ISyE DDA – Agenda for 04/04/23 Lecture (Lec.22)

I) Resampling and Ensemble Methods

(a) Resampling: Bootstrap and Jackknife (presented)

(b) **Ensemble Methods: Bagging, Stacking,** (presented)

Boosting, Random Forest (today's focus)

II) Model Assessments

III) In-Class Exam-2 Topics

IV) Past In-Class Exam-2 Problems

Detailed Lectures:

I) Ensemble Methods: Boosting, Random Forest

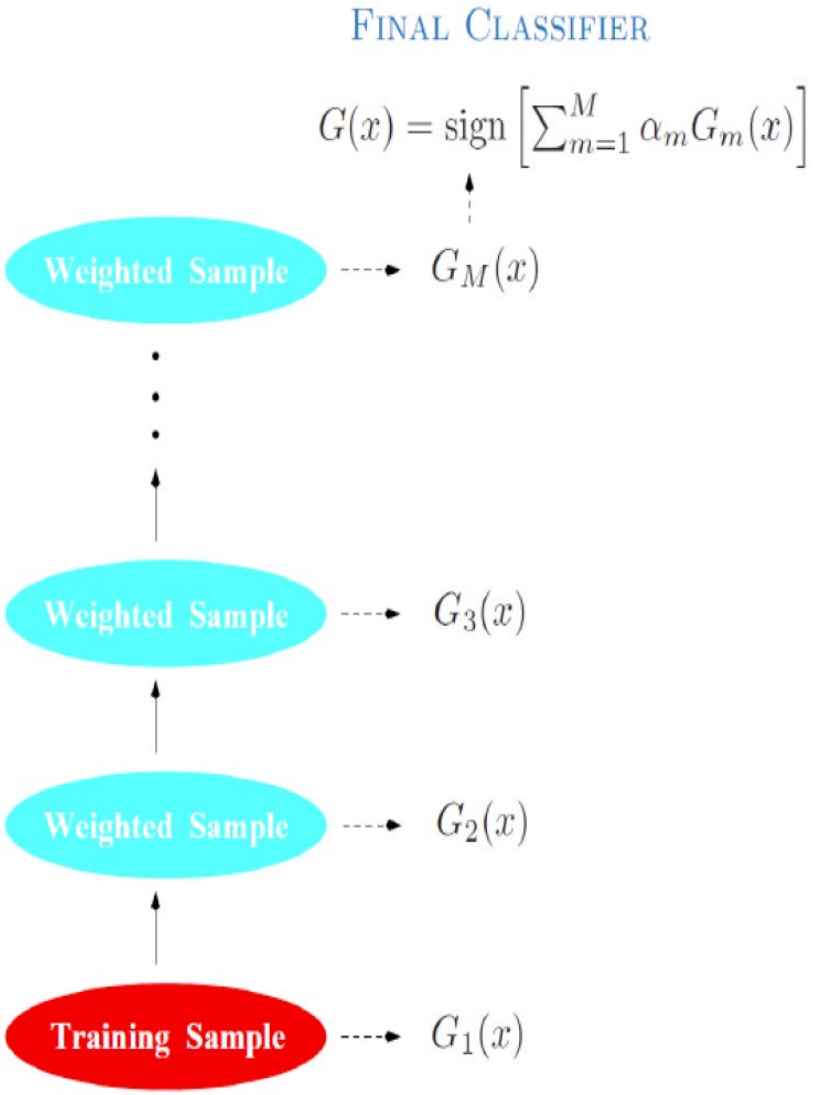


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Two Important Concepts:
(1) Learn from mistakes in building a new classification model;
(2) Trust quality-model more.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

2. For $m = 1$ to M:

Build a new classifier by emphasizing on learning from mistakes, i.e., weight (wi) the mis-classified cases more.

(a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

(b) Compute

Indicator = 1 for the mis-classification case.

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

Err = 0 is for the case "no mis-classification".

Alpha is the "learning rate".

(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

When err is zero, alpha becomes infinity; When err is ONE (largest), alpha becomes negative infinity.

When the indicator is zero (i.e., no mis-classification), exp(.) = 1, wi stays the same.

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

Only adjust the wi weight for mis-classified cases.

When alpha is infinity, exp(.) = infinity; When alpha is negative infinity, exp(.) becomes 0.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Conclusion: When a MODEL has zero err, alpha becomes infinity, i.e., the MODEL is well trusted. When the MODEL is trusted, the weight-adjustment wi will become larger. Finally, in the FINAL MODEL, the well-trusted MODEL will be weighted more.

4) Random Forest:

Random forests (Breiman, 2001) is a substantial modification of bagging that builds a large collection of *de-correlated* trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.

Boosting Weight Examples:

Err	(1-Err)/Err	Alpha_m = Log[(1-Err)/Err]	Exp(Alpha_m)		Model-Quality	
			weight-factor			
0.8	0.25	-0.602059991	0.547682253		Not a good model	
0.5	1	0	1			
0.4	1.5	0.176091259	1.192546884		Good model	

An Algorithm for Random Forest (Tree Constructions):

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

An Example – Comparison Between Bagging, Random Forest and Boosting Procedures:

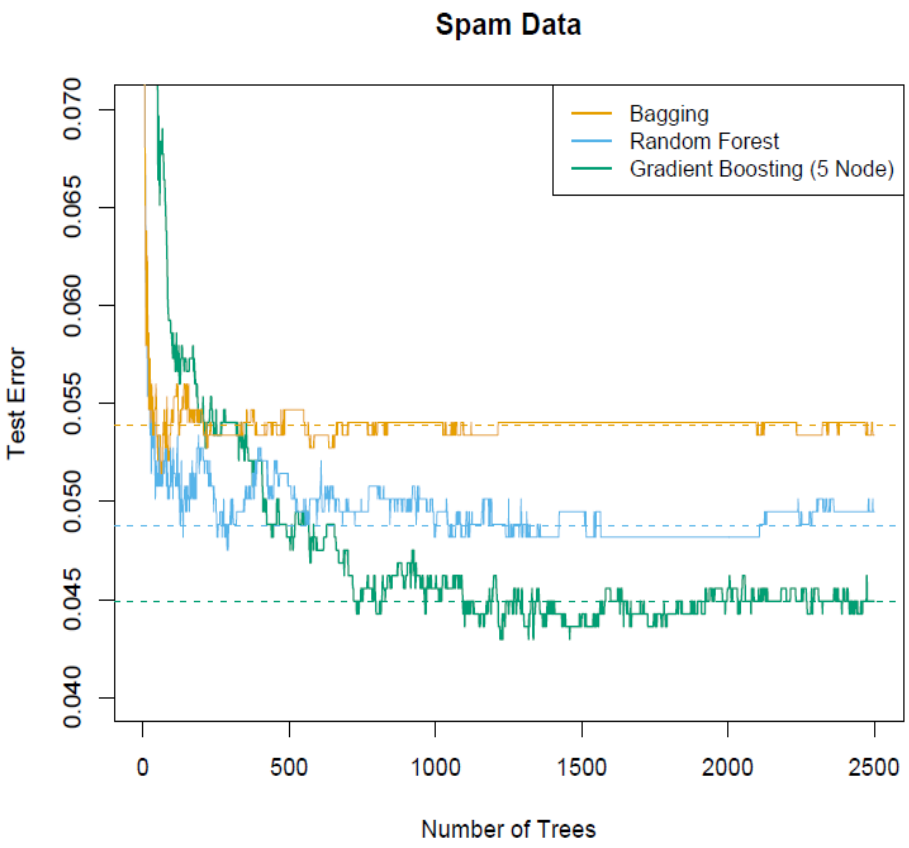


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

II) Model Assessment:

Reference: Hastie, T., Tibshirani, R., and Friedman, J. (2017), *The Elements of Statistical Learning*, Springer, New York. **Note that this book is a typical textbook used in data mining or/and machine learning classes.** Downloadable pdf file: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf

In this chapter we describe and illustrate the key methods for performance assessment, and show how they are used to select models. We begin the chapter with a discussion of the interplay between bias, variance and model complexity.

7.2 Bias, Variance and Model Complexity

Figure 7.1 illustrates the important issue in assessing the ability of a learning method to generalize. Consider first the case of a quantitative or interval scale response. We have a target variable Y, a vector of inputs X, and a prediction model $\hat{f}(X)$ that has been estimated from a training set $\mathcal{T}$. The loss function for measuring errors between Y and $\hat{f}(X)$ is denoted by $L(Y, \hat{f}(X))$. Typical choices are

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases} \tag{7.1}$$

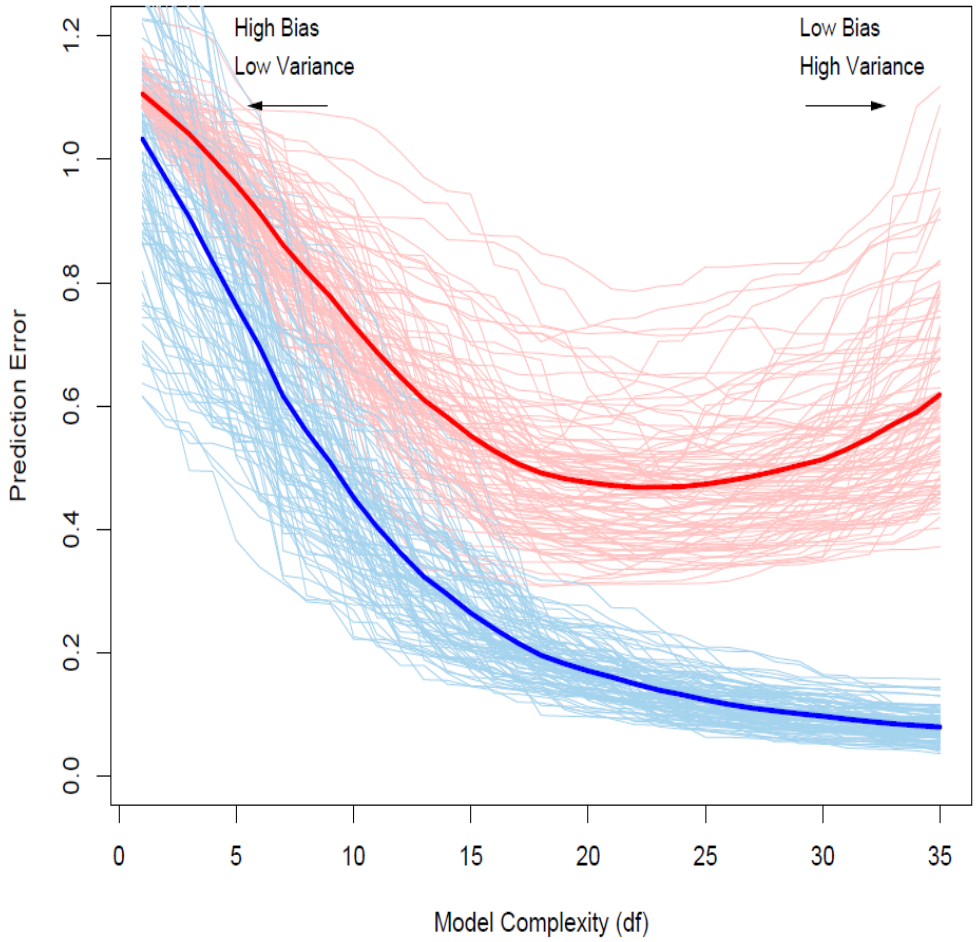


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$*.*

Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-to-noise ratio in the data and the training sample size. A typical split might be 50% for training, and 25% each for validation and testing:

Training error is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)). \tag{7.4}$$

Test error, also referred to as *generalization error*, is the prediction error over an independent test sample

$$\text{Err}_{\mathcal{T}} = \text{E}[L(Y, \hat{f}(X))|\mathcal{T}] \tag{7.2}$$

where both X and Y are drawn randomly from their joint distribution (population). Here the training set $\mathcal{T}$ is fixed, and test error refers to the error for this specific training set. A related quantity is the expected prediction error (or expected test error)

$$\text{Err} = \text{E}[L(Y, \hat{f}(X))] = \text{E}[\text{Err}_{\mathcal{T}}]. \tag{7.3}$$

Note that this expectation averages over everything that is random, including the randomness in the training set that produced $\hat{f}$.

III) In-Class Exam-2 Subjects – focus on "high level" concepts

1. **Decision Models (30%)**

   i) Decision in An Uncertain Environment (Constrained-Optimization Model Formulation)

   ii) Game-Theoretic Models

2. **Advanced Data Modeling (70%)**

   i) Classification (LDA/QDA, SVM)

   ii) Nonparametric Regression

   iii) Cluster Analysis (K-Means, Multivariate Gaussian Mixtures)

   iv) Dimension Reduction (PCA, PLS, MDS, SOM)

   v) Variable-Selections (Stepwise, All-Subsets, Ridge Regression, Lasso)

   vi) Resampling and Ensemble Methods (Bootstrap Sampling; Cross-Validation, Boosting, Random Forest)

IV) Past In-Class Exam-2 Problems:

A) **Essay questions: [50 points in total]**

11. Briefly describe boosting's three key properties discussed in lectures. [9 points; 3 points each]

Answers:

(1) "Trust **high-quality model** more" by assigning a larger value to α_m in the weighted average based (final) prediction model.

(2) "Learn from **mistakes**" by updating weights w_i to **mis-classified** cases.

(3) "Use **weighted** data-modeling or -analysis" procedure to construct G_2, G_3, …, G_m models in the boosting process.

12. Briefly describe Principal Component Analysis' (PCA) three key properties. [10 points]

Answers:

(1) All Principal Components (PCs) are mutually orthogonal to each other. [3 points]

(2) All PCs are linearly weighted sum of the original X-variables. [3 points]

(3) The variances (σ^2_i for i = 1, 2, …, p) captured by the first, second, third, …, p-th PCs are in the descending order, i.e., $\sigma^2_1 \geq \sigma^2_2 \geq \ldots \geq \sigma^2_p$. [4 points]

13. For understanding the physical meaning of the Multi-Dimensional Scaling (MDS) procedure described below (taken from Lecture-Notes #18), answer questions below.

Define $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$, the distance (dis-similarity) of two data points. MDS seeks values $z_1, z_2, \ldots, z_N \in R^k$ to minimize the so-called stress function

$$S_M(z_1, z_2, \ldots, z_N) = \Sigma_{i \neq k} (d_{ik} - \|z_i - z_k\|)^2 .$$

(1) What is the physical meaning of d_{ik} ? [3 points]

(2) What is the physical meaning of $c_{ik} = \| Z_i - Z_k \|$? [3 points]

(3) What is the physical meaning that MDS seeks values $Z_1, Z_2, \ldots, Z_N$ to minimize the above *stress function*? [5 points]

Answers:

(1) d_{ik} is the distance between two data points in the (original) X-variable scale. For example, consider an L_2-norm with p (=eg= 30) dimensional data x_i. The distance metric

$$d_{ik} = \text{sqrt}\{ (x_{i1} - x_{k1})^2 + (x_{i2} - x_{k2})^2 + \ldots + (x_{i1p} - x_{kp})^2 \}$$

measure the *similarity* between two X-variable locations, $\boldsymbol{X_i}$ and $\boldsymbol{X_k}$.

(2) Similarly, $c_{ik} = \| \boldsymbol{Z_i} - \boldsymbol{Z_k} \|$ measures the similarity of two Z-variable locations $\boldsymbol{Z_i}$ and $\boldsymbol{Z_k}$,

where in the Z-transformed scale, the Z-data variable has a much lower dimension q =eg= 7 (to 10). Again, $c_{ik} = \| \boldsymbol{Z_i} - \boldsymbol{Z_k} \|$ can be defined as the L_2-norm similar to the one defined for the X-variables.

(3) The stress function shown above aims to find the Z-variables that minimize the difference between X-variable distance metric and Z-variable distance metric such that these two sets of distance metrics are as "close" (defined in the square-error-loss format) as possible in describing the similarities in the data locations (at p- and q-dim scales, respectively).

14. Define profit Π(x1, x2; Y) as a function for two decision variables x1 and x2 and one random variable Y describing economic outlook for business.

(a) Use the notations taught in the lectures to **define a mixture-distribution model** for investors' outlook of **economic condition Y** in two directions, getting worse and getting better. [6 points]

(b) What is the typical optimization model used in the risk-neutral solution? [4 points]

(c) Formulate the **Probability-constraint-based Optimization (PCO) Model,** [5 points] and provide *detailed interpretation* **focusing on the "getting-worse" economic outlook situation** [5 points]

Answer:

(a) Denoted by Y_1 as the random variable for describing a "getting-better" economic outlook. Its distribution function (cdf) is $G_1(y_1)$. Similarly, denoted by Y_2 as the random variable for describing "**getting-worse**" economic outlook. Its distribution function (cdf) is $\mathbf{G_2(y_2)}$**.** Define an indicator random variable D = 1 if it is a better-outlook; 0, if it is a worse-outlook. Pr(D = 1) = p. Furthermore, assume that D, Y_1 and Y_2 are mutually *independent.* Then, the random variable for **any economic outlook condition Y** is a mixture of these two random variables as $Y = D^* Y_1 + (1 - D)^* Y_2$. Its distribution can be derived as $G(y) = p^* G_1(y) + (1 - p)^* G_2(y)$.

(b) The risk-neutral solution finds the optimal values of decision variables x1, x2, to maximize $E_G[\ \Pi(x1, x2; Y)\ ]$, where the expectation is taken with respect to (w.r.t.) the G(y) distribution for the **general economic outlook random variable Y**.

Grading Remarks: 2 pts assigned to the expectation-objective-function, and 2 pts assigned to the distribution G(y).

(c)

(i)The PCO model will find the optimal values of decision variables x1, x2, to maximize $E_G[\ \Pi(x1, x2; Y)\ ]$ subject to $Pr[\ \Pi(x1, x2; Y) \leq 20\% \mid$ **in the "getting-worse" economic outlook situation** $] \leq 5\%$.

(ii) The physical explanation of the constraint is that there is a very small chance (using 5% as an upper bound) for seeing the profit Π(x1, x2; Y) being lower than a value (using 20% as an example here), which is like the low-acceptable-value for business (money making) performance. The probability there is evaluated against the distribution $\mathbf{G_2(y_2)}$ for the "getting-worse" economic outlook situations.

B) **True and False Questions (5 points for each question and there is no partial credit).**

(True, False) 1. In K-means method a data-point could be in several clusters with different probabilities (sum to one).

Answer: False. There is no probability or distribution assumption for the K-means method.

(True, False) 2. AIC metric could be used in the Stepwise Regression.

Answer: False. AIC is used in the all-subsets regression. Stepwise regression is based on the partial-F-tests.

(True, False) 3. In the four ensemble methods (bagging, stacking, boosting and random-forest) taught in the lectures for regression predictions *bagging* is the only procedure uses equal-weights in the average of predictions from m = 1000 (eg) individual models fitted to re-sampled-data.

Answer: False. Random-forest also uses equal weights to average predictions.

(True, False) 4. Whether Y-data is normally or multinominal distributed, the results of PCA are the same.

Answer: True. PCA is an unsupervised learning method dealing with only X-data, where Y-data and its distribution are *not involved*.

(True, False) 5. AIC-, BIC- and PRESS-based all-subsets regression procedures are applicable to non-normal data and non-linear models. Similarly, the stepwise regression procedures have the same properties.

Answer: False. Stepwise regression depends on the normal distribution and linear model assumptions such that the partial-F-test has an F-distribution.

(True, False) 6. In the boosting procedure the weights for the mis-classified samples from the previous model are always more than the weights for the non-mis-classified samples.

Answer: False. In boosting the weights for the mis-classified samples are more than the weights for correctly classified samples – **this comparison is only valid for a set of samples**, i.e., comparison within a set of

samples, but not between two sets of samples or samples in different models. This is the property in boosting's algorithm Step-2(c). Thus, the sample-weight can be more or less than the weights given by the previous model.

(True, False) 7. When the X-data-columns are organized in different sequences (e.g., X1, X3, X4 versus X4, X3, X1), it is possible that forward-selection produces distinct models (even that alpha-to-enter is set at the same value, e.g., 15%).

Answer: True. The partial-F-test in the forward-selections takes the first (and then, the subsequent) variable(s) from the X-data-columns given by the user. Different sequences could produce distinct models.

(True, False) 8. One of the key differences in the LDA- and logistic-classification is that LDA assumes the distribution of X-variables (as a multivariate normal).

Answer: True. Logistic-classification treats X-variables as given constants, but LDA assumes them to have a

multivariate normal distribution (given a class of Y).

(True, False) 9. The weights for each one of the $m = 1000$ stacking-models and boosting-models are found based on certain optimization objectives.

Answer: False. It is true for the stacking method, but not true for the boosting method, where the model-weight is assigned as a given function of "Err (= error rate)", where no optimization objective function is used to derive the weights.

(True, False) 10. Even that the covariance between X and Y stays the same, PLS-results could change depending on the distribution assumption of Y-data, e.g., log-normal versus Weibull or Gamma.

Answer: False. PLS only works with the covariance between X and Y. The distribution (e.g., pdf) information is not involved.

(True, False) 11. The SVM-classification will minimize the mis-classification counts.

Answer: False. The SVM maximizes the **margin of support-vectors** in different classes.

(True, False) 12. In all variable-selection procedures taught in the lectures stepwise-regression is the only sequential procedure, i.e., the model selected in the latter stage will depend on the model(s) selected in the earlier stage(s).

Answer: True. The partial-F-test used in the stepwise regression is a sequential procedure. All other variable-selection procedures work with each possible model independently.

(True, False) 1. Cluster analysis works with explanatory variables (i.e., X-data). Thus, it is an unsupervised learning procedure.

(True, False) 2. In the Game-Theoretic problem presented in lectures, for a Retailer Stackelberg Decision Model, one should work on Retailer's Reaction Function

first before solving Manufacturers' optimal decision models.

Answer: False. In a Retailer Stackelberg Decision Model, the retailer is the leader and has more bargaining power. Therefore, one should first work on the manufacturer's optimal decision model before the retailer's reaction function. This is because the retailer has more power and has knowledge of how a manufacturer would react to different quantities. So, the retailer can condition on the information about the manufacturer's wholesale price and service when making their own decision to set retail price and order quantity.

(True, False) 3. A statistics-based classification procedure such as QDA will always be more accurate than a computing/optimization procedure such as SVM.

Answer: False. When all assumptions needed in a statistics-based classification procedure are satisfied, the statistical classification can be more accurate. However, when a certain assumption is not satisfied, its accuracy is not guaranteed. SVM will

do a better job in handling data with potential outliers compared to most statistical classifications.

(True, False) 4. The PCA procedure constructs *p* number of Principal Components (PCs), where *p* (=eg= 10) is the number of X-variables. Each PCs is a linearly weighted sum of the X-variables.

(True, False) 5. For optimizing objective functions involving uncertain components (i.e., random variables), one always takes expected values for random components to simplify the stochastic optimization problem into a *deterministic optimization* problem.

Answer: False. There are many alternative options in handling optimization with uncertainty. Lecture notes presented a few constrained optimization procedures to limit potential impact of uncertainty.

(True, False) 6. In spline fitting, when the tuning parameter λ is larger, the fitted regression function will be smoother.

Answer: True. One can consider the "extreme" case that when λ is equal to zero, the spline fit will go through all

data points to minimize the sum-of-squares of prediction-errors, i.e., the fitted regression will be the least smooth. Thus, when the tuning parameter λ is larger, the fitted regression function will be smoother.

(True, False) 7. Stepwise and all-subsets regressions are suitable for $p < n$ problems, where p = #x-variables and n = sample size. They are not suitable for handling "large p and small n" problems.

(True, False) 8. In clustering analysis with a Gaussian Mixture Model (GMM), it is possible that some data points have non-zero probabilities for being in a few different clusters.

Answer: True. The above characteristic is the key feature that distinguishes GMM-clustering with other clustering methods.

(True, False) 9. LDA and QDA are the only two Bayesian classification procedures discussed in lectures.

Answer: False. Bayes classifier is another Bayesian classification procedure presented in lectures.

(True, False) 10. All ensemble methods taught in lectures use bootstrapping methods to generate many data sets for fitting various data models. Then, the final model is an assembly (e.g., average) of these data models.

Answer: False. Boosting is not. *Boosting works with one data set, the original data set.* It assigns different weights in successive iterations. In each iteration, a new data model is constructed. The final model is a (weighted) average of these models.

ISyE DDA – Agenda for 03/30/23 Lecture (Lec.21)

I) Semester Project – PPT and Final Report Guidelines

II) Resampling and Ensemble Methods

(a) Resampling: Bootstrap and Jackknife (presented)

(b) **Ensemble Methods: Bagging, Stacking, Boosting, Random Forest**

III) Model Assessments

IV) In-Class Exam-2 Topics

V) Past In-Class Exam-2 Problems

Detailed Lectures:

I) Guidelines for Preparing PPTs and Final Reports

Suggestions for PPT-Designs

PPT will serve as a "skeleton" for the Final Report

According to our syllabus, 3% of credits are allocated to Presentation-PPT Designs. Even that we do not have live (or remote) presentations for your semester-projects, students need to design their presentation-PPTs. Every team is supposed to have 15 minutes for presenting their project experience. Suppose that there are **five main sections** in the project studies (e.g., Introduction and Problem Definition, LoM for Problem

Background, Decision Modeling, Data Modeling and Data Source and Conclusion/Future Work). Then, each section will have about 3 minutes. If one slide takes 20 - 30 seconds to present, there shall be about 6-8 pages per section. Totally, there might be 30-40 pages for the entire PPT. Of course, certain sections might need more pages than other sections do. If students have anything to discuss or ask me, please use email me at < JCLU@isye.gatech.edu >. Please remember to put "4034:" in your subject-line.

Suggestions for Section Contents in the Final Report

Let us use **bullets** to suggest section contents. Each bullet could be developed into one or two paragraphs. The total number of pages (without counting Appendix that includes more in-depth details) could be 30-40 pages.

Please remember to include a page of project title, brief table-of-contents and student names. Every page should be numbered.

Section 1 – Introduction

[1] Briefly describe (and motivate) your problem scope.

[2] State the **goal(s)** of your semester-project.

[3] A brief "review" of the problem background; LoM studies should be referred in the main text, but included in appendix.

[4] Briefly provide an overview of **problem solving ideas** and their rational; A brief roadmap

might be useful. Specific sections on decision models, data models and data sources could be cited in your roadmap. This roadmap also provides a structure for presenting subsequent sections.

Section 2 – Decision Models

[1] Outline the contents of your decision models and steps of formulating decision models. For example, native language-based decision modeling, notation definition for the components involved in the decision models, functional relationship for certain variables, and rigorous mathematical formulation of decision models.

[2] Use a hierarchical style for presenting **mathematically formulated decision models with critical explanations and justifications**; subsections (e.g., notations, decision variables,

objective functions and constraints) could be used to make the presentation easy to read.

[3] Briefly discuss possible method(s) (and associated software package(s)) to solve your optimization problem for decision variables.

Section 3 – Data Models

[1] Use general terms to discuss how your data models are needed for supporting optimization of your decision models.

[2] Use a hierarchical style to present data models rigorously. Please make sure notations are well defined, equations or expressions are explained and justified.

[3] Briefly discuss how to apply software package(s) to perform data modeling.

Section 4 – Data Sources

[1] Discuss what types of data are needed to execute your data modeling.

[2] Provide examples of sources for extracting or gathering needed data. For example, some data might be collected from interviewing "experts" (e.g., high school teachers).

[3] Because some data models include several common X-variables. Let us **use a subsection, "data source architecture", to organize needed data variables with respect to their background, similarity, and their potential sources or/and methods of collecting them.**

Section 5 – Conclusion, Future Work and Lesson Learned

[1] Briefly summarize key findings in this project study.

[2] Discuss future work and their potential impact to your current studies.

[3] Summarize lesson-learned from this project study.

Reference

Appendix

II) Resampling and Ensemble Methods

(b) Ensemble Methods: Bagging, Stacking, Boosting, Random Forest

1) Bagging:

Consider first the regression problem. Suppose we fit a model to our training data $\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, obtaining the prediction $\hat{f}(x)$ at input x. Bootstrap aggregation or *bagging* averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample $\mathbf{Z}^{*b}$, $b = 1, 2, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \qquad (8.51)$$

A more interesting example is a regression tree, where $\hat{f}(x)$ denotes the tree's prediction at input vector x (regression trees are described in Chapter 9). Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes. The bagged estimate is the average prediction at x from these B trees.

Now suppose our tree produces a classifier $\hat{G}(x)$ for a K-class response. Here it is useful to consider an underlying indicator-vector function $\hat{f}(x)$, with value a single one and $K-1$ zeroes, such that $\hat{G}(x) = \arg\max_k \hat{f}(x)$. Then the bagged estimate $\hat{f}_{\text{bag}}(x)$ (8.51) is a K-vector $[p_1(x), p_2(x), \ldots, p_K(x)]$, with $p_k(x)$ equal to the proportion of trees predicting class k at x. The bagged classifier selects the class with the most "votes" from the B trees, $\hat{G}_{\text{bag}}(x) = \arg\max_k \hat{f}_{\text{bag}}(x)$.

2) Stacking:

Stacked generalization, or *stacking*, is a way of doing this. Let $\hat{f}_m^{-i}(x)$ be the prediction at x, using model m, applied to the dataset with the

ith training observation removed. The stacking estimate of the weights is obtained from the least squares linear regression of y_i on $\hat{f}_m^{-i}(x_i)$, $m = 1, 2, \ldots, M$. In detail the stacking weights are given by

$$\hat{w}^{\text{st}} = \underset{w}{\text{argmin}} \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x_i) \right]^2. \tag{8.59}$$

The final prediction is $\sum_m \hat{w}_m^{\text{st}} \hat{f}_m(x)$. By using the cross-validated predictions $\hat{f}_m^{-i}(x)$, stacking avoids giving unfairly high weight to models with higher complexity. Better results can be obtained by restricting the weights to be nonnegative, and to sum to 1. This seems like a reasonable restriction

3) Boosting:

Boosting is one of the most powerful learning ideas introduced in the last twenty years. It was originally designed for classification problems, but as will be seen in this chapter, it can profitably be extended to regression as well. The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee."

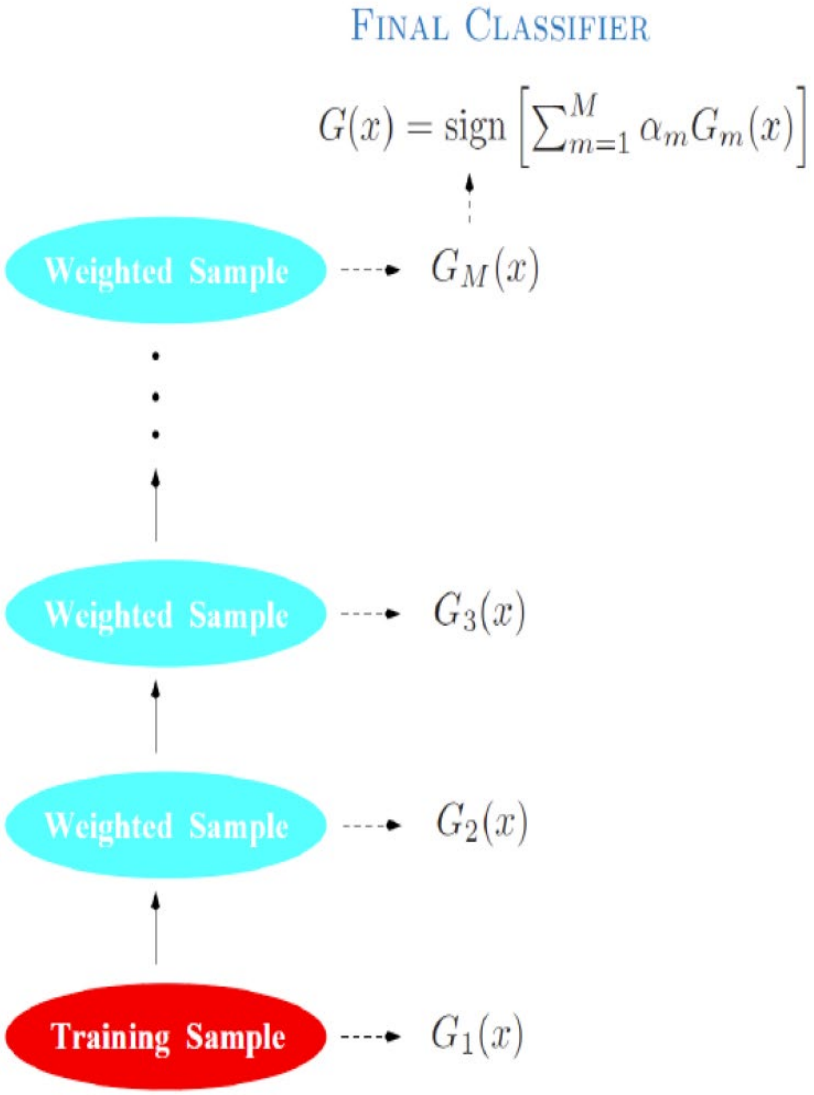


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Two Important Concepts:
(1) Learn from mistakes in building a new classification model;
(2) Trust quality-model more.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

2. For $m = 1$ to M:

Build a new classifier by emphasizing on learning from mistakes, i.e., weight (wi) the mis-classified cases more.

(a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

(b) Compute

Indicator = 1 for the mis-classification case.

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

Err = 0 is for the case "no mis-classification".

Alpha is the "learning rate".

(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

When err is zero, alpha becomes infinity; When err is ONE (largest), alpha becomes negative infinity.

When the indicator is zero (i.e., no mis-classification), exp(.) = 1, wi stays the same.

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

Only adjust the wi weight for mis-classified cases.

When alpha is infinity, exp(.) = infinity; When alpha is negative infinity, exp(.) becomes 0.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Conclusion: When a MODEL has zero err, alpha becomes infinity, i.e., the MODEL is well trusted. When the MODEL is trusted, the weight-adjustment wi will become larger. Finally, in the FINAL MODEL, the well-trusted MODEL will be weighted more.

4) Random Forest:

Random forests (Breiman, 2001) is a substantial modification of bagging that builds a large collection of *de-correlated* trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.

Boosting Weight Examples:

Err	(1-Err)/Err	Alpha_m = Log[(1-Err)/Err]	Exp(Alpha_m)		Model-Quality	
			weight-factor			
0.8	0.25	-0.602059991	0.547682253		Not a good model	
0.5	1	0	1			
0.4	1.5	0.176091259	1.192546884		Good model	

An Algorithm for Random Forest (Tree Constructions):

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

An Example – Comparison Between Bagging, Random Forest and Boosting Procedures:

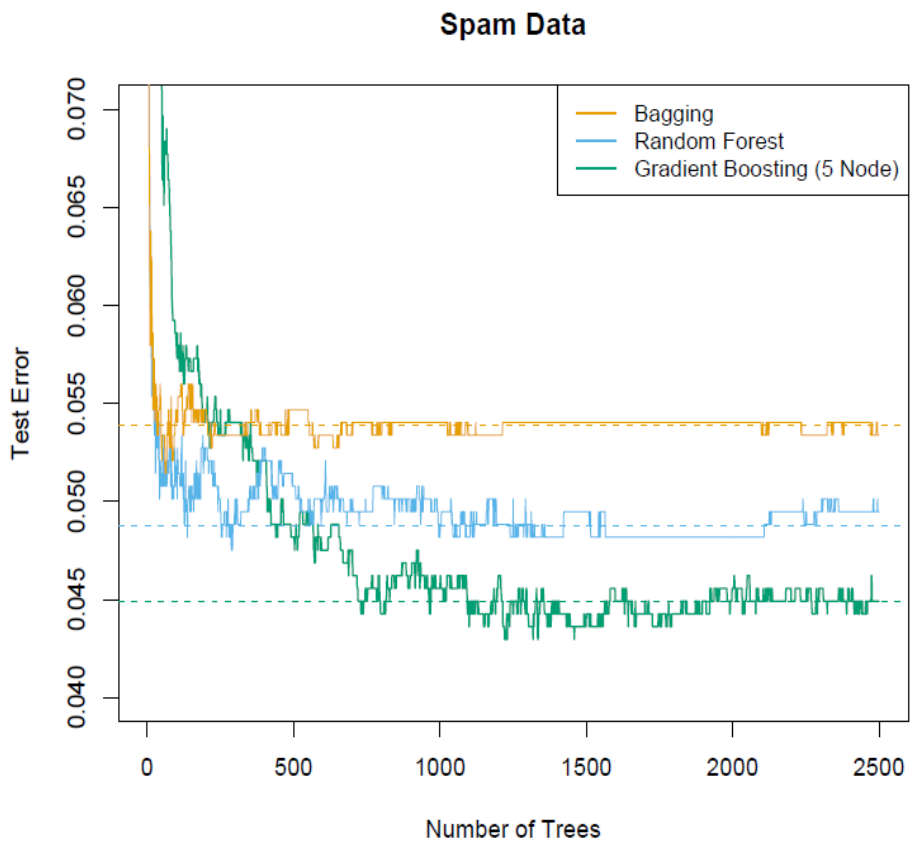


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

III) Model Assessment:

Reference: Hastie, T., Tibshirani, R., and Friedman, J. (2017), *The Elements of Statistical Learning*, Springer, New York. **Note that this book is a typical textbook used in data mining or/and machine learning classes.** Downloadable pdf file: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf

In this chapter we describe and illustrate the key methods for performance assessment, and show how they are used to select models. We begin the chapter with a discussion of the interplay between bias, variance and model complexity.

7.2 Bias, Variance and Model Complexity

Figure 7.1 illustrates the important issue in assessing the ability of a learning method to generalize. Consider first the case of a quantitative or interval scale response. We have a target variable Y, a vector of inputs X, and a prediction model $\hat{f}(X)$ that has been estimated from a training set $\mathcal{T}$. The loss function for measuring errors between Y and $\hat{f}(X)$ is denoted by $L(Y, \hat{f}(X))$. Typical choices are

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases} \tag{7.1}$$

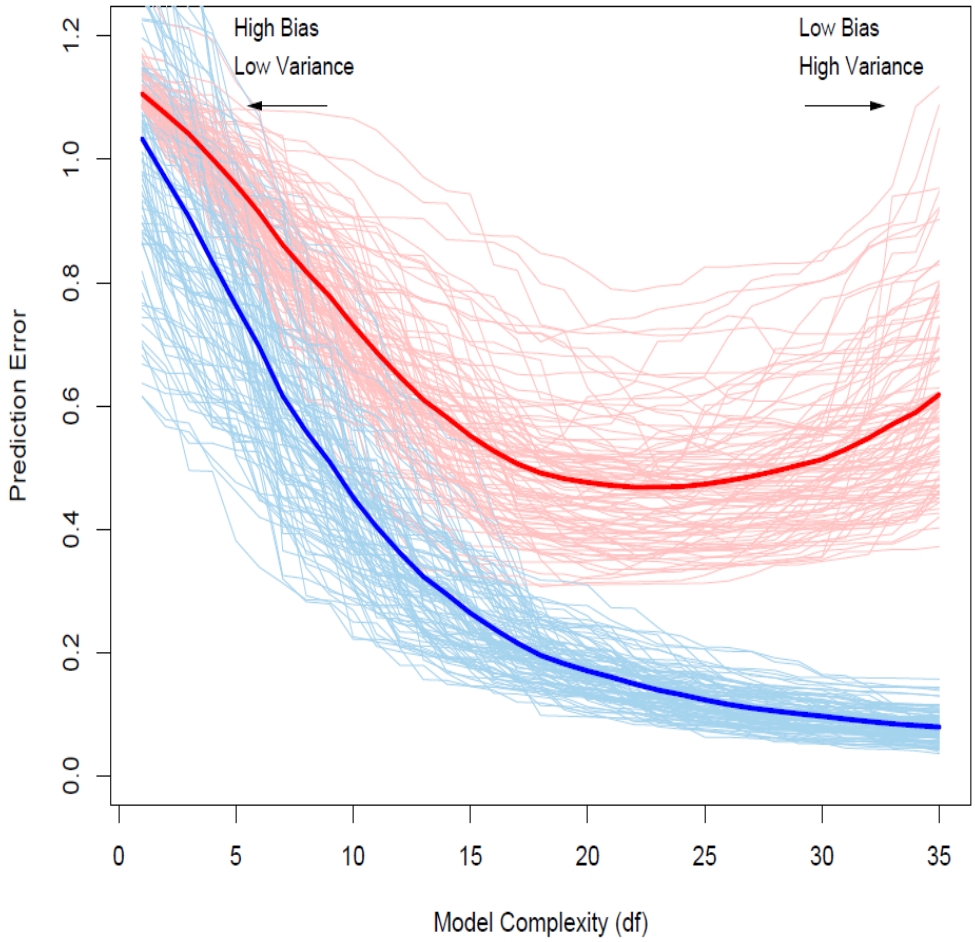


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Training error is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)). \tag{7.4}$$

Test error, also referred to as *generalization error*, is the prediction error over an independent test sample

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}[L(Y, \hat{f}(X))|\mathcal{T}] \tag{7.2}$$

where both X and Y are drawn randomly from their joint distribution (population). Here the training set $\mathcal{T}$ is fixed, and test error refers to the error for this specific training set. A related quantity is the expected prediction error (or expected test error)

$$\mathrm{Err} = \mathrm{E}[L(Y, \hat{f}(X))] = \mathrm{E}[\mathrm{Err}_{\mathcal{T}}]. \tag{7.3}$$

Note that this expectation averages over everything that is random, including the randomness in the training set that produced $\hat{f}$.

Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-to-noise ratio in the data and the training sample size. A typical split might be 50% for training, and 25% each for validation and testing:

The methods in this chapter are designed for situations where there is insufficient data to split it into three parts. Again it is too difficult to give a general rule on how much training data is enough; among other things, this depends on the signal-to-noise ratio of the underlying function, and the complexity of the models being fit to the data.

The methods of this chapter approximate the validation step either analytically (AIC, BIC, MDL, SRM) or by efficient sample re-use (cross-validation and the bootstrap). Besides their use in model selection, we also examine to what extent each method provides a reliable estimate of test error of the final chosen model.

IV) In-Class Exam-2 Subjects – focus on "high level" concepts

1. Decision Models (30%)

i) Decision in An Uncertain Environment (Constrained-Optimization Model Formulation)

ii) Game-Theoretic Models

2. Advanced Data Modeling (70%)

i) Classification (LDA/QDA, SVM)

ii) Nonparametric Regression

iii) Cluster Analysis (K-Means, Multivariate Gaussian Mixtures)

iv) Dimension Reduction (PCA, PLS, MDS, SOM)

v) Variable-Selections (Stepwise, All-Subsets, Ridge Regression, Lasso)

vi) Resampling and Ensemble Methods (Bootstrap Sampling; Cross-Validation, Boosting, Random Forest)

V) Past In-Class Exam-2 Problems:

A) **True and False Questions (5 points for each question and there is no partial credit).**

(True, False) 1. In K-means method a data-point could be in several clusters with different probabilities (sum to one).

Answer: False. There is no probability or distribution assumption for the K-means method.

(True, False) 2. AIC metric could be used in the Stepwise Regression.

Answer: False. AIC is used in the all-subsets regression. Stepwise regression is based on the partial-F-tests.

(True, False) 3. In the four ensemble methods (bagging, stacking, boosting and random-forest) taught in the

lectures for regression predictions *bagging* is the only procedure uses equal-weights in the average of predictions from m = 1000 (eg) individual models fitted to re-sampled-data.

Answer: False. Random-forest also uses equal weights to average predictions.

(True, False) 4. Whether Y-data is normally or multinominal distributed, the results of PCA are the same.

Answer: True. PCA is an unsupervised learning method dealing with only X-data, where Y-data and its distribution are *not involved*.

(True, False) 5. AIC-, BIC- and PRESS-based all-subsets regression procedures are applicable to non-normal data and non-linear models. Similarly, the stepwise regression procedures have the same properties.

Answer: False. Stepwise regression depends on the normal distribution and linear model assumptions such that the partial-F-test has an F-distribution.

(True, False) 6. In the boosting procedure the weights for the mis-classified samples from the previous model are

always more than the weights for the non-mis-classified samples.

Answer: False. In boosting the weights for the mis-classified samples are more than the weights for correctly classified samples – **this comparison is only valid for a set of samples**, i.e., comparison within a set of samples, but not between two sets of samples or samples in different models. This is the property in boosting's algorithm Step-2(c). Thus, the sample-weight can be more or less than the weights given by the previous model.

(True, False) 7. When the X-data-columns are organized in different sequences (e.g., X1, X3, X4 versus X4, X3, X1), it is possible that forward-selection produces distinct models (even that alpha-to-enter is set at the same value, e.g., 15%).

Answer: True. The partial-F-test in the forward-selections takes the first (and then, the subsequent) variable(s) from the X-data-columns given by the user. Different sequences could produce distinct models.

(True, False) 8. One of the key differences in the LDA- and logistic-classification is that LDA assumes the distribution of X-variables (as a multivariate normal).

Answer: True. Logistic-classification treats X-variables as given constants, but LDA assumes them to have a multivariate normal distribution (given a class of Y).

(True, False) 9. The weights for each one of the $m = 1000$ stacking-models and boosting-models are found based on certain optimization objectives.

Answer: False. It is true for the stacking method, but not true for the boosting method, where the model-weight is assigned as a given function of "Err (= error rate)", where no optimization objective function is used to derive the weights.

(True, False) 10. Even that the covariance between X and Y stays the same, PLS-results could change depending on the distribution assumption of Y-data, e.g., log-normal versus Weibull or Gamma.

Answer: False. PLS only works with the covariance between X and Y. The distribution (e.g., pdf) information is not involved.

(True, False) 11. The SVM-classification will minimize the mis-classification counts.

Answer: False. The SVM maximizes the **margin of support-vectors** in different classes.

(True, False) 12. In all variable-selection procedures taught in the lectures stepwise-regression is the only sequential procedure, i.e., the model selected in the latter stage will depend on the model(s) selected in the earlier stage(s).

Answer: True. The partial-F-test used in the stepwise regression is a sequential procedure. All other variable-selection procedures work with each possible model independently.

ISyE DDA – Agenda for 03/28/23 Lecture (Lec.20)

I) Large *p* Small *n* Variable-Selections

II) Resampling and Ensemble Methods

(a) Resampling: Bootstrap and Jackknife

(b) **Ensemble Methods: Bagging, Stacking, Boosting, Random Forest**

III) Model Assessments

IV) In-Class Exam-2 Topics

V) Past In-Class Exam-2 Problems

Detailed Lectures:

I) Large *p* Small *n* Variable-Selections

Review: Variable-Selections for Linear (Normal) Regressions

(1) Traditional Problems (n ≡ sample size > p ≡ #x-variables)

(a) Stepwise Regression (with Partial F-Tests) (ISyE 3030/4031 materials)

(b) All-Subsets Regression (ISyE 3030/4031 materials)

(2) Recent Problems (n < p & n << p) (our focus in ISyE 4034)

(a) Ridge Regression, Lasso, Elastic Net Regularization

(b) Adaptive Lasso and Group Lasso

(c) Clustering and Representative-Selection

New: Variable-Selections for Linear Regressions

- **Recent Problems (n < p & n << p)**
  - **(a) Ridge Regression, Lasso, Elastic Net Regularization**
  - **(b) Adaptive Lasso and Group Lasso**
  - **(c) Clustering and Representative-Selection**

Details:

Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

- In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum |\beta_j|$.

Lasso: Least Absolute Shrinkage and Selection Operator

II) Resampling and Ensemble Methods

(a) Resampling: Bootstrap and Jackknife

(b) Ensemble Methods: Bagging, Stacking, Boosting, Random Forest

Details:

[1] Purpose of Resampling Methods:

In statistics, **resampling** is any of a variety of methods for doing one of the following:

1. Estimating the precision of sample statistics (medians, variances, percentiles) by using subsets of available data (**jackknifing**) or drawing randomly with replacement from a set of data points (**bootstrapping**)
2. Exchanging labels on data points when performing significance tests (**permutation tests**, also called exact tests, randomization tests, or re-randomization tests)
3. Validating models by using random subsets (bootstrapping, cross validation).

[2] Bootstrap Resampling:

Original Samples: $\mathbf{Y}_{orig} \equiv \{Y_1, Y_2, \ldots, Y_n\}$ =eg = {2, 4, 10, 12, 7} with $n = 5$

Bootstrap Samples:

Perform a random sampling with replacement to collect n "new data" from the original samples. For example, {4, 7, 10, 4, 2} could be a set of bootstrap samples. Notice that "4" is repeated twice. This is possible due to the "sampling WITH replacement" procedure.

Denoted by the bootstrap samples as

$\mathbf{Y^{(1)}} \equiv \{Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_n^{(1)}\}$ =eg = {4, 7, 10, 4, 2};

$\mathbf{Y^{(2)}} \equiv \{Y_1^{(2)}, Y_2^{(2)}, \ldots, Y_n^{(2)}\}$;

…

$\mathbf{Y^{(m)}} \equiv \{Y_1^{(m)}, Y_2^{(m)}, \ldots, Y_n^{(m)}\}$,

where m is a large number (e.g., 10,000) to facilitate further distributional studies of the bootstrap samples.

Bias Evaluation: The procedure explains below is applicable for evaluating bias in any sample-estimate, e.g., sample-mean, -median, 90th-percentile.

Continue the example above. Denoted by θ_{orig}_hat as the sample-estimate from the original samples. Then, the **bootstrapped sample-estimates** are as follows:

$$\theta^{(1)}\_hat,\ \theta^{(2)}\_hat,\ \ldots,\ \theta^{(m)}\_hat, \qquad (1)$$

for each one of the bootstrapped samples $\mathbf{Y}^{(i)}$, $i = 1, 2, \ldots, m$, respectively.

Then, get an average ($\theta^{(i)}\_hat\_ave$) of these bootstrapped sample-estimates. Bias will be defined as

$$\text{Bias} = \theta_{orig}\_hat - (\theta^{(i)}\_hat\_ave),$$

where the average ($\theta^{(i)}\_hat\_ave$) acts like the population-average E(θ_{orig}_hat) in evaluating this bias.

Variance-Estimation: Continue the example above. Based on m quantities of bootstrapped sample-estimates given in Eq.(1), one can calculate the following bootstrap-variance. By taking a square-root, one obtains the bootstrap standard-deviation (sd).

$$\sigma_b^2\_\text{hat} = \Sigma_{i=1}^{m} \left(\theta^{(i)}\_hat - (\theta^{(i)}\_hat\_ave) \right]^2 / \left(m - 1 \right).$$

This bootstrap-variance estimates the population variance σ^2 of the original samples $\mathbf{Y}_{orig}$.

Note that many people use **bootstrap Mean Squared Error**, $\text{MSE} = (\text{Bias})^2 + \text{Variance}$, to evaluate the quality of estimating a population parameter θ (e.g., mean, median or 90$^{\text{th}}$-percentile).

Various **bootstrap-based Confidence Intervals** (CIs) can be constructed for estimating population parameter θ.

Students of interest should look into lecture-notes from ISyE-6404 (Nonparametric Statistic) or Internet-materials.

[3] Jackknife Resampling:

Bootstrap-resampling could be computing intensive (and thus, time consuming), the following jackknife-resampling method is less computing intensive. Unlike the sampling with replacement bootstrap method, Jackknife is NOT sampling with replacement.

Original Samples: $\mathbf{Y}_{orig} \equiv \{Y_1, Y_2, \ldots, Y_n\}$ =eg = {2, 4, 10, 12, 7} with $n = 5$

Delete-1 Jackknife Samples:

$\mathbf{Y}_{(1)} \equiv \{ 4, 10, 12, 7 \}$, where the first sample "2" is not included;

$\mathbf{Y}_{(2)} \equiv \{ 2, 10, 12, 7 \}$; where the second sample "4" is not included;

…

$\mathbf{Y}_{(n)} \equiv \{ 2, 4, 10, 12 \}$, where the last sample "4" is not included.

Naturally, it is required to have a large sample size *n* to create many Jackknife samples. Then, the methods introduced in the bootstrap for estimating bias, variance and MSE can be applied to these jackknife samples.

Remark: Instead of deleting one sample in creating the Jackknife re-sample distribution, some people extend this method to delete-*d*-Jackknife, where *d*-samples (e.g., $d = 2, 3, \ldots$) are deleted in each set of Jackknife sample.

[4] Cross-Validation Resampling:

Although the bootstrap method is applicable to regression problem, where each data point includes Y (outcome) and many X*j*'s (input-variables), many people prefer to use the following cross-validation method to evaluate the quality of model-predictions.

Original Samples: ($Y_{i,orig}$, $X_{ij,orig}$, $j = 1, 2, \ldots, p$) with $i = 1, 2, \ldots, n$ for ***n* sets of iid regression samples.**

Cross-validate Estimation:

Step-1: Just like the delete-1 Jackknife sampling procedure. One regression-data-point is removed to create one set of "cross-validated-samples".

Step-2: Use each set of the delete-1 "cross-validated-samples" to build a regression model; it could be linear, nonlinear, GLM or non-parametric regression model. Then, use the "delete-1-built-model" to predict the data-point, which was deleted in the model-building process. Evaluate its "*cross-validated-prediction-error*". Repeat this process for all *n* sets of delete-1 "cross-validated-samples".

Step-3: Calculate the variance from *n* "*cross-validated-prediction-errors*". This is the "cross-validation-variance".

Remark: This cross-validation method is popular in model-selections. For example, in the nonparametric regression there are tuning parameters like window-size *h* in Kernel-regression or smoothing-penalty λ in Spline. By

trying different window-sizes (e.g., $h = 0.2, 0.5, \ldots$), one obtain their corresponding cross-validation-variance (e.g., 3.5, 2.7, …). Select the h with smallest cross-validation-variance for this Kernel-regression.

(b) Ensemble Methods: Bagging, Stacking, Boosting, Random Forest

1) Bagging:

Consider first the regression problem. Suppose we fit a model to our training data $\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, obtaining the predic-

tion $\hat{f}(x)$ at input x. Bootstrap aggregation or *bagging* averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample $\mathbf{Z}^{*b}$, $b = 1, 2, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{8.51}$$

A more interesting example is a regression tree, where $\hat{f}(x)$ denotes the tree's prediction at input vector x (regression trees are described in Chapter 9). Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes. The bagged estimate is the average prediction at x from these B trees.

Now suppose our tree produces a classifier $\hat{G}(x)$ for a K-class response. Here it is useful to consider an underlying indicator-vector function $\hat{f}(x)$, with value a single one and $K-1$ zeroes, such that $\hat{G}(x) = \arg\max_k \hat{f}(x)$. Then the bagged estimate $\hat{f}_{\text{bag}}(x)$ (8.51) is a K-vector $[p_1(x), p_2(x), \ldots, p_K(x)]$, with $p_k(x)$ equal to the proportion of trees predicting class k at x. The bagged classifier selects the class with the most "votes" from the B trees, $\hat{G}_{\text{bag}}(x) = \arg\max_k \hat{f}_{\text{bag}}(x)$.

2) Stacking:

Stacked generalization, or *stacking*, is a way of doing this. Let $\hat{f}_m^{-i}(x)$ be the prediction at x, using model m, applied to the dataset with the

ith training observation removed. The stacking estimate of the weights is obtained from the least squares linear regression of y_i on $\hat{f}_m^{-i}(x_i)$, $m = 1, 2, \ldots, M$. In detail the stacking weights are given by

$$\hat{w}^{\text{st}} = \underset{w}{\text{argmin}} \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x_i) \right]^2. \tag{8.59}$$

The final prediction is $\sum_m \hat{w}_m^{\text{st}} \hat{f}_m(x)$. By using the cross-validated predictions $\hat{f}_m^{-i}(x)$, stacking avoids giving unfairly high weight to models with higher complexity. Better results can be obtained by restricting the weights to be nonnegative, and to sum to 1. This seems like a reasonable restriction

3) Boosting:

Boosting is one of the most powerful learning ideas introduced in the last twenty years. It was originally designed for classification problems, but as will be seen in this chapter, it can profitably be extended to regression as well. The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee."

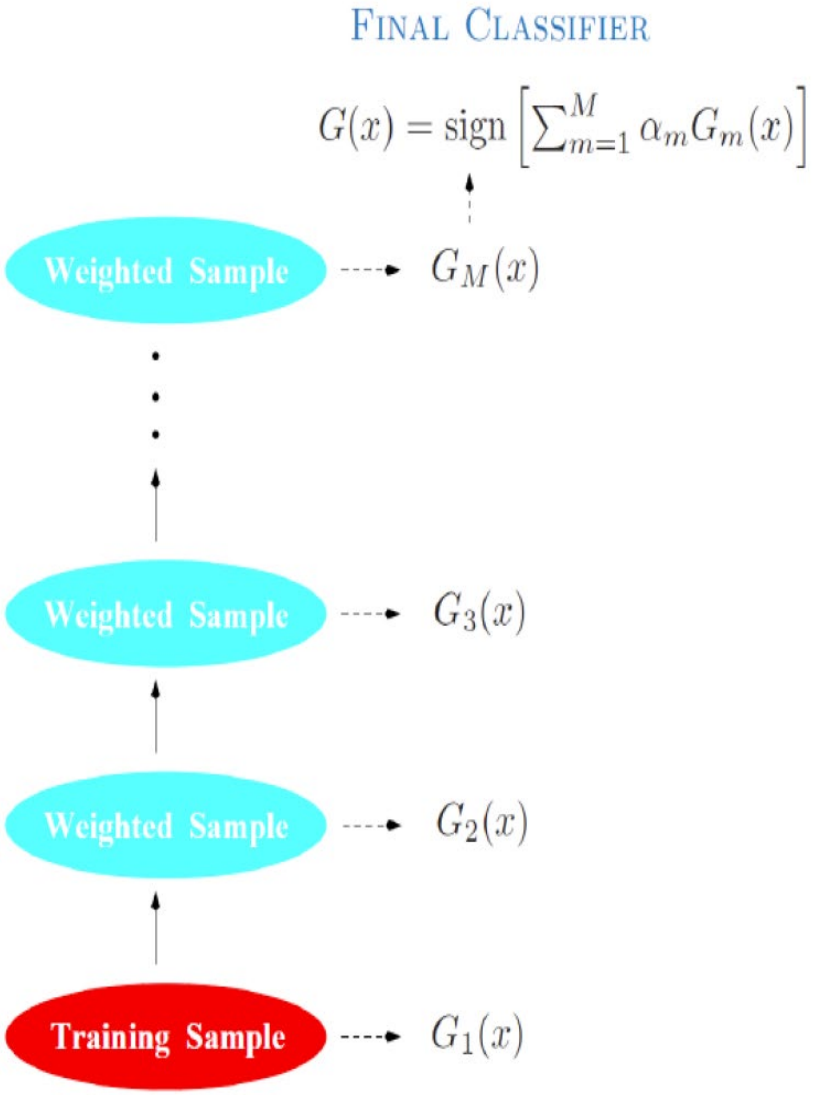


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Two Important Concepts:
(1) Learn from mistakes in building a new classification model;
(2) Trust quality-model more.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

2. For $m = 1$ to M:

Build a new classifier by emphasizing on learning from mistakes, i.e., weight (wi) the mis-classified cases more.

(a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

(b) Compute

Indicator = 1 for the mis-classification case.

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

Err = 0 is for the case "no mis-classification".

Alpha is the "learning rate".

(c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

When err is zero, alpha becomes infinity; When err is ONE (largest), alpha becomes negative infinity.

When the indicator is zero (i.e., no mis-classification), exp(.) = 1, wi stays the same.

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.

Only adjust the wi weight for mis-classified cases.

When alpha is infinity, exp(.) = infinity; When alpha is negative infinity, exp(.) becomes 0.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Conclusion: When a MODEL has zero err, alpha becomes infinity, i.e., the MODEL is well trusted. When the MODEL is trusted, the weight-adjustment wi will become larger. Finally, in the FINAL MODEL, the well-trusted MODEL will be weighted more.

4) Random Forest:

Random forests (Breiman, 2001) is a substantial modification of bagging that builds a large collection of *de-correlated* trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages.

Boosting Weight Examples:

Err	(1-Err)/Err	Alpha_m = Log[(1-Err)/Err]	Exp(Alpha_m)		Model-Quality	
			weight-factor			
0.8	0.25	-0.602059991	0.547682253		Not a good model	
0.5	1	0	1			
0.4	1.5	0.176091259	1.192546884		Good model	

An Algorithm for Random Forest (Tree Constructions):

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

An Example – Comparison Between Bagging, Random Forest and Boosting Procedures:

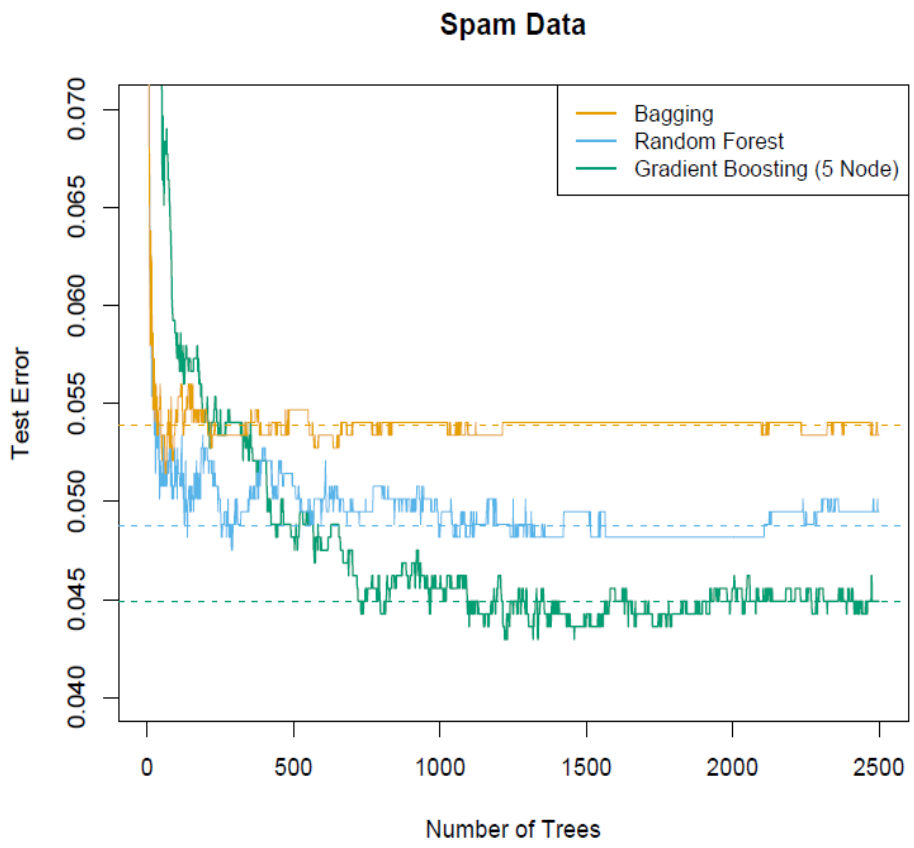


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

VI) Model Assessment:

Reference: Hastie, T., Tibshirani, R., and Friedman, J. (2017), *The Elements of Statistical Learning*, Springer, New York. **Note that this book is a typical textbook used in data mining or/and machine learning classes.** Downloadable pdf file: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf

In this chapter we describe and illustrate the key methods for performance assessment, and show how they are used to select models. We begin the chapter with a discussion of the interplay between bias, variance and model complexity.

7.2 Bias, Variance and Model Complexity

Figure 7.1 illustrates the important issue in assessing the ability of a learning method to generalize. Consider first the case of a quantitative or interval scale response. We have a target variable Y, a vector of inputs X, and a prediction model $\hat{f}(X)$ that has been estimated from a training set $\mathcal{T}$. The loss function for measuring errors between Y and $\hat{f}(X)$ is denoted by $L(Y, \hat{f}(X))$. Typical choices are

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & \text{squared error} \\ |Y - \hat{f}(X)| & \text{absolute error.} \end{cases} \tag{7.1}$$

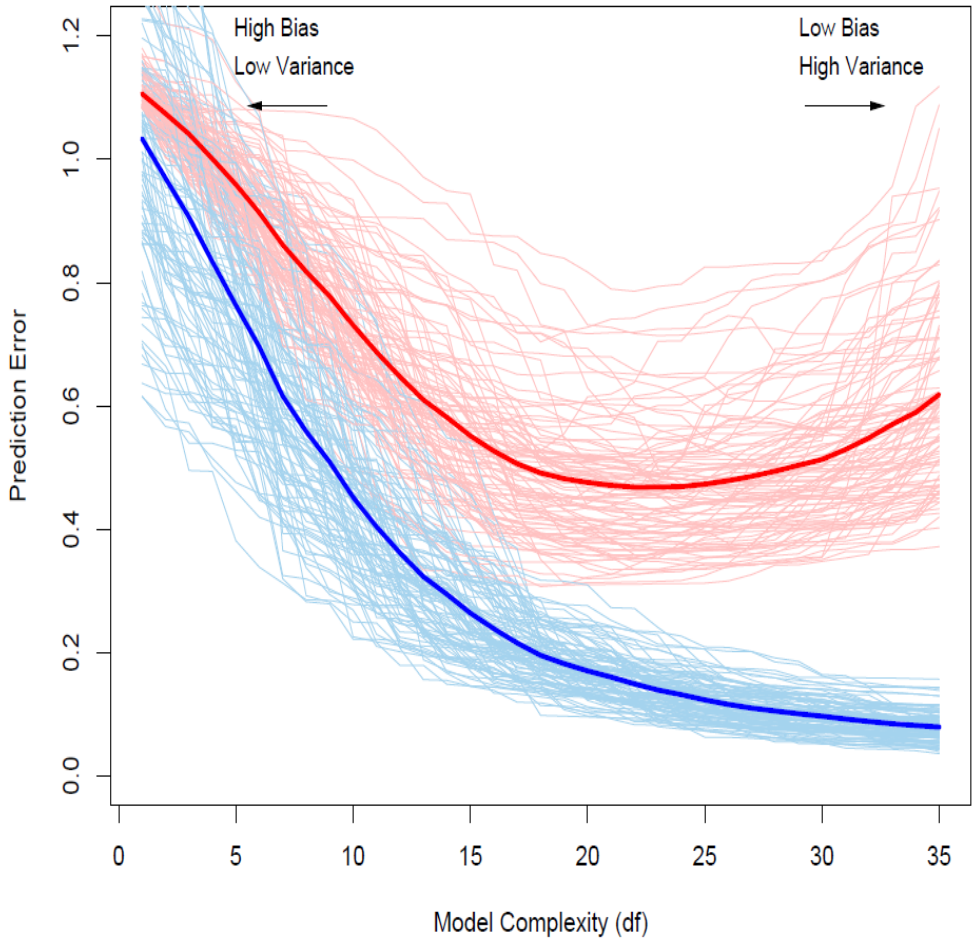


FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Training error is the average loss over the training sample

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)). \tag{7.4}$$

Test error, also referred to as *generalization error*, is the prediction error over an independent test sample

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}[L(Y, \hat{f}(X))|\mathcal{T}] \tag{7.2}$$

where both X and Y are drawn randomly from their joint distribution (population). Here the training set $\mathcal{T}$ is fixed, and test error refers to the error for this specific training set. A related quantity is the expected prediction error (or expected test error)

$$\mathrm{Err} = \mathrm{E}[L(Y, \hat{f}(X))] = \mathrm{E}[\mathrm{Err}_{\mathcal{T}}]. \tag{7.3}$$

Note that this expectation averages over everything that is random, including the randomness in the training set that produced $\hat{f}$.

Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially.

It is difficult to give a general rule on how to choose the number of observations in each of the three parts, as this depends on the signal-to-noise ratio in the data and the training sample size. A typical split might be 50% for training, and 25% each for validation and testing:

The methods in this chapter are designed for situations where there is insufficient data to split it into three parts. Again it is too difficult to give a general rule on how much training data is enough; among other things, this depends on the signal-to-noise ratio of the underlying function, and the complexity of the models being fit to the data.

The methods of this chapter approximate the validation step either analytically (AIC, BIC, MDL, SRM) or by efficient sample re-use (cross-validation and the bootstrap). Besides their use in model selection, we also examine to what extent each method provides a reliable estimate of test error of the final chosen model.

VII) In-Class Exam-2 Subjects – focus on "high level" concepts

1. **Decision Models (30%)**

   i) Decision in An Uncertain Environment (Constrained-Optimization Model Formulation)

   ii) Game-Theoretic Models

2. **Advanced Data Modeling (70%)**

   i) Classification (LDA/QDA, SVM)

   ii) Nonparametric Regression

   iii) Cluster Analysis (K-Means, Multivariate Gaussian Mixtures)

iv) Dimension Reduction (PCA, PLS, MDS, SOM)

v) Variable-Selections (Stepwise, All-Subsets, Ridge Regression, Lasso)

vi) Resampling and Ensemble Methods (Bootstrap Sampling; Cross-Validation, Boosting, Random Forest)

VIII) Past In-Class Exam-2 Problems:

A) **True and False Questions (5 points for each question and there is no partial credit).**

(True, False) 1. In K-means method a data-point could be in several clusters with different probabilities (sum to one).

Answer: False. There is no probability or distribution assumption for the K-means method.

(True, False) 2. AIC metric could be used in the Stepwise Regression.

Answer: False. AIC is used in the all-subsets regression. Stepwise regression is based on the partial-F-tests.

(True, False) 3. In the four ensemble methods (bagging, stacking, boosting and random-forest) taught in the

lectures for regression predictions *bagging* is the only procedure uses equal-weights in the average of predictions from m = 1000 (eg) individual models fitted to re-sampled-data.

Answer: False. Random-forest also uses equal weights to average predictions.

(True, False) 4. Whether Y-data is normally or multinominal distributed, the results of PCA are the same.

Answer: True. PCA is an unsupervised learning method dealing with only X-data, where Y-data and its distribution are *not involved*.

(True, False) 5. AIC-, BIC- and PRESS-based all-subsets regression procedures are applicable to non-normal data and non-linear models. Similarly, the stepwise regression procedures have the same properties.

Answer: False. Stepwise regression depends on the normal distribution and linear model assumptions such that the partial-F-test has an F-distribution.

(True, False) 6. In the boosting procedure the weights for the mis-classified samples from the previous model are

always more than the weights for the non-mis-classified samples.

Answer: False. In boosting the weights for the mis-classified samples are more than the weights for correctly classified samples – **this comparison is only valid for a set of samples**, i.e., comparison within a set of samples, but not between two sets of samples or samples in different models. This is the property in boosting's algorithm Step-2(c). Thus, the sample-weight can be more or less than the weights given by the previous model.

(True, False) 7. When the X-data-columns are organized in different sequences (e.g., X1, X3, X4 versus X4, X3, X1), it is possible that forward-selection produces distinct models (even that alpha-to-enter is set at the same value, e.g., 15%).

Answer: True. The partial-F-test in the forward-selections takes the first (and then, the subsequent) variable(s) from the X-data-columns given by the user. Different sequences could produce distinct models.

(True, False) 8. One of the key differences in the LDA- and logistic-classification is that LDA assumes the distribution of X-variables (as a multivariate normal).

Answer: True. Logistic-classification treats X-variables as given constants, but LDA assumes them to have a multivariate normal distribution (given a class of Y).

(True, False) 9. The weights for each one of the $m = 1000$ stacking-models and boosting-models are found based on certain optimization objectives.

Answer: False. It is true for the stacking method, but not true for the boosting method, where the model-weight is assigned as a given function of "Err (= error rate)", where no optimization objective function is used to derive the weights.

(True, False) 10. Even that the covariance between X and Y stays the same, PLS-results could change depending on the distribution assumption of Y-data, e.g., log-normal versus Weibull or Gamma.

Answer: False. PLS only works with the covariance between X and Y. The distribution (e.g., pdf) information is not involved.

(True, False) 11. The SVM-classification will minimize the mis-classification counts.

Answer: False. The SVM maximizes the **margin of support-vectors** in different classes.

(True, False) 12. In all variable-selection procedures taught in the lectures stepwise-regression is the only sequential procedure, i.e., the model selected in the latter stage will depend on the model(s) selected in the earlier stage(s).

Answer: True. The partial-F-test used in the stepwise regression is a sequential procedure. All other variable-selection procedures work with each possible model independently.

ISyE DDA – Agenda for 03/16/23 Lecture (Lec.19)

I) Cluster Analysis – Multivariate Gaussian Mixtures

II) Dimension Reduction (DR)

III) Large *p* Small *n* Variable-Selections

IV) Resampling and Ensemble Methods

(a) Resampling: Bootstrap and Jackknife

(b) **Ensemble Methods: Bagging, Stacking, Boosting, Random Forest**

Detailed Lectures:

vii) Statistical Distribution Models: Multivariate Gaussian Mixtures

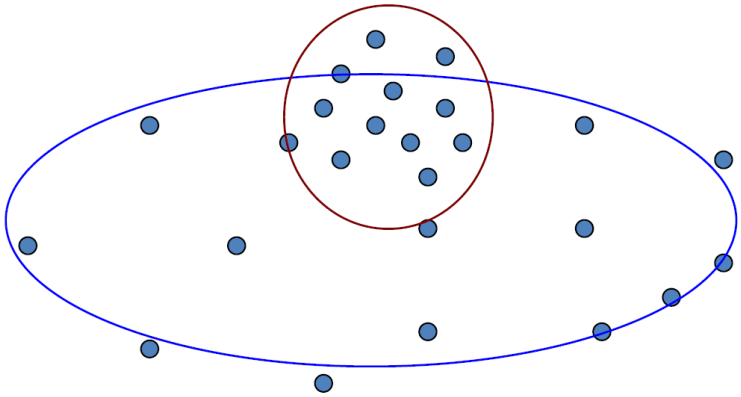
The clustering model most closely related to statistics is based on distribution models, e.g., multivariate Gaussian mixtures. Clusters can then easily be defined as objects belonging most likely to the same distribution. While the theoretical foundation of these methods is excellent, they suffer from one key problem known as overfitting, unless constraints (e.g., #clusters are known/fixed) are put on the model complexity. A more complex model will usually be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult.

Distribution-based clustering produces complex models for clusters that can capture correlation and dependence between attributes. However, these algorithms put an extra burden on the user: for many real data sets, there may be no concisely defined mathematical model (e.g. assuming Gaussian distributions is a rather strong assumption on the data).

The Gaussian Mixture Model (GMM) provides "soft clustering" or "probability clustering" results.

- Soft clustering gives probabilities that an instance belongs to each of a set of clusters

Probabilistic Clustering



- Clusters of different shapes and sizes
- Clusters can overlap! (k-means doesn't allow this)

Finite Mixture Models

- Given a dataset: $x^{(1)}, \ldots, x^{(N)}$
- **Mixture model**: $\Theta = \{\lambda_1, \ldots, \lambda_k, \theta_1, \ldots, \theta_k\}$

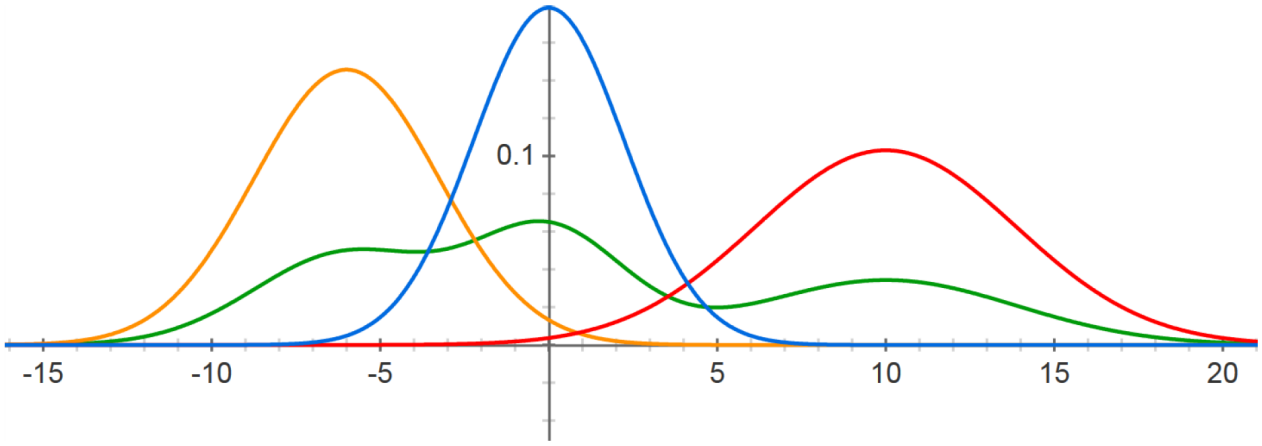
$$p(x|\Theta) = \sum_{y=1}^{k} \lambda_y p_y(x|\theta_y)$$

where $p_y(x|\theta_y)$ is a mixture component from some family of probability distributions parameterized by θ_y and $\lambda \geq 0$ such that $\sum_y \lambda_y = 1$ are the mixture weights

Remark: Please view "y" notation as the index for *K* distributions, i.e., y = 1, 2, …, *K. Students can treat this index y as "j".* We will use the "*j*" index from now on for representing the *j*-th probability distribution in the mixture model.

Examples:

Finite Mixture Models



Uniform mixture of 3 Gaussians

Typically, in the GMMs, the following multivariate normal are used to model the probability distributions $P_j(x|\theta_j), j = 1, 2, \ldots, K$.

Multivariate Gaussian

- A d-dimensional multivariate Gaussian distribution is defined by a $d \times d$ covariance matrix Σ and a mean vector μ

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

- The covariance matrix describes the degree to which pairs of variables vary together
  - The diagonal elements correspond to variances of the individual variables

Remark: The mean μ and variance-covariance $\sum$ in the probability distributions $P_j(x|\theta_j), j = 1, 2, \ldots, K$, should be μ_j and $\sum_j$ from the j-th multivariate normal distribution. Note that

these means μ_j and variance-covariances $\sum_j$ from different distributions $P_j(x|\theta_j)$, $j = 1, 2, \ldots, K$, are usually different.

Using the following Expectation-Maximization (EM) Algorithm to estimate these model parameters λ_j, μ_j and $\sum_j$ for $j = 1, 2, \ldots, K$. Details of the theoretical background and its derivation are skipped here. We provide a graphical presentation of the implementation of the EM algorithm in the GMM parameter estimation process.

