Regression and Classification

Part 2: Classification

Sourish Das

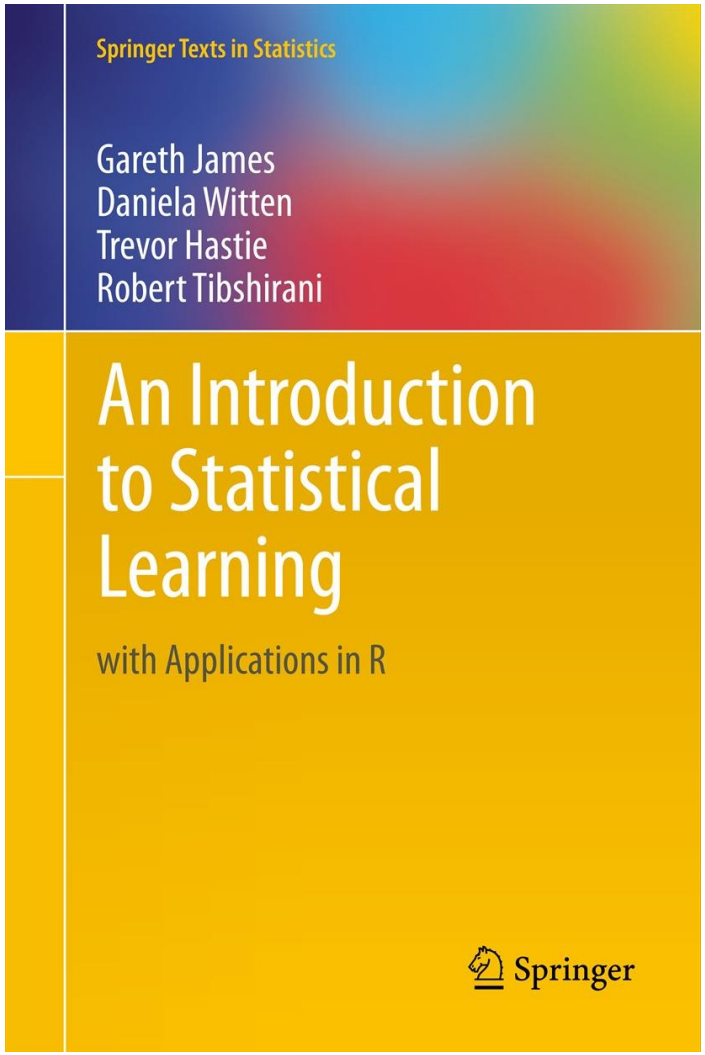
Chennai Mathematical Institute

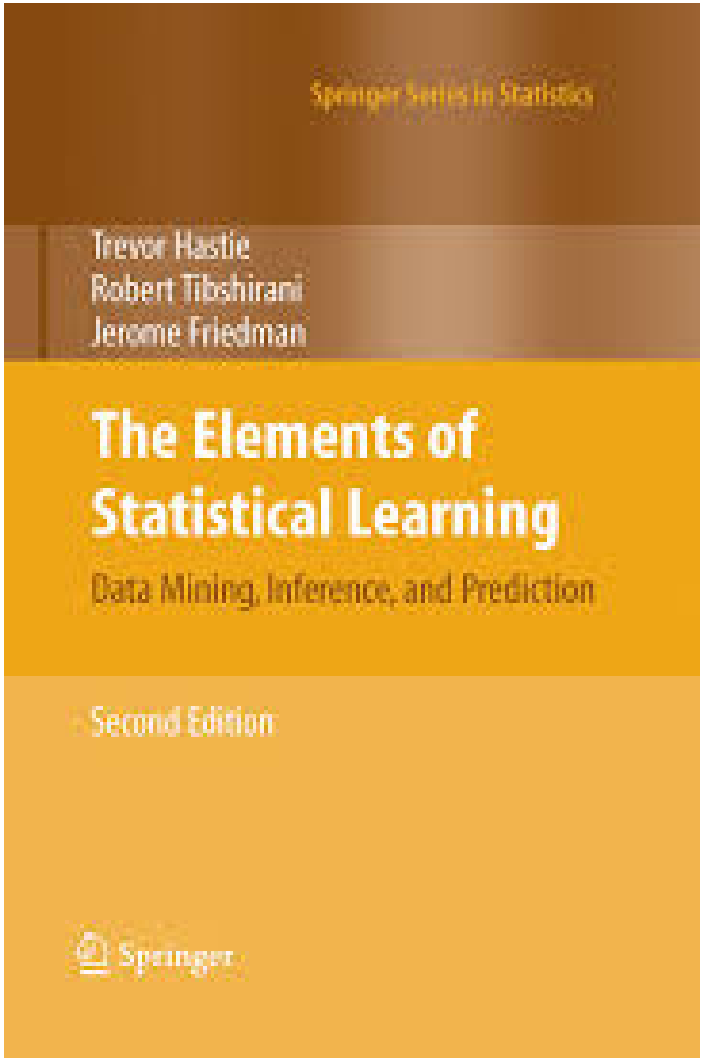
Aug-Nov, 2019

Outline

Introduction

Reference


Springer Texts in Statistics
Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani
An Introduction to Statistical Learning
with Applications in R
Springer


Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
Springer

Reading material

- *Data Mining; Concepts and Techniques*, Jiawei Han and Micheline Kamber, Morgan Kaufman (2006).
- *Web Data Mining*, Bing Liu, Springer Verlag (2007).

For a good introduction to text mining and information retrieval, please see.

- *An Introduction to Information Retrieval*, Christopher D Manning, Prabhakar Raghavan and Hinrich Schütze, Cambridge University Press (2009). (Available online at `http://www-nlp.stanford.edu/IR-book`).

Motivating Examples of Supervised Learning

Ex 2 Given the credit history and other features of a loan applicant, a bank manager want to predict if loan application would become good or bad loan!!

- Note that your objective is to predict the label of the loan good or bad!

Motivating Examples of Supervised Learning

Ex 3 Can you predict the Air Pressure Failue of Scania Truck?

- We are going to use `aps_failure_training_set.csv` and `aps_failure_test_set.csv` in `Data` folder.

Motivating Examples of Supervised Learning

Ex 4 Can you identify the correlation pathways between gene expression and the disease?

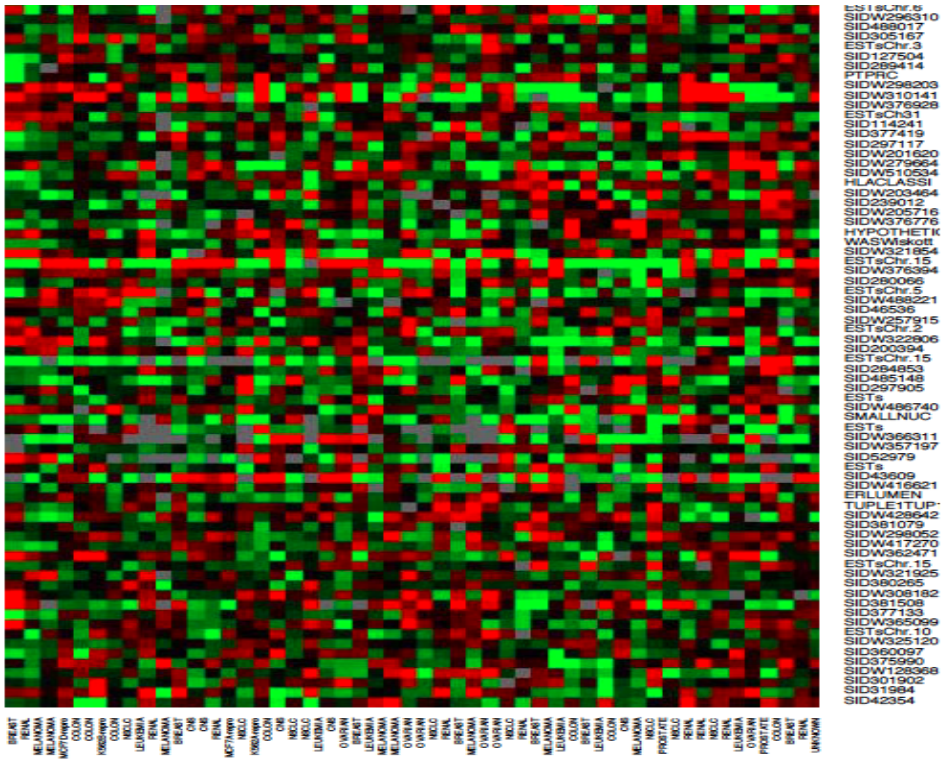


FIGURE 1.3. *DNA microarray data: expression matrix of* 6830 *genes (rows) and* 64 *samples (columns), for the human tumor data. Only a random sample of* 100 *rows are shown. The display is a heat map, ranging from bright green (negative, under expressed) to bright red (positive, over expressed). Missing values are gray. The rows and columns are displayed in a randomly chosen order.*

Supervised learning

- Supervised learning algorithms are trained using labeled data.
- For example, a piece of equipment could have data points labeled either “F” (failed) or “R” (runs).
- Typically,

$$y = f(X),$$

where y is target variable and X is feature matrix

- Objective: Learn $f(.)$

Supervised learning

- Supervised learning

$$y = f(X)$$

typically are of two types:

1. Regression : target variable y is continuous variable - e.g., income, blood pressure, distance etc.

2. Classification: target variable y is categorical or label variable - e.g., species type, color, class etc.

Data : Quantitative Response

x_{11}	x_{12}	$\cdots$	x_{1p}	y_1
x_{21}	x_{22}	$\cdots$	x_{2p}	y_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{n1}	x_{n2}	$\cdots$	x_{np}	y_n
x^*_{11}	x^*_{12}	$\cdots$	x^*_{1p}	$y^*_1 =?$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x^*_{m1}	x^*_{m2}	$\cdots$	x^*_{mp}	$y^*_m =?$

- $D_{train} = (X, y)$, is the traing dataset, where X is the matrix of predictors or features, y is the dependent or target variable.
- $D_{test} = (X^*, y^* =?)$ is the test dataset, where X^* is the matrix of predictors or features, and y^* is missing and we want to forecast or predict y^*

Data : Qualitative Response

x_{11}	x_{12}	$\cdots$	x_{1p}	G_1
x_{21}	x_{22}	$\cdots$	x_{2p}	G_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{n1}	x_{n2}	$\cdots$	x_{np}	G_n
x^*_{11}	x^*_{12}	$\cdots$	x^*_{1p}	$G^*_1 = ?$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x^*_{m1}	x^*_{m2}	$\cdots$	x^*_{mp}	$G^*_m = ?$

- Qualitative variables are also referred to as *categorical* or *discrete* variables as well as *factors*.

