what situations does this method work well? What are its limitations?

Simple Text Model

We represent the document as

$\mathbf{H} = (H_1, \ldots, H_d)$ where $H_j = \#$ occurences of term j in document.

Note:

- d is the number of all terms (distinct words) in the dictionary i.e. d is identical for all documents.
- $n = \sum_j H_j$ can change from document to document.

Multinomial model

To define a simple probabilistic model of document generation, we can use a multinomial distribution $P(\mathbf{H}|\mathbf{t}, n)$. That means:

- Each word in the document is sampled independently of the other words.
- The probabilities of occurrence are

$$\text{Pr}\{ \text{ word } = \text{ term } j \} = t_j \ .$$

With Context: Bigram Models

Bigram model

A bigram model represents the conditional distribution

$$\text{Pr}(\text{word}|\text{previous word}) =: \text{Pr}(w_l|w_{l-1}) \ ,$$

where w_l is the lth word in a text.

Representation by multinomial distributions

A bigram model is a *family* of d multinomial distributions, one for each possible previous word.

Estimation

For each term k, find all terms in the corpus which are preceeded by k and record their number of occurences in a vector

$\mathbf{H}_k = (H_{k1}, \ldots, H_{kd})$ where $H_{kj} =$ number of times term j follows on term k

Then compute the maximum likelihood estimate $_k$ from the sample $\mathbf{H}_k$.
Note: Both j and k run through $\{1, \ldots, d\}$.