EM for Gaussian Mixtures

- **E-step:**

$$q_i^t(y) = \frac{\lambda_y^t \cdot p\left(x^{(i)} \middle| \mu_y^t, \Sigma_y^t\right)}{\sum_{y'} \lambda_{y'}^t \cdot p\left(x^{(i)} \middle| \mu_{y'}^t, \Sigma_{y'}^t\right)}$$

Probability of $x^{(i)}$ under the appropriate multivariate normal distribution

- **M-step:**

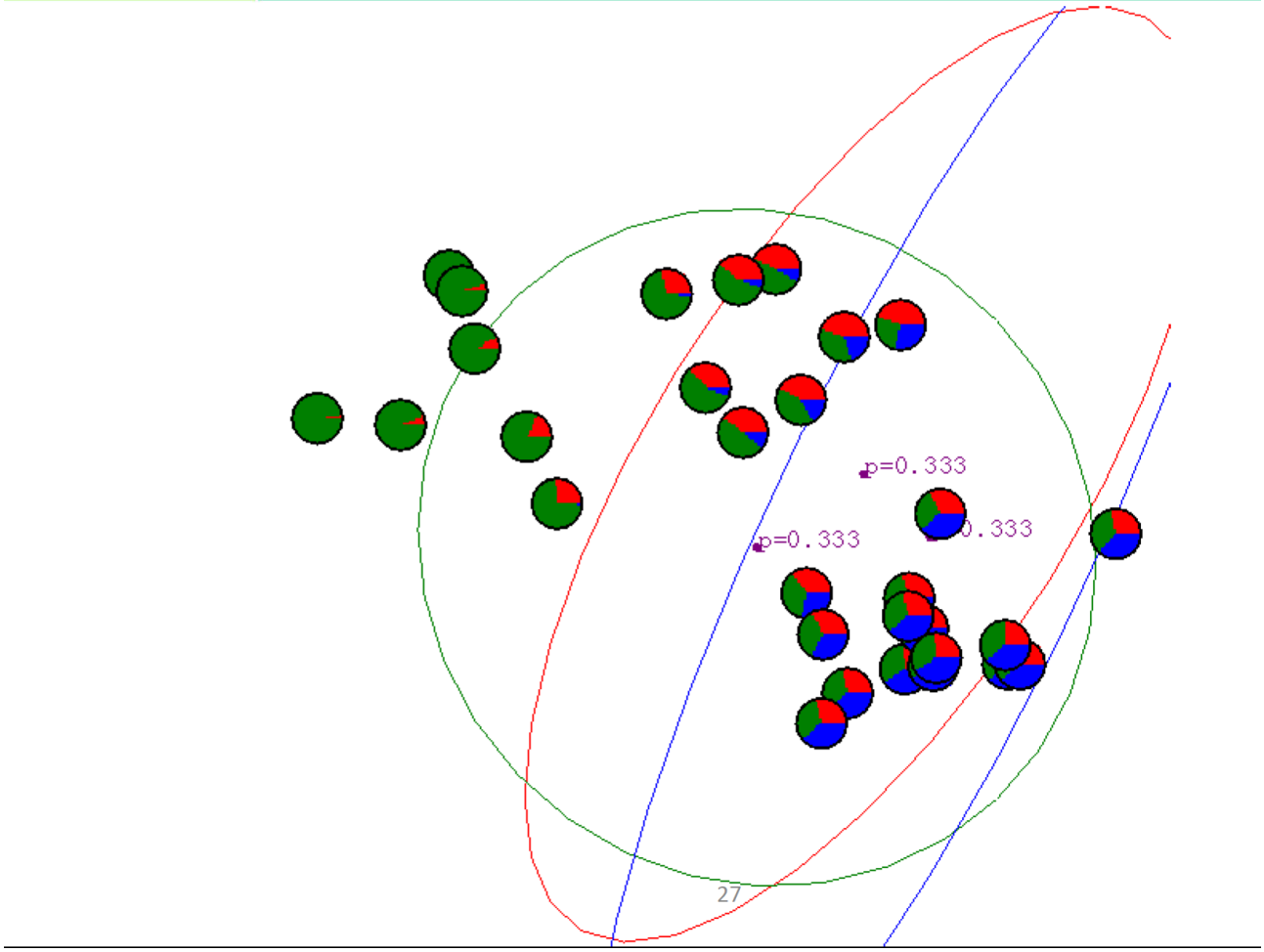
$$\mu_y^{t+1} = \frac{\sum_{i=1}^N q_i^t(y) x^{(i)}}{\sum_{i=1}^N q_i^t(y)}$$

$$\Sigma_y^{t+1} = \frac{\sum_{i=1}^N q_i^t(y)\left(x^{(i)} - \mu_y^{t+1}\right)\left(x^{(i)} - \mu_y^{t+1}\right)^T}{\sum_{i=1}^N q_i^t(y)}$$

$$\lambda_y^{t+1} = \frac{1}{N}\sum_{i=1}^N q_i^t(y)$$

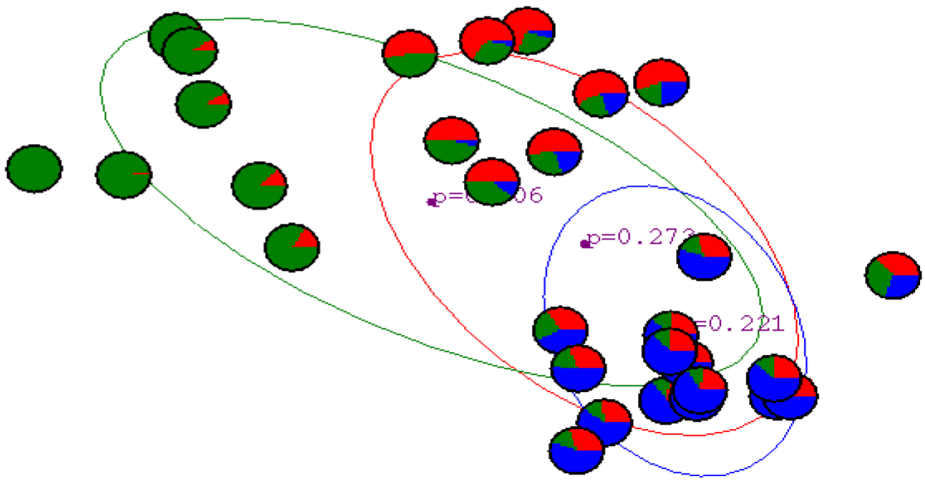
Graphical Presentation of Implementing the EM-Algorithm in GMMs' Parameter Estimation with a $K = 3$ example:

Gaussian Mixture Example: Start

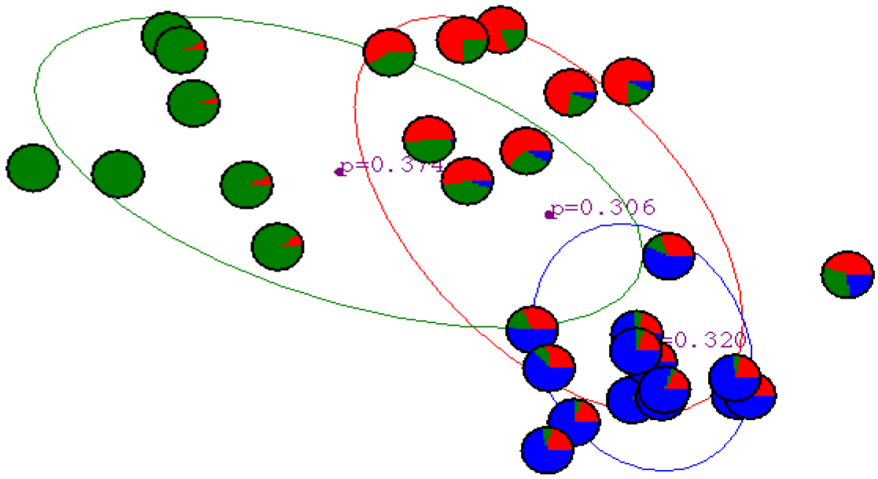


Remark: Here, "p_j =eg= 0.333" is the "weight" λ_j, j = 1, 2, …, K =eg= 3. The initial weights are usually equal weights.

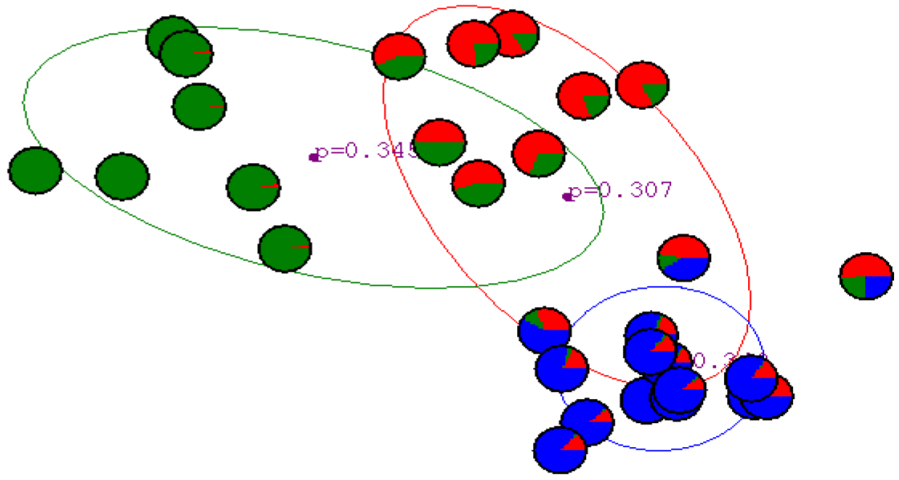
After first iteration



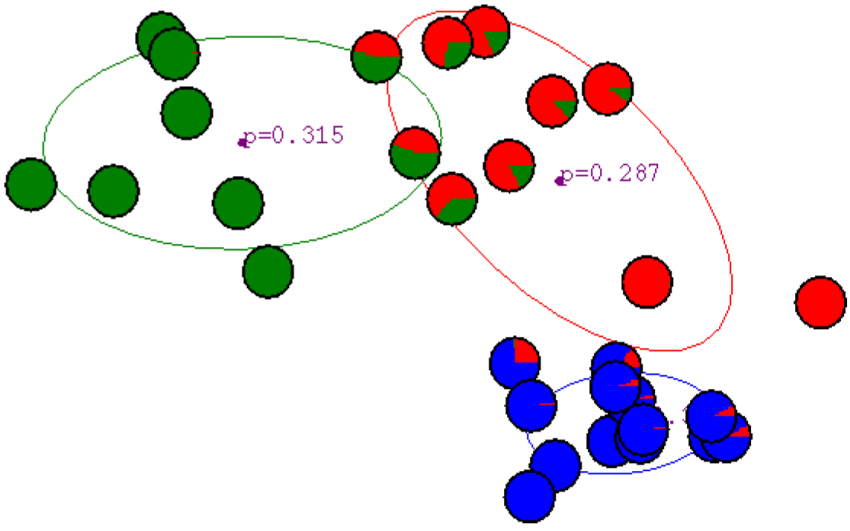
After 2nd iteration



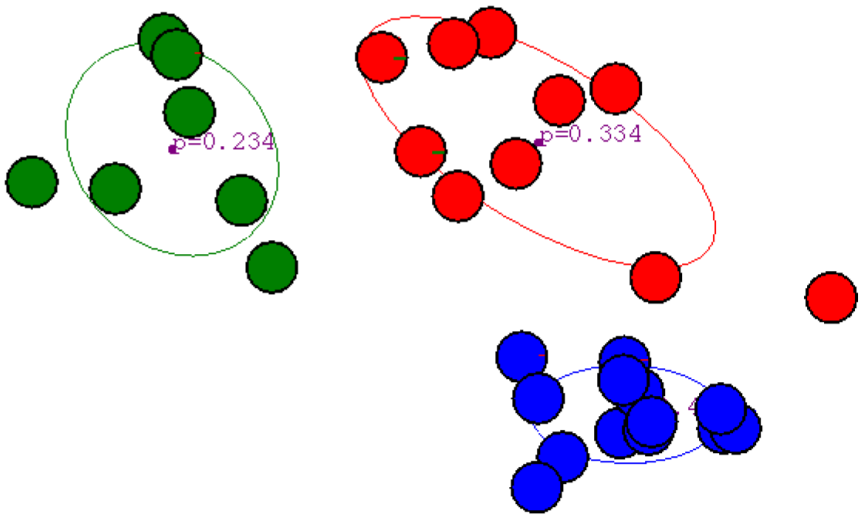
After 3rd iteration



After 6th iteration



After 20th iteration



Properties of EM

- EM converges to a local optima
  - This is because each iteration improves the log-likelihood

---

I) Dimension Reductions (DRs)

Dimension Reduction (DR) is a school of **Unsupervised Learning Procedures** (i.e., there is **no Y-outcomes data**; only

analyze X-data). This class focuses on high-level concepts. The following are three popular DR procedures.

(1) Principal Component Analysis (PCA)

(2) Multi-Dimensional Scaling (MDS)

(3) Partial Least Squares (PLS)

Details:

(1) Principal Component Analysis (PCA)

Principal components **(PC)** of a set of X-data in R^p provide **a sequence of best *linear approximations* to that data**, of all ranks $q \leq p$. These **principal components are *mutually uncorrelated and ordered in variance.*** See the following figure for an example.

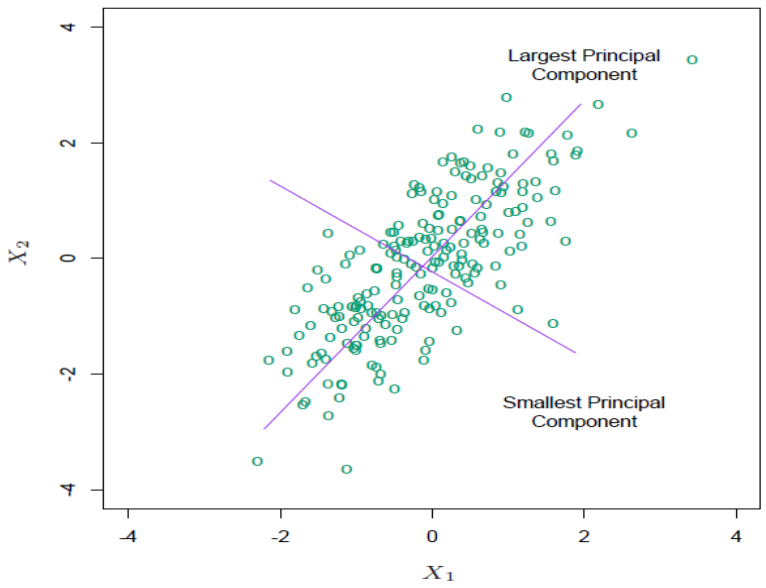


FIGURE 3.9. *Principal components of some input data points. The largest principal component is the direction that maximizes the variance of the projected data, and the smallest principal component minimizes that variance. Ridge regression projects* $\mathbf{y}$ *onto these components, and then shrinks the coefficients of the low–variance components more than the high-variance components.*

Denote the observations by $x_1, x_2, \ldots, x_N$, (each of them is a p-dim vector) and consider the following rank-q linear model for representing (or approximating) them:

$$f(\lambda) = \boldsymbol{\mu} + \mathbf{V}_q \boldsymbol{\lambda},$$

Vq is the "direction" for the projection (from a higher-dim to a lower-dim. See Figure 14.20 for a graphical presentation.

where $\boldsymbol{\mu}$ is a *p-dim* location vector, $\mathbf{V}_q$ is a *p-row times q-column* matrix with q orthogonal unit-vectors as columns, and $\boldsymbol{\lambda}$ is a q-element vector of parameters. See Figure 4.20 and 4.21 for q = 1 (page 534) and 2 (page 536), respectively, illustration of the rank-q representing function $f(\lambda)$.

mu is the "center" (or "origin")

parameters is the q-dim projected data-value, where data is in p-dim; p > q.

Example #1 (2-D X-data are projected to the first PC-line):

Figure 14.20 shows a set of 2-dimensional data points $x_1, x_2, \ldots, x_N$, plotted in the R^2 space. PC-Line projects these 2D data x_i's onto one-dimensional line with v_1 as the **direction with a value (U_{i1} * d_1)**, and black-dot-point as the center. Define an objective function as the sum-of-squares of distances from each point to its orthogonal projection onto the "PC-line" for locating the PC-line

with the direction v_1. The parameter-vectors μ and λ, and also the resulted lower-rank matrix $\boldsymbol{Vq}$ can be obtained.

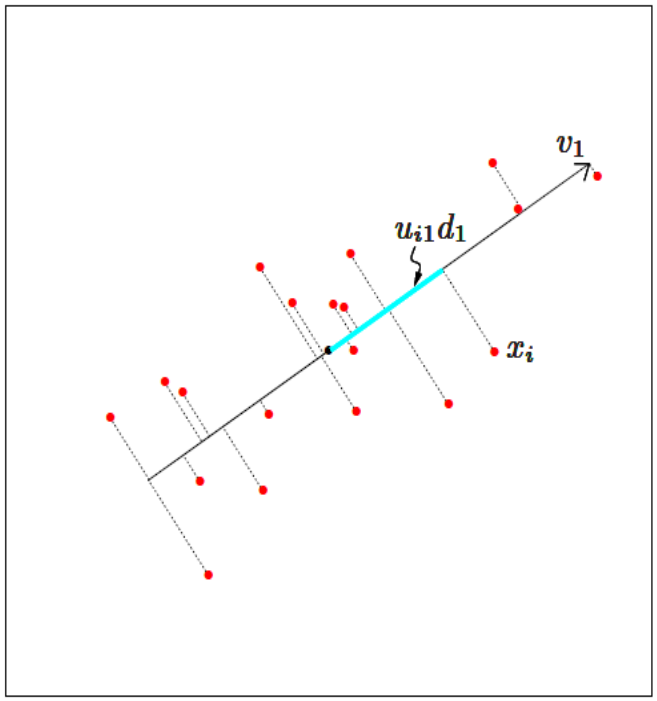


FIGURE 14.20. *The first linear principal component of a set of data. The line minimizes the total squared distance from each point to its orthogonal projection onto the line.*

The solution of the matrix $\boldsymbol{Vq}$ can be expressed as follows. Stack the (centered) observations into the rows of an $N \times p$ matrix $\boldsymbol{X}$. We construct the ***singular value decomposition*** of $\boldsymbol{X}$: $\boldsymbol{X} = \boldsymbol{UDV}^T$, where $\boldsymbol{U}$ is an $N \times p$ orthogonal matrix ($\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{Ip}$) whose columns $\boldsymbol{u}_j$ are called the left singular vectors; $\boldsymbol{V}$ is a $p \times p$ orthogonal matrix ($\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{Ip}$) with columns $\boldsymbol{v}_j$ called the right singular vectors, and $\boldsymbol{D}$ is a $p \times p$ diagonal matrix, with diagonal

elements $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$ known as the singular values. For each rank q, the solution Vq consists of the first q columns of V. **The columns of UD are called the principal components of X.** The N optimal λ_i's (elements in the vector $\boldsymbol{\lambda}$) are given by the first q principal components (the N rows of the $N \times q$ matrix $UqDq$).

Important Remark: Each principal component is a linearly weighted sum of the original p-data-columns X. Note that the original data variances is decomposed into p variances $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$ **"orthogonally."** The data represented by the j-th principal component captures d_j amount of variance.

Example #2 (3-D X-data are projected to the first two PCs):

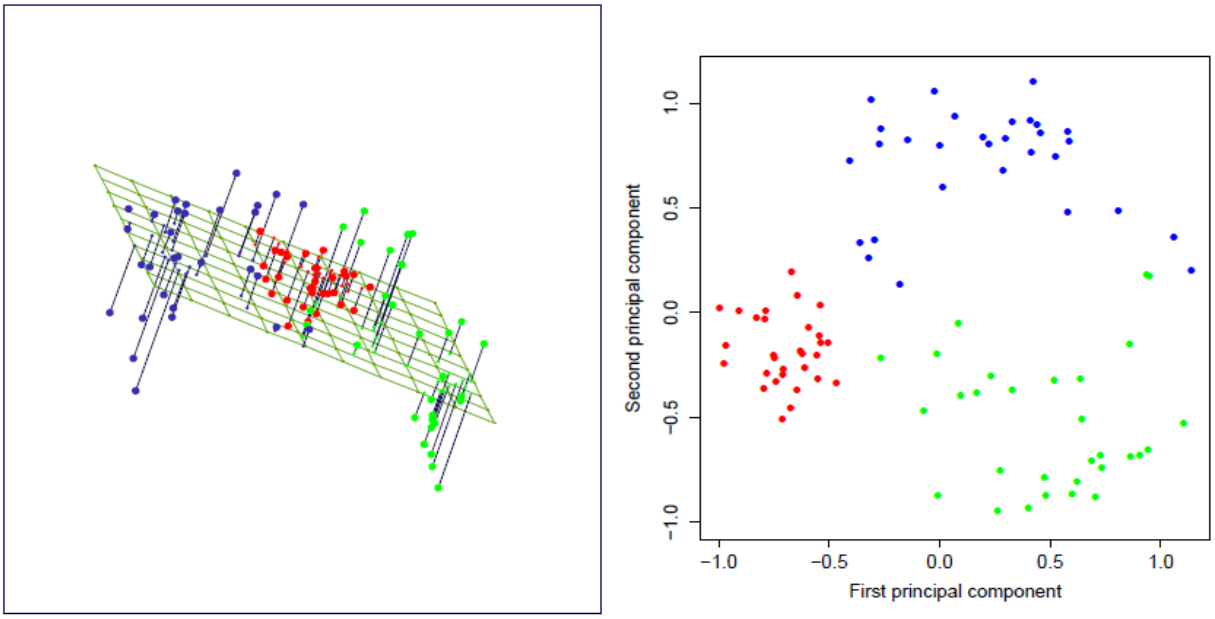


FIGURE 14.21. *The best rank-two linear approximation to the half-sphere data. The right panel shows the projected points with coordinates given by* $\mathbf{U}_2\mathbf{D}_2$, *the first two principal components of the data.*

v) Multi-Dimensional Scaling (MDS): (Important procedure)

MDS has the same goal as the one studied in PCA and SOM. However, MDS finds a lower-dimensional representation of the data that preserves the **pairwise distances as well as possible**. The following provides technical details.

Define $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$, the distance (dis-similarity) of two data points. MDS seeks values $z_1, z_2, \ldots, z_N \in \mathrm{R}^k$ to minimize the so-called stress function

$$S_M(z_1, z_2, \ldots, z_N) = \Sigma_{i \neq k} (d_{ik} - \|z_i - z_k\|)^2 .$$

This is known as least squares or Kruskal–Shephard scaling.

Remark: MDS only requires the dis-similarity data, but not the original data, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$. This property extends the application of dimension reduction to non-standard data such as qualitative data, e.g., color of

clothes. As long as one can provide the dis-similarity information, MDS can be applied.

vi) Partial least squares regression (PLS regression): (Important sub-topic)

Concept: PLS is a statistical method that bears some relation to principal components regression; instead of finding hyperplanes of maximum variance between the response and independent variables, it finds a linear regression model by projecting the predicted variables and the observable variables to a new space for modeling the covariance structures in these two spaces.

PCAs

Technical Details: For the following regression model

$$\boldsymbol{Y} = \boldsymbol{X}\,\boldsymbol{\beta} + \boldsymbol{e}. \quad \text{Model (1)}$$

Tk is the k-th PC.

PCA: $\boldsymbol{X} = \boldsymbol{T}\,\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{T}_k\,\boldsymbol{P}_k^{\mathrm{T}} + \boldsymbol{E}$, where $k = 1, 2, \ldots, K$ are the first, second, …, principal components that capture most, second-most, …, amount of X-data variances.

From SVD (Single-Value-Decomposition) $\boldsymbol{X} = (\boldsymbol{US})\,\boldsymbol{V}^{\mathrm{T}}$

with $\boldsymbol{T} = \boldsymbol{US}$ and $\boldsymbol{P} = \boldsymbol{V}$.

In application of PCA in regression consider the following regression estimates for model (1):

$\beta\_hat = (X^T X)^{-1} X^T Y = H\ Y$, where H is the "head" matrix.

PCA-Regression (PCR): Regress Y with the first k Principal Components (PCs). The head matrix becomes

$$H_k = P_k\ (\ T_k^T\ T_k\ )^{-1}\ T_k^T.$$

a lower-dim space

PLS tried to project both X- and **Y**-data into spaces that maximizes the covariance between X and Y. It requires an addition of weights W to achieve such goal. The resulted head-matrix in regression estimate is

$$H_{k,PLS} = W_k\ (P_k^T\ W_k\ )^{-1}\ (\ T_k^T\ T_k\ )^{-1}\ T_k^T. \qquad \text{Model (2)}$$

There are several competing algorithms to find the head-matrix with various computing speed, accuracy and easiness-for-interpretation (in the algorithm).

weighted sum of the original variables (like PCA)

Note: As seen from the PCR the PLS-variables found by the head-matrix in Model (2) might not have much **physical meaning** (due to its goal (and also the method) of constructing these PLS-variables). Moreover, based on Model (2), it seems that every PLS variable includes ALL "original" variables.

Please look into the past-exam problems for comparisons between PLS, PCA and MDS.

Some of these variables could be **correlated**. In the *large p small n problems*, there are many original-variables.

Dimension-Reduction in the large p small n problems is our next topic for lectures.

II) Large *p* Small *n* Variable-Selections

Review: Variable-Selections for Linear (Normal) Regressions

(1) Traditional Problems (n ≡ sample size > p ≡ #x-variables)

(a) Stepwise Regression (with Partial F-Tests) (ISyE 3030/4031 materials)

(b) All-Subsets Regression (ISyE 3030/4031 materials)

(2) Recent Problems (n < p & n << p) (our focus in ISyE 4034)

(a) Ridge Regression, Lasso, Elastic Net Regularization

(b) Adaptive Lasso and Group Lasso

(c) Clustering and Representative-Selection

New: Variable-Selections for Linear Regressions

- **Recent Problems (n < p & n << p)**

(a) Ridge Regression, Lasso, Elastic Net Regularization

(b) Adaptive Lasso and Group Lasso

(c) Clustering and Representative-Selection

Details:

Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

- In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum |\beta_j|$.

Lasso: Least Absolute Shrinkage and Selection Operator

IV) Resampling and Ensemble Methods

(a) Resampling: Bootstrap and Jackknife

(b) Ensemble Methods: Bagging, Stacking, Boosting, Random Forest

Details:

[1] Purpose of Resampling Methods:

In statistics, **resampling** is any of a variety of methods for doing one of the following:

1. Estimating the precision of sample statistics (medians, variances, percentiles) by using subsets of available data (**jackknifing**) or drawing randomly with replacement from a set of data points (**bootstrapping**)
2. Exchanging labels on data points when performing significance tests (**permutation tests**, also called exact tests, randomization tests, or re-randomization tests)
3. Validating models by using random subsets (bootstrapping, cross validation).

[2] Bootstrap Resampling:

Original Samples: $\mathbf{Y}_{orig} \equiv \{Y_1, Y_2, \ldots, Y_n\}$ =eg = {2, 4, 10, 12, 7} with $n = 5$

Bootstrap Samples:

Perform a random sampling with replacement to collect n "new data" from the original samples. For example, {4, 7, 10, 4, 2} could be a set of bootstrap samples. Notice that "4" is repeated twice. This is possible due to the "sampling WITH replacement" procedure.

Denoted by the bootstrap samples as

$\mathbf{Y^{(1)}} \equiv \{Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_n^{(1)}\}$ =eg = {4, 7, 10, 4, 2};

$\mathbf{Y^{(2)}} \equiv \{Y_1^{(2)}, Y_2^{(2)}, \ldots, Y_n^{(2)}\}$;

…

$\mathbf{Y^{(m)}} \equiv \{Y_1^{(m)}, Y_2^{(m)}, \ldots, Y_n^{(m)}\}$,

where m is a large number (e.g., 10,000) to facilitate further distributional studies of the bootstrap samples.

Bias Evaluation: The procedure explains below is applicable for evaluating bias in any sample-estimate, e.g., sample-mean, -median, 90th-percentile.

Continue the example above. Denoted by θ_{orig}_hat as the sample-estimate from the original samples. Then, the **bootstrapped sample-estimates** are as follows:

$$\theta^{(1)}\_hat,\ \theta^{(2)}\_hat,\ \ldots,\ \theta^{(m)}\_hat, \qquad (1)$$

for each one of the bootstrapped samples $\mathbf{Y}^{(i)}$, $i = 1, 2, \ldots, m$, respectively.

Then, get an average ($\theta^{(i)}\_hat\_ave$) of these bootstrapped sample-estimates. Bias will be defined as

$$\text{Bias} = \theta_{orig}\_hat - (\theta^{(i)}\_hat\_ave),$$

where the average ($\theta^{(i)}\_hat\_ave$) acts like the population-average E(θ_{orig}_hat) in evaluating this bias.

Variance-Estimation: Continue the example above. Based on m quantities of bootstrapped sample-estimates given in Eq.(1), one can calculate the following bootstrap-variance. By taking a square-root, one obtains the bootstrap standard-deviation (sd).

$$\sigma_b^2\_\text{hat} = \Sigma_{i=1}^{m} \left(\theta^{(i)}\_hat - (\theta^{(i)}\_hat\_ave) \right]^2 / \left(m - 1 \right).$$

This bootstrap-variance estimates the population variance σ^2 of the original samples $\mathbf{Y}_{orig}$.

Note that many people use **bootstrap Mean Squared Error**, MSE = $(\text{Bias})^2$ + Variance, to evaluate the quality of estimating a population parameter θ (e.g., mean, median or 90^{th}-percentile).

Various **bootstrap-based Confidence Intervals** (CIs) can be constructed for estimating population parameter θ. Students of interest should look into lecture-notes from ISyE-6404 (Nonparametric Statistic) or Internet-materials.

[3] Jackknife Resampling:

Bootstrap-resampling could be computing intensive (and thus, time consuming), the following jackknife-resampling method is less computing intensive. Unlike the sampling with replacement bootstrap method, Jackknife is NOT sampling with replacement.

Original Samples: $\mathbf{Y}_{orig} \equiv \{Y_1, Y_2, \ldots, Y_n\}$ =eg = {2, 4, 10, 12, 7} with $n = 5$

Delete-1 Jackknife Samples:

$\mathbf{Y}_{(1)} \equiv$ { 4, 10, 12, 7 }, where the first sample "2" is not included;

$\mathbf{Y_{(2)}} \equiv \{ 2, 10, 12, 7 \}$; where the second sample "4" is not included;

…

$\mathbf{Y_{(n)}} \equiv \{ 2, 4, 10, 12 \}$, where the last sample "4" is not included.

Naturally, it is required to have a large sample size *n* to create many Jackknife samples. Then, the methods introduced in the bootstrap for estimating bias, variance and MSE can be applied to these jackknife samples.

Remark: Instead of deleting one sample in creating the Jackknife re-sample distribution, some people extend this method to delete-*d*-Jackknife, where *d*-samples (e.g., *d* = 2, 3, …) are deleted in each set of Jackknife sample.

[4] Cross-Validation Resampling:

Although the bootstrap method is applicable to regression problem, where each data point includes Y (outcome) and many X*j*'s (input-variables), many people prefer to use the

following cross-validation method to evaluate the quality of model-predictions.

Original Samples: ($Y_{i,orig}$, $X_{ij,orig}$, $j = 1, 2, \ldots, p$) with *i = 1, 2, …, n* for ***n*** **sets of iid regression samples.**

Cross-validate Estimation:

Step-1: Just like the delete-1 Jackknife sampling procedure. One regression-data-point is removed to create one set of "cross-validated-samples".

Step-2: Use each set of the delete-1 "cross-validated-samples" to build a regression model; it could be linear, nonlinear, GLM or non-parametric regression model. Then, use the "delete-1-built-model" to predict the data-point, which was deleted in the model-building process. Evaluate its "*cross-validated-prediction-error*". Repeat this process for all *n* sets of delete-1 "cross-validated-samples".

Step-3: Calculate the variance from *n* "*cross-validated-prediction-errors*". This is the "cross-validation-variance".

Remark: This cross-validation method is popular in model-selections. For example, in the nonparametric regression there are tuning parameters like window-size h in Kernel-regression or smoothing-penalty λ in Spline. By trying different window-sizes (e.g., $h = 0.2, 0.5, \ldots$), one obtain their corresponding cross-validation-variance (e.g., 3.5, 2.7, …). Select the h with smallest cross-validation-variance for this Kernel-regression.

(b) Ensemble Methods: Bagging, Stacking, Boosting, Random Forest

1) Bagging:

Consider first the regression problem. Suppose we fit a model to our training data $\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, obtaining the predic-

tion $\hat{f}(x)$ at input x. Bootstrap aggregation or *bagging* averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample $\mathbf{Z}^{*b}$, $b = 1, 2, \ldots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{8.51}$$

A more interesting example is a regression tree, where $\hat{f}(x)$ denotes the tree's prediction at input vector x (regression trees are described in Chapter 9). Each bootstrap tree will typically involve different features than the original, and might have a different number of terminal nodes. The bagged estimate is the average prediction at x from these B trees.

Now suppose our tree produces a classifier $\hat{G}(x)$ for a K-class response. Here it is useful to consider an underlying indicator-vector function $\hat{f}(x)$, with value a single one and $K-1$ zeroes, such that $\hat{G}(x) = \arg\max_k \hat{f}(x)$. Then the bagged estimate $\hat{f}_{\text{bag}}(x)$ (8.51) is a K-vector $[p_1(x), p_2(x), \ldots, p_K(x)]$, with $p_k(x)$ equal to the proportion of trees predicting class k at x. The bagged classifier selects the class with the most "votes" from the B trees, $\hat{G}_{\text{bag}}(x) = \arg\max_k \hat{f}_{\text{bag}}(x)$.

2) Stacking:

Stacked generalization, or *stacking*, is a way of doing this. Let $\hat{f}_m^{-i}(x)$ be the prediction at x, using model m, applied to the dataset with the

ith training observation removed. The stacking estimate of the weights is obtained from the least squares linear regression of y_i on $\hat{f}_m^{-i}(x_i)$, $m = 1, 2, \ldots, M$. In detail the stacking weights are given by

$$\hat{w}^{\text{st}} = \underset{w}{\text{argmin}} \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} w_m \hat{f}_m^{-i}(x_i) \right]^2. \tag{8.59}$$

The final prediction is $\sum_m \hat{w}_m^{\text{st}} \hat{f}_m(x)$. By using the cross-validated predictions $\hat{f}_m^{-i}(x)$, stacking avoids giving unfairly high weight to models with higher complexity. Better results can be obtained by restricting the weights to be nonnegative, and to sum to 1. This seems like a reasonable restriction

3) Boosting:

Boosting is one of the most powerful learning ideas introduced in the last twenty years. It was originally designed for classification problems, but as will be seen in this chapter, it can profitably be extended to regression as well. The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee."

ISyE DDA – Agenda for 03/14/23 Lecture (Lec.18)

I) Nonparametric (NP) Regressions - Spline

II) Unsupervised Learning - Cluster Analysis

III) Dimension Reduction (DR)

IV) Large *p* Small *n* Variable-Selections

Detailed Lectures:

I) Nonparametric Regressions:

(1) Kernel Regression (presented)
(2) Nearest-Neighbor NP-Regression (presented)
(3) Splines (new)

Details:

(3) Splines

Interpolating splines and **smoothing splines** are motivated from a different perspective than kernels and local polynomials; in the latter case, we started off with a special kind of local averaging and moved our way up to a higher-order local models. With splines, we build up our estimate globally, from a set of select **basis functions**.

These basis functions, as you might guess, are splines. Let's assume that d = 1 for one input variable, for simplicity. A **k-**

th order spline is a **piecewise polynomial function of degree k**, that is continuous and has continuous derivatives of orders 1, . . ., k − 1, **at its knot points**. Please see below for details.

13.4.1 Interpolating Splines

There are many varieties of splines. Although piecewise constant, linear, and quadratic splines easy to construct, cubic splines are most commonly used because they have a desirable extremal property.

Denote the cubic spline function by $m(x)$. Assume $X_1, X_2, \ldots, X_n$ are ordered and belong to a finite interval $[a, b]$. We will call $X_1, X_2, \ldots, X_n$ *knots*. On each interval $[X_{i-1}, X_i]$, $i = 1, 2, \ldots, n+1, X_0 = a, X_{n+1} = b$, the spline $m(x)$ is a polynomial of degree less than or equal to 3. In addition, these polynomial pieces are connected in such a way that the second derivatives are continuous. That means that at the knot points $X_i, i = 1, \ldots, n$ where the two polynomials from the neighboring intervals

meet, the polynomials have common tangent and curvature. We say that such functions belong to $\mathbb{C}^2[a,b]$, the space of all functions on $[a,b]$ with continuous second derivative.

The cubic spline is called *natural* if the polynomial pieces on the intervals $[a,X_1]$ and $[X_n,b]$ are of degree 1, that is, linear. The following two properties distinguish natural cubic splines from other functions in $\mathbb{C}^2[a,b]$.

Unique Interpolation. Given the n pairs, $(X_1,Y_1),\ldots,(X_n,Y_n)$, with distinct knots X_i there is a *unique* natural cubic spline m that interpolates the points, that is, $m(X_i)=Y_i$.

Extremal Property. Given n pairs, $(X_1,Y_1),\ldots,(X_n,Y_n)$, with distinct and ordered knots X_i, the natural cubic spline $m(x)$ that interpolates the points also minimizes the curvature on the interval $[a,b]$, where $a<X_1$ and $X_n<b$. In other words, for any other function $g\in\mathbb{C}^2[a,b]$,

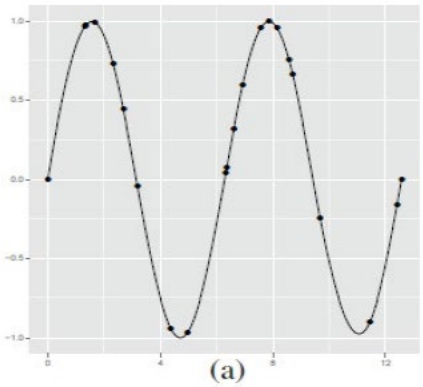
$$\int_a^b (m''(t))^2 dt \le \int_a^b (g''(t))^2 dt.$$

Remarks: It turns out that the cubic spline is in some sense asymptotically equivalent to a kernel regression, with an unusual choice of kernel, "Silverman kernel" (Silverman, 1984).

EXAMPLE 13.4

In R, the function `splinefun` and `bicubic` (in `akima` package) compute the cubic spline interpolant, and for the following x and y,

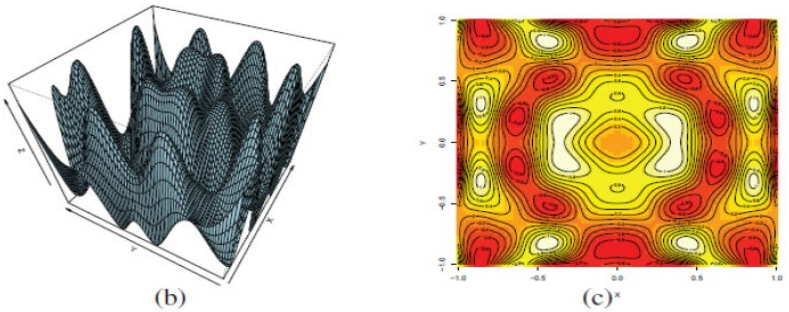
```
> x <- 4*pi*c(0,1,runif(20));
> y <- sin(x);
> fit <- splinefun(x,y);
> xx <- seq(0,max(x),length=100);yy<-fit(xx);
> p <- ggplot() + geom_point(aes(x=x,y=y),size=3)
> p <- p + geom_line(aes(x=xx,y=yy)) + xlab("") + ylab("")
> print(p)
```



(a)

the interpolation is plotted in Figure 13.8(a), along with the data. A surface interpolation by 2-d splines is demonstrated by the following R code and Figure 13.8(b) and (c).

```
> x <- seq(-1,1,by=0.2);
> y <- seq(-1,1,by=0.25);
> z <- outer(x,y,function(x,y){sin(10*(x^2+y^2));});
> xy<-expand.grid(seq(-1,1,length=100),seq(-1,1,length=80));
> fit<-bicubic(x,y,z,xy[,1],xy[,2]);
>
> xx <- seq(-1,1,length=100); yy <- seq(-1,1,length=80);
> zz <- matrix(fit$z,nrow=100);
> persp(x=xx,y=yy,z=zz,col="lightblue",phi=45,theta=-60,xlab="X",
+ ylab="Y",zlab="Z");
>
> image(x=seq(-1,1,length=100),y=seq(-1,1,length=80),
+       z=matrix(fit$z,nrow=100),xlab="X",ylab="Y");
> contour(x=seq(-1,1,length=100),y=seq(-1,1,length=80),
+       z=matrix(fit$z,nrow=100),add=TRUE);
```



(b) (c)

Figure 13.8 (a) Interpolating sine function; (b) Interpolating a surface; (c) Interpolating a contour.

13.4.2 Smoothing Splines

Smoothing splines, unlike interpolating splines, may not contain the points of a scatterplot, but are rather a form of nonparametric regression. Suppose we are given bivariate observations (X_i, Y_i), $i = 1, \ldots, n$. The continuously differentiable function $\hat{m}$ on $[a, b]$ that minimizes the functional

$$\sum_{i=1}^{n} (Y_i - m(X_i))^2 + \lambda \int_a^b (m''(t))^2 dt \tag{13.8}$$

is exactly a natural cubic spline. The cost functional in (13.8) has two parts: $\sum_{i=1}^{n} (Y_i - m(X_i))^2$ is minimized by an interpolating spline, and $\int_a^b (m''(t))^2 dt$ is minimized by a straight line. The parameter λ trades off the importance of these two competing costs in (13.8). For small λ, the minimizer is close to an interpolating spline. For λ large, the minimizer is closer to a straight line.

Although natural cubic smoothing splines do not appear to be related to kernel-type estimators, they can be similar in certain cases. For a value of x that is away

Smoothing Splines as Linear Estimators. The spline estimator is linear in the observations, $\hat{\mathbf{m}} = S(\lambda)\mathbf{Y}$, for a smoothing matrix $S(\lambda)$. The Reinsch algorithm (Reinsch, 1967) efficiently calculates S as

$$S(\lambda) = \left(I + \lambda Q R^{-1} Q'\right)^{-1}, \tag{13.11}$$

See textbook page 257 for the expressions for Q and R.

"Textbook": Kvam, Vidakovic and Kim (2016), Nonparametric Statistics with Applications in Science and Engineering, John Wiley. (ISBN: 978-0-470-08147-1).

II) Unsupervised Learning

(1) Clustering Analysis

a) Computing Methods:

i) **Connectivity Models:** Hierarchical Clustering

ii) **Centroid Models:** K-Means

iii) **Density Models:** DBSCAN and OPTICS

iv) **Subspace or Models:** Biclustering or Co-Clustering

v) **Neural Models:** Self-Organization Map and Multi-Dimensional Scaling

vi) **Graph-based Models:** Clique

b) Statistical Methods:

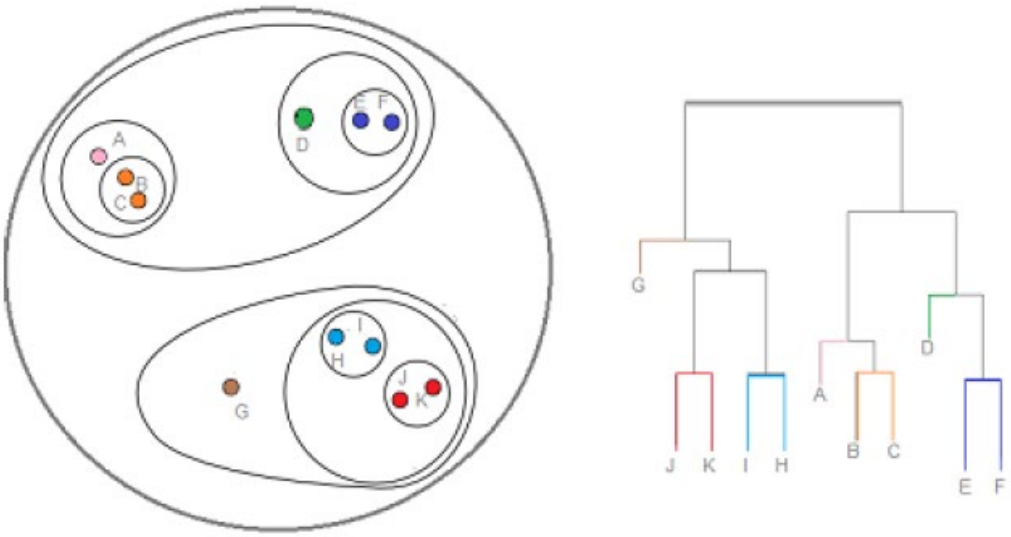
vii) **Distribution Models: Multivariate Normal Mixtures**

---

3. Details for Cluster Analysis

i) Connectivity Models: Hierarchical Clustering

Connectivity based clustering, also known as *hierarchical clustering*, is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a dendrogram, which explains where the common name "hierarchical clustering"



which explains where the common name "hierarchical clustering" comes from: these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. In a dendrogram, the y-axis marks the distance at which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix.