Practical Session with R

- Open `mtcars_analysis.R`
- Check out how we split the data into `train_data` and `test_data`.

Outline

Classification

Statistical Decision Theory for Classification

- What do we do when the output is a categorical variable G?
- Same strategy works here
- Define proper loss function: zero-one loss function

$$L(G, \hat{G}) = \begin{cases} 1 & g_k \neq \hat{g}_k \\ 0 & g_k = \hat{g}_k \end{cases}$$

for $k = 1, 2, \ldots, K$

Statistical Decision Theory for Classification

- The expected prediction error is

$$EPE = \mathbb{E}[L(G, \hat{G}(X))],$$

where again the expectation is taken with respect to the joint distribution $\mathbb{P}(G, X)$, where

$$EPE = \mathbb{E}_X \sum_{k=1}^{K} L(g_k, \hat{G}(X)) \mathbb{P}(g_k|X)$$

Statistical Decision Theory for Classification

- Suffices to minimize *EPE* pointwise

$$\hat{G}(x) = \mathit{argmin}_{g \in G}[1 - \mathbb{P}(g|X = x)]$$

or simply

$$\hat{G}(X) = g_k \quad \mathit{if} \quad \mathbb{P}(g_k|X = x) = \max_{g \in G} \mathbb{P}(g|X = x)$$

- This reasonable solution is known as the Bayes classifier

- Classify to the most probable class, using the conditional (discrete) distribution $\mathbb{P}(G|X)$

Statistical Decision Theory for Classification

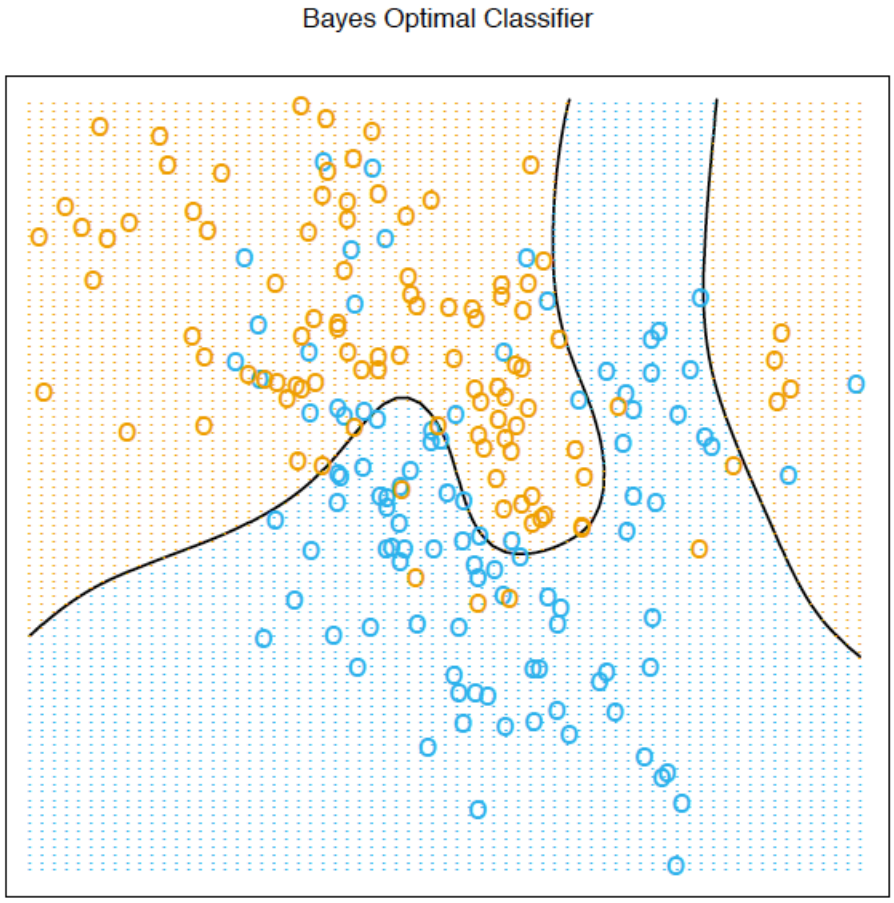


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

Linear Models in Classification

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (BLUE $= 0$, ORANGE $= 1$), and then fit by linear regression. The line is the decision boundary defined by $x^T\hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.*

- scatter plot of training data on pair of input X_1 and X_2

Linear Models in Classification

- scatter plot of training data on pair of input X_1 and X_2
- Simulated data - please ignore the simulation model for now.
- output class variable G has the values BLUE or ORANGE.
- There are 100 points in each of the two classes.
- The response Y coded as 0 for BLUE and 1 for ORANGE.

Linear Models in Classification

- The fitted values $\hat{Y}$ are converted to a fitted class variable $\hat{G}$ according to the rule

$$\hat{G} = \begin{cases} \textit{ORANGE} & \text{if } \hat{Y} > 0.5 \\ \textit{BLUE} & \text{if } \hat{Y} \leq 0.5 \end{cases}$$

- The set of points in $\mathbb{R}^2$ classified as ORANGE corresponds to $\{x : x^T \hat{\beta} > 0.5\}$

- **Decision boundary**: $\{x : x^T \hat{\beta} = 0.5\}$

- There are several misclassifications on both sides of the decision boundary.

Linear Models in Classification

- Scenario 1 The training data in each class were generated from bivariate Gaussian distributions with uncorrelated components and different means.

- Scenario 2 The training data in each class came from a mixture of 10 lowvariance Gaussian distributions, with individual means themselves distributed as Gaussian.

- A mixture of Gaussians is best described in terms of the generative model.
- A linear decision boundary is unlikely to be optimal, and in fact is not.
- The optimal decision boundary is nonlinear and disjoint, and as such will be much more difficult to obtain.

Nearest-Neighbor Methods

- the k-nearest neighbor fit for $\hat{Y}$ is defined as follows:

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

  where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training sample.
- Closeness implies a metric, which for the moment we assume is Euclidean distance.

Nearest-Neighbor Methods

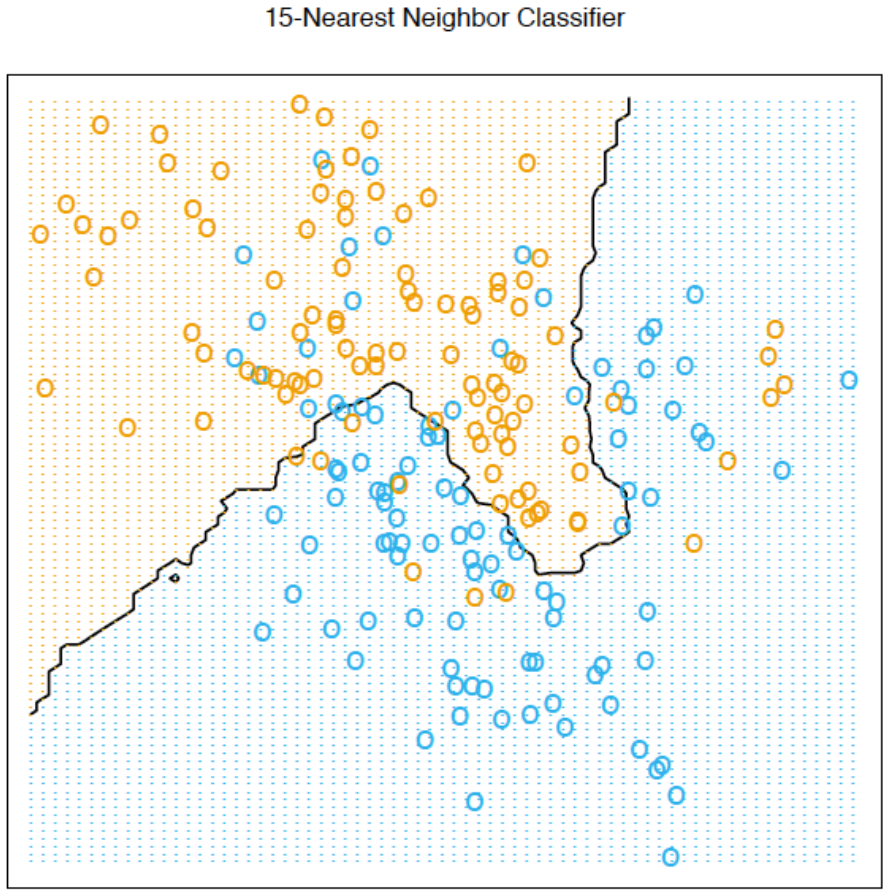


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

- Far fewer training observations are misclassified

Nearest-Neighbor Methods

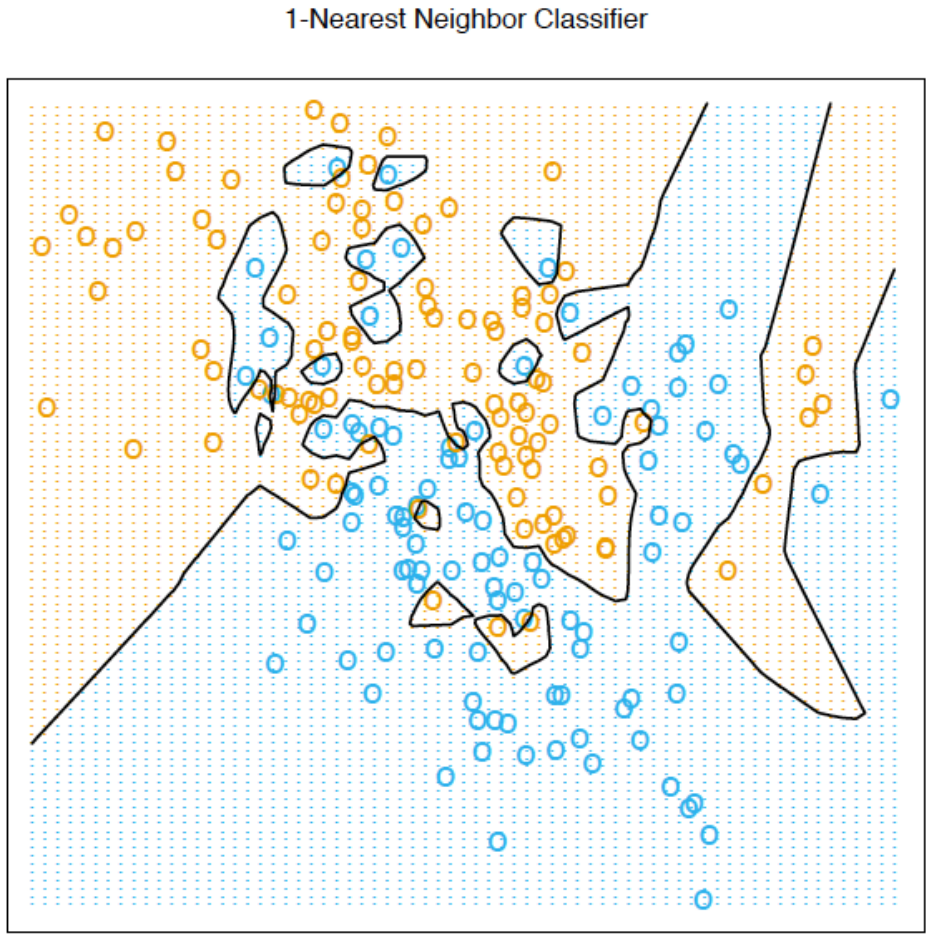


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

- None are misclassified

Nearest-Neighbor Methods

- k-nearest-neighbor fits have a single parameter, the number of neighbors k, compared to the p parameters in least-squares fits.
- The effective number of parameters of k-nearest neighbors is n/k and is generally bigger than p.
- If the neighborhoods were nonoverlapping, there would be n/k neighborhoods and we would fit one parameter (a mean) in each neighborhood.

From Least Squares to Nearest Neighbors

- The linear decision boundary from least squares is very smooth (work nicely in Scenario 1)
- Major assumption: decision boundary is linear (low variance and high bias)
- The k-nearest-neighbor procedures do not appear to rely on any stringent assumptions about the underlying data (work nicely in Scenario 2)
- However, any particular subregion of the decision boundary depends on a handful of input points (high variance and low bias)

- Each method has its own situations for which it works best

Practical Session with R

- R session on Nearest Neighbors

Discriminant Analysis

- Suppose $f_k(x)$ is the class-conditional density of X in class $G = k$
- π_k be the prior probability of class k, with $\sum_{k=1}^{K} \pi_k = 1$.
- Using Bayes Theorem:

$$\mathbb{P}(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

- In terms of ability to classify, having the $f_k(x)$ is almost equivalent to having the quantity $\mathbb{P}(G = k|X = x)$.

Discriminant Analysis

- Many techniques are there to model $f_k(x)$
- linear and quadratic discriminant analysis use Gaussian densities
- Finite mixture models (some what complicated)

$$f_k(x) = \sum_{i=1}^{I} p_i N(\mu_i, \Sigma_i)$$

- Nonparametric density estimation (very complicated)

$$f_k(x) = \sum_{i=1}^{\infty} p_i N(\mu_i, \Sigma_i) = \lim_{I \to \infty} \sum_{i=1}^{I} p_i N(\mu_i, \Sigma_i)$$

Linear Discriminant Analysis

- We model each class density as multivariate Gaussian

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

- Linear discriminant analysis (LDA) arises in the special case when we assume

$$\Sigma_k = \Sigma \quad \forall k$$

Linear Discriminant Analysis

- We want to compare two classes k and l,
- Let's look at the ratio

$$\begin{aligned}\log \frac{\mathbb{P}(G=k|X=x)}{\mathbb{P}(G=l|X=x)} &= \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k-\mu_l) \\ &\quad + x^T\Sigma^{-1}(\mu_k-\mu_l)\end{aligned}$$

  is an equation linear in x.

Linear Discriminant Analysis

- $\Sigma_k = \Sigma \ \forall k$ cause the normalization factors to cancel, as well as the quadratic part in the exponents.
- The decision boundary between classes k and l is linear
- From above the linear discriminant functions

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \Sigma^{-1} \mu_k + \log \pi_k$$

- Best decision rule:

$$G(x) = \text{argmax}_k \, \delta_k(x)$$

Linear Discriminant Analysis

- In practice we do not know the parameters of the Gaussian distributions
- Need to estimate using our training data
  - $\hat{\pi}_k = \frac{f_k}{n}$, where f_k is the number of class-k observations
  - $\hat{\mu}_k = \sum_{g_i=k} x_n / f_k$
  - $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (n - K)$
- These estimates are MLE

Two Classes LDA

- The LDA for two classes are very simple.
- The LDA rule classifies to class 2 if

$$x^T \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > c$$

where

$$c = \frac{1}{2}\hat{\mu}_2^T \hat{\Sigma}^{-1}\hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\Sigma}^{-1}\hat{\mu}_1 + \log(f_1/n) - \log(f_2/n)$$

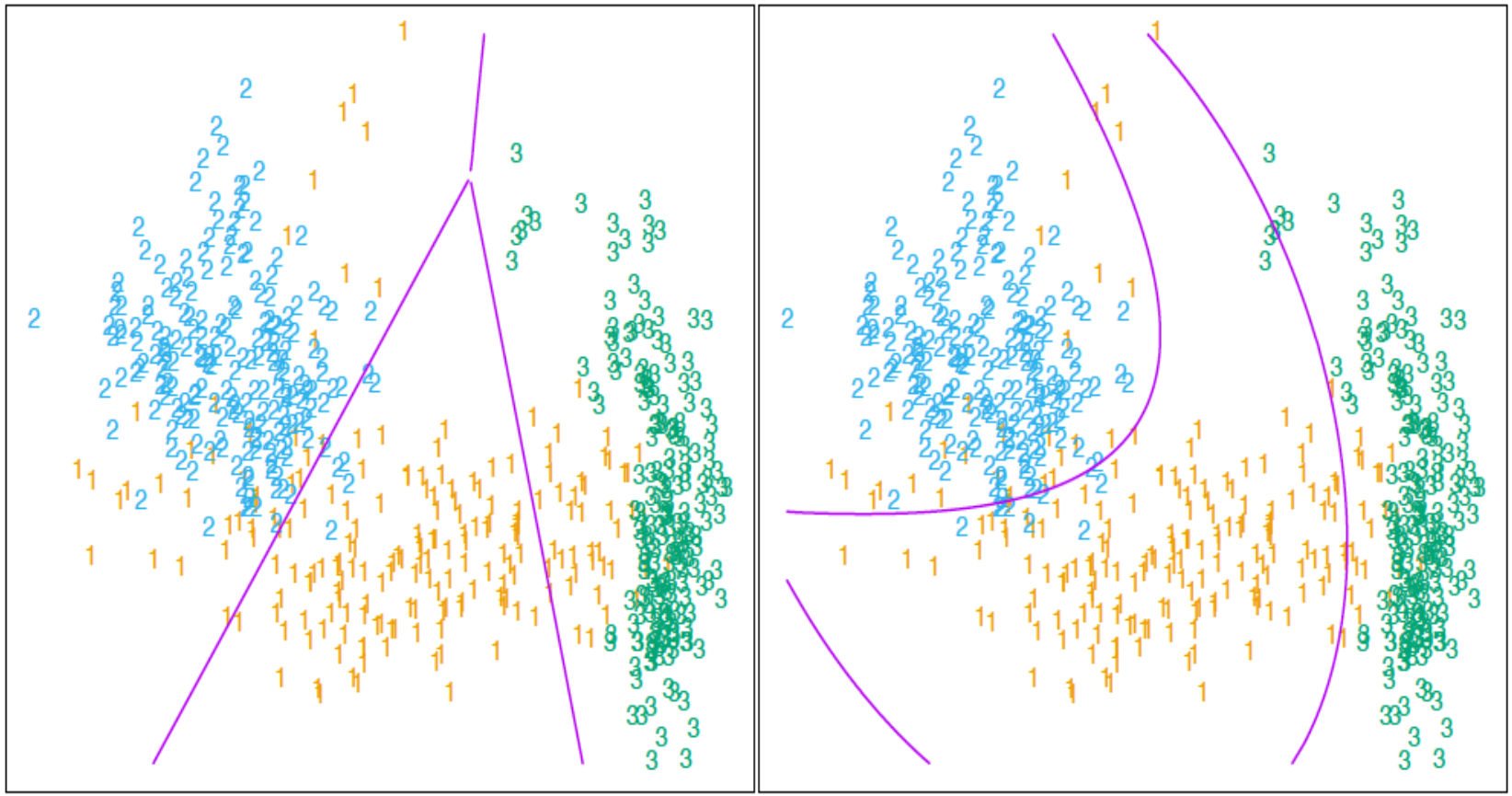
Quadratic Discriminant Analysis

- $\Sigma_k \neq \Sigma$ at least for one k
- Convinient cancellation will not work any more
- Then QDA function is