Connectivity based clustering is a whole family of methods that differ by the way distances are computed. Apart from the usual choice of distance functions, the user also needs to decide on the linkage criterion (since a cluster consists of multiple objects, there are multiple candidates to compute the distance) to use. Popular choices are known as single-linkage clustering (the minimum of

object distances), complete linkage clustering (the maximum of object distances) or UPGMA ("Unweighted Pair Group Method with Arithmetic Mean", also known as average linkage clustering). Furthermore, hierarchical clustering can be ("bottom-up") agglomerative (starting with single elements and aggregating them into clusters) or ("top-down") divisive (starting with the complete data set and dividing it into partitions).

Remark: These methods will not produce a unique partitioning of the data set, but a hierarchy from which the user still needs to choose appropriate clusters. They are not very robust towards outliers, which will either show up as additional clusters or even cause other clusters to merge (known as "chaining phenomenon", in particular with single-linkage clustering). In the general case, the complexity is $O(n^3)$ for agglomerative clustering and $O(2^{n-1})$ for divisive clustering which makes them too slow for large data sets.

ii) Centroid Models: K-Means

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such

that the squared distances (for the *p*-dim variables) from the cluster are minimized.

The optimization problem itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximate method is Lloyd's algorithm,[8] often just referred to as "*k-means algorithm*". It does however only find a local optimum, and is commonly run multiple times with different random initializations. Variations of *k*-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set (*k*-medoids), choosing medians (*k*-medians clustering), choosing the initial centers less randomly (*k*-means++) or allowing a fuzzy cluster assignment (fuzzy c-means).

Most *k*-means-type algorithms require the number of clusters - *k* - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders of clusters (which is not surprising since the algorithm optimizes cluster centers, not cluster borders).

Remark: K-means has a number of interesting theoretical properties. First, it partitions the data space into a structure known as a Voronoi diagram. Second, it is conceptually close to nearest

neighbor classification, and as such is popular in machine learning. Third, it can be seen as a variation of model based clustering, and Lloyd's algorithm as a variation of the Expectation-maximization algorithm for this model discussed in Multivariate Normal Mixture Models.

Neural Models: Self-Organization Map and Multi-Dimensional Scaling

A **self-organizing map** (**SOM**) or **self-organizing feature map** (**SOFM**) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a **map**, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense

that they use a neighborhood function to preserve the topological properties of the input space.

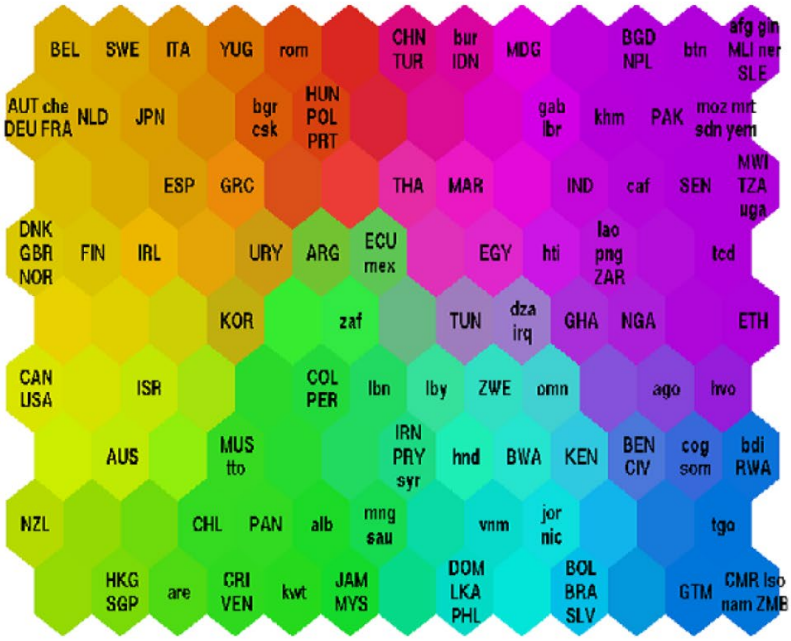
SOMs are useful for visualizing low-dimensional views of high-dimensional data, akin to multidimensional scaling. Like most

artificial neural networks, SOMs operate in two modes: training and mapping. "Training" builds the map using input examples, while "mapping" automatically classifies a new input vector.

A self-organizing map consists of components called nodes or neurons. Associated with each node are a weight vector of the same dimension as the input data vectors, and a position in the map space. The usual arrangement of **nodes is a two-dimensional regular spacing in a hexagonal or rectangular grid**. The self-organizing map describes a mapping from a higher-dimensional input space to a lower-dimensional map space. ***The procedure for placing a vector from data space onto the map is to find the node with the closest (smallest distance metric) weight vector to the data space vector.***

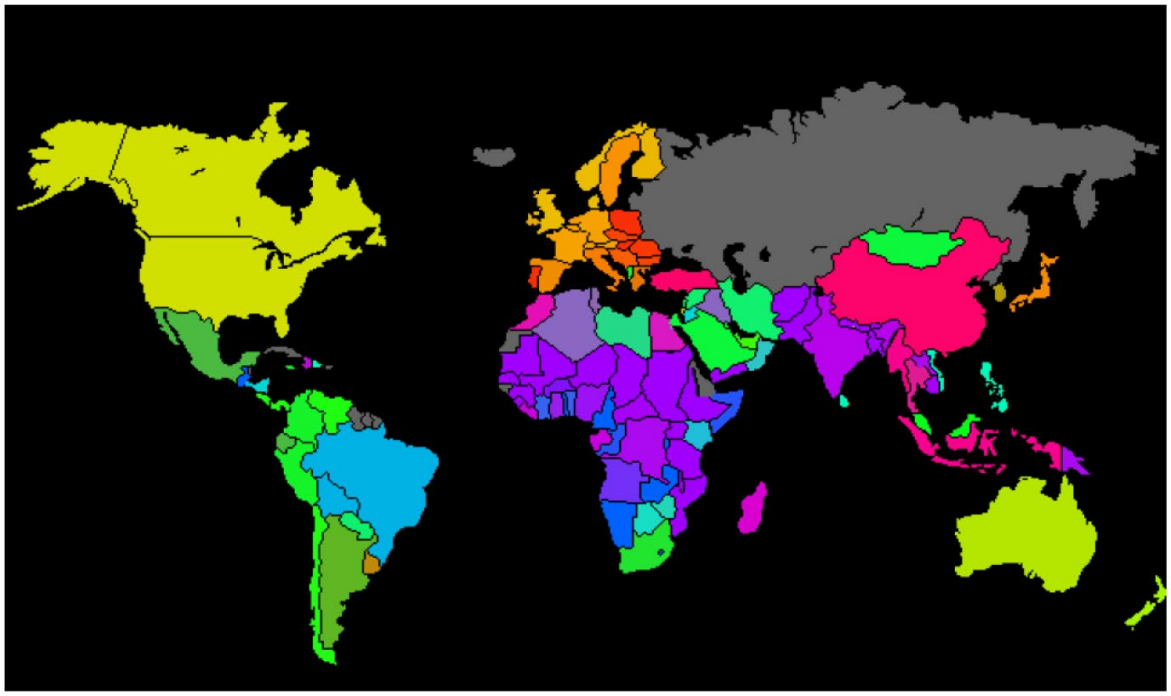
It has been shown that while self-organizing maps with a small number of nodes behave in a way that is similar to K-means, larger self-organizing maps rearrange data in a way that is fundamentally topological in character.

Classifying World Poverty



The Country Names

AFG	Afghanistan	GTM	Guatemala	NZL	New Zealand
AGO	Angola	HKG	Hong Kong	OAN	Taiwan, China
ALB	Albania	HND	Honduras	OMN	Oman
ARE	United Arab Emirates	HTI	Haiti	PAK	Pakistan
ARG	Argentina	HUN	Hungary	PAN	Panama
AUS	Australia	HVO	Burkina Faso	PER	Peru
AUT	Austria	IDN	Indonesia	PHL	Philippines
BDI	Burundi	IND	India	PNG	Papua New Guinea
BEL	Belgium	IRL	Ireland	POL	Poland
BEN	Benin	IRN	Iran, Islamic Rep.	PRT	Portugal
BGD	Bangladesh	IRQ	Iraq	PRY	Paraguay
BGR	Bulgaria	ISR	Israel	ROM	Romania
BOL	Bolivia	ITA	Italy	RWA	Rwanda
BRA	Brazil	JAM	Jamaica	SAU	Saudi Arabia
BTN	Bhutan	JOR	Jordan	SDN	Sudan
BUR	Myanmar	JPN	Japan	SEN	Senegal
BWA	Botswana	KEN	Kenya	SGP	Singapore
CAF	Central African Rep.	KHM	Cambodia	SLE	Sierra Leone
CAN	Canada	KOR	Korea, Rep.	SLV	El Salvador
CHE	Switzerland	KWT	Kuwait	SOM	Somalia
CHL	Chile	LAO	Lao PDR	SWE	Sweden
CHN	China	LBN	Lebanon	SYR	Syrian Arab Rep.
CIV	Cote d'Ivoire	LBR	Liberia	TCD	Chad
CMR	Cameroon	LBY	Libya	TGO	Togo
COG	Congo	LKA	Sri Lanka	THA	Thailand
COL	Colombia	LSO	Lesotho	TTO	Trinidad and Tobago
CRI	Costa Rica	MAR	Morocco	TUN	Tunisia
CSK	Czechoslovakia	MDG	Madagascar	TUR	Turkey
DEU	Germany	MEX	Mexico	TZA	Tanzania
DNK	Denmark	MLI	Mali	UGA	Uganda
DOM	Dominican Rep.	MNG	Mongolia	URY	Uruguay
DZA	Algeria	MOZ	Mozambique	USA	United States
ECU	Ecuador	MRT	Mauritania	VEN	Venezuela
EGY	Egypt, Arab Rep.	MUS	Mauritius	VNM	Viet Nam
ESP	Spain	MWI	Malawi	YEM	Yemen, Rep.
ETH	Ethiopia	MYS	Malaysia	YUG	Yugoslavia
FIN	Finland	NAM	Namibia	ZAF	South Africa
FRA	France	NER	Niger	ZAR	Zaire
GAB	Gabon	NGA	Nigeria	ZMB	Zambia
GBR	United Kingdom	NIC	Nicaragua	ZWE	Zimbabwe
GHA	Ghana	NLD	Netherlands		
GIN	Guinea	NOR	Norway		
GRC	Greece	NPL	Nepal		



vii) Statistical Distribution Models: Multivariate Gaussian Mixtures

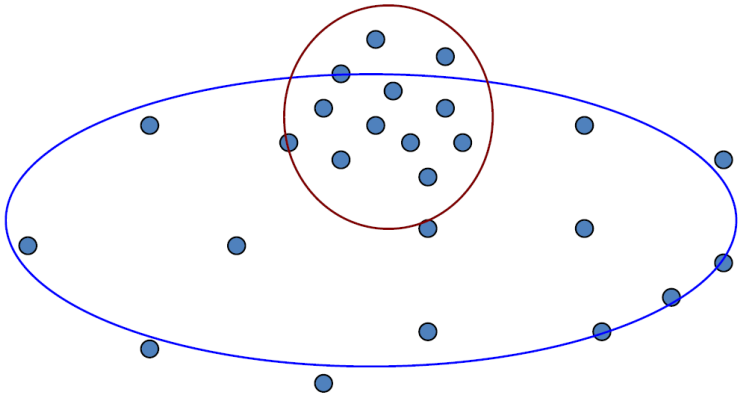
The clustering model most closely related to statistics is based on distribution models, e.g., multivariate Gaussian mixtures. Clusters can then easily be defined as objects belonging most likely to the same distribution. While the theoretical foundation of these methods is excellent, they suffer from one key problem known as overfitting, unless constraints (e.g., #clusters are known/fixed) are put on the model complexity. A more complex model will usually be able to explain the data better, which makes choosing the appropriate model complexity inherently difficult.

Distribution-based clustering produces complex models for clusters that can capture correlation and dependence between attributes. However, these algorithms put an extra burden on the user: for many real data sets, there may be no concisely defined mathematical model (e.g. assuming Gaussian distributions is a rather strong assumption on the data).

The Gaussian Mixture Model (GMM) provides "soft clustering" or "probability clustering" results.

- Soft clustering gives probabilities that an instance belongs to each of a set of clusters

Probabilistic Clustering



- Clusters of different shapes and sizes
- Clusters can overlap! (k-means doesn't allow this)

Finite Mixture Models

- Given a dataset: $x^{(1)}, \ldots, x^{(N)}$
- **Mixture model**: $\Theta = \{\lambda_1, \ldots, \lambda_k, \theta_1, \ldots, \theta_k\}$

$$p(x|\Theta) = \sum_{y=1}^{k} \lambda_y p_y(x|\theta_y)$$

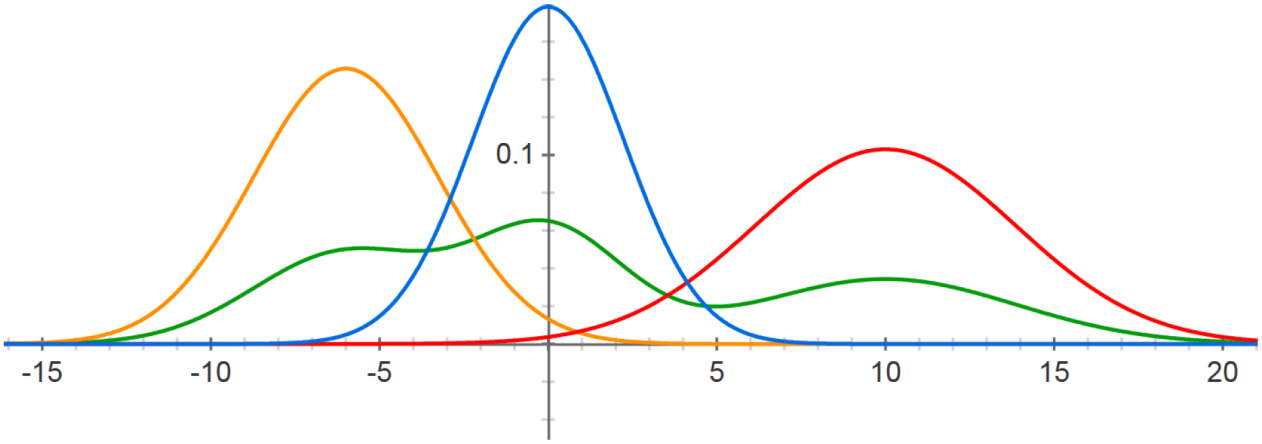
where $p_y(x|\theta_y)$ is a mixture component from some family of probability distributions parameterized by θ_y and $\lambda \geq 0$ such that $\sum_y \lambda_y = 1$ are the mixture weights

Remark: Please view "y" notation as the index for *K* distributions, i.e., y = 1, 2, …, *K. Students can treat this index y as "j".* We will

use the "*j*" index from now on for representing the *j*-th probability distribution in the mixture model.

Examples:

Finite Mixture Models



Uniform mixture of 3 Gaussians

Typically, in the GMMs, the following multivariate normal are used to model the probability distributions $P_j(x|\theta_j)$, $j = 1, 2, \ldots, K$.

Multivariate Gaussian

- A d-dimensional multivariate Gaussian distribution is defined by a $d \times d$ covariance matrix Σ and a mean vector μ

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

- The covariance matrix describes the degree to which pairs of variables vary together
  - The diagonal elements correspond to variances of the individual variables

Remark: The mean μ and variance-covariance $\sum$ in the probability distributions $P_j(x|\theta_j)$, $j = 1, 2, \ldots, K$, should be μ_j and $\sum_j$ from the j-th multivariate normal distribution. Note that these means μ_j and variance-covariances $\sum_j$ from different distributions $P_j(x|\theta_j)$, $j = 1, 2, \ldots, K$, are usually different.

Using the following Expectation-Maximization (EM) Algorithm to estimate these model parameters λ_j, μ_j and $\sum_j$ for $j = 1, 2, \ldots, K$. Details of the theoretical background and its derivation are skipped here. We provide a graphical presentation of the implementation of the EM algorithm in the GMM parameter estimation process.

EM for Gaussian Mixtures

- **E-step:**

$$q_i^t(y) = \frac{\lambda_y^t \cdot p\left(x^{(i)} \middle| \mu_y^t, \Sigma_y^t\right)}{\sum_{y'} \lambda_{y'}^t \cdot p\left(x^{(i)} \middle| \mu_{y'}^t, \Sigma_{y'}^t\right)}$$

Probability of $x^{(i)}$ under the appropriate multivariate normal distribution

- **M-step:**

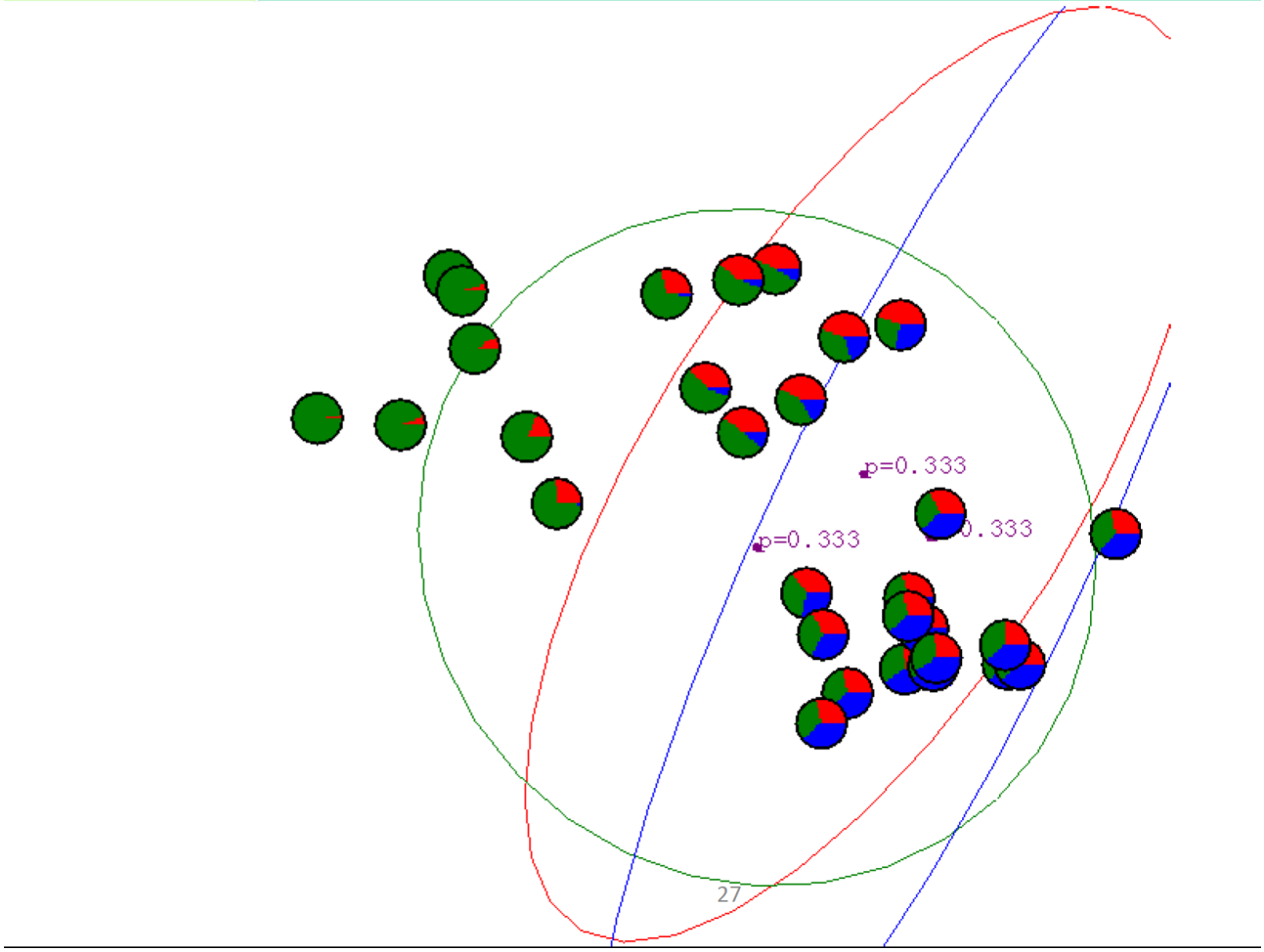
$$\mu_y^{t+1} = \frac{\sum_{i=1}^N q_i^t(y) x^{(i)}}{\sum_{i=1}^N q_i^t(y)}$$

$$\Sigma_y^{t+1} = \frac{\sum_{i=1}^N q_i^t(y)\left(x^{(i)} - \mu_y^{t+1}\right)\left(x^{(i)} - \mu_y^{t+1}\right)^T}{\sum_{i=1}^N q_i^t(y)}$$

$$\lambda_y^{t+1} = \frac{1}{N}\sum_{i=1}^N q_i^t(y)$$

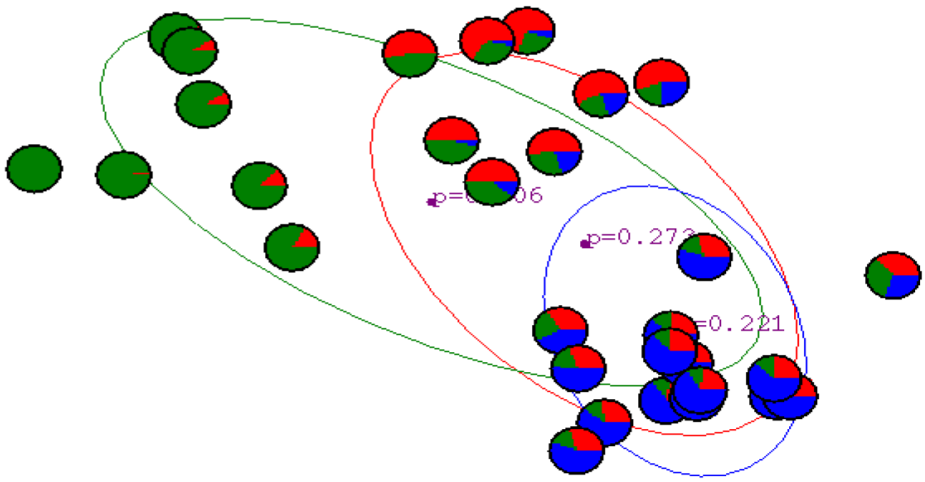
Graphical Presentation of Implementing the EM-Algorithm in GMMs' Parameter Estimation with a $K = 3$ example:

Gaussian Mixture Example: Start

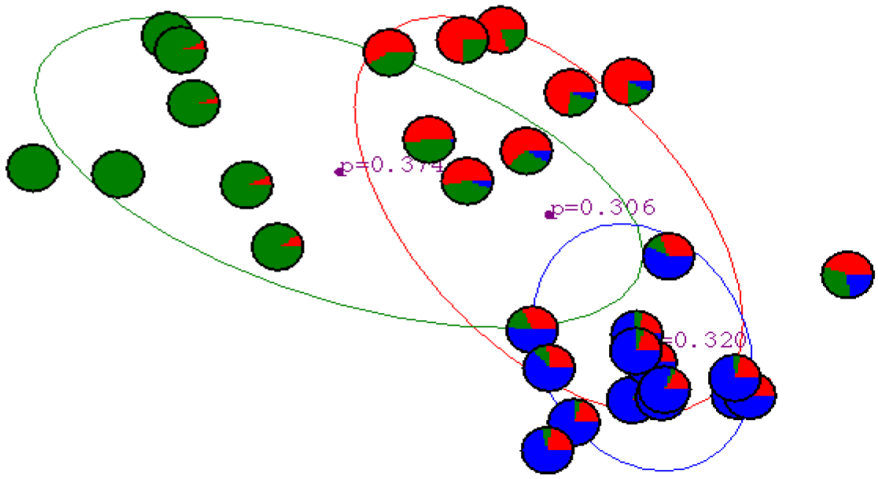


Remark: Here, "p_j =eg= 0.333" is the "weight" λ_j, j = 1, 2, …, K =eg= 3. The initial weights are usually equal weights.

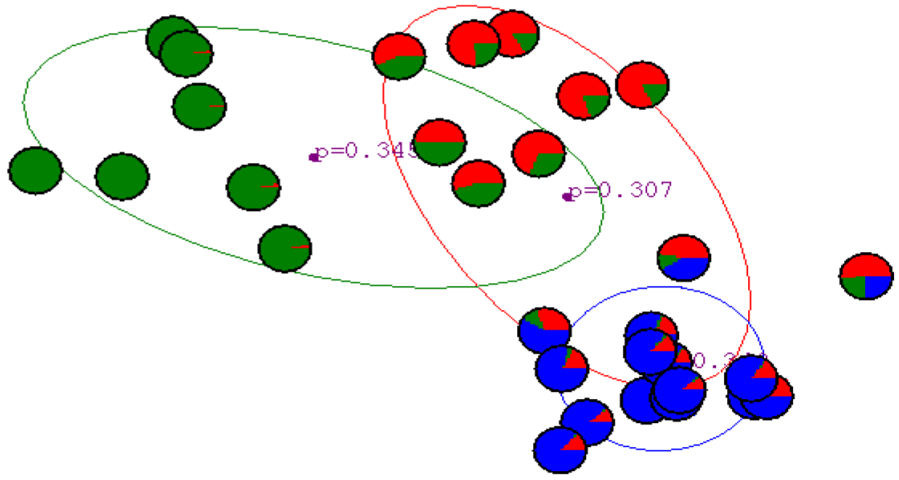
After first iteration



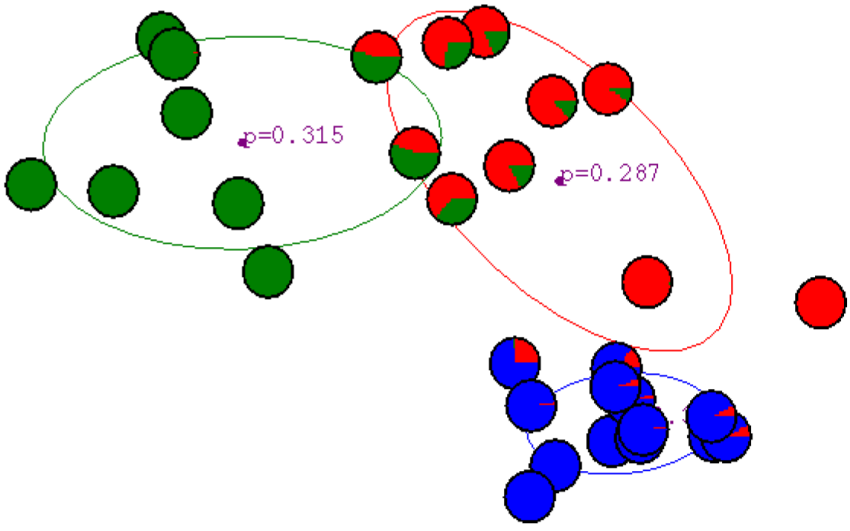
After 2nd iteration



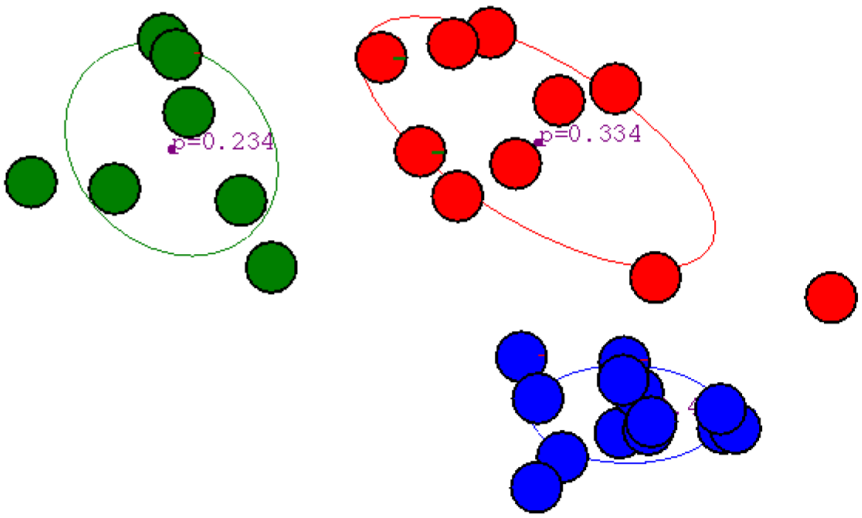
After 3rd iteration



After 6th iteration



After 20th iteration



Properties of EM

- EM converges to a local optima
  - This is because each iteration improves the log-likelihood

---

III) Dimension Reductions (DRs)

Dimension Reduction (DR) is a school of **Unsupervised Learning Procedures** (i.e., there is **no Y-outcomes data**; only

analyze X-data). This class focuses on high-level concepts. The following are three popular DR procedures.

(1) Principal Component Analysis (PCA)

(2) Multi-Dimensional Scaling (MDS)

(3) Partial Least Squares (PLS)

Details:

(1) Principal Component Analysis (PCA)

Principal components **(PC)** of a set of X-data in R^p provide **a sequence of best *linear approximations* to that data**, of all ranks $q \leq p$. These **principal components are *mutually uncorrelated and ordered in variance.*** See the following figure for an example.

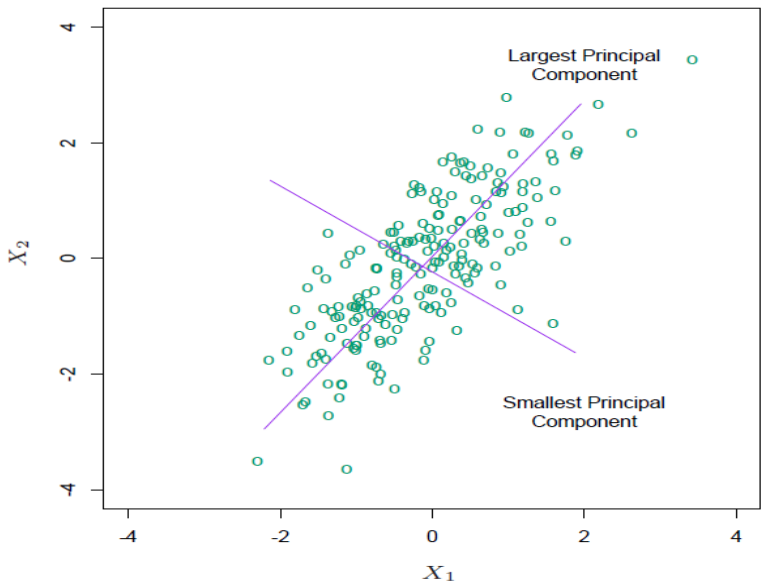


FIGURE 3.9. *Principal components of some input data points. The largest principal component is the direction that maximizes the variance of the projected data, and the smallest principal component minimizes that variance. Ridge regression projects* $\mathbf{y}$ *onto these components, and then shrinks the coefficients of the low–variance components more than the high-variance components.*

Denote the observations by $x_1, x_2, \ldots, x_N$, (each of them is a p-dim vector) and consider the following rank-q linear model for representing (or approximating) them:

$$f(\lambda) = \boldsymbol{\mu} + \mathbf{V}_q \boldsymbol{\lambda},$$

Vq is the "direction" for the projection (from a higher-dim to a lower-dim. See Figure 14.20 for a graphical presentation.

where $\boldsymbol{\mu}$ is a *p-dim* location vector, $\mathbf{V}_q$ is a *p-row times q-column* matrix with q orthogonal unit-vectors as columns, and $\boldsymbol{\lambda}$ is a q-element vector of parameters. See Figure 4.20 and 4.21 for q = 1 (page 534) and 2 (page 536), respectively, illustration of the rank-q representing function $f(\lambda)$.

mu is the "center" (or "origin")

parameters is the q-dim projected data-value, where data is in p-dim; p > q.

Example #1 (2-D X-data are projected to the first PC-line):

Figure 14.20 shows a set of 2-dimensional data points $x_1, x_2, \ldots, x_N$, plotted in the R^2 space. PC-Line projects these 2D data x_i's onto one-dimensional line with v_1 as the **direction with a value (U_{i1} * d_1)**, and black-dot-point as the center. Define an objective function as the sum-of-squares of distances from each point to its orthogonal projection onto the "PC-line" for locating the PC-line

with the direction v_1. The parameter-vectors μ and λ, and also the resulted lower-rank matrix $\boldsymbol{Vq}$ can be obtained.

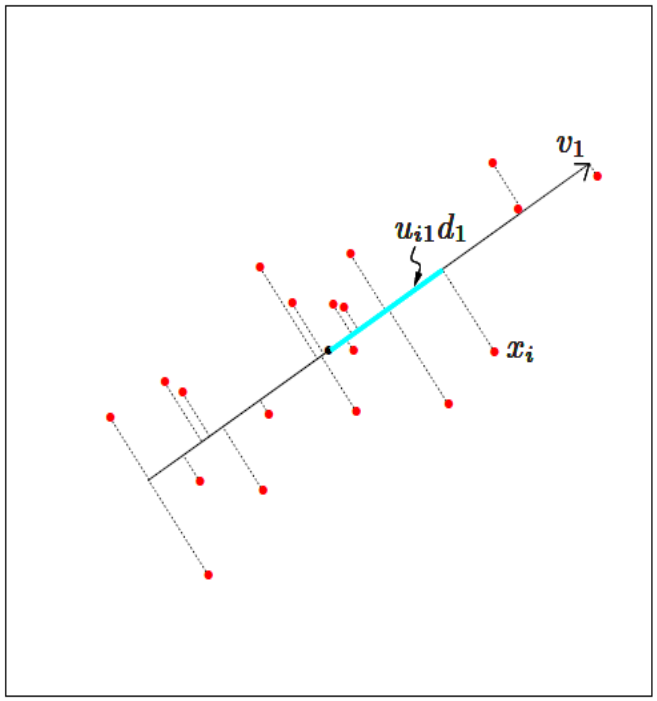


FIGURE 14.20. *The first linear principal component of a set of data. The line minimizes the total squared distance from each point to its orthogonal projection onto the line.*

The solution of the matrix $\boldsymbol{Vq}$ can be expressed as follows. Stack the (centered) observations into the rows of an $N \times p$ matrix $\boldsymbol{X}$. We construct the ***singular value decomposition*** of $\boldsymbol{X}$: $\boldsymbol{X} = \boldsymbol{UDV}^{\mathrm{T}}$, where $\boldsymbol{U}$ is an $N \times p$ orthogonal matrix ($\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{Ip}$) whose columns $\boldsymbol{u}_j$ are called the left singular vectors; $\boldsymbol{V}$ is a $p \times p$ orthogonal matrix ($\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{Ip}$) with columns $\boldsymbol{v}_j$ called the right singular vectors, and $\boldsymbol{D}$ is a $p \times p$ diagonal matrix, with diagonal

elements $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$ known as the singular values. For each rank q, the solution Vq consists of the first q columns of V. **The columns of UD are called the principal components of X.** The N optimal λ_i's (elements in the vector $\boldsymbol{\lambda}$) are given by the first q principal components (the N rows of the $N \times q$ matrix $UqDq$).

Important Remark: Each principal component is a linearly weighted sum of the original p-data-columns X. Note that the original data variances is decomposed into p variances $d_1 \geq d_2 \geq \cdots \geq d_p \geq 0$ **"orthogonally."** The data represented by the j-th principal component captures d_j amount of variance.

Example #2 (3-D X-data are projected to the first two PCs):

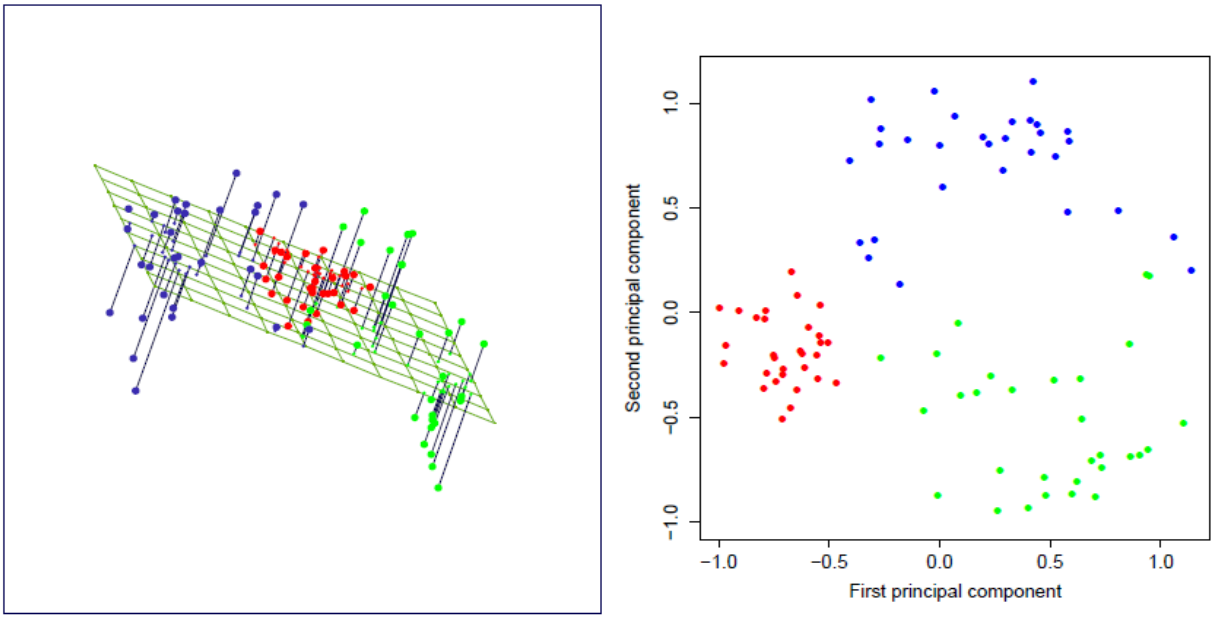


FIGURE 14.21. *The best rank-two linear approximation to the half-sphere data. The right panel shows the projected points with coordinates given by* $\mathbf{U}_2\mathbf{D}_2$, *the first two principal components of the data.*

v) Multi-Dimensional Scaling (MDS): (Important procedure)

MDS has the same goal as the one studied in PCA and SOM. However, MDS finds a lower-dimensional representation of the data that preserves the **pairwise distances as well as possible**. The following provides technical details.

Define $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$, the distance (dis-similarity) of two data points. MDS seeks values $z_1, z_2, \ldots, z_N \in \mathrm{R}^k$ to minimize the so-called stress function

$$S_M(z_1, z_2, \ldots, z_N) = \Sigma_{i \neq k} (d_{ik} - \|z_i - z_k\|)^2 .$$

This is known as least squares or Kruskal–Shephard scaling.

Remark: MDS only requires the dis-similarity data, but not the original data, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$. This property extends the application of dimension reduction to non-standard data such as qualitative data, e.g., color of

clothes. As long as one can provide the dis-similarity information, MDS can be applied.

vi) Partial least squares regression (PLS regression): (Important sub-topic)

Concept: PLS is a statistical method that bears some relation to principal components regression; instead of finding hyperplanes of maximum variance between the response and independent variables, it finds a linear regression model by projecting the predicted variables and the observable variables to a new space for modeling the covariance structures in these two spaces.

PCAs

Technical Details: For the following regression model

$$\boldsymbol{Y} = \boldsymbol{X}\,\boldsymbol{\beta} + \boldsymbol{e}. \quad \text{Model (1)}$$

Tk is the k-th PC.

PCA: $\boldsymbol{X} = \boldsymbol{T}\,\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{T}_k\,\boldsymbol{P}_k^{\mathrm{T}} + \boldsymbol{E}$, where $k = 1, 2, \ldots, K$ are the first, second, …, principal components that capture most, second-most, …, amount of X-data variances.

From SVD (Single-Value-Decomposition) $\boldsymbol{X} = (\boldsymbol{US})\,\boldsymbol{V}^{\mathrm{T}}$

with $\boldsymbol{T} = \boldsymbol{US}$ and $\boldsymbol{P} = \boldsymbol{V}$.

In application of PCA in regression consider the following regression estimates for model (1):

$\beta\_hat = (X^T X)^{-1} X^T Y = H\ Y$, where H is the "head" matrix.