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k.$$

- The decision boundary between each pair of classes k and l is described by quadratic equation $\{x : \delta_k(x) = \delta_l(x)\}$

LDA or QDA



Practical Session with R

- R session on Discriminant Analysis

Feature Extraction/Transformed Predictors

- **Feature extraction** starts from an initial set of measured data and builds derived values (features)
- Suppose $X = \{X_1, X_2, \ldots, X_p\}$ is the available data
- Variable transformation such as X_j^k, or $\log(X_j)$ etc. can be adapted and feature space can be extended.
- Feature extraction extends the feature space into higher-dimension space.
- Often this may lead to overfitting and hence poor performance of the model.

Practical Session with R

- R session on Feature Extraction

Logistic Regression

- Logistic Regression model for K class problem is

$$\begin{aligned}
\log \frac{\mathbb{P}(G=1|X=x)}{\mathbb{P}(G=K|X=x)} &= x^T\beta_1 \\
\log \frac{\mathbb{P}(G=2|X=x)}{\mathbb{P}(G=K|X=x)} &= x^T\beta_2 \\
&\vdots \\
\log \frac{\mathbb{P}(G=K-1|X=x)}{\mathbb{P}(G=K|X=x)} &= x^T\beta_{K-1}
\end{aligned}$$

Logistic Regression

- A simple calculation shows that

$$\begin{aligned}\mathbb{P}(G=k|X=x) &= \frac{\exp(x^T\beta_k)}{1+\sum_{l=1}^{K-1}\exp(x^T\beta_l)}, \quad k=1,2,\ldots,K-1\\ \mathbb{P}(G=K|X=x) &= \frac{1}{1+\sum_{l=1}^{K-1}\exp(x^T\beta_l)},\end{aligned}$$

  and they clearly sum to one.

- To emphasize the dependence on the entire parameter set $\theta=\{\beta_1,\ldots,\beta_{K-1}\}$; we denote the probabilities

$$\mathbb{P}(G=k|X=x)=p_k(x;\theta)$$

- Appropriate distribution for the K class Logistic Regression implies **multinomial distribution**.

Logistic Regression for Two Class

- Two-class case:

$$y_i = \begin{cases} 1 & \text{if } g_i = 1 \\ 0 & \text{if } g_i = 2 \end{cases}$$

- The log-likelihood can be written

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{n} \left\{ y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta)) \right\} \\ &= \sum_{i=1}^{n} \left\{ y_i (x_i^T \beta) - \log(1+e^{x_i^T \beta}) \right\} \end{aligned}$$

Logistic Regression for Two Class

- To maximize the log-likelihood, we set its derivatives to zero.

$$\frac{\partial l(\beta)}{\partial \beta} = 0$$

- Newton's update in Fisher's scoring algorithm

$$\beta^{new} = \beta^{old} - H^{-1}\frac{\partial l(\beta)}{\partial \beta},$$

where H is the Hessian matrix,

$$H = \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T}$$

Practical Session with R

- R session on Logistic Regression

Random Forest

- Apply decision tree on bootstrap samples

- Suppose $\mathcal{D} = \{(y_i, x_i) | i = 1, \ldots, n\}$ is the dataset

1. Draw a simple random sample of size n with replaement from $\mathcal{D}$ (aka. bootstrap sample)

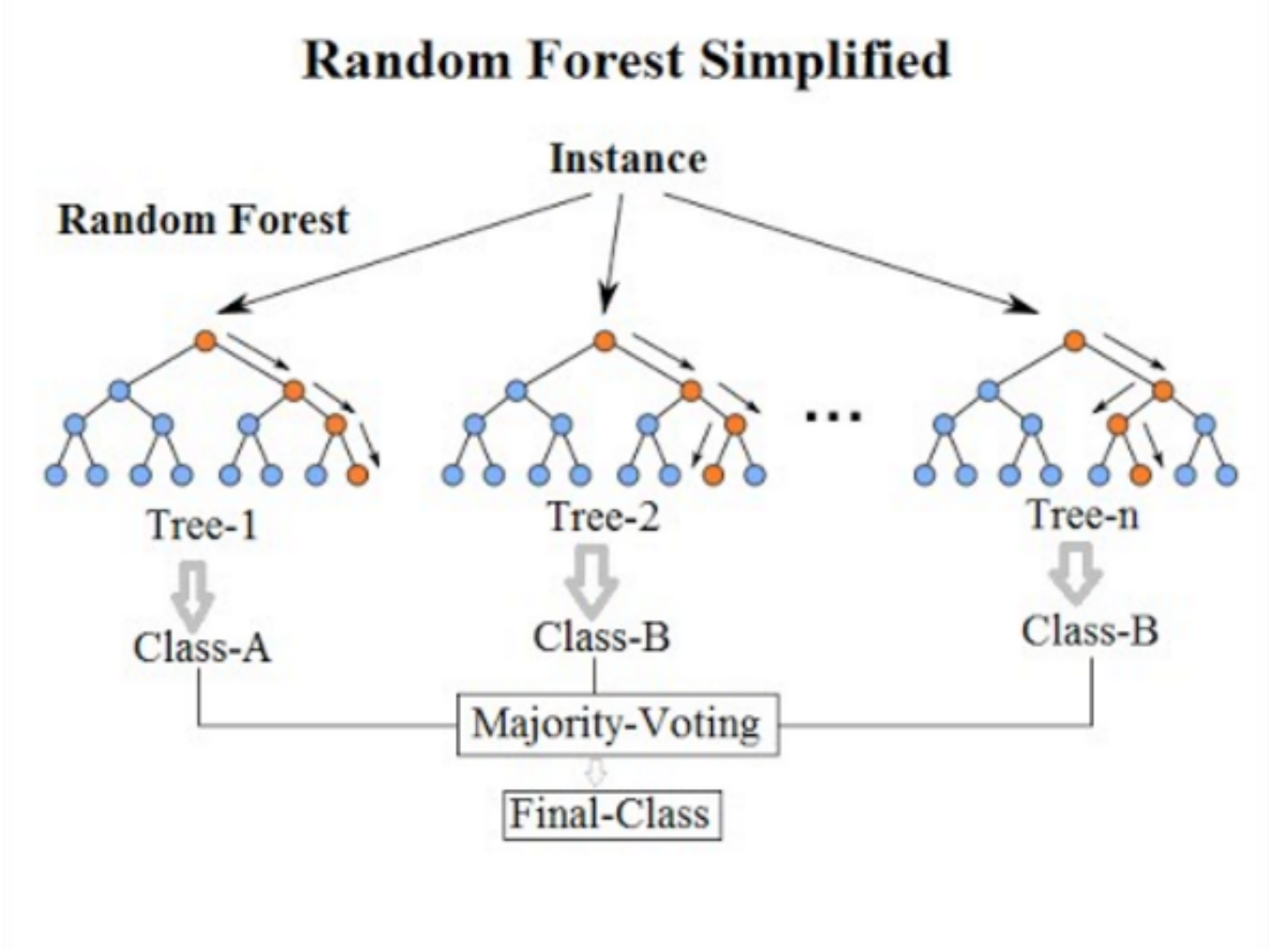
2. Build a decision tree on the bootstrap sample

3. Repeat step 1 and step 2 - B times - choose B large (maybe 5000)

Random Forest

- Now you have B many Decision Tree (DT) models $DT_1, \ldots, DT_B$
- For new data point x^* we have prediction B many predictions $y_1^* = DT_1(x^*), \ldots, y_B^* = DT_B(x^*)$
- Take the label as predition which gets the maximum vote - or mode