PCA-Regression (PCR): Regress Y with the first k Principal Components (PCs). The head matrix becomes

$$H_k = P_k\ (\ T_k^T\ T_k\ )^{-1}\ T_k^T.$$

PLS tried to project both X- and **Y**-data into spaces (a lower-dim space) that maximizes the covariance between X and Y. It requires an addition of weights W to achieve such goal. The resulted head-matrix in regression estimate is

$$H_{k,PLS} = W_k\ (P_k^T\ W_k\ )^{-1}\ (\ T_k^T\ T_k\ )^{-1}\ T_k^T. \qquad \text{Model (2)}$$

There are several competing algorithms to find the head-matrix with various computing speed, accuracy and easiness-for-interpretation (in the algorithm).

Note: As seen from the PCR the PLS-variables (weighted sum of the original variables (like PCA)) found by the head-matrix in Model (2) might not have much **physical meaning** (due to its goal (and also the method) of constructing these PLS-variables). Moreover, based on Model (2), it seems that every PLS variable includes ALL "original" variables.

Please look into the past-exam problems for comparisons between PLS, PCA and MDS.

Some of these variables could be **correlated**. In the *large p small n problems*, there are many original-variables.

Dimension-Reduction in the large p small n problems is our next topic for lectures.

IV) Large *p* Small *n* Variable-Selections

Review: Variable-Selections for Linear (Normal) Regressions

(1) Traditional Problems (n ≡ sample size > p ≡ #x-variables)

(a) Stepwise Regression (with Partial F-Tests) (ISyE 3030/4031 materials)

(b) All-Subsets Regression (ISyE 3030/4031 materials)

(2) Recent Problems (n < p & n << p) (our focus in ISyE 4034)

(a) Ridge Regression, Lasso, Elastic Net Regularization

(b) Adaptive Lasso and Group Lasso

(c) Clustering and Representative-Selection

New: Variable-Selections for Linear Regressions

- **Recent Problems (n < p & n << p)**

(a) Ridge Regression, Lasso, Elastic Net Regularization

(b) Adaptive Lasso and Group Lasso

(c) Clustering and Representative-Selection

Details:

Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

- The *Lasso* is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|.$$

- In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum |\beta_j|$.

Lasso: Least Absolute Shrinkage and Selection Operator

ISyE DDA – Agenda for 03/09/23 Lecture (Lec.17)

I) Advanced Data Modeling – Classifications

A) Statistical Classification Procedures:

B) Computing/Optimization Based Classification Procedures

- **(a) Separating Hyperplane (done)**
- **(b) Support Vector Machine (done)**
- **(c) Decision Tree (done)**
- **(d) Artificial Neural Network (ANN) (done)**
- **(e) *K*-Nearest Neighbor (*k*NN)**

II) Nonparametric (NP) Regressions

III) Cluster Analysis

Detailed Lectures:

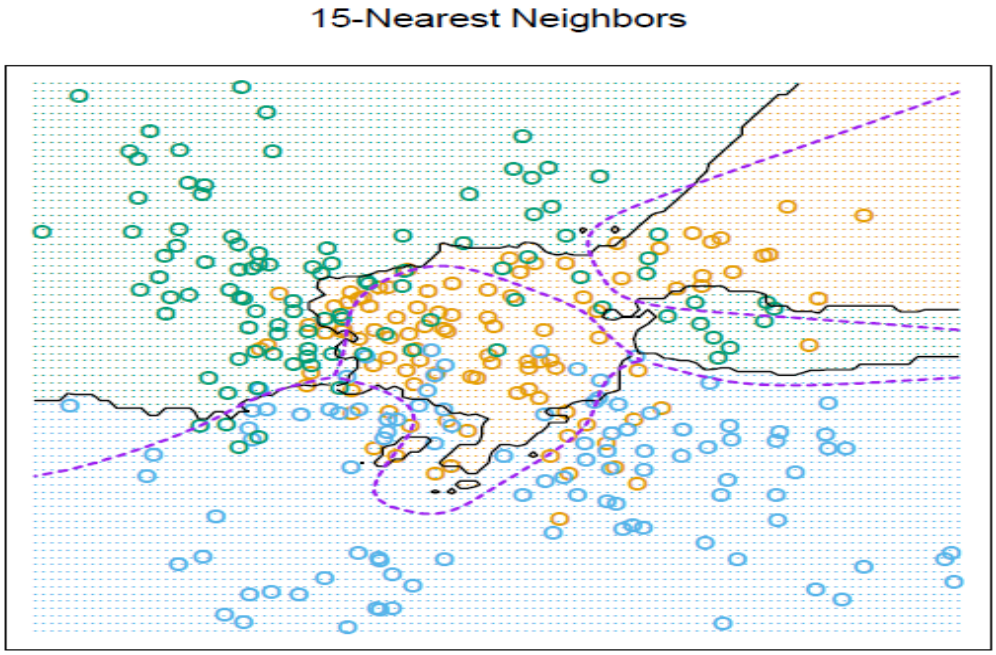
I) Classifications - *K*-Nearest Neighbor (*k*NN)

Given a query point $\boldsymbol{x}_0$**-vec** (to be classified), the *k*NN-classifier finds the k data points $\boldsymbol{x}_{(r)}$**-vec**, $r = 1, 2, \ldots, k$ (=eg= 5) closest in distance to $\boldsymbol{x}_0$**-vec**, and then classify using majority vote among the k neighbors.

In selecting these k neighbors, data with distances that are tied will be selected at random. For simplicity, we will assume that all x-variables are real-valued, and use the following Euclidean distance in the x-variable space:

$$d_{(i)} = \| \boldsymbol{x}_{(i)}\text{-vec} - \boldsymbol{x}_{0}\text{-vec} \|$$
$$= \text{Sqrt}[\ (x_{(i)1} - x_{01})^2 + (x_{(i)2} - x_{02})^2 + \ldots + (x_{(i)p} - x_{0p})^2\ ].$$

The following shows two examples with $k = 15$ and 1. Note that the figure in the bottom with $k = 1$ has very rugged decision-boundaries. The decision-boundaries for the $k = 15$ case is much smoother. One can use either cross-validation or test-data to evaluate their #mis-classifications to decide the best k to use for a particular data set. We will introduce these model-assessment procedures later.



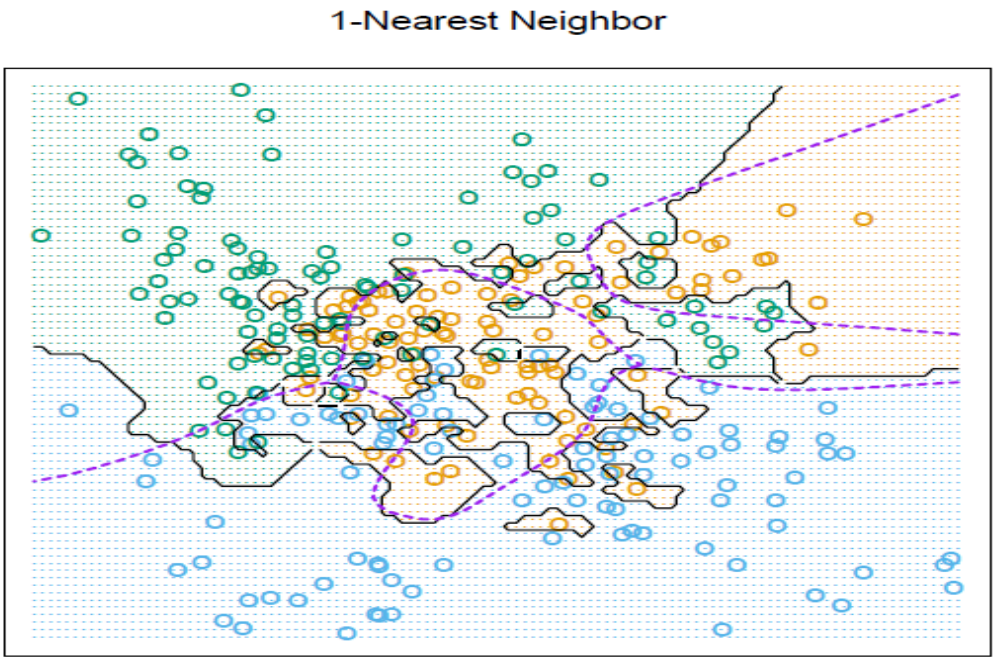


FIGURE 13.3. *k-nearest-neighbor classifiers applied to the simulation data of Figure 13.1. The broken purple curve in the background is the Bayes decision boundary.*

II) **Nonparametric Regressions:**

(1) Kernel Regression
(2) Nearest-Neighbor NP-Regression
(3) Splines

Details:

13.1 Kernel Estimators

Let $K(x)$ be a real-valued function for assigning local weights to the linear estimator, that is,

$$y(x) = \sum K\left(\frac{x - x_i}{h}\right) y_i.$$

If $K(u) \propto \mathbf{1}(|u| \leq 1)$ then a fitted curve based on $K(\frac{x-x_i}{h})$ will estimate $m(x)$ using only design points within h units of x. Usually it is assumed that $\int_R K(x)dx = 1$, so any bounded probability density could serve as a kernel. Unlike kernel functions used in density estimation, now $K(x)$ also can take negative values, and in fact such unrestricted kernels are needed to achieve optimal estimators in the asymptotic sense.

Note: $m(x)$ is the mean function.

The kernel function K has five important properties –

1.	$K(x)$	$\geq 0 \quad \forall x$	
2.	$K(x)$	$= K(-x)$	for $x > 0$
3.	$\int K(u)du$	$= 1$	
4.	$\int uK(u)du$	$= 0$	
5.	$\int u^2 K(u)du$	$= \sigma_K^2 < \infty.$	

Figure 11.4 shows four basic kernel functions:

1. Normal (or Gaussian) kernel $K(x) = \phi(x)$,
2. Triangular kernel $K(x) = c^{-2}(c - |x|)\mathbf{1}(-c < x < c),\ c > 0.$
3. Epanechnickov kernel (described below).
4. Box kernel, $K(x) = \mathbf{1}(-c < x < c)/(2c),\ c > 0.$

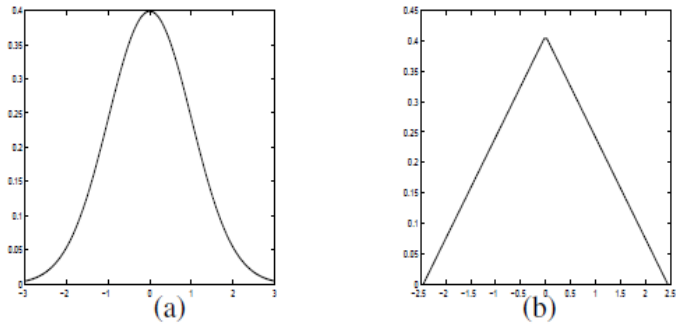


Figure 11.4: (a) Normal and (b) Triangular Kernel Functions

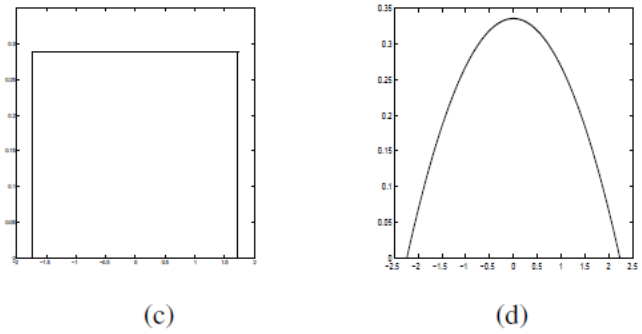


Figure 11.4: (c) Box and (b) Epanechnickov Kernel Functions

The following introduces a few popular kernel regression **estimators**.

13.1.1 Nadaraya-Watson Estimator

Nadaraya (1964) and Watson (1964) independently published the earliest results on for smoothing functions (but this is debateable), and the Nadaraya-Watson Estimator (NWE) of $m(x)$ is defined as

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K_h(X_i - x)Y_i}{\sum_{i=1}^{n} K_h(X_i - x)}. \tag{13.3}$$

For x fixed, the value $\hat{\theta}$ that minimizes

$$\sum_{i=1}^{n} (Y_i - \theta)^2 K_h(X_i - x), \tag{13.4}$$

is of the form $\sum_{i=1}^{n} a_i Y_i$. The Nadaraya-Watson estimator is the minimizer of (13.4) with $a_i = K_h(X_i - x)/\sum_{i=1}^{n} K_h(X_i - x)$.

Although several competing kernel-based estimators have been derived since, the NWE provided the basic framework for kernel estimators, including local polynomial fitting which is described later in this section. The R function

```
ksmooth(x,y,kernel,bandwidth)
```

computes the Nadaraya-Watson kernel estimate. Here, (X,Y) are input data, `kernel` is the kernel function, and `bandwidth` is the bandwidth.

EXAMPLE 13.1

Noisy pairs (X_i,Y_i), $i = 1,\ldots,200$ are generated in the following way:

```
> x <- sort(runif(200));
> y <- sort(4*pi*sort(runif(200))+0.3*rnorm(200));
```

Three bandwidths are selected $h = 0.07, 0.14,$ and 0.21. The three Nadaraya-Watson Estimators are shown in Figure 13.3. As expected, the estimators constructed with the larger bandwidths appear smoother than those with smaller bandwidths.

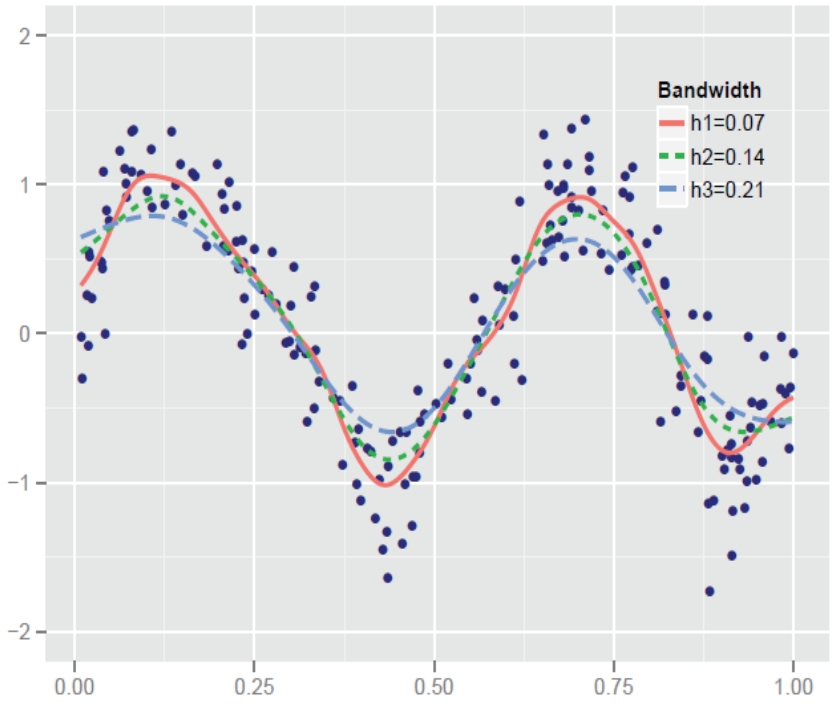


Figure 13.3 Nadaraya-Watson Estimators for different values of bandwidth.

Remarks for Kernel Bandwidth - Bias and Variance Tradeoffs:

1) **Smaller width**: few observations in the "window/band", each contribution is closer to x_0: Leads to higher variance

(estimated function will vary a lot), but lower bias (i.e., captures more details of data patterns).

2) **Larger width** leads to lower variance due to average of more observations; but the bias is larger - observations from further away contribute to the estimated function at x_0.

Remarks:

[1] **Cross-Validation** is a common tool to select the "best" **tuning parameter *h*.**

Let $\hat{m}_{(i)h}(x)$ be the estimator of $m(x)$, based on bandwidth parameter h, obtained by using all the observation pairs except the pair (X_i, Y_i). Define the cross-validation score $CV(h)$ depending on the bandwith/trade-off parameter h as

$$CV(h) = \frac{1}{n}\sum_{i=1}^{n}\left[Y_i - \hat{m}_{(i)h}(x)\right]^2. \qquad (13.12)$$

Find h to minimize the above cross-validated MSE (cMSE).

[2] Other method for deciding h involves asymptotic MISE (mean integrated square error) defined as follows.

While K controls the shape, h_n controls the spread of the kernel. The accuracy of a density estimator can be evaluated using the mean integrated squared error, defined as

$$\begin{aligned} \text{MISE} &= \mathbb{E}\left(\int (f(x)-\hat{f}(x))^2 dx\right) \\ &= \int \text{Bias}^2(\hat{f}(x))dx + \int \mathbb{V}\text{ar}(\hat{f}(x))dx. \end{aligned} \tag{11.2}$$

Details of MISE calculation is skipped. Because there is a lot of work involved in estimating the MISE for model validation and comparisons. Thus, **this class uses cross-validation to select the optimal bandwidth *h***.

---

(2) Nearest-Neighbor NP-Regression

13.2.1 LOESS

William Cleveland (1979), Figure 13.4(a), introduced a curve fitting regression technique called LOWESS, which stands for *locally weighted regression scatter plot smoothing*. Its derivative, LOESS[1], stands more generally for a local regression, but many researchers consider LOWESS and LOESS as synonyms.

Consider a multiple linear regression set up with a set of regressors $X_i = X_{i1}, \ldots, X_{ik}$ to predict Y_i, $i = 1, \ldots, n$. If $Y = f(x_1, \ldots, x_k) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Adjacency of the regressors is defined by a distance function $d(X, X^*)$. For $k = 2$, if we are fitting a curve at (X_{r1}, X_{r2}) with $1 \leq r \leq n$, then for $i = 1, \ldots, n$,

$$d_i = \sqrt{(X_{i1} - X_{r1})^2 + (X_{i2} - X_{r2})^2}.$$

Each data point influences the regression at (X_{r1}, X_{r2}) according to its distance to that point. In the LOESS method, this is done with a tri-cube weight function

$$w_i = \begin{cases} \left(1 - \left(\frac{d_i}{d_q}\right)^3\right)^3 & d_i \le d_q \\ 0 & d_i > d_q \end{cases},$$

where only q of n points closest to X_i are considered to be "in the neighborhood" of X_i, and d_q is the distance of the furthest X_i that is in the neighborhood. Actually, many other weight functions can serve just as well as the tri-weight function; requirements for w_i are discussed in Cleveland (1979).

If q is large, the LOESS curve will be smoother but less sensitive to nuances in the data. As q decreases, the fit looks more like an interpolation of the data, and the curve is zig-zaggy. Usually, q is chosen so that $0.10 \le q/n \le 0.25$. Within the window of observations in the neighborhood of X, we construct the LOESS curve $Y(X)$ using either linear regression (called first order) or quadratic (second order).

There are great advantages to this curve estimation scheme. LOESS does not require a specific function to fit the model to the data; only a smoothing parameter ($\alpha = q/n$) and local polynomial (first or second order) are required. Given that complex functions can be modeled with such a simple precept, the LOESS procedure is popular for constructing a regression equation with cloudy, multidimensional data.

On the other hand, LOESS requires a large data set in order for the curve-fitting to work well. Unlike least-squares regression (and, for that matter, many non-linear regression techniques), the LOESS curve does not give the user a simple math formula to relate the regressors to the response. Because of this, one of the most valuable uses of LOESS is as an exploratory tool. It allows the practitioner to visually check the relationship between a regressor and response no matter how complex or convoluted the data appear to be.

In R, use the function

```
loess(formula, span, degree)
```

EXAMPLE 13.2

Consider the motorcycle accident data found in Schmidt, Matter and Schüler (1981). The first column is time, measured in milliseconds, after a simulated impact of a motorcycle. The second column is the acceleration factor of the driver's head (accel), measured in g ($9.8m/s^2$). Time versus accel is graphed in Figure 13.5. The R code below creates a LOESS curve to model acceleration as a function of time (also in the figure). Note how the smoothing parameter influences the fit of the curve.

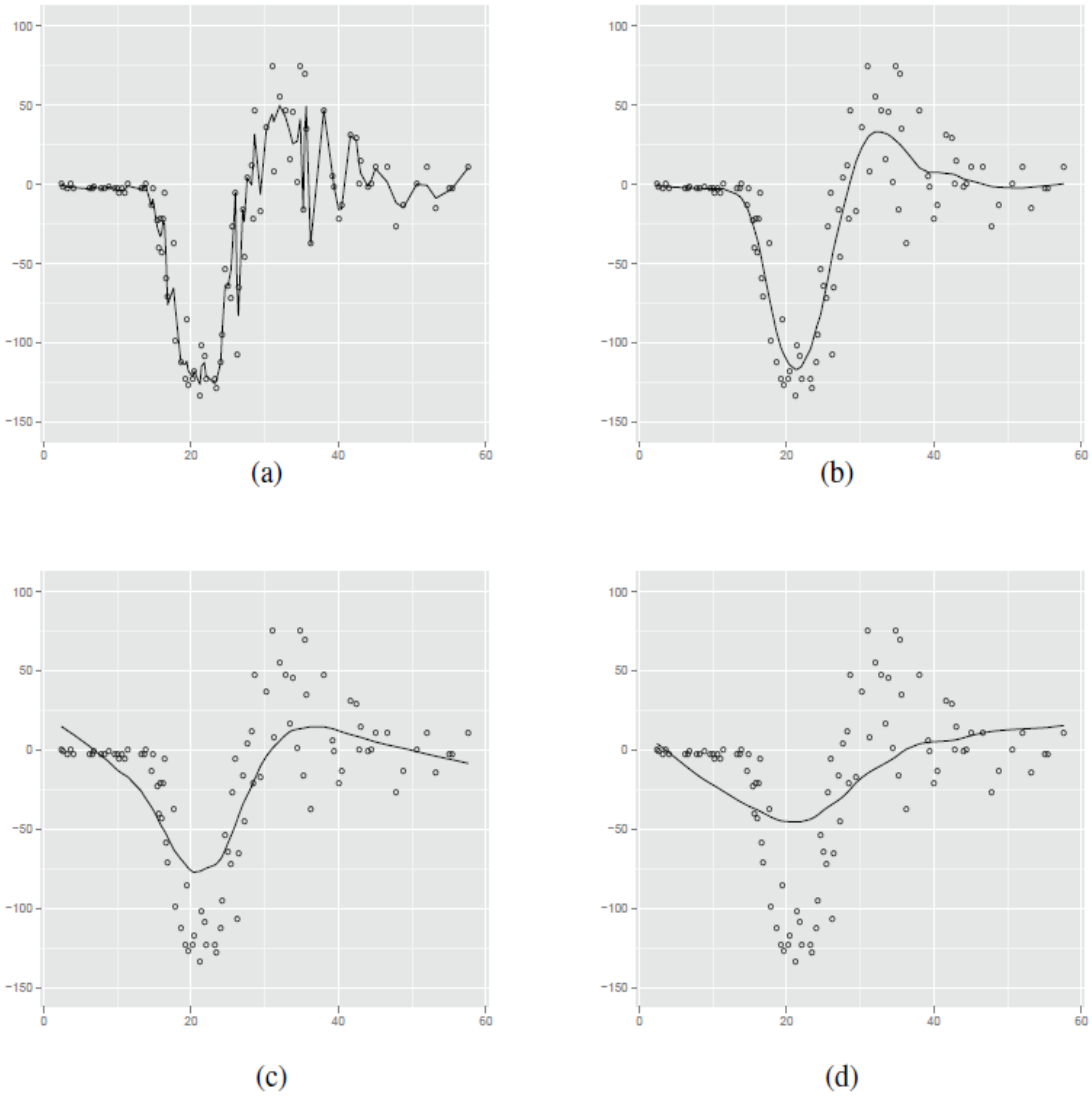


Figure 13.5 Loess curve-fitting for Motorcycle Data using (a) $\alpha = 0.05$, (b) $\alpha = 0.20$, (c) $\alpha = 0.50$, and (d) $\alpha = 0.80$.

```
> motor <- read.table("./motorcycle.dat");
> time <- motor[,1];
> accel <- motor[,2];
> fit<-loess(accel~time,span=0.2,degree=1);
> plot(time,accel,ylim=c(-150,100))
> lines(time,fitted(fit),type="l")
>
> motor.plot <- function(alpha){
+ fit <- loess(accel~time,span=alpha,degree=1);
+ dat <- data.frame(x=time,y=fitted(fit));
+     p <- ggplot() + geom_point(aes(x=time,y=accel),shape=1)
+     p <- p + geom_line(aes(x=x,y=y),data=dat)
+     p <- p + xlab("") + ylab("") + ylim(c(-150,100))
+     print(p);
+ }
>
> motor.plot(0.05)
> motor.plot(0.2)
> motor.plot(0.5)
> motor.plot(0.8)
```

---

(3) Splines

Interpolating splines and **smoothing splines** are motivated from a different perspective than kernels and local polynomials; in the latter case, we started off with a special kind of local averaging and moved our way up to a higher-order local models. With

splines, we build up our estimate globally, from a set of select **basis functions**.

These basis functions, as you might guess, are splines. Let's assume that d = 1 for one input variable, for simplicity. A **k-th order spline** is a **piecewise polynomial function of degree k**, that is continuous and has continuous derivatives of orders 1, . . ., k − 1, **at its knot points**. Please see below for details.

13.4.1 Interpolating Splines

There are many varieties of splines. Although piecewise constant, linear, and quadratic splines easy to construct, cubic splines are most commonly used because they have a desirable extremal property.

Denote the cubic spline function by $m(x)$. Assume $X_1, X_2, \ldots, X_n$ are ordered and belong to a finite interval $[a, b]$. We will call $X_1, X_2, \ldots, X_n$ *knots*. On each interval $[X_{i-1}, X_i]$, $i = 1, 2, \ldots, n+1, X_0 = a, X_{n+1} = b$, the spline $m(x)$ is a polynomial of degree less than or equal to 3. In addition, these polynomial pieces are connected in such a way that the second derivatives are continuous. That means that at the knot points $X_i, i = 1, \ldots, n$ where the two polynomials from the neighboring intervals

meet, the polynomials have common tangent and curvature. We say that such functions belong to $\mathbb{C}^2[a,b]$, the space of all functions on $[a,b]$ with continuous second derivative.

The cubic spline is called *natural* if the polynomial pieces on the intervals $[a,X_1]$ and $[X_n,b]$ are of degree 1, that is, linear. The following two properties distinguish natural cubic splines from other functions in $\mathbb{C}^2[a,b]$.

Unique Interpolation. Given the n pairs, $(X_1,Y_1),\ldots,(X_n,Y_n)$, with distinct knots X_i there is a *unique* natural cubic spline m that interpolates the points, that is, $m(X_i)=Y_i$.

Extremal Property. Given n pairs, $(X_1,Y_1),\ldots,(X_n,Y_n)$, with distinct and ordered knots X_i, the natural cubic spline $m(x)$ that interpolates the points also minimizes the curvature on the interval $[a,b]$, where $a<X_1$ and $X_n<b$. In other words, for any other function $g\in\mathbb{C}^2[a,b]$,

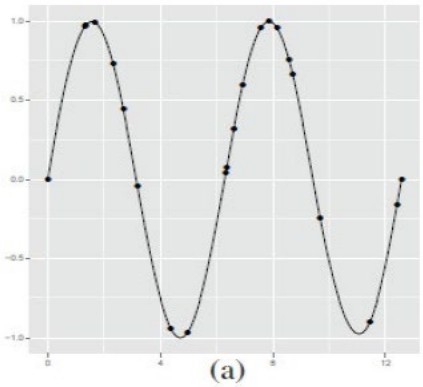
$$\int_a^b (m''(t))^2 dt \le \int_a^b (g''(t))^2 dt.$$

Remarks: It turns out that the cubic spline is in some sense asymptotically equivalent to a kernel regression, with an unusual choice of kernel, "Silverman kernel" (Silverman, 1984).

EXAMPLE 13.4

In R, the function `splinefun` and `bicubic` (in `akima` package) compute the cubic spline interpolant, and for the following x and y,

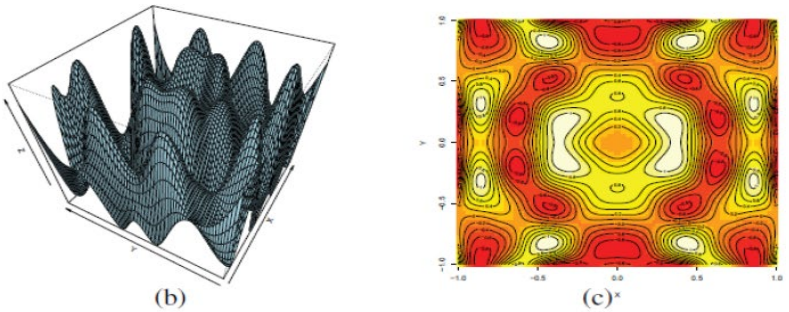
```
> x <- 4*pi*c(0,1,runif(20));
> y <- sin(x);
> fit <- splinefun(x,y);
> xx <- seq(0,max(x),length=100);yy<-fit(xx);
> p <- ggplot() + geom_point(aes(x=x,y=y),size=3)
> p <- p + geom_line(aes(x=xx,y=yy)) + xlab("") + ylab("")
> print(p)
```



(a)

the interpolation is plotted in Figure 13.8(a), along with the data. A surface interpolation by 2-d splines is demonstrated by the following R code and Figure 13.8(b) and (c).

```
> x <- seq(-1,1,by=0.2);
> y <- seq(-1,1,by=0.25);
> z <- outer(x,y,function(x,y){sin(10*(x^2+y^2));});
> xy<-expand.grid(seq(-1,1,length=100),seq(-1,1,length=80));
> fit<-bicubic(x,y,z,xy[,1],xy[,2]);
>
> xx <- seq(-1,1,length=100); yy <- seq(-1,1,length=80);
> zz <- matrix(fit$z,nrow=100);
> persp(x=xx,y=yy,z=zz,col="lightblue",phi=45,theta=-60,xlab="X",
+ ylab="Y",zlab="Z");
>
> image(x=seq(-1,1,length=100),y=seq(-1,1,length=80),
+       z=matrix(fit$z,nrow=100),xlab="X",ylab="Y");
> contour(x=seq(-1,1,length=100),y=seq(-1,1,length=80),
+       z=matrix(fit$z,nrow=100),add=TRUE);
```



(b) (c)

Figure 13.8 (a) Interpolating sine function; (b) Interpolating a surface; (c) Interpolating a contour.

13.4.2 Smoothing Splines

Smoothing splines, unlike interpolating splines, may not contain the points of a scatterplot, but are rather a form of nonparametric regression. Suppose we are given bivariate observations (X_i, Y_i), $i = 1, \ldots, n$. The continuously differentiable function $\hat{m}$ on $[a, b]$ that minimizes the functional

$$\sum_{i=1}^{n} (Y_i - m(X_i))^2 + \lambda \int_a^b (m''(t))^2 dt \tag{13.8}$$

is exactly a natural cubic spline. The cost functional in (13.8) has two parts: $\sum_{i=1}^{n} (Y_i - m(X_i))^2$ is minimized by an interpolating spline, and $\int_a^b (m''(t))^2 dt$ is minimized by a straight line. The parameter λ trades off the importance of these two competing costs in (13.8). For small λ, the minimizer is close to an interpolating spline. For λ large, the minimizer is closer to a straight line.

Although natural cubic smoothing splines do not appear to be related to kernel-type estimators, they can be similar in certain cases. For a value of x that is away

Smoothing Splines as Linear Estimators. The spline estimator is linear in the observations, $\hat{\mathbf{m}} = S(\lambda)\mathbf{Y}$, for a smoothing matrix $S(\lambda)$. The Reinsch algorithm (Reinsch, 1967) efficiently calculates S as

$$S(\lambda) = \left(I + \lambda Q R^{-1} Q'\right)^{-1}, \tag{13.11}$$

See textbook page 257 for the expressions for Q and R.

"Textbook": Kvam, Vidakovic and Kim (2016), Nonparametric Statistics with Applications in Science and Engineering, John Wiley. (ISBN: 978-0-470-08147-1).

III) Unsupervised Learning

(1) Clustering Analysis

a) Computing Methods:

i) **Connectivity Models:** Hierarchical Clustering

ii) **Centroid Models:** K-Means

iii) **Density Models:** DBSCAN and OPTICS

iv) **Subspace or Models:** Biclustering or Co-Clustering

v) **Neural Models:** Self-Organization Map and Multi-Dimensional Scaling

vi) **Graph-based Models:** Clique

b) Statistical Methods:

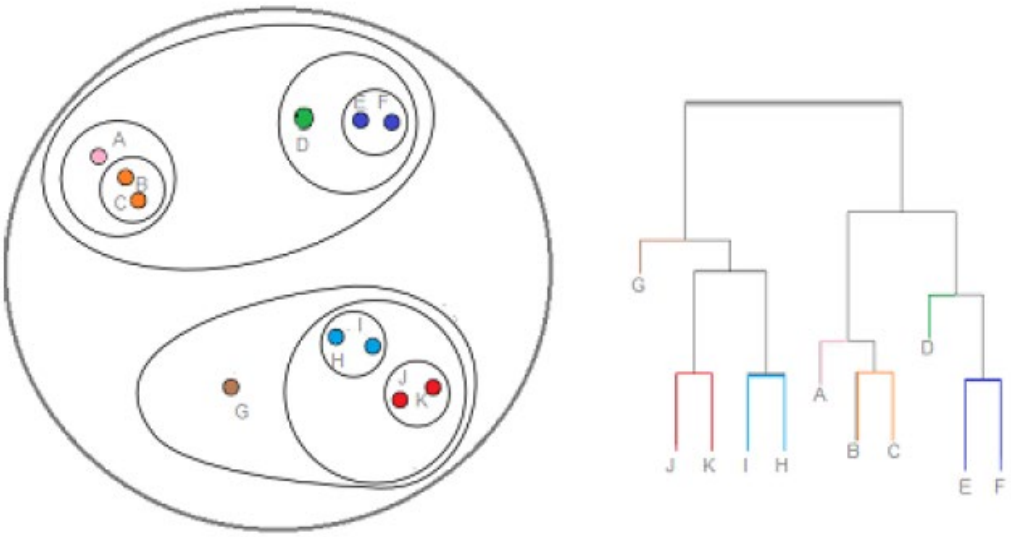
vii) **Distribution Models: Multivariate Normal Mixtures**

---

3. Details for Cluster Analysis

i) Connectivity Models: Hierarchical Clustering

Connectivity based clustering, also known as *hierarchical clustering*, is based on the core idea of objects being more related to nearby objects than to objects farther away. These algorithms connect "objects" to form "clusters" based on their distance. A cluster can be described largely by the maximum distance needed to connect parts of the cluster. At different distances, different clusters will form, which can be represented using a dendrogram, which explains where the common name "hierarchical clustering"



which explains where the common name "hierarchical clustering" comes from: these algorithms do not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances. In a dendrogram, the y-axis marks the distance at which the clusters merge, while the objects are placed along the x-axis such that the clusters don't mix.

Connectivity based clustering is a whole family of methods that differ by the way distances are computed. Apart from the usual choice of distance functions, the user also needs to decide on the linkage criterion (since a cluster consists of multiple objects, there are multiple candidates to compute the distance) to use. Popular choices are known as single-linkage clustering (the minimum of

object distances), complete linkage clustering (the maximum of object distances) or UPGMA ("Unweighted Pair Group Method with Arithmetic Mean", also known as average linkage clustering). Furthermore, hierarchical clustering can be ("bottom-up") agglomerative (starting with single elements and aggregating them into clusters) or ("top-down") divisive (starting with the complete data set and dividing it into partitions).

Remark: These methods will not produce a unique partitioning of the data set, but a hierarchy from which the user still needs to choose appropriate clusters. They are not very robust towards outliers, which will either show up as additional clusters or even cause other clusters to merge (known as "chaining phenomenon", in particular with single-linkage clustering). In the general case, the complexity is $O(n^3)$ for agglomerative clustering and $O(2^{n-1})$ for divisive clustering which makes them too slow for large data sets.

ii) Centroid Models: K-Means

In centroid-based clustering, clusters are represented by a central vector, which may not necessarily be a member of the data set. When the number of clusters is fixed to k, k-means clustering gives a formal definition as an optimization problem: find the k cluster centers and assign the objects to the nearest cluster center, such

that the squared distances (for the *p*-dim variables) from the cluster are minimized.

The optimization problem itself is known to be NP-hard, and thus the common approach is to search only for approximate solutions. A particularly well known approximate method is Lloyd's algorithm,[8] often just referred to as "*k-means algorithm*". It does however only find a local optimum, and is commonly run multiple times with different random initializations. Variations of *k*-means often include such optimizations as choosing the best of multiple runs, but also restricting the centroids to members of the data set (*k*-medoids), choosing medians (*k*-medians clustering), choosing the initial centers less randomly (*k*-means++) or allowing a fuzzy cluster assignment (fuzzy c-means).

Most *k*-means-type algorithms require the number of clusters - *k* - to be specified in advance, which is considered to be one of the biggest drawbacks of these algorithms. Furthermore, the algorithms prefer clusters of approximately similar size, as they will always assign an object to the nearest centroid. This often leads to incorrectly cut borders of clusters (which is not surprising since the algorithm optimizes cluster centers, not cluster borders).

Remark: K-means has a number of interesting theoretical properties. First, it partitions the data space into a structure known as a Voronoi diagram. Second, it is conceptually close to nearest

neighbor classification, and as such is popular in machine learning. Third, it can be seen as a variation of model based clustering, and Lloyd's algorithm as a variation of the Expectation-maximization algorithm for this model discussed in Multivariate Normal Mixture Models.

ISyE DDA – Agenda for 03/07/23 Lecture (Lec.16)

Advanced Data Modeling – Classifications

A) **Statistical Classification Procedures:**

- (a) **Bayes Classifier** (done)
- (b) **Discriminant Analysis: LDA/QDA/RDA** (done)
- (c) **Naïve Bayes Classifier** (done)
- (d) **Multinomial/Logistic Regression Based Classification**

B) **Computing/Optimization Based Classification Procedures**

- (e) **Separating Hyperplane**
- (f) **Support Vector Machine**
- (g) **Decision Tree**
- (h) **Artificial Neural Network (ANN)**
- (i) ***K*-Neararest Neighbor (*k*NN)**

Detailed Lectures:

1. **Advanced Data Modeling - Classifications**

Details:

(d) Multinomial/Logistic Regression Based Classifications

Step-1: Focus on an example with $K = 5$ classes. When there are more than two classes, instead of using a logistic regression to

fit the class Y-data, multinomial regression should be used to fit the data. There will be multiple logit-transformation to link π_j = Pr(Y = class j =eg= office dress) to regression functions for classes, $j = 1, 2, \ldots, K$ and $\sum_{j=1}^{K} \pi_j = 1$. For example,

$$\log_e[\ \Pr(Y = \text{class} = j) / \Pr(Y = \text{class} = 1)\ ]$$

$$= \beta_{0j} + \beta_{1j}\, x_1 + \beta_{2j}\, x_2 + \ldots + \beta_{pj}\, x_1,$$

where $x_1, x_2, \ldots, x_p$, are p explanatory variables associated for all classes Y-data, and $\beta_{0j}, \beta_{1j}, \beta_{2j}, \ldots, \beta_{pj}$ are (p + 1) coefficients in the above logit-regression function for class-j against class-1 for $j = 2, 3, \ldots, K$ (=eg= 5). Note that we use class-1 as the "base" for the "odd-ratio" Pr(Y = class = j) / Pr(Y = class = 1) against all other classes. With $K = 5$, there will be FOUR logit-transformations with j = 2, 3, 4, 5 = K in this multinomial regression. In each one of these logit-regressions, their regression coefficients $\beta_{0j}, \beta_{1j}, \beta_{2j}, \ldots, \beta_{pj}$ will be different for distinct class, $j = 2, 3, \ldots, K$.

Similar to the logistic regression, parameters are estimated via the maximum likelihood estimation (MLE) method. There, one needs to replace the Bernoulli distribution with the multinomial distribution and replace the distribution parameters $\pi_j = \Pr(Y = \text{class } j)$, $j = 1, 2, \ldots, K$, $\sum_{j=1}^{K} \pi_j = 1$, with those multiple logit-regressions discussed above. Then, employ a numerical method such as Newton-Raphson method to optimize log-likelihood with respect to the regression coefficients. Predictions π_j_hat $= \Pr(Y = \text{class } j)$_hat, $j = 1, 2, \ldots, K$, can then be obtained by replacing the unknown beta-coefficients with their MLEs.

For classifying a new data point $(Y_0, x_{10}, x_{20}, \ldots, x_{p0})$ to a class, one use the new set of explanatory variables $(x_{10}, x_{20}, \ldots, x_{p0})$ to predict this case's class-probabilities π_j_hat $= \Pr(Y_0 = \text{class } j)$_hat, for all K classes with $j = 1, 2, \ldots, K$. Then, a simple decision rule for classification is

"assign this data case to class m if its class-probability

$$\pi_m\_\text{hat} = \Pr(Y_0 = \text{class } m)\_\text{hat}$$ is the highest

compared to all other class-probabilities",

where m is one of the members in $j = 1, 2, \ldots, K$.

Remark:

[1] Multinomial/Logistic regression based classification procedures are statistical procedures (why?), but they are not Bayes Classifiers (why?).