Random Forest



courtesy: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

Practical Session with R

- R session on Decision Tree and Random Forest

Classification using Hyperplane

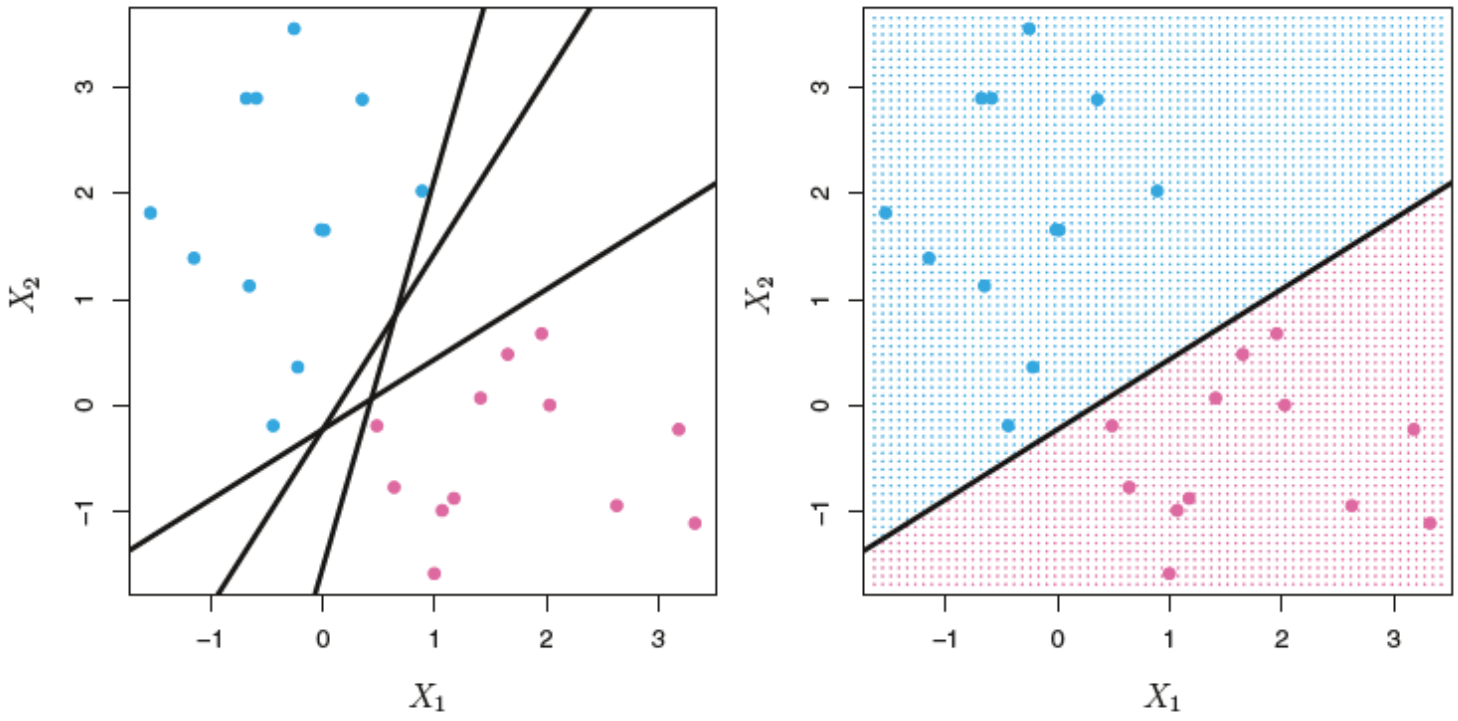


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

Classification using Hyperplane

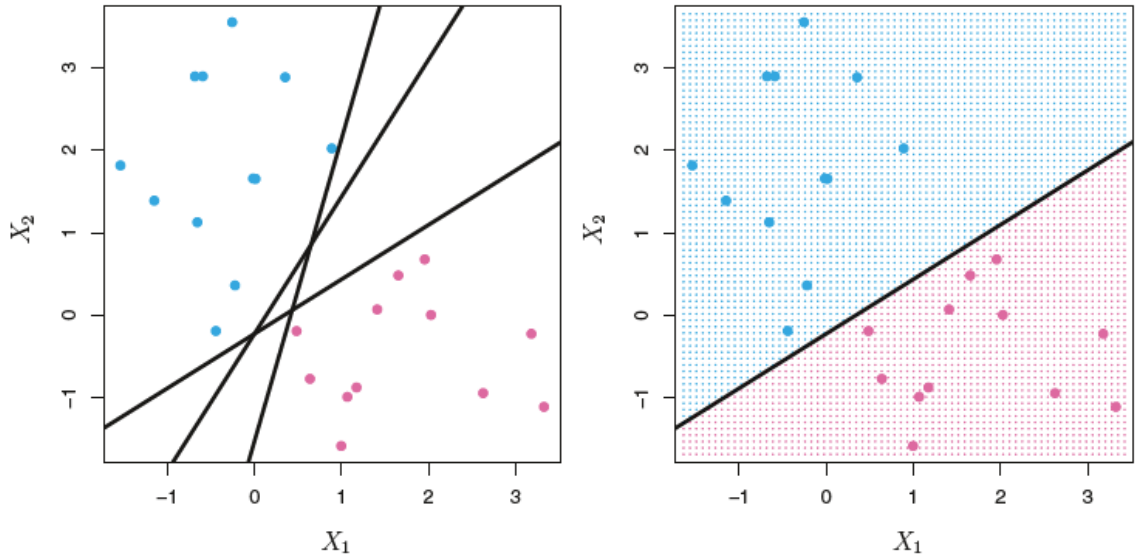


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

- A point is on the hyperplane if

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

Classification using Hyperplane

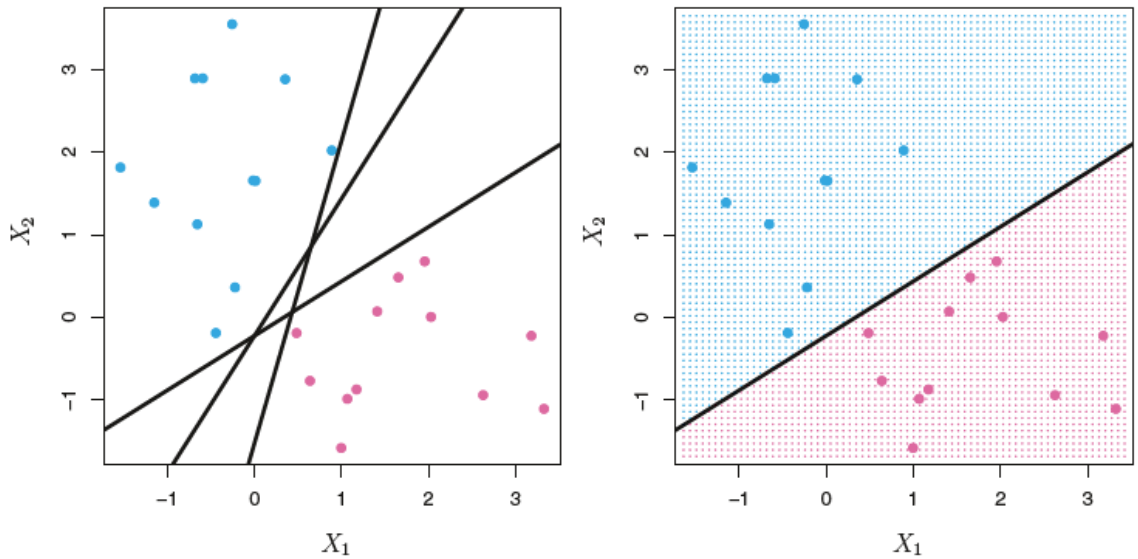


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

- If blue then $y_i = 1$
- If purple then $y_i = -1$

Classification using Hyperplane

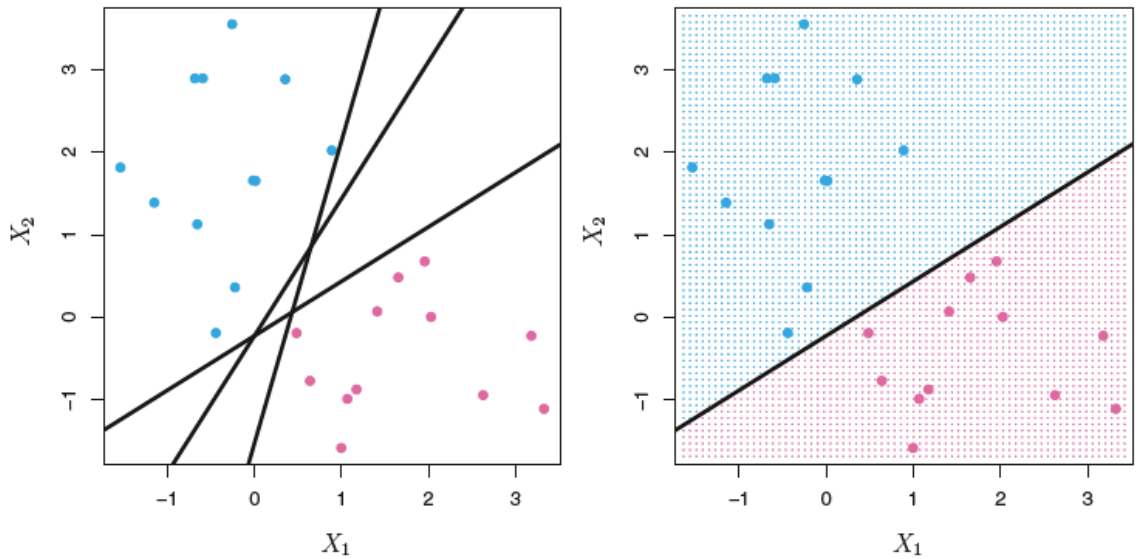


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

- If $\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} > 0$ then $y_i = 1$
- If $\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} < 0$ then $y_i = -1$