[2] When misclassification costs are provided, one can develop a cost-based decision rule other than the one assign a data class with the highest class-probability.

General Remarks for Statistical Classification procedures:

[1] Statistical classifications are more rigorous but require more assumptions. Thus, it might not be as flexible as computing procedures to be presented next. However, statistical

classifications address the "population" problem. Computing procedures tend to address "sample" problem, i.e., the classification results vary according to data sampled.

[2] Statistical classifications involves distributions (e.g., prior, data-likelihood, posterior in Bayes classifiers, and multinomial distribution in "Multinomial Regression Based Classifications"). Computing procedures do not involve any distribution, but they have objective functions and constraints.

The following procedures are computing or optimization-based procedures.

#4) Separating Hyperplane: This is a "computing" procedure that finds linear or nonlinear "hyperplanes" (i.e., function of x-variables in a p-dimension space) to "separate" data in different classes. The following present three examples.

4.1) **Rosenblatt's (1958)** ***perceptron learning algorithm*** tries to find a separating hyperplane by **minimizing the distance of misclassified points to the decision boundary**. For instance, if a response $y_i = 1$ is misclassified (as $y_i\_hat = -1$), then $(x_i^T \beta\_hat + \beta_0\_hat) < 0$, and the opposite for a misclassified response with $y_i = -1$. Thus, the goal in finding this hyperplane is to find β_0_hat and β_hat to minimize

Two-class example with yi = +1 or -1. Its classification yi_hat will also be +1 or -1.

prediction

D is positive for a minus in front of the negative-summations.

$$D = -\sum_{i \text{ in } C} [\, y_i * (x_i^T \beta\_hat + \beta_0\_hat) \,],$$

data = yi; classification/prediction = yi_hat

If a data-case is mis-classified, then, yi * yi_hat will be equal to -1 due to their opposite signs.

Note that both data yi and its classification yi_hat will be either +1 or -1.

where C is the set includes *all misclassified points*. The algorithm implements a stochastic gradient descent to minimize the objective function D.

4.2) **Optimal Separating Hyperplanes (OSH):** The OSH separates the (two-) classes and maximizes the distance to the closest point from either class (Vapnik, 1996). Not only does this provide a **unique solution to**

the separating hyperplane problem, but by maximizing the margin between the two classes on the training data – this leads to better classification performance on test data. This is the idea in the **Support Vector Machine (SVM)**. A simple formulation of the above optimization problem is as follows.

The decision-boundary of this classification procedure is the middle of the "Margin".

In the original SVM development, it is assumed that there is no mis-classification case.

The idea of maximizing the MARGIN in SVM is rather new and innovative. It only needs to deal with data around the decision-boundary. These data are called "support-vectors".

Find β_0_hat and β_hat to maximize M

Subject to $[y_i * (x_i^T \beta\_hat + \beta_0\_hat)] \geq M$ = "margin"

(data: y_i; prediction: $x_i^T \beta\_hat + \beta_0\_hat$)

for $i = 1, 2, \ldots, n$, and

$\| \beta\_vec\_hat \| = 1$ (i.e., normalize the beta-coefficients).

Thus, if data yi is in the +1 case, and its prediction is positive, then, their multiply is positive. Similarly, if the data yi is "-1" and its prediction is negative, their multiply is also positive.

Note that $[y_i * (x_i^T \beta\_hat + \beta_0\_hat)]$ is the distance from a non-misclassified data-point to the decision boundary. Thus, the above optimization model find the "margin" M such that the all data cases are "outside" the (-M, M) region - this is what the above constraint for. The middle of the (-M, M) region is the decision-boundary for the two cases (one is in the negative side and the other is in the positive side). The data-cases that are closest to the margin "-M" or "+M" are called "support-vectors". Even that there might be outliers or millions of data points, only these few support-vectors decide the Margin and thus, the decision-boundary. Thus, SVM can be used to deal with a large number of data (with outliers).

#5) Support Vector Machine (SVM)

Let us use the following figures to explain the construction of decision boundary in the SVM. In the left figure, there is no mis-classification case. The two green dots and one red dot in the borders of the yellow box are called "support vectors". The size and location of the yellow box are decided by these support vectors. An optimization program will find these support vectors for a data set to maximize the size (i.e., margins) of the yellow box. The decision boundary for this SVM is the middle line in the yellow box.

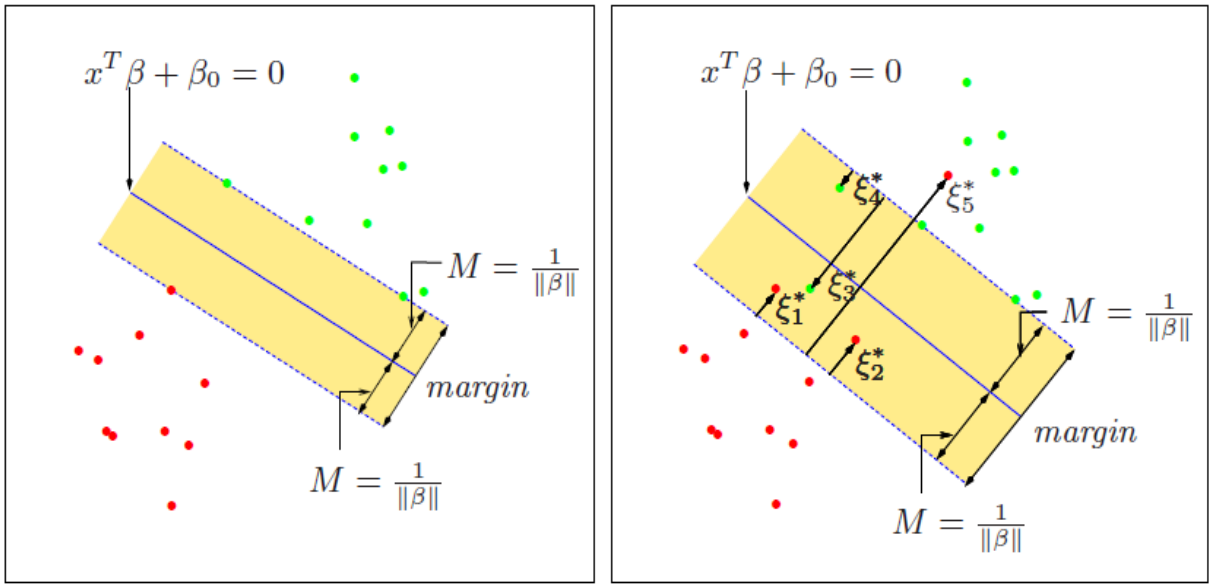


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

The right figure extends the simple SVM described above to handle potential "mis-classification" cases. Note that there are one red-dot classified into green-decision-region, and there is one green-dot (with ξ_3^* notation) classified in the left-side of the decision-boundary, i.e., red-decision-region. These are mis-classification cases. The idea of SVM in this "mis-classification" situation is to set a total budget to allow a small amount of mis-classifications. For example, in the right figure, there are FIVE cases that can be considered as "mis-classifications". The sum of their distances to the "correct-margins" is ξ_1^* (for the red-dot in the yellow-box and its distance to "red-margin") + ξ_2^* (similar to ξ_1^* in the red-zone) + ξ_3^* (for the green-dot in the yellow-box and its distance to "green-margin") + ξ_4^* (similar to ξ_3^*) + ξ_5^* (for the red-dot in the green-zone and its distance to "red-margin").

The budget of these "mis-classification" cases controls how many "mis-classified" points will be IGNORED in searching for the maximum margin. For example, when these FIVE cases addressed above are ignored, the support vectors are two green-dots on the margin of the yellow-box. The decision boundary for this SVM classification procedure is the middle line of the yellow-box. When the budget of "mis-classification" cases is changed, the

support vectors will change. Then, the decision boundary will change correspondingly.

#6) Mathematical Programming (MP) Based Separating Hyperplanes

This school of computing algorithms extends the above optimization-model to other optimization objective functions for allowing a limited amount of misclassifications, etc. Moreover, there could be different types of distances defined in various x-variable spaces.

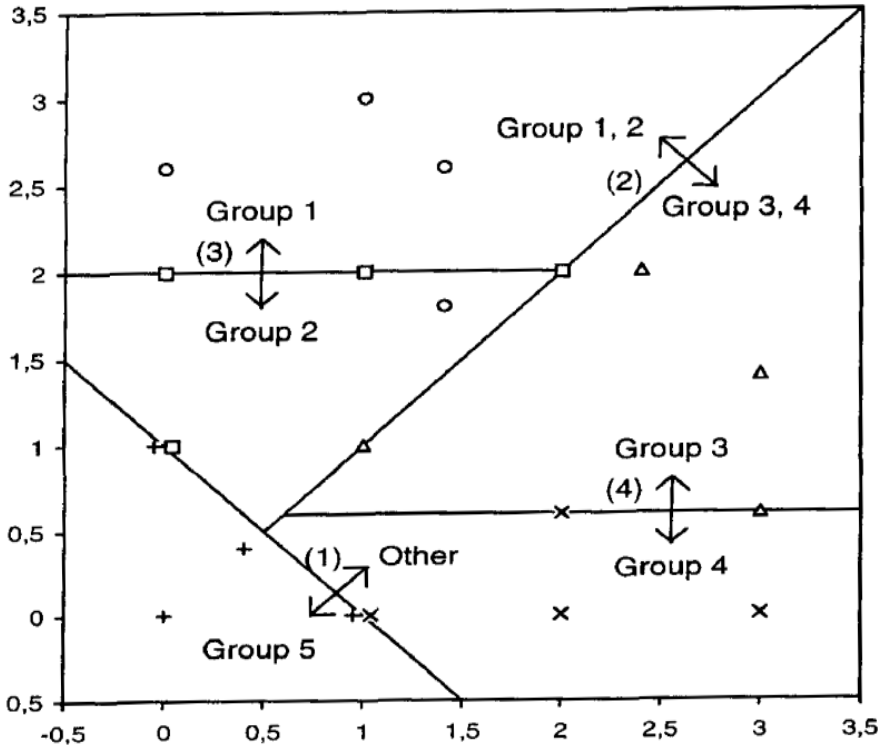
MP-based classification

Examples of objective functions:

- Minimize the sum of deviations
- Minimize the number of misclassifications
- Minimize the maximum deviation

Multi-group MP-based classification (cont')

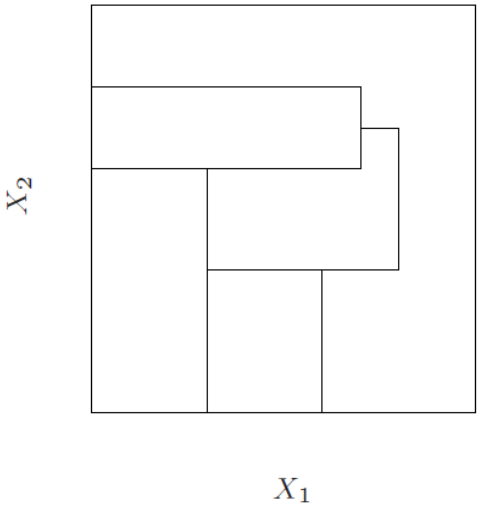
This example shows that MP-based classification procedure is very flexible in dealing with complicated situations with *non-standard shape of class-regions.*

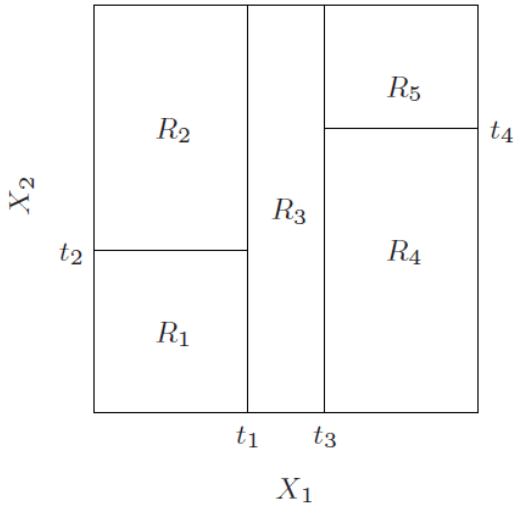


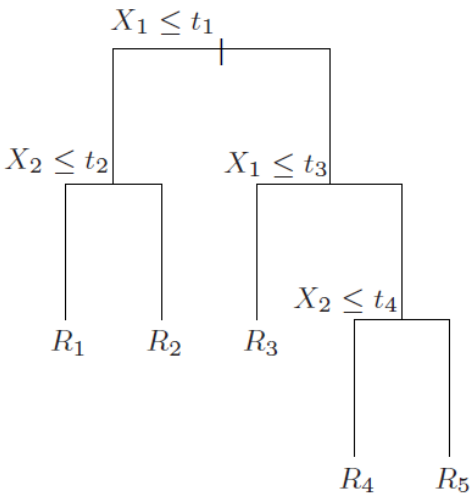
#7) Decision Tree: This school of procedures could be statistical or computational. The example presented below is a computing based decision tree.

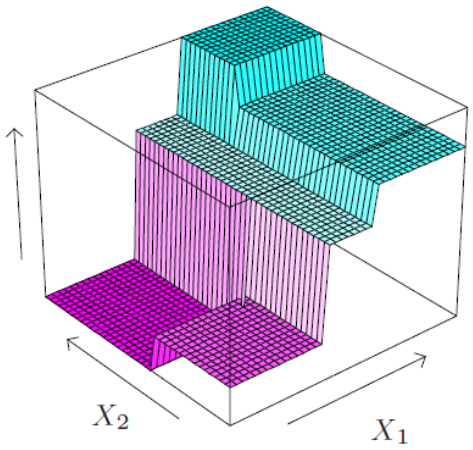
The let-bottom figure presents one example with five-nodes in a binary-split decision tree. Its decision boundaries in the two-x-variable (X_1 and X_2) domain is shown in the upper-right figure. The bottom right figure is the 3-D representation of these decision boundaries and their predictive surfaces for Y-class-outcomes.

The left-top figure shows that this type of non-binary-split decision boundaries cannot be handled by the traditional decision-tree procedures, which are usually binary-split trees.









Objectives **in the optimization procedures for building decision trees will be presented next.**

Notations:

[1] X-vec = $(X_1, X_2)^T$ in the above example; y-data = a collection of all y ≡ outcomes;

Note that outcomes in the decision-trees can be **classes (for classification) or numerical values (for regression)**; This is why decision-trees are called "**Classification and Regression Trees**";

[2] Decision-Regions R_m, m = 1, 2, …, 5, are decision-regions defined in the upper-right figure;

[3] (L_1, U_1), (L_2, U_2) are lower and upper bounds for X_1 and X_2, respectively;

[4] T is the collection of all terminal nodes defined by decision-regions R_m, m = 1, 2, …, 5; $|T|$ ≡ the size of the decision-tree = the number of terminal nodes =eg= 5;

[5] N_m = #{data in R_m}; Note that because R_m is defined for X-variables, #{data in R_m} is evaluated in terms of X-variable data in a specific region R_m;

[6] c_m_hat = average(y-data with their X-vec in R_m).

Classification/Prediction Function:

$$\boldsymbol{f\_hat}(\text{ X-vec}) = \sum_{m=1}^{5} c_m\_hat * I(\text{ X-vec in } R_m)$$

$$= \text{ average(y-data with their X-vec in } R_1) *$$

$$I(R_1 \equiv L_1 \le X_1 \le t_1, L_2 \le X_2 \le t_2)$$

$$+ \text{ average(y-data with their X-vec in } R_2) *$$

$$I(R_2 \equiv L_1 \le X_1 \le t_1, \ t_2 \le X_2 \le U_2)$$

$$+ \text{ average(y-data with their X-vec in } R_3) *$$

$$I(R_3 \equiv t_1 \le X_1 \le t_3, L_2 \le X_2 \le U_2)$$

$$+ \text{ average(y-data with their X-vec in } R_4) *$$

$$I(R_4 \equiv t_3 \le X_1 \le U_1, L_2 \le X_2 \le t_4)$$

$$+ \text{ average(y-data with their X-vec in } R_5) *$$

$$I(R_5 \equiv t_3 \le X_1 \le U_1, t_4 \le X_2 \le U_2).$$

***Objective Function* to Construct Decision-Trees:**

Find all decision-regions R_m (including which x-variable at what split-node of the tree and t_m-value of the split-decision), **m = 1, 2, …, M ≡ | T | (=eg= 5) to**

Minimize

$$C_\alpha(T) = \sum_{m=1}^{M} \sum_{xi\text{-}vec\ in\ Rm} (y_i - c_m\_hat)^2 + \alpha \mid T \mid,$$

where the first component represents the model-fitting quality and is the sum-of-squares (SS) of prediction-errors (SSE). Note that in the classification problems, the above SSE can be changed to some metrics related to #mis-classification cases. The second component represents the size of the tree, and α is a tuning parameter for balancing the fitting-quality against the complexity of the tree. Typically, α is decided by a cross-validation procedure, which will be presented later. The complexity of the tree is presented by the size of the tree $| T |$ = #terminal nodes.

There are many algorithms for constructing decision-trees to decide which x-variable should be used at what split-node and what should be its threshold-value (i.e., t_m-value) for the split-decision. HW-2 will ask students to review software available for decision-trees.

(h) Neural Network (or Artificial Neural Network (ANN)):

ANN is a two-stage regression or classification procedure. See Figure 11.2 below for a schematic diagram of a typical ANN.

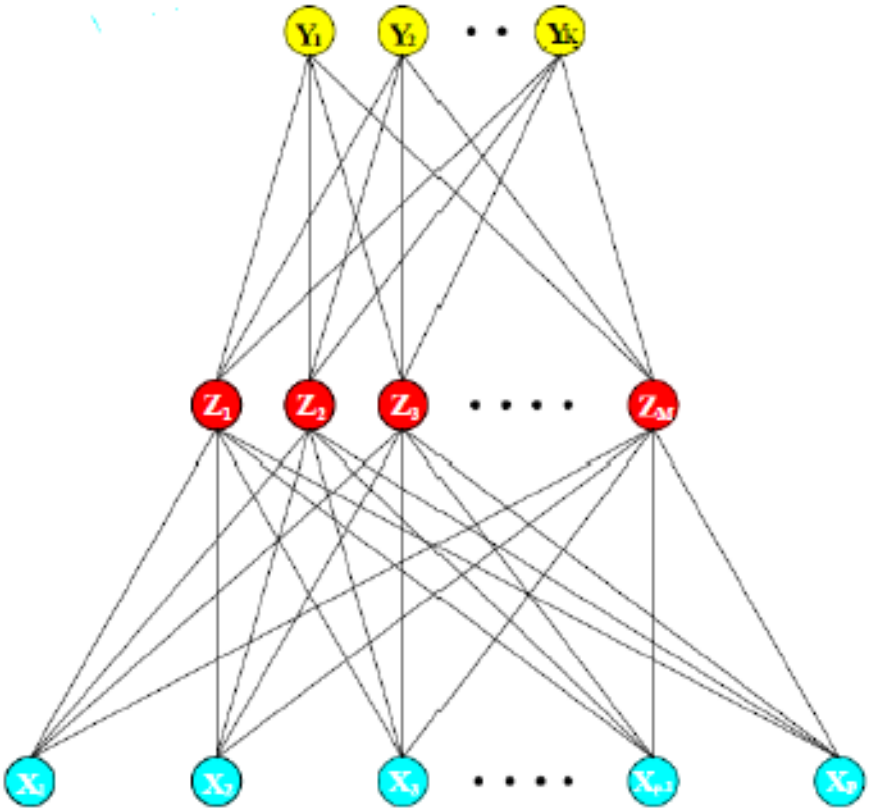


FIGURE 11.2. *Schematic of a single hidden layer, feed-forward neural network.*

For K-class classification there are K units at the top (which is the output layer in the ANN-diagram), with the k-th unit modeling the probability of class k in $\{1, 2, \ldots, K\}$. There are K target measurements Y_k, $k = 1, 2, \ldots, K$, each being coded as 0-1 variable for the k-th class.

The middle layer is called "hidden-layer". "Derived features" Z_m are created from linear combinations of the inputs X-vec = $(X_1, X_2, \ldots, X_p)$. Then the target Y_k is modeled/predicted as f_k(X-vec) = $g_k(T\text{-vec})$, which is a function of linear combinations of Z_m's.

$$Z_m = \sigma(\, \alpha_{0m} + \alpha_m^{\mathrm{T}}\, X\text{-vec}\,),\ m = 1, 2, \ldots, M,$$

$$T_k = \beta_{0k} + \beta_k^{\mathrm{T}}\, Z\text{-vec},\ \ k = 1, 2, \ldots, K,$$

$$f_k(X\text{-vec}) = g_k(T\text{-vec}),\ \ k = 1, 2, \ldots, K,$$

where

$$Z\text{-vec} = (Z_1, Z_2, \ldots, Z_M) \text{ and } T\text{-vec} = (T_1, T_2, \ldots, T_K).$$

The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = 1/[1 + \exp(-v)]$. See Figure 11.3 below for a plot of this function. Sometimes Gaussian radial basis functions are used for the activation functions. Then, the ANN is called radial basis function neural network.

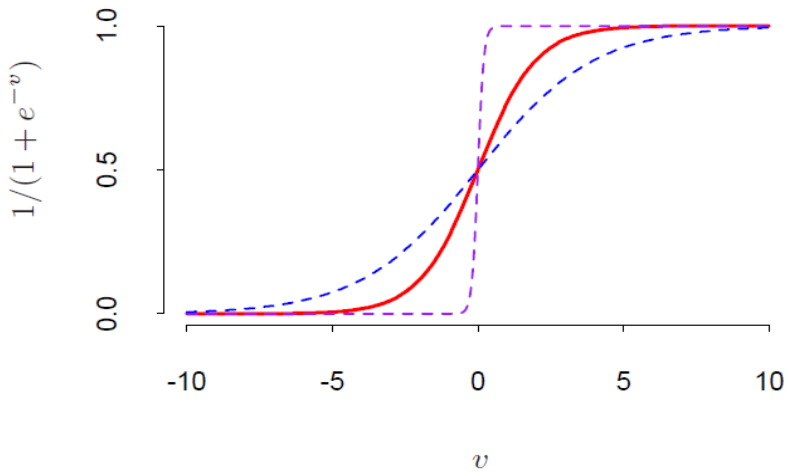


FIGURE 11.3. *Plot of the sigmoid function $\sigma(v) = 1/(1+\exp(-v))$ (red curve), commonly used in the hidden layer of a neural network. Included are $\sigma(sv)$ for $s = \frac{1}{2}$ (blue curve) and $s = 10$ (purple curve). The scale parameter s controls the activation rate, and we can see that large s amounts to a hard activation at $v = 0$. Note that $\sigma(s(v - v_0))$ shifts the activation threshold from 0 to v_0.*

There are many computing algorithms developed for calculating these ANN parameters α_{0m}, $\boldsymbol{\alpha_m}$ (a p-dim vector), β_{0k}, $\boldsymbol{\beta_K}$ (a M-dim vector), where $m = 1, 2, \ldots, M \equiv$ #hidden-nodes = #Neurons, $k = 1, 2, \ldots, K \equiv$ #classes.

There are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers, such as the following:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

These three rules provide a starting point for you to consider. Ultimately, the selection of an architecture for your neural network will come down to trial and error for getting a quality ANN for reasonable % mis-classifications or SS-prediction errors.

ISyE DDA – Agenda for 03/02/23 Lecture (Lec.15)

Advanced Data Modeling – Classifications

A) Statistical Classification Procedures:

- **(a) Bayes Classifier**
- **(b) Discriminant Analysis: LDA/QDA/RDA**
- **(c) Naïve Bayes Classifier**
- **(d) Multinomial/Logistic Regression Based Classification**

B) Computing/Optimization Based Classification Procedures

- **(e) Separating Hyperplane**
- **(f) Support Vector Machine**
- **(g) Decision Tree**
- **(h) Artificial Neural Network (ANN)**
- **(i) *K*-Neararest Neighbor (*k*NN)**

Detailed Lectures:

1. **Advanced Data Modeling - Classifications**

Details:

Classification is similar to regression. A classification model is constructed/estimated for predicting a set of data with **input variables** x-vec (e.g., x_1 = long sleeves, x_2 = \$600, x_3 = cotton in

Example-2 below) to a class-**outcome** (e.g., "office dress" (as Class-1), "party dress" (as Class-2)). This type of model could be useful in developing decision rules for sharing mom's clothes with others. This is like Airbnb.com for sharing rooms with others.

Our presentation for classification procedures will **focus on concepts** such that students know when to use which procedure. HW-2 asks students to learn how to use software packages to perform classifications.

(a) Bayes Classifier:

Bayes classifier is a statistical procedure, which classifies a case to the *most probable* class, using the (posterior-) conditional distribution Pr(G | X = x). Note that G has a discrete distribution with probability mass assigned to a few outcomes ≡ classes, $k = 1, 2, \ldots, K$.

Review – Prior, Likelihood, Posterior Distribution and Bayes Theorem:

1. Structure **prior (probability) distribution Pr(G = *k*) for classes** $k = 1, 2, \ldots, K$.

Example-1: A typical prior distribution for $K = 2$ classes is the **Bernoulli distribution** of $\Pr(G = 1) = \pi$ and $\Pr(G = 2) = 1 - \pi$ with probability $0 \le \pi \le 1$. Extending this concept, a popular prior distribution for general K =eg= 5 classes is the **Multinomial distribution** with "cell probabilities" $\Pr(G = 1) = \pi_1$, $\Pr(G = 2) = \pi_2$, …, $\Pr(G = 5) = \pi_5$, with $0 \le \pi_k \le 1$, $k = 1, 2, \ldots, 5$ and $\sum_{k=1}^{5} \pi_k = 1$. Note that in applications, there is a coding process needed to define these nature classes such as "red", "black", …, "blue" as class-1, class-2, …, class-5, respectively.

2. Given/conditioning on a particular class "$G = k$" (e.g., "red" in class-1), collect **data** for understanding regressor/classifier's probability distribution, $\mathbf{Pr(X = x \mid G = k)}$, which is usually called "**data likelihood**".

   Example-2: One example of the input variables could be style, price and material of mom's clothes. For instance, for a **"office dress" in class-1**, x_1 = long sleeves, x_2 = \$600, x_3 = cotton, …. For a **"party dress" in class-2**, x_2 = short sleeves, x_2 = \$1000, x_3 = expensive materials, ….

3. Use the Bayes Theorem to construct/derive the conditional probability distribution for Pr(G = k | X_vec = x_vec). Then, given a set of "inputs" X_vec = x_vec, predict/classify this case into a particular class, e.g., G = k. The conditional probability Pr(G = k | X = x) is called the **posterior distribution** in Bayes Classifiers.

Example-3: Classification is similar to regression. A classification model will be constructed to link input variables X_vec = x_vec to their outcomes, which are classes G = k, k = 1, 2, …, K =eg=5. In **Bayes Classifiers**, Bayes Theorem is used to construct a posterior distribution Pr(G = k given that X = x), using prior distribution and data-likelihood. Please see note #4 below for construction of posterior distribution.

Let us show an example of using posterior distribution to make classifications. If Pr(G = **Class-1 = "office dress"** given that x_1 = long sleeves, x_2 = \$600, x_3 = cotton, …) **is the highest among all posterior probabilities for all classes**, e.g., Pr(G = Class-2 = "party dress" given that x_1 = long sleeves, x_2 = \$600, x_3 = expensive materials, …) and Pr(G = Class-3, …, or Class-5 given the same set of inputs), this Bayes Classifier classifies a dress with inputs x_1 = long sleeves, x_2 = \$600, x_3 = cotton, … as a "office dress" in Class-1.

4. **Bayes Theorem:** $\Pr(G = k \mid X = x) = \Pr(G = k \text{ and } X = x) / \Pr(X = x) = [\Pr(X = x \mid G = k) * \Pr(G = k)] / \Pr(X = x)$, where the **numerator is a multiplication between the prior probability** $\Pr(G = k)$ **and the "data likelihood"** $\Pr(X = x \mid G = k)$ as explained above. They are the essential components to construct/derive the posterior distribution in Bayes Classifiers.

   The denominator is usually the "normalizing constant" for making the conditional pdf $\Pr(G = k \text{ and } X = x)$ a proper density, i.e., integration of this density over its domain is equal to one. There are computational methods to find this denominator, and thus, it is not necessary to "derive" its expression.

(b) Discriminant Analysis (LDA/QDA/RDA):

Discriminant analysis (DA) is one of the oldest classification procedures. LDA/QDA/RDA are special cases of **Bayes Classifiers** stated in #3 above.

Model Assumptions:

In these Discriminant Analysis procedures, Multivariate Normal Distributions are assumed for the **joint distribution** of p-dimensional X-variables with density $\boldsymbol{f_k(x_k\text{-}matrix) = Pr(X_k\text{-}matrix = x_k\text{-}matrix \mid G = k)}$ (i.e., the "**data likelihood**"). Note that for each class G = k, k = 1, 2, …, K, its inputs are considered as **random variables** and they are different from the inputs for the other classes.

Focus on Class-k data. Denoted by $\mathbf{X_k}$**-matrix a matrix of** $\boldsymbol{p}$**-dimensional input-variables**, i.e., $\mathbf{X_k}$**-matrix** = ($\mathbf{X_{1k}}$***-vec*****,** $\mathbf{X_{2k}}$***-vec*****, …,** $\mathbf{X_{pk}}$***-vec***), where each $\mathbf{X_{jk}}$***-vec*** has ***n*** **rows of data-samples**, and j = 1 (short/long sleeves), 2 (price), …, p.

For a class G = k, the **mean parameter-vectors** is $\boldsymbol{\mu_k} = (\mu_1, \mu_2, \ldots, \mu_p)^T$ for $\mathbf{X_k}$**-matrix** = ($\mathbf{X_{1k}}$***-vec*****,** $\mathbf{X_{2k}}$***-vec*****, …,** $\mathbf{X_{pk}}$***-vec***). Note that mean-vectors from different classes are usually **different,** i.e., $\boldsymbol{\mu_k \neq \mu_{k'}}$ for $k \neq k'$ in $\{1, 2, \ldots, K\}$.

The matrix $\boldsymbol{\Sigma_k}$ of their variance-covariances **in the general situations are different.** For example, the first element in $\boldsymbol{\Sigma_k}$ **is Var($\boldsymbol{X_{1k}}$-*vec*),** which could have **different values for different classes** of X_{1k}-*vec* data-vectors. The second element in $\boldsymbol{\Sigma_k}$ **is Cov($\boldsymbol{X_{1k}}$-*vec*, $\boldsymbol{X_{2k}}$-*vec*),** which could also have **different values for different classes** of X_{1k}-*vec* and X_{2k}-*vec* data-vectors.

In the Linear Discriminant Analysis (LDA), there is an additional assumption that **all variance-covariance matrices have an equal variance-covariance structure, i.e., $\boldsymbol{\Sigma_k = \Sigma}$ for all classes $\boldsymbol{k = 1, 2, \ldots, K}$.** For the Quadratic Discriminant Analysis (QDA), there is no equal variance-covariance assumption. Regularized Discriminant Analysis (RDA) assumes that $\boldsymbol{\Sigma_k(\alpha)} = \alpha\, \boldsymbol{\Sigma_k} + (1 - \alpha)\, \boldsymbol{\Sigma}$ with α given as a tuning parameter decided by a cross-validation procedure.

Details of Discriminant Analysis – LDA and QDA:

1. Assume that given a particular class G = k, the pdf of the input-variables X_k-matrix = x_k-*matrix*, $\boldsymbol{f_k(x_k\text{-}matrix) = Pr(X_k\text{-}matrix = x_k\text{-}matrix \mid G = k)}$, has a multivariate normal distribution with the following pdf.

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

With distinct G = k class, the mean-vector mu_k is different; similarly, variance-covariance matrix Sigma_k could be different.

2. For **LDA (Linear Discriminant Analysis)** further assume that the variance-covariance matrix $\boldsymbol{\Sigma_k = \Sigma}$ is the same for all classes k = 1, 2, …, K. Then,

LDA – Decision Boundary (for two classes)

In comparing two classes k and l, set their posterior prob. to be equal, i.e.,

$$0 = log\frac{\Pr(G=l \mid X=x)}{\Pr(G=k \mid X=x)} = log\frac{f_k(x)}{f_l(x)} + log\frac{\pi_k}{\pi_l}$$

That is, let us assume that the posterior probability for the G = L and G = k classes are the same, i.e., Pr(G = L | X = x) = Pr(G = k | X = x). Take a log-ratio of them, we get the equation in the left here.

$$0 = log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu_l) + \boldsymbol{x}^T\Sigma^{-1}(\mu_k - \mu_l).$$

This equation describes data-cases in the boundary of G = L and G = k classes (with equal posterior probability given above). With this boundary, data-cases in one side will be classified into the G = L class, and the data in the other side will be classified into the G = k class.

- Discriminant Function:
  - $\delta k(x) = log\frac{\pi_k}{} - \frac{1}{2}(\mu_k)^T\Sigma^{-1}(\mu_k) + \boldsymbol{x}^T\Sigma^{-1}(\mu_k).$

The deciison-boundary is a linear function of input X = x. All other mean (mu) and variance-covariance (Sigma) quantities are known.

- Decision boundary is the solution of x, which is a ***linear hyperplane*** in a p-dim x-space.

This is why this procedure is called "Linear Discriminant Analysis" (LDA). See a graphical presentation next.

3. Linear Discriminant Analysis

G = k in 1 (green), 2 (blue), 3 (orange) classes

- an idealized example with three classes and $p = 2$

  Two-dimensional X-inputs.
- the data arises from three Gaussian distributions with a common covariance matrix
- the contours corresponding to 95% highest probability density, as well as the class centroids

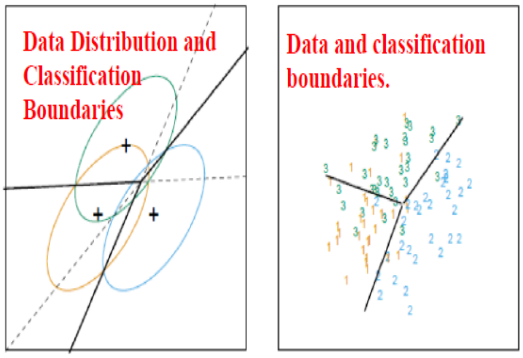


FIGURE 4.5. *The left panel shows three Gaussian distributions, with the same covariance and different means. Included are the contours of constant density enclosing 95% of the probability in each case. The Bayes decision boundaries between each pair of classes are shown (broken straight lines), and the Bayes decision boundaries separating all three classes are the thicker solid lines (a subset of the former). On the right we see a sample of 30 drawn from each Gaussian distribution, and the fitted LDA decision boundaries.*

Estimates of Unknown Parameters

Note that these estimates are from X-input-data at different classes. That is, for different classes k = 1, 2, ..., K, the mu_k are different.

In practice we do not know the parameters of the Gaussian distributions, and will need to estimate them using our training data:

- $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the number of class-k observations;
- $\hat{\mu}_k = \sum_{g_i=k} x_i / N_k$
- $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} \frac{(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{N - K}$

Because LDA assumes that Sigma_1 = Sigma_2 = ... = Sigma_K, we pool all variance-covariance information from all classes (k = 1, 2, ..., K) together to estimate the common variance Sigma.

3. For QDA (Quadratic Discriminant Analysis) there is no assumption like LDA has on Σ_k, i.e., Σ_k can be unequal.

Quadratic Discriminant Analysis

Get back to the general discriminant problem,

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

What if the Σ_k are not assumed to be equal ?

Quadratic discriminant functions (QDA)

By going through the similar derivation (skipped) like the one shown in the LDA (page 4), we have the QDA function here.

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

One can see the the decision-boundary in the QDA is a quadratic (nd-order) function.

Decision boundary between classes k and l,

$$\{x: \delta_k(x) = \delta_l(x)\}$$

Plots of Discriminant Analyses

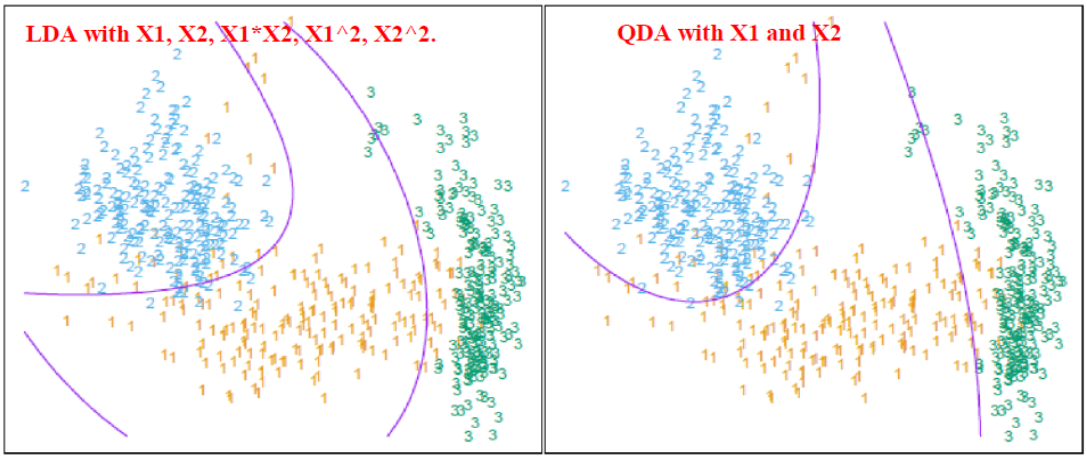


FIGURE 4.6. *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

Note that the left-plot is the classification from the LDA in a five-dimensional space (X_1, X_2, X_1*X_2, X_1^2, X_2^2). Then, projects LDA's linear decision boundaries into a 2-dimensional space (X_1, X_2) for a visualization plot. The right-plot is the classification from the QDA using X_1 and X_2 variables (2-dimensions) only. The results from these two procedures are similar, but different slightly.

(c) Naïve Bayes Classifier

This is a special case of (general) Bayes Classifier defined in #3 above. It can also be a special case of LDA/QDA/RDA.

Instead of using a (dependent) multivariate distribution to model the data-likelihood, all X-variables are assumed to be independent, and thus,

$$f_k(x_k\text{-}matrix) = \Pr(X_k\text{-matrix} = x_k\text{-}matrix \mid G = k)$$

is a product of marginal densities, i.e., pdfs for each X_{1k}-*vec*, X_{2k}-*vec*, …, X_{pk}-*vec*. In the example of LDA, QDA and RDA, **correlations between all X-variables are all zeros**. This implies that the variance-covariance matrix Σ_k only has diagonal elements (i.e., variances) and all off-diagonal elements (i.e., covariances) are all zeros.

(d) Multinomial/Logistic Regression Based Classifications

Step-1: Focus on an example with $K = 5$ classes. When there are more than two classes, instead of using a logistic regression to fit the class Y-data, multinomial regression should be used to fit the data. There will be multiple logit-transformation to link π_j = Pr(Y = class j =eg= office dress) to regression functions for classes, $j = 1, 2, \ldots, K$ and $\sum_{j=1}^{K} \pi_j = 1$. For example,

$$\log_e[\ \Pr(Y = \text{class} = j) / \Pr(Y = \text{class} = 1)\ ]$$

$$= \beta_{0j} + \beta_{1j}\, x_1 + \beta_{2j}\, x_2 + \ldots + \beta_{pj}\, x_1,$$

where $x_1, x_2, \ldots, x_p$, are p explanatory variables associated for all classes Y-data, and $\beta_{0j}, \beta_{1j}, \beta_{2j}, \ldots, \beta_{pj}$ are (p + 1) coefficients in the above logit-regression function for class-j against class-1 for $j = 2, 3, \ldots, K$ (=eg= 5). Note that we use class-1 as the "base" for the "odd-ratio" Pr(Y = class = j) / Pr(Y = class = 1) against all other classes. With $K = 5$, there will be FOUR logit-transformations with j = 2, 3, 4, 5 = K in

this multinomial regression. In each one of these logit-regressions, their regression coefficients β_{0j}, β_{1j}, β_{2j}, …, β_{pj} will be different for distinct class, $j = 2, 3, \ldots, K$.

Similar to the logistic regression, parameters are estimated via the maximum likelihood estimation (MLE) method. There, one needs to replace the Bernoulli distribution with the multinomial distribution and replace the distribution parameters $\pi_j = \Pr(Y = \text{class } j), j = 1, 2, \ldots, K, \sum_{j=1}^{K} \pi_j = 1$, with those multiple logit-regressions discussed above. Then, employ a numerical method such as Newton-Raphson method to optimize log-likelihood with respect to the regression coefficients. Predictions π_j_hat = Pr(Y = class j)_hat, $j = 1, 2, \ldots, K$, can then be obtained by replacing the unknown beta-coefficients with their MLEs.

For classifying a new data point $(Y_0, x_{10}, x_{20}, \ldots, x_{p0})$ to a class, one use the new set of explanatory variables $(x_{10}, x_{20}, \ldots,$

x_{p0}) to predict this case's class-probabilities π_j_hat = Pr(Y_0 = class j)_hat, for all K classes with j = 1, 2, …, K. Then, a simple decision rule for classification is

"assign this data case to class m if its class-probability

π_m_hat = Pr(Y_0 = class m)_hat is the highest

compared to all other class-probabilities",

where m is one of the members in j = 1, 2, …, K.