Classification using Hyperplane

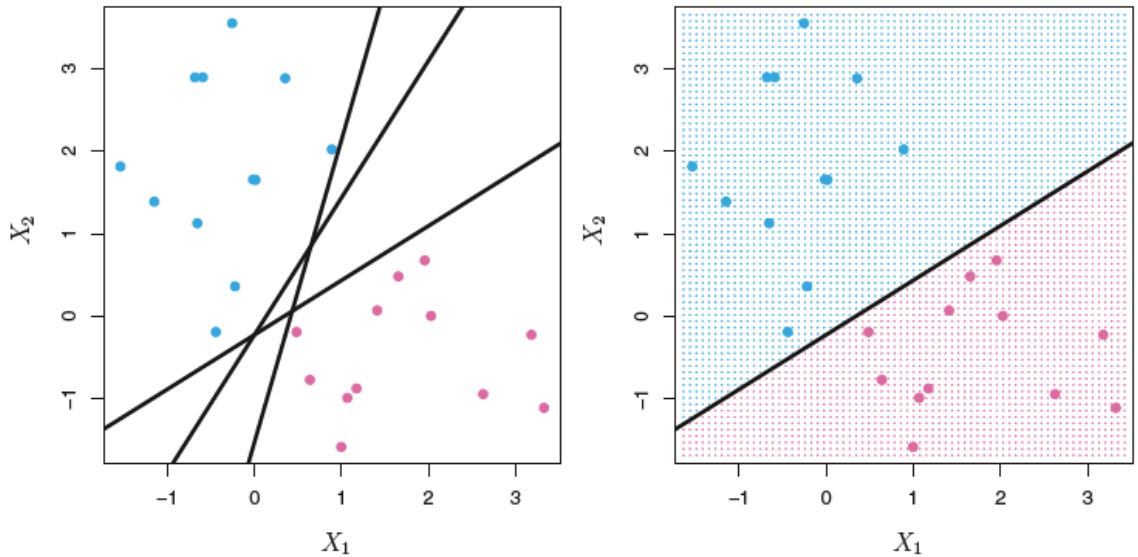


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

- $y_i(\beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i}) > 0 \; i = 1, 2, \ldots, n$

Classification using Hyperplane

- If seperating hyperplane exists and $x^* = (x_1^*, x_2^*)$ is a new test point - we can classify based on sign of $f(x^*) = \beta_0 + \beta_1 x_1 * + \beta_2 x_2^*$
- If $f(x^*) > 0$ then we assign $y^* = 1$ or blue
- If $f(x^*) < 0$ then we assign $y^* = -1$ or purple

Maximal Margin Classifier

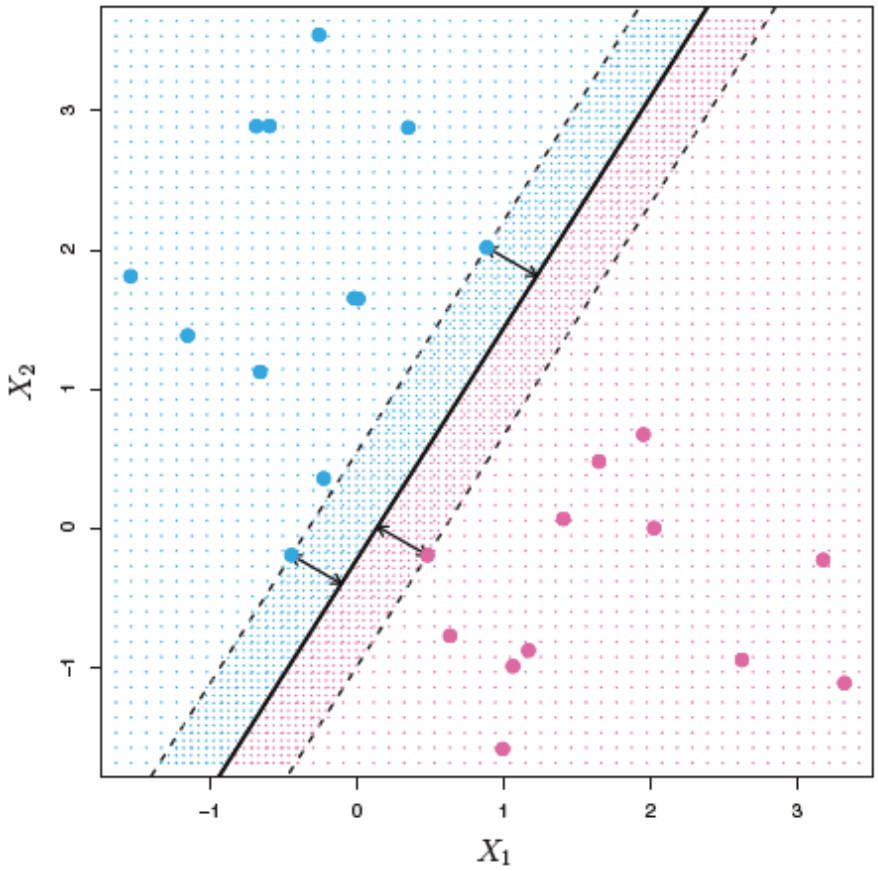


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

Maximal Margin Classifier

$$\max_{\beta_0,\ldots,\beta_p} M$$

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

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) > M \ \forall \ i = 1, 2, \ldots, n.$$

Non-separable Cases

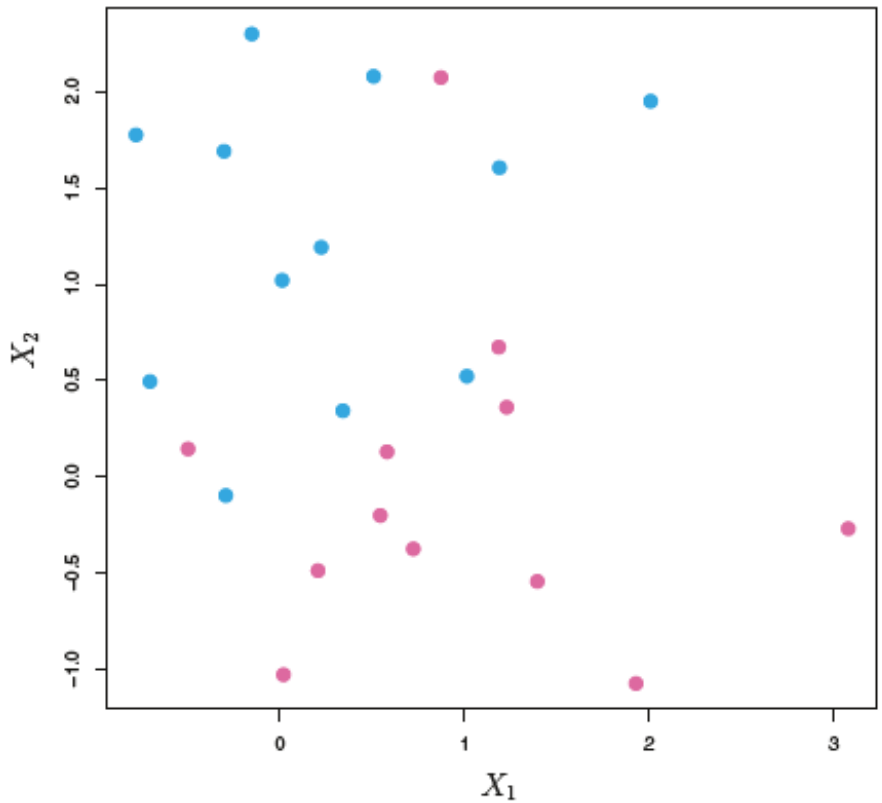


FIGURE 9.4. *There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.*

Support Vector Classifier

$$\max_{\beta_0,\ldots,\beta_p} M$$

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

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) > M(1 - \epsilon_i) \ \forall \ i = 1, 2, \ldots, n$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C$$

C is nonnegative tuning parameter; often known as total cost of error.

- If $\epsilon_i = 0$ then the observation is on the correct side of the margin
- If $\epsilon_i > 0$ then the observation is on the wrong side of the margin
- If $\epsilon_i > 1$ then the observation is on the wrong side of the hyperplance

Support Vector Machine

- SVM is an extension of Support Vector Classifier
- It tries to model the lonlinear decision boundary
- For instance, rather than fitting a support vector classifier using p features

$$X_1, X_2, \ldots, X_p$$

we could instead fit a support vector classifier using $2p$ features

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

Support Vector Machine

$$\max_{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2}} M$$

$$\text{subject to } \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1$$

$$y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right) > M(1-\epsilon_i)\ \forall\ i = 1, 2, \ldots, n$$

$$\epsilon_i \geq 0,\quad \sum_{i=1}^{n}\epsilon_i \leq C$$

C is nonnegative tuning parameter; often known as total cost of error.

Support Vector Machine

A non-linear hyperplane can be considered

$$\max_{\beta_0,\beta_{11},\beta_{12},\ldots,\beta_{p1},\beta_{p2}} M$$

$$\text{subject to } \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1$$

$$y_i f(x) > M(1-\epsilon_i) \;\forall\; i = 1, 2, \ldots, n$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n}\epsilon_i \leq C$$

C is nonnegative tuning parameter; often known as total cost of error.

Support Vector Machine

- The inner product of two r-vectors a and b is defined as

$$< a, b >= \sum_{i=1}^{r} a_i b_i$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$$

- a generalization of the inner product of the form

$$f(x) = K(x_i, x_i'),$$

where K is some function that we will refer to as a *kernel*.

Support Vector Machine

- Linear kernel: $K(x_i, x_i') = \sum_{j=1}^{p} x_{ij} x_{i'j}$,
- Plynomial kernel: $K(x_i, x_i') = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d; \quad d > 1$
- Radial Basis Kernel: $K(x_i, x_i') = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^r)$, $\gamma > 0$

Support Vector Machine

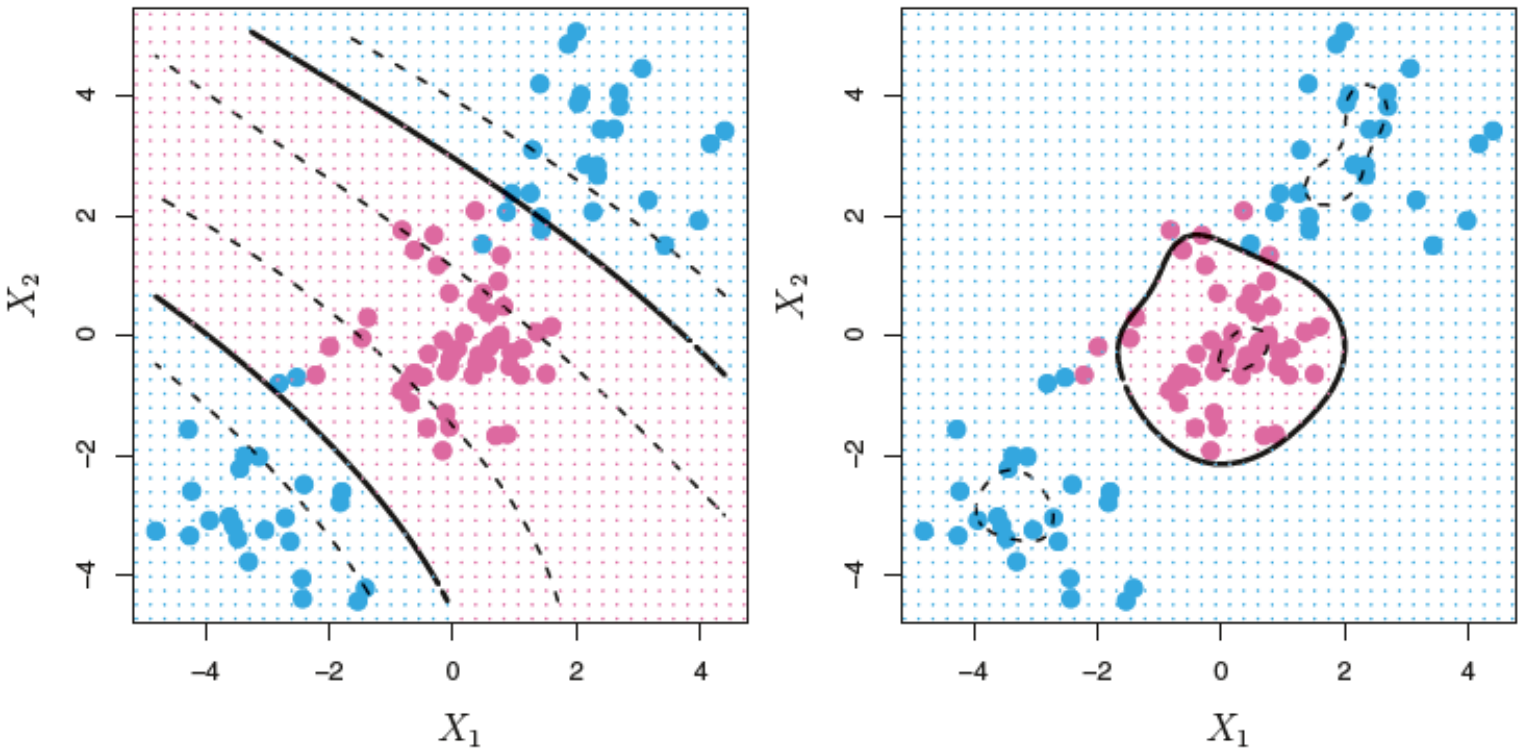


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

Practical Session with R

- R session on Support Vector Machine

Single Layer Neural Network

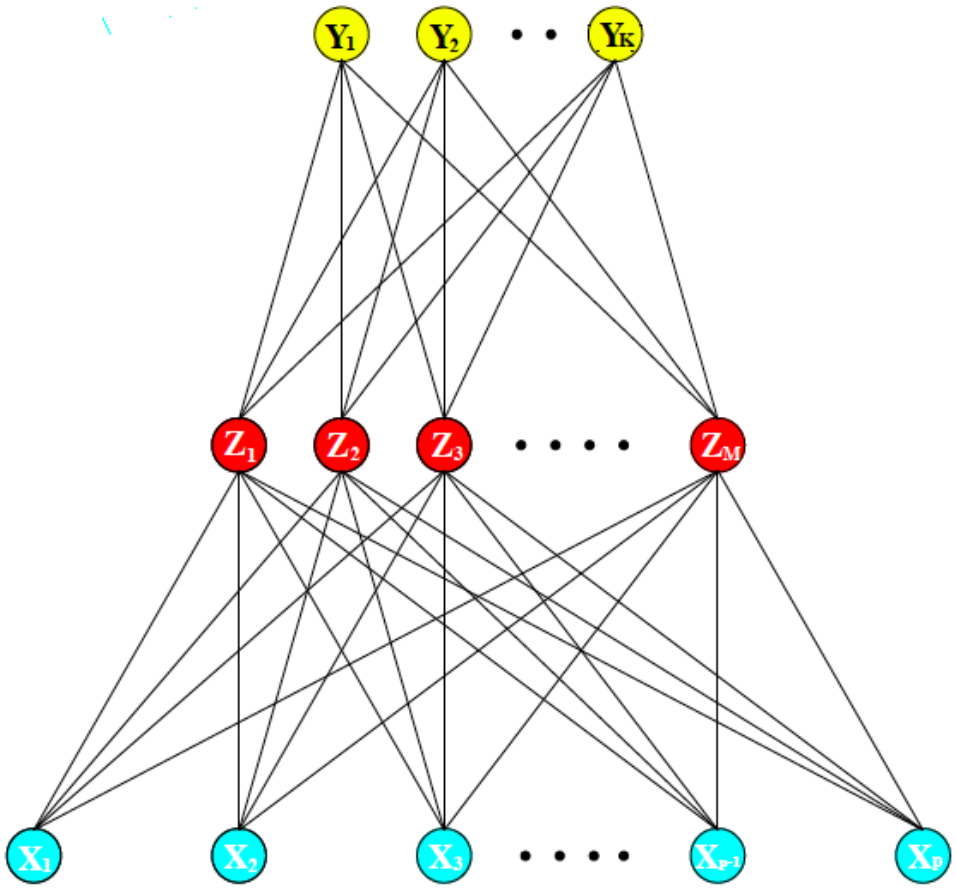
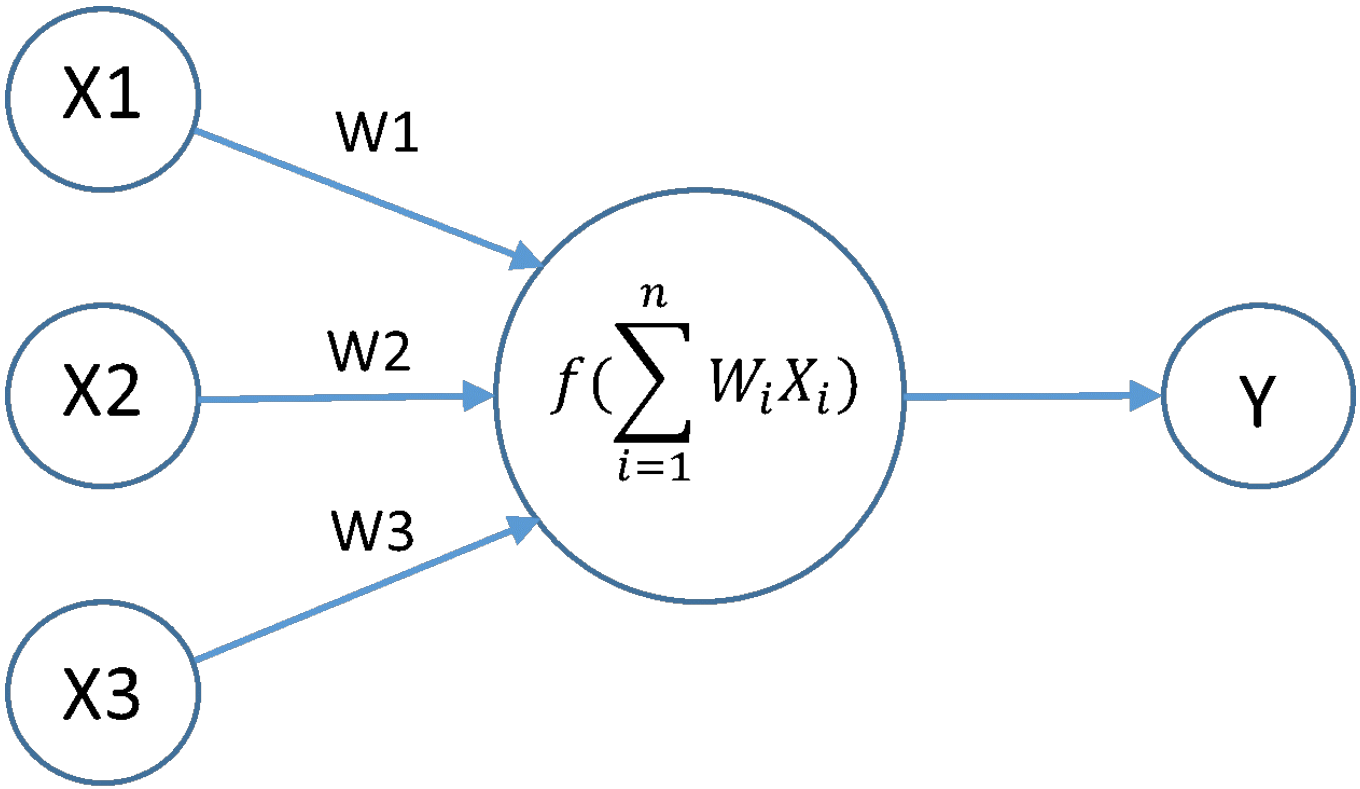


FIGURE 11.2. *Schematic of a single hidden layer, feed-forward neural network.*

Single Layer Neural Network



Neural Network

- For K-class classification, with the k^{th} unit modeling the probability of class k.
- There are K target measurements $Y_k,\ k = 1, \ldots, K$, each being coded as a $0-1$ variable for the k^{th} class.
- Derived features Z_m are created from linear combinations of the inputs, and then the target Y_k is modeled as a function of linear combinations of the Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T),\ m = 1, \ldots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K, \\ f_k(X) &= g_k(T),\ k = 1, \ldots, K, \end{aligned}$$

  where $Z = (Z_1, Z_2, \ldots, Z_m)$, and $T = (T_1, T_2, \ldots, T_K)$.