Remark:

[1] Multinomial/Logistic regression based classification procedures are statistical procedures (why?), but they are not Bayes Classifiers (why?).

[2] When misclassification costs are provided, one can develop a cost-based decision rule other than the one assign a data class with the highest class-probability.

General Remarks for Statistical Classification procedures:

[1] Statistical classifications are more rigorous but require more assumptions. Thus, it might not be as flexible as computing procedures to be presented next. However, statistical classifications address the "population" problem. Computing procedures tend to address "sample" problem, i.e., the classification results vary according to data sampled.

[2] Statistical classifications involves distributions (e.g., prior, data-likelihood, posterior in Bayes classifiers, and multinomial distribution in "Multinomial Regression Based Classifications"). Computing procedures do not involve any distribution, but they have objective functions and constraints.

The following procedures are computing or optimization-based procedures.

#4) Separating Hyperplane: This is a "computing" procedure that finds linear or nonlinear "hyperplanes" (i.e., function of x-variables in a p-dimension space) to "separate" data in different classes. The following present three examples.

4.1) **Rosenblatt's (1958)** ***perceptron learning algorithm*** tries to find a separating hyperplane by **minimizing the distance of misclassified points to the decision boundary**. For instance, if a response $y_i = 1$ is misclassified (as y_i_hat $= -1$), then $(x_i^T \beta\_hat + \beta_0\_hat) < 0$, and the opposite for a misclassified response with $y_i = -1$. Thus, the goal in finding this hyperplane is to find β_0_hat and β_hat to minimize

Two-class example with yi = +1 or -1. Its classification yi_hat will also be +1 or -1.

prediction

D is positive for a minus in front of the negative-summations.

$$D = -\sum_{i \text{ in } C} [\, y_i * (x_i^T \beta\_hat + \beta_0\_hat) \,],$$

data = yi; classification/prediction = yi_hat

If a data-case is mis-classified, then, yi * yi_hat will be equal to -1 due to their opposite signs.

Note that both data yi and its classification yi_hat will be either +1 or -1.

where C is the set includes *all misclassified points*. The algorithm implements a stochastic gradient descent to minimize the objective function D.

4.2) **Optimal Separating Hyperplanes (OSH):** The OSH separates the (two-) classes and maximizes the distance to the closest point from either class (Vapnik, 1996). Not only does this provide a **unique solution to**

the separating hyperplane problem, but by maximizing the margin between the two classes on the training data – this leads to better classification performance on test data. This is the idea in the **Support Vector Machine (SVM)**. A simple formulation of the above optimization problem is as follows.

The decision-boundary of this classification procedure is the middle of the "Margin".

In the original SVM development, it is assumed that there is no mis-classification case.

The idea of maximizing the MARGIN in SVM is rather new and innovative. It only needs to deal with data around the decision-boundary. These data are called "support-vectors".

Find β_0_hat and β_hat to maximize M

Subject to $[y_i * (x_i^T \beta\_hat + \beta_0\_hat)] \geq M$ = "margin"

(y_i: data; $x_i^T \beta\_hat + \beta_0\_hat$: prediction)

for $i = 1, 2, \ldots, n$, and

$\| \beta\_vec\_hat \| = 1$ (i.e., normalize the beta-coefficients).

Thus, if data yi is in the +1 case, and its prediction is positive, then, their multiply is positive. Similarly, if the data yi is "-1" and its prediction is negative, their multiply is also positive.

Note that $[y_i * (x_i^T \beta\_hat + \beta_0\_hat)]$ is the distance from a non-misclassified data-point to the decision boundary. Thus, the above optimization model find the "margin" M such that the all data cases are "outside" the (-M, M) region - this is what the above constraint for. The middle of the (-M, M) region is the decision-boundary for the two cases (one is in the negative side and the other is in the positive side). The data-cases that are closest to the margin "-M" or "+M" are called "support-vectors". Even that there might be outliers or millions of data points, only these few support-vectors decide the Margin and thus, the decision-boundary. Thus, SVM can be used to deal with a large number of data (with outliers).

#5) Support Vector Machine (SVM)

Let us use the following figures to explain the construction of decision boundary in the SVM. In the left figure, there is no mis-classification case. The two green dots and one red dot in the borders of the yellow box are called "support vectors". The size and location of the yellow box are decided by these support vectors. An optimization program will find these support vectors for a data set to maximize the size (i.e., margins) of the yellow box. The decision boundary for this SVM is the middle line in the yellow box.

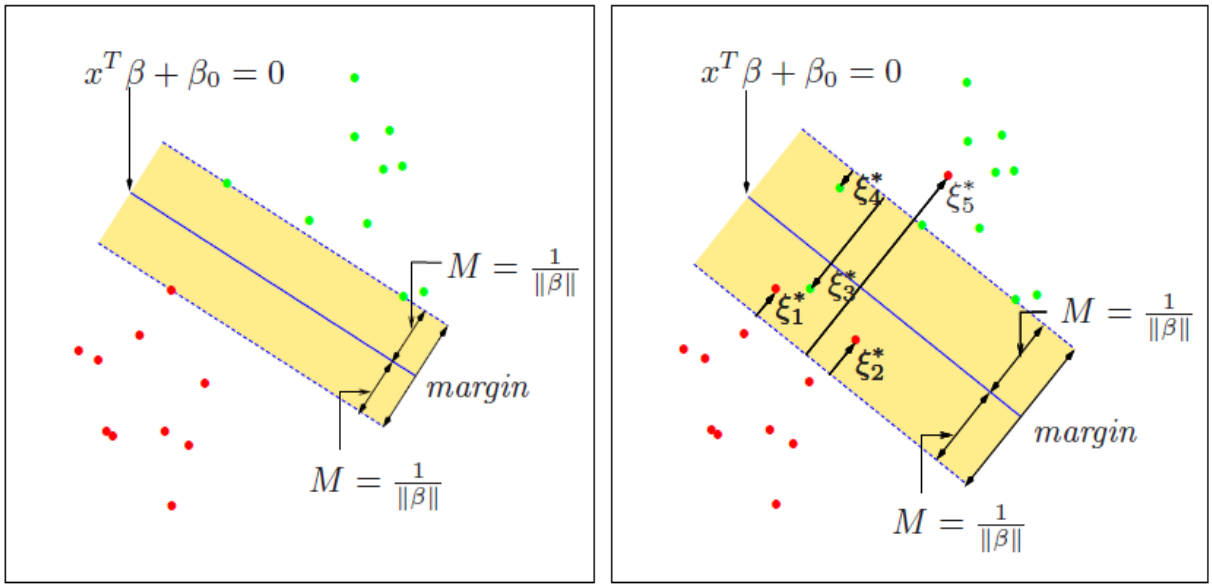


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

The right figure extends the simple SVM described above to handle potential "mis-classification" cases. Note that there are one red-dot classified into green-decision-region, and there is one green-dot (with ξ_3^* notation) classified in the left-side of the decision-boundary, i.e., red-decision-region. These are mis-classification cases. The idea of SVM in this "mis-classification" situation is to set a total budget to allow a small amount of mis-classifications. For example, in the right figure, there are FIVE cases that can be considered as "mis-classifications". The sum of their distances to the "correct-margins" is ξ_1^* (for the red-dot in the yellow-box and its distance to "red-margin") + ξ_2^* (similar to ξ_1^* in the red-zone) + ξ_3^* (for the green-dot in the yellow-box and its distance to "green-margin") + ξ_4^* (similar to ξ_3^*) + ξ_5^* (for the red-dot in the green-zone and its distance to "red-margin").

The budget of these "mis-classification" cases controls how many "mis-classified" points will be IGNORED in searching for the maximum margin. For example, when these FIVE cases addressed above are ignored, the support vectors are two green-dots on the margin of the yellow-box. The decision boundary for this SVM classification procedure is the middle line of the yellow-box. When the budget of "mis-classification" cases is changed, the

support vectors will change. Then, the decision boundary will change correspondingly.

#6) Mathematical Programming (MP) Based Separating Hyperplanes

This school of computing algorithms extends the above optimization-model to other optimization objective functions for allowing a limited amount of misclassifications, etc. Moreover, there could be different types of distances defined in various x-variable spaces.

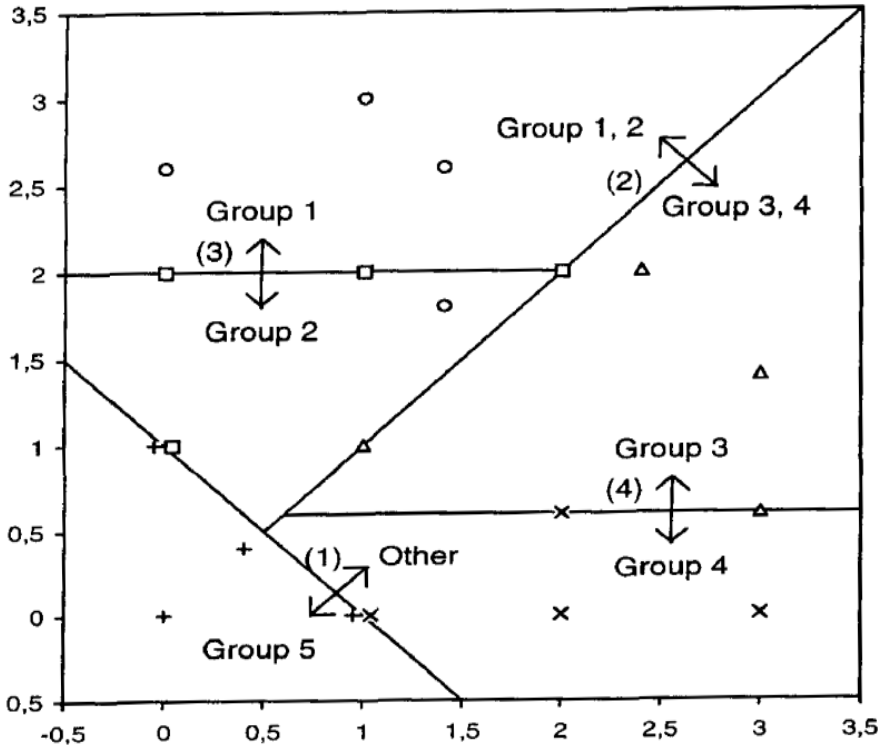
MP-based classification

Examples of objective functions:

- Minimize the sum of deviations
- Minimize the number of misclassifications
- Minimize the maximum deviation

Multi-group MP-based classification (cont')

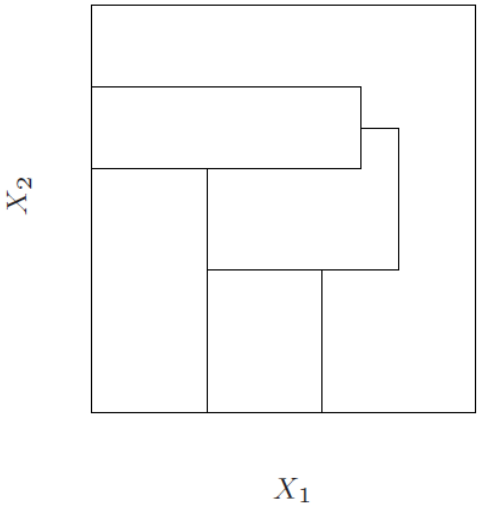
This example shows that MP-based classification procedure is very flexible in dealing with complicated situations with *non-standard shape of class-regions.*

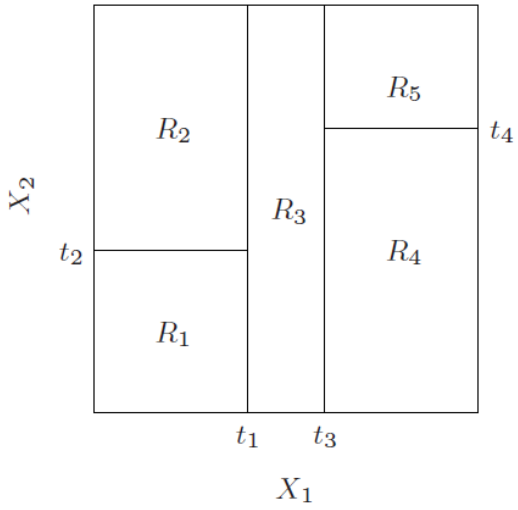


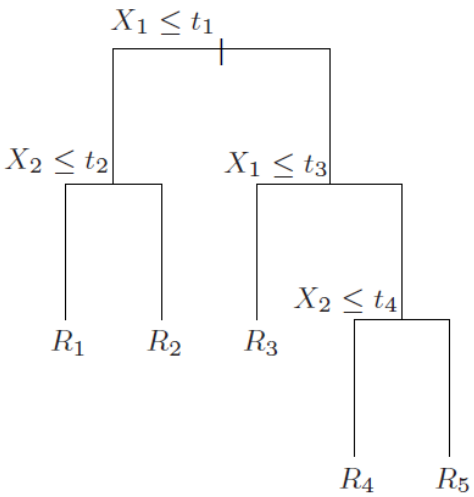
#7) Decision Tree: This school of procedures could be statistical or computational. The example presented below is a computing based decision tree.

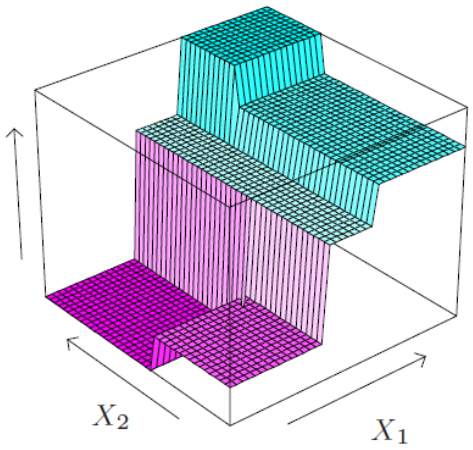
The let-bottom figure presents one example with five-nodes in a binary-split decision tree. Its decision boundaries in the two-x-variable (X_1 and X_2) domain is shown in the upper-right figure. The bottom right figure is the 3-D representation of these decision boundaries and their predictive surfaces for Y-class-outcomes.

The left-top figure shows that this type of non-binary-split decision boundaries cannot be handled by the traditional decision-tree procedures, which are usually binary-split trees.









Objectives **in the optimization procedures for building decision trees will be presented next.**

Notations:

[1] X-vec = $(X_1, X_2)^T$ in the above example; y-data = a collection of all y ≡ outcomes;

Note that outcomes in the decision-trees can be **classes (for classification) or numerical values (for regression)**; This is why decision-trees are called "**Classification and Regression Trees**";

[2] Decision-Regions R_m, m = 1, 2, …, 5, are decision-regions defined in the upper-right figure;

[3] (L_1, U_1), (L_2, U_2) are lower and upper bounds for X_1 and X_2, respectively;

[4] T is the collection of all terminal nodes defined by decision-regions R_m, m = 1, 2, …, 5; $|T|$ ≡ the size of the decision-tree = the number of terminal nodes =eg= 5;

[5] N_m = #{data in R_m}; Note that because R_m is defined for X-variables, #{data in R_m} is evaluated in terms of X-variable data in a specific region R_m;

[6] c_m_hat = average(y-data with their X-vec in R_m).

Classification/Prediction Function:

$$\boldsymbol{f\_hat}(\text{ X-vec}) = \sum_{m=1}^{5} c_m\_hat * I(\text{ X-vec in } R_m)$$

$$= \text{ average(y-data with their X-vec in } R_1) *$$

$$I(R_1 \equiv L_1 \le X_1 \le t_1, L_2 \le X_2 \le t_2)$$

$$+ \text{ average(y-data with their X-vec in } R_2) *$$

$$I(R_2 \equiv L_1 \le X_1 \le t_1, \ t_2 \le X_2 \le U_2)$$

$$+ \text{ average(y-data with their X-vec in } R_3) *$$

$$I(R_3 \equiv t_1 \le X_1 \le t_3, L_2 \le X_2 \le U_2)$$

$$+ \text{ average(y-data with their X-vec in } R_4) *$$

$$I(R_4 \equiv t_3 \le X_1 \le U_1, L_2 \le X_2 \le t_4)$$

$$+ \text{ average(y-data with their X-vec in } R_5) *$$

$$I(R_5 \equiv t_3 \le X_1 \le U_1, t_4 \le X_2 \le U_2).$$

***Objective Function* to Construct Decision-Trees:**

Find all decision-regions R_m (including which x-variable at what split-node of the tree and t_m-value of the split-decision), **m = 1, 2, …, M ≡ | T | (=eg= 5) to**

Minimize

$$C_\alpha(T) = \sum_{m=1}^{M} \sum_{xi\text{-}vec\ in\ Rm} (y_i - c_m\_hat)^2 + \alpha \mid T \mid,$$

where the first component represents the model-fitting quality and is the sum-of-squares (SS) of prediction-errors (SSE). Note that in the classification problems, the above SSE can be changed to some metrics related to #mis-classification cases. The second component represents the size of the tree, and α is a tuning parameter for balancing the fitting-quality against the complexity of the tree. Typically, α is decided by a cross-validation procedure, which will be presented later. The complexity of the tree is presented by the size of the tree $| T |$ = #terminal nodes.

There are many algorithms for constructing decision-trees to decide which x-variable should be used at what split-node and what should be its threshold-value (i.e., t_m-value) for the split-decision. HW-2 will ask students to review software available for decision-trees.

(h) Neural Network (or Artificial Neural Network (ANN)):

ANN is a two-stage regression or classification procedure. See Figure 11.2 below for a schematic diagram of a typical ANN.

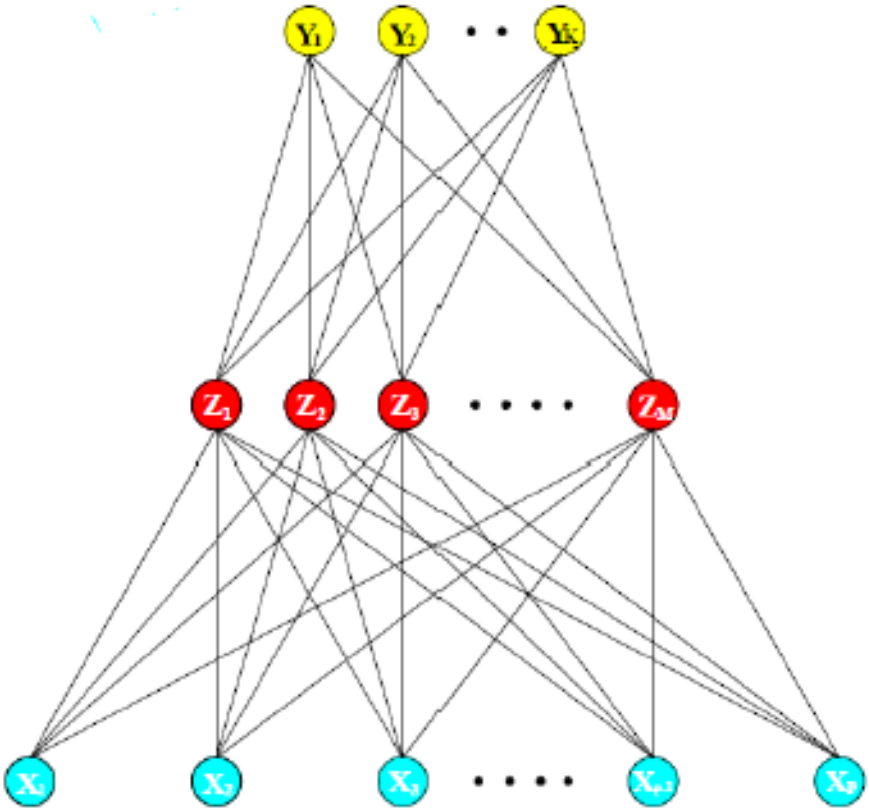


FIGURE 11.2. *Schematic of a single hidden layer, feed-forward neural network.*

For K-class classification there are K units at the top (which is the output layer in the ANN-diagram), with the k-th unit modeling the probability of class k in $\{1, 2, \ldots, K\}$. There are K target measurements Y_k, $k = 1, 2, \ldots, K$, each being coded as 0-1 variable for the k-th class.

The middle layer is called "hidden-layer". "Derived features" Z_m are created from linear combinations of the inputs X-vec = $(X_1, X_2, \ldots, X_p)$. Then the target Y_k is modeled/predicted as f_k(X-vec) = g_k(T-vec), which is a function of linear combinations of Z_m's.

$$Z_m = \sigma(\, \alpha_{0m} + \alpha_m^{\mathrm{T}}\, X\text{-vec}\,),\ m = 1, 2, \ldots, M,$$

$$T_k = \beta_{0k} + \beta_k^{\mathrm{T}}\, Z\text{-vec},\ \ k = 1, 2, \ldots, K,$$

$$f_k(X\text{-vec}) = g_k(T\text{-vec}),\ \ k = 1, 2, \ldots, K,$$

where

$$Z\text{-vec} = (Z_1, Z_2, \ldots, Z_M) \text{ and } T\text{-vec} = (T_1, T_2, \ldots, T_K).$$

The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = 1/[1 + \exp(-v)]$. See Figure 11.3 below for a plot of this function. Sometimes Gaussian radial basis functions are used for the activation functions. Then, the ANN is called radial basis function neural network.

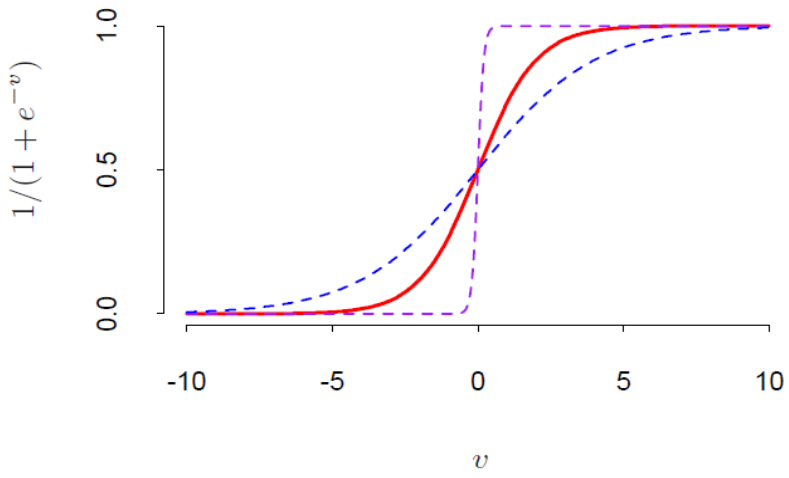


FIGURE 11.3. *Plot of the sigmoid function* $\sigma(v) = 1/(1+\exp(-v))$ *(red curve), commonly used in the hidden layer of a neural network. Included are* $\sigma(sv)$ *for* $s = \frac{1}{2}$ *(blue curve) and* $s = 10$ *(purple curve). The scale parameter* s *controls the activation rate, and we can see that large* s *amounts to a hard activation at* $v = 0$*. Note that* $\sigma(s(v - v_0))$ *shifts the activation threshold from* 0 *to* v_0*.*

There are many computing algorithms developed for calculating these ANN parameters α_{0m}, $\boldsymbol{\alpha_m}$ (a p-dim vector), β_{0k}, $\boldsymbol{\beta_K}$ (a M-dim vector), where $m = 1, 2, \ldots, M \equiv$ #hidden-nodes = #Neurons, $k = 1, 2, \ldots, K \equiv$ #classes.

There are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers, such as the following:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

These three rules provide a starting point for you to consider. Ultimately, the selection of an architecture for your neural network will come down to trial and error for getting a quality ANN for reasonable % mis-classifications or SS-prediction errors.

ISyE DDA – Agenda for 02/28/23 Lecture (Lec.14)

1. **Economic Decision Models & Game Theoretic Model**
2. **Advanced Data Modeling** (switch to data analytics here)
   - i) **Classifications**
     - **(a) Bayes Classifier**
     - **(b) Discriminant Analysis: LDA/QDA/RDA**
     - **(c) Naïve Bayes Classifier**

Detailed Lectures:

1. **Economic Decision Models & Game Theoretic Model**

Reference: Lu *et al.* (2011), "Competition Under Manufactuer Service and Retail Price", Economic Modeling 28, 1256-1264.

Multiple decision-makers are involved. **Each one has his/her own decision-variable(s), objective-function(s) and constraint(s).** Consider the situation with two decision-makers, e.g., one manufacturer and one retailer, where the manufacturer supplies retailer products to sell. The following are four typical decision models.

1. "Integrated" decisions:

The decision-variables, objective-functions and constraints from both decision-makers are integrated (e.g., added) together to formulate a single set of decision problems. Physically, this implies that there is ONE "super-manager" making decisions with an integrated decision function. In the supply-chain example, the super-manager owns both manufacturing and retail operations. In this model, individual's objectives are "integrated" together into one objective function.

2. Manufacturer Stackelberg decisions:

In this model, manufacturer (the leader) has more bargaining power in making decisions based on his/her objective function, conditioning on retailer's (the follower) response function. Usually, the optimal decisions for both manufacturer and retailers are solved backwards. See the following example for details.

The System of Two Manufacturers and One Retailer:

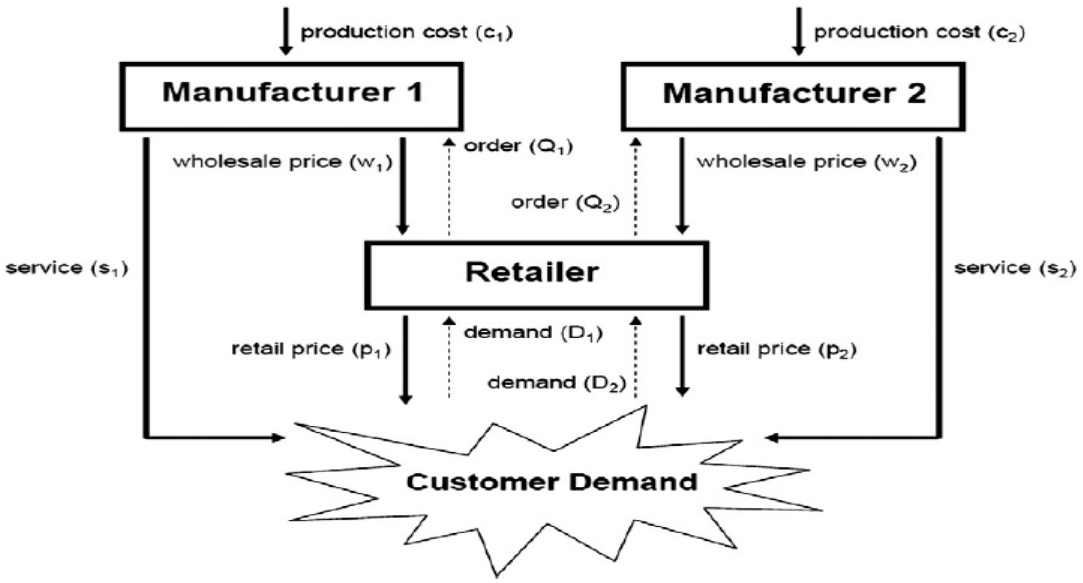


Fig. 1. Main model.

Retailer's Reaction Function: Decide his/her optimal prices p_1^* and p_2^* to maximize his/her equilibrium profit. That is,

$p_i^* = \arg \text{Max (w.r.t. } p_i) \prod_R (p_1, p_2^* \mid w_1, w_2, s_1, s_2), i, j = 1, 2,$

where w_1, w_2, s_1, s_s are decision-variables made by two manufacturers. See Figure 1 for the definition of notations. Note that Retailer's profit function $\prod_R$ has two decision-variables p_1 and p_2, which have to be optimized simultaneously.

Example-R1: Demand function for Product i supplied by Manufacturer i (= 1, 2) is

$$Q_i(p_i, p_j, s_i, s_j) = a_i - b_p p_i + \theta_p(p_j - p_i) + b_s s_i - \theta_s(s_j - s_i). \qquad (2)$$

which is a linear function of its product's price p_i and service level s_i, and also difference of two prices $(p_j - p_i)$ and two service levels $(s_j - s_i)$. Retailer's profit function is a function of demand times the difference of retail price (p_i) to the wholesale price (w_i). The contributions from two manufacturers are added together equally to form retailer's profit function.

$$\Pi_R = \sum_{i=1}^{2} (p_i - w_i) Q_i\left(p_i, p_j, s_i, s_j\right), \tag{3}$$

Given the above first and second order optimality conditions, the following expression provides the retailer's reaction function

$$p_i^* = \frac{w_i}{2} + \frac{\left(b_p + \theta_p\right) a_i + \theta_p a_j}{2b_p\left(b_p + 2\theta_p\right)} - \frac{\theta_s\left(s_j - s_i\right)}{2\left(b_p + 2\theta_p\right)} + \frac{\left(b_p + \theta_p\right) b_s s_i + \theta_p b_s s_j}{2b_p\left(b_p + 2\theta_p\right)}, \tag{7}$$

where $i \in \{1,2\}$ and $j = 3 - i$. From Eqs. (7) and (1), the demand quantities for products i is

$$Q_i^* = \frac{a_i}{2} - \frac{\left(b_p + \theta_p\right)}{2} w_i + \frac{\theta_p}{2} w_j + \frac{\left(b_s + \theta_s\right)}{2} s_i - \frac{\theta_s}{2} s_j, \tag{8}$$

where $i \in \{1,2\}$ and $j = 3 - i$. We can see that the equilibrium quantities p_i^* and Q_i^* for each product are linear functions of the wholesale prices and service levels by the manufacturers and the market bases (a_1 and a_2). Note that the wholesale price and service levels are a function of production costs and other model parameters

Manufacturers' Decisions:

Using retailer's reaction function, we can derive each manufacturer's optimal wholesale price (w_i^*) and service level (s_i^*). This is carried out by maximizing each manufacturer's profit

$\prod_M(w_i^*, w_j, s_i^*, s_j \mid p_1^*, p_2^*), i, j = 1, 2,$

given optimal prices p_1^*, p_2^* and optimal demands Q_1^* and Q_2^*. Note that manufacturer's profit function depends on Q_i^* ($i = 1$ or 2), which is a function of (w_i, w_j, s_i, s_j). Moreover, the optimal prices, p_1^*, p_2^*, are also functions of (w_i, w_j, s_i, s_j).

Recall that the two manufacturers have equal bargaining power, and thus, they make decisions for their decision-variables according to Nash Equilibrium (see #3 below for details).

Example-R2: Similar to Retailer's profit function, manufacturer's profit function is given as

$$\Pi_{M_i} = (w_i - c_i)Q_i\left(p_i, p_j, s_i, s_j\right) - \eta_i s_i^2 / 2, \; i = 1,2, \qquad (4)$$

where $w_i - c_i$ is the difference between the manufacturer's wholesale price and production cost and η_i is the service cost coefficient of manufacturer i.

Remarks: Service level s_i is involved in manufacturer's profit function in a "negative" quadratic form. That is, if a

manufacturer provides more after-sale service to customers, his/her profit will be smaller.

To find the optimal wholesale price, w_i, we first look at the first order condition.

$$0 = \frac{\partial \Pi_{M_i}}{\partial w_i} = a_i - b_p \left[w_i + \frac{\left(b_p + \theta_p\right)a_i + \theta_p a_j}{2b_p\left(b_p + 2\theta_p\right)} - \frac{\theta_s\left(s_j - s_i\right)}{2\left(b_p + 2\theta_p\right)} + \frac{\left(b_p + \theta_p\right)b_s s_i + \theta_p b_s s_j}{2b_p\left(b_p + 2\theta_p\right)} \right]$$

$$+ \theta_p \left[\frac{a_j - a_i}{2\left(b_p + 2\theta_p\right)} + \frac{w_j - 2w_i}{2} + \frac{(2\theta_s + b_s)\left(s_j - s_i\right)}{2\left(b_p + 2\theta_p\right)} \right] + b_s s_i - \theta_s\left(s_j - s_i\right)$$

$$+ \frac{c_i b_p}{2} + \frac{c_i \theta_p}{2}, \quad 0 = \frac{\partial \Pi_{M_i}}{\partial s_i} = (w_i - c_i) \left[-\frac{b_p \theta_s}{2\left(b_p + 2\theta_p\right)} - \frac{b_p\left(b_p + \theta_p\right)b_s}{2b_p\left(b_p + 2\theta_p\right)} \right.$$

$$\left. - \frac{\theta_p(b_s + 2\theta_s)}{2\left(b_p + 2\theta_p\right)} + b_s + \theta_s \right] - \eta_i s_i.$$

The second order condition, $\partial \Pi_{M_i}^2 / \partial w_i^2 = -b_p - \theta_p$, $\partial \Pi_{M_i}^2 / \partial w_i \partial s_i = b_s + \theta_s/2$, $\partial \Pi_{S_i}^2 / \partial w_i^2 = -\eta_i$, is then calculated to check the optimality. For $b_p > 0$ and $\theta_p > 0$ we have a negative definite Hessian. Therefore, the w_i and s_i calculated above are the optimal reaction functions for the manufacturer i.

After a series of tedious algebraic manipulations (not presented here), the following proposition gives the closed form solution of wholesale price and service level.

Proposition 4.1. *The manufacturer's equilibrium wholesale price and service level are:*

$$w_i^* = \frac{2\eta_i A_j}{A_1A_2 - B_1B_2}\left[a_i + D_j a_j + (E_i + F_i D_i)c_i + \left(F_j + E_j D_j\right)c_j\right], \quad (11)$$

$$s_i^* = (b_s + \theta_s)\left\{\frac{A_j}{A_1A_2 - B_1B_2}\left[a_i + D_j a_j + \left(F_j + E_j D_j\right)c_j\right] + \left[\frac{A_j(E_i + F_i D_i)}{A_1A_2 - B_1B_2} - \frac{1}{2\eta_i}\right]c_i\right\}, \quad (12)$$

where $i \in \{1,2\}$ *and* $j = 3 - i$, *and* $A_i = 4\eta_i(b_p + \theta_p) + (b_s + \theta_s)^2$, $B_i = 2\eta_i\theta_p - \theta_s(b_s + \theta_s)\left[(b_p - b_s + 2\theta_p)/(b_p + 2\theta_p)\right]$, $D_i = B_i/A_i$, $E_i = (b_p + \theta_p) - (b_s + \theta_s)^2/2\eta_i$, $F_i = [\theta_s(b_s + \theta_s)/2\eta_i] - [\theta_p b_s(b_s + \theta_s)/2\eta_i(b_p + 2\theta_p)]$.

3. "Nash equilibrium" decisions:

In this model, **every decision-maker has equal bargaining power and thus makes his/her decisions simultaneously**. In the supply chain example, this scenario arises in a market in which there are relatively small to medium-sized manufacturers and retailers. In this market it is reasonable to assume that a manufacturer may not know the competitor's wholesale price but can observe its retail price. Since a manufacturer cannot

dominate the market over the retailer, his price decision is conditioned on how the retailer prices the product. On the other hand, the retailer must also condition its retail price decisions on the wholesale price.

Game-theoretic framework is employed to derive the reaction function (see below for examples) of each decision-maker in the supply chain. Solving these reaction functions simultaneously yields the Nash equilibrium solution. Note that the solution process here is different from the one considered in #1, "integrated" decisions". Here, there are two sets of objective functions involved in these reaction functions; but, in the integrated decisions, there is only one set of objective function.

Example N-1: By solving reactive functions from both retailer and manufacturers simultaneously, we have

Proposition 4.4. *In the Vertical Nash case, the equilibrium retail price* p_1^* *and* p_2^* *chosen by the retailer are*

$$p_i = \frac{\left(\gamma_j\kappa_i - \lambda_i\kappa_j\right)a_1 + \left(\gamma_j\nu_i - \lambda_i\nu_j\right)a_2 + \gamma_j\psi_i c_1 + \lambda_i\psi_j c_2}{\gamma_i\gamma_j - \lambda_i\lambda_j}, \qquad (27)$$

where κ_i, λ_i, ν_i *and* ψ_i *for* $i \in \{1,2\}$ *and* $j = 3 - i$ *are constants defined in* Appendix B.

4. **Retailer Stackelberg decisions:**

In this model, retailer has more bargaining power in making decisions. The Retailer Stackelberg scenario arises in markets where retailers' sizes are large compared to their suppliers. For example, large retailers like Walmart and Target can influence each product's sales by lowering the price. Because of their sizes, the retailers can maintain their margin on sales while squeezing profit from their suppliers. The suppliers are mostly concerned with receiving orders from the retail giants.

Similar game-theoretic framework as applied in the Manufacturer Stackelberg case is implemented to solve this problem; i.e., the problem is solved backwards. First, the manufacturer/suppliers' problem is solved to derive the response function conditional on the retail prices chosen by the retailer. The retailer problem is then solved given that the retailer knows how the manufacturers would react to the retail prices he sets. Details are skipped here.

(New Data Analytics Materials for Exam-2)

2. Advanced Data Modeling - Classifications

Detailed Lectures – Classifications:

(a) Bayes Classifier

(b) Discriminant Analysis: LDA/QDA/RDA

(c) Naïve Bayes Classifier

(d) Logistic-Regression Based Classifier

(e) Separating Hyperplane

(f) Support Vector Machine (SVM)

(g) Mathematical Programming Based Classification

(h) Decision Tree

Details:

Classification is similar to regression. A classification model is constructed/estimated for predicting a set of data with **input variables** x-vec (e.g., x_1 = short sleeves, x_2 = \$800, x_3 = cotton in Example-2 below) to a class-**outcome** (e.g., "red dress" (as Class-1), "black jacket (as Class-2)). This type of model could be useful in developing decision rules for sharing mom's clothes with others. This is similar to Airbnb.com for sharing rooms with others.

Our presentation for classification procedures will **focus on concepts** such that students know when to use which procedure.

HW-2 asks students to learn how to use software packages to perform classifications.

[1] Introduction – 10 Classification Procedures

Statistical Procedure 1. Logistic Regression (important)

Statistical Procedure 2. Linear Discriminant Analysis (LDA) (and Quadratic Discriminant Analysis (QDA), Regulated Discriminant Analysis (RDA)) (important)

Statistical Procedure 3. Bayes Classifier and Naïve Bayes Classifier (Ideas)

Computation/optimization procedure 4. Separating Hyperplane (secondary)

Computation/optimization procedure 5. Support Vector Machine (SVM) (important)

Computation/optimization procedure 6. Mathematical Programming Based Classification (secondary)

Computation/optimization procedure 7. Decision Tree (ideas)

Computation/optimization procedure 8. Neural Network (ideas)

Statistical Procedure 9. Kernel-Based Classification (ideas)

Computation/optimization procedure 10. Nearest Neighbor (local) Classification (ideas)

Notes: Because logistic-regression based classification is closer to separating hyperplane, support vector machine, we will present it after Bayes classifier and LDA/QDA/RDA.

(a) Bayes Classifier:

Bayes classifier is a statistical procedure, which classifies a case to the *most probable* class, using the (posterior-) conditional distribution Pr(G | X = x). Note that G has a discrete distribution with probability mass assigned to a few outcomes ≡ classes, $k = 1, 2, \ldots, K$.

Review – Prior, Likelihood, Posterior Distribution and Bayes Theorem:

1. Structure **prior (probability) distribution Pr(G = *k*) for classes** $k = 1, 2, \ldots, K$.

   Example-1: A typical prior distribution for $K = 2$ classes is the **Bernoulli distribution** of $\Pr(G = 1) = \pi$ and $\Pr(G = 2) = 1 - \pi$ with probability $0 \le \pi \le 1$. Extending this concept, a popular prior distribution for general K =eg= 5 classes is the **Multinomial distribution** with "cell probabilities" $\Pr(G = 1) = \pi_1$, $\Pr(G = 2) = \pi_2$, …, $\Pr(G = 5) = \pi_5$, with $0 \le \pi_k \le 1$, $k = 1, 2, \ldots, 5$ and $\sum_{k=1}^{5} \pi_k = 1$. Note that in applications, there is a coding process needed to define these nature classes such as "red", "black", …, "blue" as class-1, class-2, …, class-5, respectively.

2. Given/conditioning on a particular class "G = *k*" (e.g., "red" in class-1), collect **data** for understanding regressor/classifier's

probability distribution, **Pr(X = *x* | G = *k*)**, which is usually called "**data likelihood**".

Example-2: One example of the input variables could be style, price and material of mom's clothes. For instance, for a **"red dress" in class-1**, x_1 = short sleeves, x_2 = \$800, x_3 = cotton, …. For a **"black jacket" in class-2**, x_2 = long sleeves, x_2 = \$1000, x_3 = leather, ….

3. Use the Bayes Theorem to construct/derive the conditional probability distribution for Pr(G = *k* | X_vec = *x_vec*). Then, given a set of "inputs" X_vec = *x_vec*, predict/classify this case into a particular class, e.g., G = *k*. The conditional probability Pr(G = *k* | X = *x*) is called the **posterior distribution** in Bayes Classifiers.