- The activation function $\sigma(v) = \frac{1}{(1+e^{-v})}$

Fitting Neural Network

- We denote the complete set of weights by θ, which consists of

$$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}\ M(p+1)\ \textit{weights}$$
$$\{\beta_0, \beta_k; k = 1, 2, \ldots, K\}\ K(M+1)\ \textit{weights}$$

- For regression, use sum-of-squared errors as objective function

$$R(\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2$$

- For classification, use cross-entropy as objective function

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i),$$

and the corresponding classifier is $G(x) = \textit{argmax}_k f_k(x)$

Fitting Neural Network

- We denote the complete set of weights by θ, which consists of

$$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}\ M(p+1)\ \textit{weights}$$
$$\{\beta_0, \beta_k; k = 1, 2, \ldots, K\}\ K(M+1)\ \textit{weights}$$

- For regression, use sum-of-squared errors as objective function

$$R(\theta) = \sum_{i=1}^{N} \underbrace{\Big[\sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \Big]}_{R_i}$$

- For classification, use cross-entropy as objective function

$$R(\theta) = -\sum_{i=1}^{N} \underbrace{\Big[\sum_{k=1}^{K} y_{ik} \log f_k(x_i) \Big]}_{R_i},$$

and the corresponding classifier is $G(x) = \textit{argmax}_k f_k(x)$

Fitting Neural Network

- The generic approach to minimizing $R(\theta)$ is by gradient descent, called back-propagation in this setting.
- Because of the compositional form of the model, the gradient can be easily derived using the chain rule for differentiation.
- This can be computed by a forward and backward sweep over the network model, keeping track only of quantities local to each unit.

Fitting Neural Network

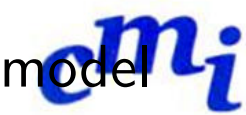
- Here is back-propagation in detail for squared error loss.
- Let $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ and $z_i = (Z_{1i}, z_{2i}, \ldots, z_{Mi})$. Then we have

$$R(\theta) = \sum_{i=1}^{N} R_i$$

with derivatives

$$\begin{aligned}
\frac{\partial R_i}{\partial \beta_{km}} &= -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi} = \delta_{ki} z_{mi}, \\
\frac{\partial R_i}{\partial \alpha_{ml}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il} \\
&= s_{mi} x_{il}
\end{aligned}$$

- The quantities δ_{ki} and s_{mi} are "**errors**" from the current model at the output and hidden layer units, respectively.



Back-propagation equation

$$\begin{aligned}
\frac{\partial R_i}{\partial \beta_{km}} &= -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi} = \delta_{ki} z_{mi}, \\
\frac{\partial R_i}{\partial \alpha_{ml}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il} \\
&= s_{mi} x_{il}
\end{aligned}$$

- The quantities δ_{ki} and s_{mi} are "**errors**" from the current model at the output and hidden layer units, respectively.
- From their definitions, these errors satisfy

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km}\delta_{ki},$$

known as the *back-propagation* equations.

Gradient Descent

- Given these *back-propagation* equations, a gradient descent update at the $(r+1)st$ iteration has the form

$$\begin{aligned}\beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}} \\ \alpha_{ml}^{(r+1)} &= \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}},\end{aligned}$$

  where γ_r is the learning rate.

- With the *back-propagation* equations; these *gradient descent* update can be implemented with a *two-pass* algorithm.

- If $r \to \infty$, $\gamma_r \to 0$, $\sum_r \gamma_r = \infty$ and $\sum_r \gamma_r^2 < \infty$ ensures convergence. For example, $\gamma_r = 1/r$

Two-pass algorithm

- In the forward pass, the current weights are fixed and the predicted values $\hat{f}_k(x_i)$ are computed from the NN model.
- In the backward pass, the errors δ_{ki} are computed, and then back-propagated via *back-propagation* equations to give the errors s_{mi}.
- Both sets of errors are then used to compute the gradients for the updates in gredient descent equations.
- This two-pass procedure is what is known as back-propagation algorithm.
- It has also been called the *delta rule*.
- The algorithm can be implemented efficiently on a parallel architecture computer.

Practical Session with R

- We will use `neuralnet` package.
- The `neuralnet` only deals with quantitative variables.
- We can transform all the qualitative variables (factors) to binary ("dummy") variables

Performance Measure

Overfitting

- Model is too specific
  - Tailored to fit anomalies in training data
  - Performs suboptimally on general data
- Variable Selection
  - Forward selection:
    - In this approach, one adds variables to the model one at a time.
    - The most significant of these variables is added to the model, so long as it's P-value is below some pre-set level.
  - Backward selection:
    - one starts with fitting a model with all the variables of interest. Then the least significant variable is dropped
    - continue by successively re-fitting reduced models and applying the same rule until all remaining variables are statistically significant.

Evaluating a classifier

- *Accuracy* What fraction of predictions are correct?
  - Need access to an "oracle" to validate answers
- Classification is often asymmetric
  - Suppose 1% of email traffic constitutes phishing
  - An email filter that always says "No" is 99% accurate, but totally useless!
- *Note:* Conventional to assume that "Yes" is the minority answer
- Need a finer classification of correct predictions and errors

Evaluating a classifier

- *Accuracy* What fraction of predictions are correct?
  - Need access to an "oracle" to validate answers
- Classification is often asymmetric
  - Suppose 1% of email traffic constitutes phishing
  - An email filter that always says "No" is 99% accurate, but totally useless!
- *Note:* Conventional to assume that "Yes" is the minority answer
- Need a finer classification of correct predictions and errors

Evaluating a classifier

- *Accuracy* What fraction of predictions are correct?
  - Need access to an "oracle" to validate answers
- Classification is often asymmetric
  - Suppose 1% of email traffic constitutes phishing
  - An email filter that always says "No" is 99% accurate, but totally useless!
- *Note:* Conventional to assume that "Yes" is the minority answer
- Need a finer classification of correct predictions and errors

Evaluating a classifier . . .

Confusion matrix

	Classified positive	Classified negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Precision

What fraction of positive classifications are correct?

$$p = \frac{TP}{TP + FP}$$

Recall

What fraction of actual positive cases are correctly classified?

$$r = \frac{TP}{TP + FN}$$

Evaluating a classifier . . .

	Classified positive	Classified negative
Actual Positive	1	99
Actual Negative	0	1000

Here $p = 1$ but $r = 0.01$

- No functional relationship between p and r
- In practice, they are typically inversely related—increasing p reduces r and vice versa
  - Conservative classifier — higher precision, ignores valid cases
  - Permissive classifier — higher recall, more mistakes

Evaluating a classifier . . .

	Classified positive	Classified negative
Actual Positive	1	99
Actual Negative	0	1000

Here $p = 1$ but $r = 0.01$

- No functional relationship between p and r
- In practice, they are typically inversely related—increasing p reduces r and vice versa
  - Conservative classifier — higher precision, ignores valid cases
  - Permissive classifier — higher recall, more mistakes

Evaluating a classifier . . .

Combine p, r into a single *F-Score*–weighted harmonic mean

$$F = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{r}} = \frac{(\beta^2+1)pr}{\beta^2 p + r}$$

where $\alpha \in [0,1]$ and $\beta^2 = \frac{1-\alpha}{\alpha}$

$$F_{\beta=1} = \frac{2pr}{p+r}$$

Evaluating a classifier . . .

- Who provides the "oracle" to validate answers?
- *Holdout sets (aka. Test Set)*
  - Exclude a random sample of training data
  - Build classifier on remaining data, check answers on holdout set
  - Suitable if we have a large volume of training data
- *Cross validation*
  - Systematically exclude $1/n$ of training data
  - Build classifier on remaining data and check answers on excluded set
  - Repeat n times to span entire training data
  - Aggregate the scores obtained

Evaluating a classifier . . .

- Who provides the "oracle" to validate answers?
- *Holdout sets (aka. Test Set)*
  - Exclude a random sample of training data
  - Build classifier on remaining data, check answers on holdout set
  - Suitable if we have a large volume of training data
- *Cross validation*
  - Systematically exclude $1/n$ of training data
  - Build classifier on remaining data and check answers on excluded set
  - Repeat n times to span entire training data
  - Aggregate the scores obtained

Evaluating a classifier . . .

- Who provides the "oracle" to validate answers?
- *Holdout sets (aka. Test Set)*
  - Exclude a random sample of training data
  - Build classifier on remaining data, check answers on holdout set
  - Suitable if we have a large volume of training data
- *Cross validation*
  - Systematically exclude $1/n$ of training data
  - Build classifier on remaining data and check answers on excluded set
  - Repeat n times to span entire training data
  - Aggregate the scores obtained

Thank You

sourish@cmi.ac.in