Example-3: Classification is similar to regression. A classification model will be constructed to link input variables X_vec = *x_vec* to their outcomes, which are classes G = *k*, *k* = 1, 2, …, *K* =eg=5. In **Bayes Classifiers**, Bayes Theorem is used to construct a posterior distribution Pr(G = *k* given that X = *x*), using prior distribution and data-likelihood. Please see note #4 below for construction of posterior distribution.

Let us show an example of using posterior distribution to make classifications. If Pr(G = **Class-1 = "red-dress"** given that x_1 = short sleeves, x_2 = \$800, x_3 = cotton, …) **is the highest among all posterior probabilities for all classes**, e.g., Pr(G = Class-2 = "Black jacket" given that x_1 = short sleeves, x_2 = \$800, x_3 = cotton, …) and Pr(G = Class-3, …, or Class-5 given the same set of inputs), this Bayes Classifier classifies a dress with inputs x_1 = short sleeves, x_2 = \$800, x_3 = cotton, … as a "red dress" in Class-1.

4. **Bayes Theorem:** Pr(G = k | X = x) = Pr(G = k and X = x) / Pr(X = x) = [Pr(X = x | G = k) * Pr(G = k)] / Pr(X = x), where the **numerator is a multiplication between the prior probability** Pr(G = k) **and the "data likelihood"** Pr(X = x | G = k) as explained above. They are the essential components to construct/derive the posterior distribution in Bayes Classifiers.

   The denominator is usually the "normalizing constant" for making the conditional pdf Pr(G = k and X = x) a proper density, i.e., integration of this density over its domain is equal to one. There are computational methods to find this denominator, and thus, it is not necessary to "derive" its expression.

(b) Discriminant Analysis (LDA/QDA/RDA):

Discriminant analysis (DA) is one of the oldest classification procedures. LDA/QDA/RDA are special cases of **Bayes Classifiers** stated in #3 above.

Model Assumptions:

In these Discriminant Analysis procedures, Multivariate Normal Distributions are assumed for the **joint distribution** of *p*-dimensional X-variables with density $\boldsymbol{f_k(x_k\text{-}matrix) = Pr(X_k\text{-}matrix = x_k\text{-}matrix \mid G = k)}$ (i.e., the "**data likelihood**"). Note that for each class G = *k*, *k* = 1, 2, …, *K*, its inputs are considered as **random variables** and they are different from the inputs for the other classes.

Focus on Class-*k* data. Denoted by **X_k-matrix a matrix of *p*-dimensional input-variables**, i.e., $\boldsymbol{X_k\text{-}matrix = (X_{1k}\text{-}vec, X_{2k}\text{-}vec, \ldots, X_{pk}\text{-}vec)}$, where each $\boldsymbol{X_{jk}\text{-}vec}$ has ***n* rows of data-samples**, and *j* = 1 (short/long sleeves), 2 (price), …, *p*.

For a class G = k, the **mean parameter-vectors** is $\boldsymbol{\mu_k} = (\mu_1, \mu_2, \ldots, \mu_p)^T$ for $\mathbf{X_k}$**-matrix** = ($\mathbf{X_{1k}}$***-vec***, $\mathbf{X_{2k}}$***-vec***, **…,** $\mathbf{X_{pk}}$***-vec***). Note that mean-vectors from different classes are usually **different,** i.e., $\boldsymbol{\mu_k} \neq \boldsymbol{\mu_{k'}}$ for $k \neq k'$ in $\{1, 2, \ldots, K\}$.

The matrix $\boldsymbol{\Sigma_k}$ of their variance-covariances **in the general situations are different.** For example, the first element in $\boldsymbol{\Sigma_k}$ **is Var(**$\mathbf{X_{1k}}$***-vec*****),** which could have **different values for different classes** of X_{1k}-*vec* data-vectors. The second element in $\boldsymbol{\Sigma_k}$ **is Cov(**$\mathbf{X_{1k}}$***-vec***, $\mathbf{X_{2k}}$***-vec*****),** which could also have **different values for different classes** of X_{1k}-*vec* and X_{2k}-*vec* data-vectors.

In the Linear Discriminant Analysis (LDA), there is an additional assumption that **all variance-covariance matrices have an equal variance-covariance structure, i.e.,** $\boldsymbol{\Sigma_k} = \boldsymbol{\Sigma}$ **for all classes** $\boldsymbol{k = 1, 2, \ldots, K}$**.** For the Quadratic Discriminant Analysis (QDA), there is no equal variance-covariance assumption. Regularized Discriminant Analysis (RDA) assumes

that $\boldsymbol{\Sigma_k(\alpha)} = \alpha\, \boldsymbol{\Sigma_k} + (1 - \alpha)\, \boldsymbol{\Sigma}$ with α given as a tuning parameter decided by a cross-validation procedure.

Details of Discriminant Analysis – LDA and QDA:

1. Assume that given a particular class G = *k*, the pdf of the input-variables X_k-matrix = x_k-*matrix*, $\boldsymbol{f_k(x_k\text{-}matrix) = Pr(X_k\text{-}matrix = x_k\text{-}matrix \mid G = k)}$, has a multivariate normal distribution with the following pdf.

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

With distinct G = k class, the mean-vector mu_k is different; similarly, variance-covariance matrix Sigma_k could be different.

2. For **LDA (Linear Discriminant Analysis)** further assume that the variance-covariance matrix $\boldsymbol{\Sigma_k = \Sigma}$ is the same for all classes *k* = 1, 2, …, *K*. Then,

LDA – Decision Boundary (for two classes)

In comparing two classes k and l, set their posterior prob. to be equal, i.e.,

$$0 = log\frac{\Pr(G=l \mid X=x)}{\Pr(G=k \mid X=x)} = log\frac{f_k(x)}{f_l(x)} + log\frac{\pi_k}{\pi_l}$$

That is, let us assume that the posterior probability for the G = L and G = k classes are the same, i.e., Pr(G = L | X = x) = Pr(G = k | X = x). Take a log-ratio of them, we get the equation in the left here.

$$0 = log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu_l) + \boldsymbol{x}^T\Sigma^{-1}(\mu_k - \mu_l).$$

This equation describes data-cases in the boundary of G = L and G = k classes (with equal posterior probability given above). With this boundary, data-cases in one side will be classified into the G = L class, and the data in the other side will be classified into the G = k class.

- Discriminant Function:
  - $\delta k(x) = log\frac{\pi_k}{} - \frac{1}{2}(\mu_k)^T\Sigma^{-1}(\mu_k) + \boldsymbol{x}^T\Sigma^{-1}(\mu_k).$

The deciison-boundary is a linear function of input X = x. All other mean (mu) and variance-covariance (Sigma) quantities are known.

- Decision boundary is the solution of x, which is a ***linear hyperplane*** in a p-dim x-space.

This is why this procedure is called "Linear Discriminant Analysis" (LDA). See a graphical presentation next.

3. Linear Discriminant Analysis

G = k in 1 (green), 2 (blue), 3 (orange) classes

- an idealized example with three classes and $p = 2$

  Two-dimensional X-inputs.
- the data arises from three Gaussian distributions with a common covariance matrix
- the contours corresponding to 95% highest probability density, as well as the class centroids

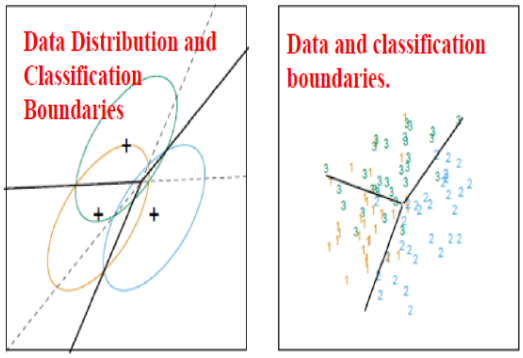


FIGURE 4.5. *The left panel shows three Gaussian distributions, with the same covariance and different means. Included are the contours of constant density enclosing 95% of the probability in each case. The Bayes decision boundaries between each pair of classes are shown (broken straight lines), and the Bayes decision boundaries separating all three classes are the thicker solid lines (a subset of the former). On the right we see a sample of 30 drawn from each Gaussian distribution, and the fitted LDA decision boundaries.*

Estimates of Unknown Parameters

Note that these estimates are from X-input-data at different classes. That is, for different classes k = 1, 2, ..., K, the mu_k are different.

In practice we do not know the parameters of the Gaussian distributions, and will need to estimate them using our training data:

- $\hat{\pi}_k = \frac{N_k}{N}$, where N_k is the number of class-k observations;
- $\hat{\mu}_k = \sum_{g_i=k} x_i / N_k$
- $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} \frac{(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{N - K}$

Because LDA assumes that Sigma_1 = Sigma_2 = ... = Sigma_K, we pool all variance-covariance information from all classes (k = 1, 2, ..., K) together to estimate the common variance Sigma.

3. For QDA (Quadratic Discriminant Analysis) there is no assumption like LDA has on Σ_k, i.e., Σ_k can be unequal.

Quadratic Discriminant Analysis

Get back to the general discriminant problem,

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

What if the Σ_k are not assumed to be equal ?

Quadratic discriminant functions (QDA)

By going through the similar derivation (skipped) like the one shown in the LDA (page 4), we have the QDA function here.

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

One can see the the decision-boundary in the QDA is a quadratic (nd-order) function.

Decision boundary between classes k and l,

$$\{x: \delta_k(x) = \delta_l(x)\}$$

Plots of Discriminant Analyses

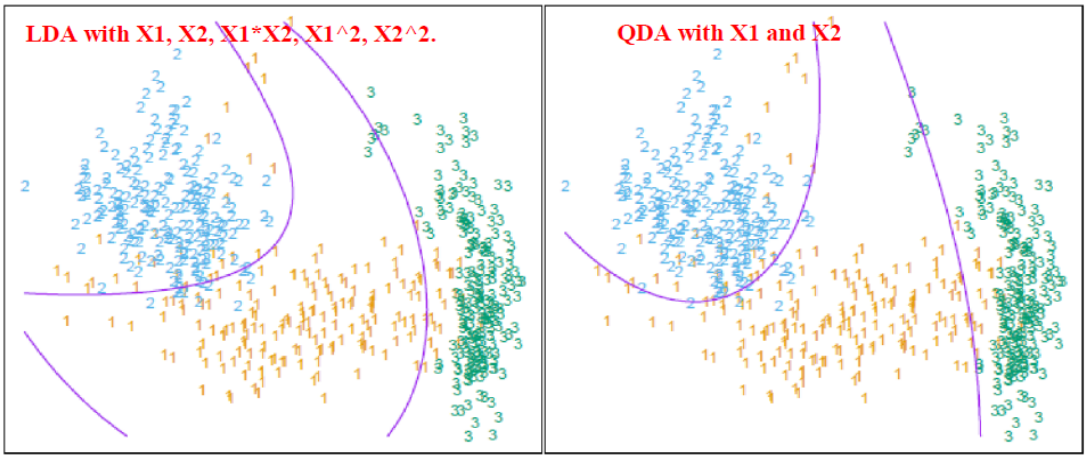


FIGURE 4.6. *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

Note that the left-plot is the classification from the LDA in a five-dimensional space (X_1, X_2, X_1*X_2, X_1^2, X_2^2). Then, projects LDA's linear decision boundaries into a 2-dimensional space (X_1, X_2) for a visualization plot. The right-plot is the classification from the QDA using X_1 and X_2 variables (2-dimensions) only. The results from these two procedures are similar, but different slightly.

(c) Naïve Bayes Classifier

This is a special case of (general) Bayes Classifier defined in #3 above. It can also be a special case of LDA/QDA/RDA.

Instead of using a (dependent) multivariate distribution to model the data-likelihood, all X-variables are assumed to be independent, and thus,

$$f_k(x_k\text{-}matrix) = \Pr(X_k\text{-matrix} = x_k\text{-}matrix \mid G = k)$$

is a product of marginal densities, i.e., pdfs for each X_{1k}-*vec*, X_{2k}-*vec*, …, X_{pk}-*vec*. In the example of LDA, QDA and RDA, **correlations between all X-variables are all zeros**. This implies that the variance-covariance matrix Σ_k only has diagonal elements (i.e., variances) and all off-diagonal elements (i.e., covariances) are all zeros.

ISyE DDA – Agenda for 02/23/23 Lecture (Lec.13)

1. **Semester Project's Decision-Model-1 (DM-1) Feedback**
   a) **General comments for DM-1 and DM-2 reports**
   b) **General Decision-Model Structures**
   c) **How is a decision model linked to a data model?**
   d) **Layers-of-Presentation (LoP)**
2. **Decisions in an Uncertain Environment (DiUE) – Data Modeling**

Details:

1. Semester Project's Decision-Model-1 Feedback

a) General Comments for DM-1 and DM-2 reports:

In students' decision-model-2 (DM-2) reports, the following should be presented in layers-of-presentation sequence. Some details can be included in an appendix to keep the main content flow smoothly. In DM-2 reports, all notations need to be defined, and all equations need to be explained or justified.

DM-2 reports need to include a "transition" paragraph for linking the presentation of decision models to LoM studies of problem background. It is important to understand the rational of how students formulate a "focused project problem" out of many possibilities in the problem background studies.

b) General Decision-Model Structures

A decision model includes:

(1) decision variable(s),

(2) objective function(s) to be optimized,

(3) constraint(s),

(4) assumptions.

Three types of quantities might be involved in objective function(s) and constraint(s).

[1] decision variables to be found via an optimization of the objective function(s) and constraint(s),

[2] "needed quantities" to be predicted, estimated, or calculated via data models,

[3] "given constants" provided by users of the decision model(s).

c) How is a decision model linked to a data model?

A decision model is linked to a data model via the "needed quantities" discussed above. See examples below.

d) Layers-of-Presentation (LoP)

Let us see two top DM-1 reports for illustrating their applications of the LoP technique.

Travel Itinerary Creation:

Objective of Project

Our project this semester aims to provide travelers with tailored travel itineraries based on traveler's preferences, time constraints, and budget. It will be applicable to a single traveler, not a group of people with different preferences. Focusing on convenience, accessibility, and efficiency, this project hopes to optimize the best routes by cutting down time and money for tourists while also supporting local businesses and attractions.

Decision Variables Good

- $d_{ijk} = 1$ *if one uses transportation mode i to get from location j to location k,* 0 *otherwise*
- $h_i = 1$ *if hotel i is selected out of o options,* 0 *otherwise*
- $f_i = 1$ *if flight i is selected out of l options,* 0 *otherwise*
- $a_{ij} = 1$ *if attraction i is visited on day j,* 0 *otherwise*
- $r_i = 1$ *if restaurant i is selected out of q options,* 0 *otherwise*

Parameters

These "parameters" are "given constants" provided by the user (or other "system inputs").

- t_{ijk} = *cost per mile of using transportation mode i to get from location j to location k*
- p_i = *cost of attraction i*
- m_{ijk} = *# of miles traveled using transportation mode i to get from location j to location k*
- n = *number of attractions that an individual is likely to visit*

If there are some "needed quantities" to be predicted/calculated by data models, please specify them in the end of Decision-Model-2. Then, data model(s) can be connected to your decision models via these "needed quantities".

Objective Functions

Why does one want to minimize their transportation distance in traveling? Note that air-travel can cover a long distance. You do provide a good example below. Yes, this objective will be "compensated" against other objectives involving cost and satisfaction.

1. Minimize Transportation Distance

This objective function aims to take into account distance from a traveler's starting point to the recommended locations within a particular time constraint set by the traveler. For example, the traveler can ask the service for an 8-hour travel itinerary from 9AM- 5PM starting and ending at their hotel. The objective function here would be able to provide x activities for them to do while taking into account the starting/ending locations, time constraint, and modes of transportation.

$$min \sum_{i=1}^{s}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ijk}\, m_{ijk}\ (total\ miles\ traveled),\ \ j \neq k$$

An example of the scenario would be that there are three locations - hotel to L1 (car) to L2 (train) and back to the hotel (bus). The objective function will minimize the total transportation distance - $d_{hotel,\,L1,car}\, m_{hotel,\,L1,car} + d_{L1,\,L2,train}\, m_{L1,\,L2,train} + d_{L2,\,hotel,bus}\, m_{L2,\,hotel,bus}$

I see a good example here.

2. Minimize Itinerary Costs

This objective looks at the financial characteristics of the itinerary created. The traveler is able to set an upper bound constraint for his or her budget (for example $250 for one day) and utilize the information of the recommended locations and other possible financial factors to minimize overall costs.

$$min \sum_{i=1}^{s}\sum_{j=1}^{n}\sum_{k=1}^{n} t_{ijk} d_{ijk}\, m_{ijk}\ (total\ transportation\ cost) + z\sum_{i=1}^{o} b_i h_i\ (total\ hotel\ cost) + \sum_{i=1}^{l} f_i w_i\ (flight\ cost)$$

good

$$+ \sum_{i=1}^{n}\sum_{j=1}^{z} a_{ij} p_i\ (total\ attractions\ cost) + \sum_{i=1}^{q} r_i c_i\ (total\ food\ cost)$$

3. Maximize User Satisfaction of Attractions Visited

For this objective, we can determine n, the number of attractions that a traveler is likely to enjoy, through data modeling techniques such as logistic regression or other classification methods. Data modeling can also help us predict the satisfaction that a person may get from certain attractions - which could be measured on a 1-10 scale.

We want to maximize the satisfaction provided by the attractions that they can visit throughout the course of a trip spanning z days.

$$max \sum_{i=1}^{n} \sum_{j=1}^{z} u_i a_{ij} \quad (total\ satisfaction\ from\ attractions)$$

Constraints:

1. Budget

Assumptions

- When estimating costs for restaurants for budget reasons, we will use the average of the prices of all the dishes + some amount to account for dessert or drinks. Our model will try to be on the more conservative side when estimating prices to ensure that the user will be able to stay within budget.
- When drawing the boundary between handicap accessible and inaccessible locations and activities, we assume that physically handicapped users all require the same level of accessible features in facilities they visit.
- We assume that a traveler will only visit an attraction/restaurant once throughout the duration of a trip.
- We assume that a traveler will only stay in one hotel at a destination.

Rental Property:

Decision Model - Objective Functions

For our decision model, our objective would be to maximize profit from these rental properties. Profit is made up of several components, but as a whole is defined as revenue less costs. In terms of revenue, we would receive revenue in the form of rental payments. This would also depend on the rent risk, involving the tenant information as well as the contract itself. Costs are much more complicated because we would have to account for a multitude of costs involving maintenance, labor for the upkeep of the property, taxes, HOA fees, insurance, interest on mortgage. These payments are not incurred at the same intervals, so it is important to modify the revenue and costs accordingly to reflect the same time period. For example, we could choose to calculate profit on a yearly basis, so if the rental payments are incurred monthly, we would multiply number of units * monthly payment * 12 months.

$$maximize\ Profit\ Cash\ Flow\ =\ Revenue\ -\ Costs$$

***Decision Variables are bolded** excellent presentation with the bolded decision variables

I assume that "vacancy rate" will be predicted/estimated or calculated from a data model to be discussed later. Note that the rate might also depend on the rental price.

Gross rental revenue = $\sum_i$ **RentPrice_i** * (1 - Vacancy Rate)

Tax is a "given constant" to be provided by the user - correct?

Operating Expenses = $\sum_i \sum_j \sum_k \sum_p$ Property Taxes + **InsuranceProvider_j** *insurance provider j cost + HOA fees + **Contractor_k** * **Repair_p** *contractor provider k cost+ **Water_i** *unit cost of water at property i +**Gas_i***unit cost of gas at property i + Property Management Fees + **Advertising_i** + **Legal_i**

Typo for two "insuranceprovider_j" ?

Rental Payments =$\sum_i$ **RentPrice_i** * if property i is occupied or not + **Pet_i*** pet cost at property i

How does your rental payments come into your gross rental income?

Net Operating Income = Gross Rental Income - Operating Expenses

Overall: Cash Flow = Net Operating Income - $\sum_i$ Mortgage Payment for property i (if applicable)

---

2. Decisions in an Uncertain Environment (DiUE) – Data Modeling

(continuation of materials presented before Exam-1)

The Supply Chain System:

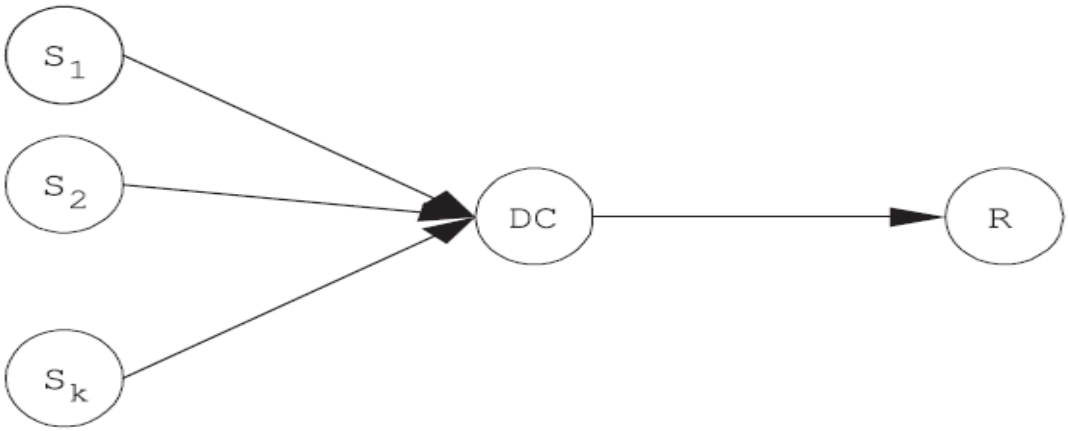


Fig. 1. Supply-chain network with *k* suppliers and one DC.

Step-1: Decision Model:

A. Decision Variable: Q_T = Total Order-Quantity for all suppliers

B. Objective Function:

Retailer's Profit =

Π(Q) = sales revenue – procurement cost – inventory cost – "stock-out" cost

$$= r\,\text{Min}[\,\xi, (1 - Y)Q\,] - c(1 - Y)Q - h[(1 - Y)Q - \xi]^{+} - \pi[\xi - (1 - Y)Q]^{+}, \qquad \text{Eq. (2.2)}$$

where $[A]^{+}$ represents max[A, 0].

Notations:

[1] r = unit-profit for products sold, ξ = amount of products sold;

[2] Y is the average (random) proportion of defects for all products arrived at the retail store; Thus, $(1 - Y)Q_T$ = #products (ordered and arrived at the store) without defects;

[3] c = unit-procurement cost, h = unit-holding-cost, and π = unit-stockout-cost, where stock-out means that no product to sell.

Step-2: Data Models for Y:

A. **Defect-Data (Y) Modeling** includes the following **four stages**:
   (1) #defects occurred during shipping from a Supplier S_j (j = 1, 2, …, K) to DC;
   (2) Aggregation of products at the DC;
   (3) #defects occurred during shipping from DC to Store (R);
   (4) Total #defects occurred during (1) – (3) above. See detailed discussion of this stage in Layer-2 modeling in file 1.2) Details_Supplier_to-DC-to-Store…..

B. "Contingent" versus "Typical" Situations:

In this problem our focus of the *uncertainty* is that there are two situations for the shipping process: (i) contingent situation and (ii) typical situation. In the contingent situation, the proportion of #defects in the shipped products is large, e.g., 40%, but the chance to face this situation is small, e.g., 0 – 5%. On the other hand, in the typical situation, the proportion of #defects is small, e.g., 5%, but the chance for this to happen is large, e.g., 95%.

The following provides an example of modeling the #defects in the process of **shipping products from a supplier S_j ($j = 1, 2, \ldots, K$) to DC**. Denoted by X_{jC} and X_{jN} (for $j = 1, 2, \ldots, K$) the (random) proportion of defective products in a unit-product shipment during the contingent and typical situations, respectively. Define an indicator function I_j as $I_j = 1$ if it is a contingent situation; $I_j = 1$, otherwise. For a supplier S_j the proportion of defects for a unit of products is then

$$P_{jDC} = (1 - I_j)*X_{jN} + I_j*X_{jC}, \quad j = 1, 2, \ldots, K, \qquad (1)$$

which is a random variable consisting of two cases with their own corresponding proportions of defects. Discussion of the

distributions of these *three* random variables are skipped here. See file 1.2) Example_Supplier_to-DC-to-Store….. for details.

[2] Details of Defect-Data (Y) Modeling:

The following provide details of probability-based Y-data modeling for the *four stages* (discussed in page 4 above) from suppliers to the retail store.

1> Stage-1: Shipping from Supplier S_j ($j = 1, 2, \ldots, K$) to DC:

For a supplier S_j **the proportion of defects** for a unit of products is then

$$P_{jDC} = (1 - I_j)*X_{jN} + I_j*X_{jC}, \quad j = 1, 2, \ldots, K, \qquad (1)$$

which is a random variable with a mixture distribution of two cases with their own corresponding proportions of defects.

2> Stage-2: Aggregation of Products at DC - "Average-Aggregation":

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

Because for each supplier there are $Q_{Sj}*P_{jDC}$ #defects among Q_{Sj} #ordered-products, the **average-aggregated proportion of defects from all suppliers** is

$$P_{au,DC} = \Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC}) / Q_T, \qquad (2)$$

where $Q_T = \Sigma_{j=1}^{K} Q_{Sj}$ is the total number of ordered-products. Note that the numerator in Eq.(2) is the total number of defects from all suppliers' products.

3> **Stage-3: Shipping from DC to Store:**

Our goal is to formulate **Y* ≡ the *average proportion of defects* for one unit of products shipped from suppliers through DC and to the store,** which was the goal of the defect-data-modeling studies.

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

Similar to the definitions of the three random variables (1) above, for **shipping from DC to store** we have X_C^* and X_N^* (for $j = 1, 2, \ldots, K$) the proportion of defective products in a

unit-product shipment during the contingent and typical situations, respectively. The indicator function I_0 is defined as $I_0 = 1$ if it is a contingent situation; $I_0 = 1$, otherwise. Then, the proportion (P_R) of defects from **shipping between DC and store** is

$$P_R = (1 - I_0)*X_N^* + I_0*X_C^*. \quad (3)$$

Note that there are $\Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC})$ amount of defects out of Q_T total number of products-ordered (and shipped from suppliers). Assume that there is NO INSPECTION at the DC for taking out defective products. There are

$$Q_T - \Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC})$$

$$= Q_T - Q_T *P_{au,DC} = Q_T *(1 - P_{au,DC}) \quad (4)$$

amount of non-defective products shipped from DC to store. See Eq. (2) for understanding the first equality above.

Since the $Q_T *(1 - P_{au,DC})$ amount of non-defective products shipped from DC to store will subject to further shipping damage (with the probability P_R defined in Eq.(3)), the #defects for non-defective products shipped from DC and arrived at the store is

$Q_T * (1 - P_{au,DC}) * P_R$.

Note that there are only $Q_T * (1 - P_{au,DC}) * (1 - P_R)$ amount of non-defective products arrived at the store eventually.

4> Stage-4: Calculate total #defects occurred at all stages.

From supplier's (or logistics service provider's) point of view, the total #defects (and its proportion) occurred at Stage (1) and (3)'s shipping processes is important for calculating the cost of product-supply. From retailer's (store manager's) point of view, the #good-products arrived at the store and also the proportion (#good-products arrived at the store / #total-ordered-products) is important for supporting their sales (to meet customer demands).

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

There are $\Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC})$ amount of defects in Stage-1. Among $Q_T * (1 - P_{au,DC})$ good-products (see Eq.(4))

shipped in Stage-3, there are $Q_T * (1 - P_{au,DC}) * P_R$ amount of defects. Thus, the total #defects is

$$\Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC}) + Q_T * (1 - P_{au,DC}) * P_R.$$

If one is interested in the **average-proportion of defects in all stages**, it is then

$$Y = [\ \Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC}) + Q_T * (1 - P_{au,DC}) * P_R\ ] / Q_T\ .$$

Step-3: Optimization-Solution Procedures

1) **Risk Neutral: Find $Q \equiv Q_{T,opt}$ = total order-quantity to maximize Expected Profit.**
2) **Risk Averse:**
   i) **Constrained Optimization I – Profit Constraint**

   Maximize the Expected Profit

   subject to that the expected profit in the contingent situation is over a certain threshold. That is,

$$\max_{Q \geq 0} \; E_G[\Pi(Q,Y)]$$

$$\text{s.t.} \quad E_{G_C}[\Pi(Q,Y)] \equiv E_G[\Pi(Q,Y)|I=1] \geq \Pi_0, \qquad (3.4)$$

where I is the indicator function for a contingency.

Notations:

[1] G is the distribution for the random variable Y (= average proportion of defects for products arrived at the store). G has the mixture distribution mixing from distributions in both contingent and typical situations; The expectation in the main objective is taken with respect to (w.r.t.) the distribution G; See File 1.2) Example_Supplier…. for details of G.

[2] G_c is the distribution for the random variable Y in the contingent situation; The expectation in the constraint is taken w.r.t. G_c.

ii) Constrained Optimization II – Probability Constraint

Maximize Expected Profit

subject to that in the contingent situation the probability of (profit being less than a given amount) is less than or equal to a certain threshold.

$$\max_{Q \geq 0} \quad E_g[\Pi(Q,Y)]$$

$$\text{s.t.} \quad P_g(\Pi(Q,Y) \leq \Pi_1) \leq \gamma, \tag{3.5}$$

where

$$\Pi(Q,Y) = \begin{cases} r\xi - c(1-Y)Q - h((1-Y)Q - \xi), & (1-Y)Q \geq \xi, \\ r\xi - c(1-Y)Q - (r+\pi)(\xi - (1-Y)Q), & (1-Y)Q \leq \xi. \end{cases}$$

Remarks:

[1] The expectation for the objective should be taken w.r.t. to G defined in (i) above.

[2] The probability evaluation for the constraint should be taken w.r.t. to G_c for the contingent situation.

iii) Mean-Variance Solution

$$\max_{Q \geq 0} E_G[\Pi(Q,Y)] - \alpha Var_G[\Pi(Q,Y)]$$

or

$$\max_{Q \geq 0} (E_{G_N}[\Pi(Q,Y)] - \alpha Var_{G_N}[\Pi(Q,Y)])P[I=0] + (E_{G_C}[\Pi(Q,Y)] - \alpha Var_{G_C}[\Pi(Q,Y)])P[I=1].$$

where G_N is the distribution for the random variable Y in the typical situation;

iv) Max-Min Solution

$$\max_{Q} \min_{G} E_G[\Pi(Q,Y)].$$

The Max-Min procedure provides a solution which maximizes the expected profit under the worst case scenario. Figures

Remarks: The worse-case of the expected-profit occurs in the contingent situation, where 40% of products shipped

are defective. Thus, the optimal total order-quantity $Q_{T,opt}$ is located to maximize this worse-case expected-profit.

Final Remarks:

When a procedure is developed, one needs to explore properties of this procedure and compare them with respect to a few alternative solutions (i.e., optimization models/methods). In exploring these properties, data models' distribution parameter values should be changed to represent various real-world situations. Moreover, several constants involved in the decision-model (e.g., unit-profit, and unit-procurement, - holding, - stockout costs) need to be varied for exploring different situations possibly occurred in real-life supply-chain practices.

The original studies Kim, Lu, Kvam and Tsao (2011) have applied analytical methods and numerical calculations to explore properties of the developed models and optimization methods. The following provides an example of a brief summary of these properties.

ISyE DDA – Agenda for 02/16/23 Lecture (Lec.12)

1. Exam-1 Review – Exam-1 Topics

2. 2022 & 2020 Past Exam-1 Problems and Solutions

Details:

ISyE-DDA- Exam-1 Subjects

Exams in 2023 Spring will all be in-class exams.

1. **Data Modeling (50%)** – focus on

   i) Logistic Regression, GLM and Nonlinear Regression

1. **Decision Modeling (40%)** – focus on formulating simple real-world problems into decision-models using the following techniques.
   i) Linear Programming (including Linear Integer and Linear Mixed Integer Programming) (skipped in your exam)
   ii) Non-Linear Programming (high-level conceptual questions)
   iii) Multi-Objective Optimization

2. **Semester-Project Questions (10%):**
   i) Problem Formulation Techniques: LoM
   ii) Decision Model Formulation Techniques: Formulating objective function(s), decision variables and constraints with English.

There will be a few high-level conceptual from various semester projects. Examples will be presented in lectures.

==========

2022 Fall ISyE 4034 Exam #1 (V1) Name: Solution

A) **True and False Questions (5 points for each question).**

(True, False) 1. R^2 is used in Logistic Regression and Generalized Linear Models for evaluating how well a model fits the data.

Answer: False. Because the ANOVA decomposition, SSTO = SSR + SSE does not hold in the logistic, GLM and Nonlinear regressions, R^2 = SSR/SSTO does not exist.

(True, False) 2. The following is a nonlinear regression model:

$$Y = \beta_0 + \exp(\beta_1)*(x^2) + \varepsilon.$$

Answer: False. One can redefine $\beta_2 = \exp(\beta_1)$ and $x_2 = (x^2)$ to re-formulate the above as the typical linear regression model: $Y = \beta_0 + \beta_2 * x_2 + \varepsilon$.

(True, False) 3. The following is a nonlinear-programming model for optimizing decision variables, x_1, x_2, x_3: $g(x_1, x_2, x_3) = \exp(x_1) * (x_2)^2 + x_3$.

Answer: True. Notice the nonlinear form of the decision variable x_1, and also the multiplication of two decision variables x_1 and x_2, which makes the above model nonlinear.

(True, False) 4. In ε-constraint and Lexicographic multi-objective optimization methods, the objective function or

constraint must be linear, i.e., it cannot be a nonlinear objective/constraint.

Answer: False. Some of the objective functions or/and constraints can be nonlinear in both ε-constraint and Lexicographic multi-objective optimization methods. One just needs to follow the definitions of these two methods to find thresholds/bounds ε_j or ε_i, respectively.

(True, False) 5. The maximum likelihood estimation (MLE) is used in all GLM distributions including Poisson, Exponential, Gamma and Normal, where in the Normal distribution case, MLE is equivalent as the LSE (Least Squares Estimation).

Answer: True. All GLM regressions require formulations of their likelihoods, which are derived from their respective Y-data distributions.

(True, False) 6. In logistic regressions, MLE is the one responsible for relaxing the constraint that the regression mean = E(Y) = π = Pr(Y = 1) is in (0, 1) interval.

Answer: False. Logit-transformation (not the MLE) is the one responsible for relaxing the constraint that the regression mean = E(Y) = π = Pr(Y = 1) is in (0, 1) interval. MLE is used to estimate unknown parameters.

(True, False) 7. The LoM (Layer-of-Modeling) tool is helpful in directing the steps that students should follow in their semester-project studies.

Answer: False. "Study Blueprint" is the tool helpful in directing the steps that students should follow in their semester-

project studies. LoM is the tool for understanding breadth and depth of the problem background.

(True, False) 8. In this DDA class, our data model is only used to support decision models. That is, if analyzing a data set does not help completing decision models, it is not needed in this class.

Answer: True. This "requirement" is emphasized a few times in lectures.

(True, False) 9. For selecting variables in logistic or GLM regressions, explanatory/input variables are selected to maximize a function of likelihood.

Answer: True. AIC is a negative and linear function of log-likelihood. One selects regression variables to minimize the AIC, which implies that the log-likelihood is maximized. Thus, the likelihood is maximized correspondingly.

(True, False) 10. In logistic and GLM regressions, standard deviations of regression parameter estimates are derived from a large-sample CLT (Central Limit Theory).

Answer: True. Because the unknown parameters including regression coefficients are estimated via the MLEs. The large-sample CLT theory is used to derive the estimates of the standard deviations of regression parameter estimates.

B) **Fill-In Blanks (5 points for each question):**

11. In optimizing multi-objectives with the ε-Constraint Method. One will

Minimize $f_j(x\text{-}vec)$ with respect to *x-vec in X-space*

Subject to $f_i(x\text{-}vec) \leq \varepsilon_j$ for $i = 1, 2, \ldots, K$ except j.

What are the two key words in locating a "single" upper bound ε_j for *all* secondary objectives (e.g., #job openings) that have *different* metric values?

Answer: Distribution Percentile .

Detailed Answers: One use the same percentile from the distributions of secondary objective functions $f_i(x\text{-}vec)$, $i = 1, 2, \ldots, K$ except j. The distribution of an objective function is constructed from all values of the objective evaluated with all combinations of its decision variables.

12. What is the distribution used to calculate p-values for the regression parameter estimates in the Nonlinear Regressions?

Answer: Normal distribution

Detailed Answers: Note that finite-sample student-*t* distributions for regression parameter estimates obtained from the *Nonlinear Least Squares method* taught in the class do not exist. If the data (or errors) are distributed as normal, the nonlinear least squares estimates are equivalent to the MLEs. Then, the large-sample CLT theory for the MLEs are used to approximate the distribution of the regression parameter estimates. This implies that the MLEs of regression parameter estimates have normal

distributions approximately. Thus, normal distributions should be used to calculate p-values for testing whether them are statistically significant.

C) Essay questions: (40 points in total)

13. What are the steps in formulating a Gamma regression procedure? Please provide details including likelihood and link function. [30 points]

Answer:

Step-1: Formulate the Gamma regression model.

Let us use regression over **mean** of a GLM distribution here to formulate a Gamma regression model.

[1] $Y_i = E(Y_i) + \varepsilon_i$, $i = 1, 2, \ldots, n$,

where Y_i has a Gamma distribution with two parameters (α, δ_i), and the parameter α is an unknown parameter. The second unknown parameter δ_i is linked to the regression function (see Item [3] below for details). Also, see pages 18-19 Lecture-Notes W2.1.3) for details of Gamma regressions. Note that I change the notation β to δ in this presentation to avoid confusions to the regression coefficients $\beta_0, \beta_1, \ldots, \beta_{m-1}$.

The mean (≡ expected value) is a deterministic component in this regression, and the modeling error ε_i is a random variable describing the randomness of Y_i-data.

[2] According to the Gamma(α, δ_i) distribution posted in Lecture-Notes W2.1.3),

$$E(Y_i) = \alpha / \delta_i,\ i = 1, 2, \ldots, n,$$

where the parameter δ_i is unknown and is related to the i-th subject, i.e., different data Y_i has a distinct parameter δ_i.

[3] In the GLM regression, some distribution parameters are "linked" to regression coefficients. In Gamma regressions, based on Lecture-Notes W2.1.3), the link function is

$$-\delta_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i},$$

where there could be one intercept and $(m - 1)$ explanatory/input variables, $x_{1i}, x_{2i}, \ldots, x_{m-1i}$, and their corresponding UNKNOWN regression coefficients are $\beta_0, \beta_1, \ldots, \beta_{m-1}$.

[4] Because the mean $E(Y_i) = \alpha / \delta_i$ involves the Gamma distribution parameter δ_i in the denominator, we need to "transform" the link-function to $1/\delta_i$. Note that from the link-function above

$$1/\delta_i = -1 / [\ \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i}\ ].$$

Thus,

$$E(Y_i) = \alpha / \delta_i$$

$$= -\alpha / [\ \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i}\ ],$$

which is a NONLINEAR regression just like the logistic regression.

[5] In conclusion, a Gamma regression model is

$$Y_i = -\alpha / [\, \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i} \,] + \varepsilon_i, \;\; i = 1, 2, \ldots, n,$$

where the unknown distribution parameter α and the regression coefficients $\beta_0, \beta_1, \ldots, \beta_{m-1}$ will be estimated via the following MLE method.

Prediction of this Gamma regression is

$$Y_i\_hat = E(Y_i)\_hat$$

$$= -\alpha\_hat / [\, \beta_0\_hat + \beta_1\_hat\, x_{1i} + \ldots + \beta_{m-1}\_hat\, x_{m-1i} \,].$$

Step-2: Formulate the Likelihood Function preparing the ML estimation of regression coefficients.

[1] There are $Y_1, Y_2, \ldots, Y_n$ data points in this regression problem. Assumption of mutual independence of these data implies that the *joint distribution* of these *n* random variables is a product of their individual "marginal distributions". Denoted by $f_{Yi}(.)$ the pdf of the distribution for data point Y_i, $i = 1, 2, \ldots, n$.

If these data points all have the same distribution (other than their parameters), we say that these *n* random variables are identically distributed, i.e., $f_{Yi}(.) = f_Y(.)$ without the specific subscript "*i*" in the pdf. Then, the joint distribution of these data, which is the likelihood of the data becomes

Likelihood = $\prod_{i=1}^{n} f_Y(.)$.

In the Negative Binomial distribution, according to page 22 of Lecture-Notes W2.1.3), NB's distribution pdf is

$$f_Y(.) = [\ \delta^{\alpha}\, y^{\alpha-1} / \Gamma(\alpha)\ ] \exp\ \{ -\delta\, y \}.$$

Apply this pdf to $Y_1, Y_2, \ldots, Y_n$ data points, which have observations $y_1, y_2, \ldots, y_n$ and their corresponding parameters $(\alpha, \delta_1), (\alpha, \delta_2), \ldots, (\alpha, \delta_n)$. Note that the (shape) parameter α is the same for all data point due to the identical distribution assumption. The unknown parameter δ_i, $i = 1, 2, \ldots, n$, "acts" like Normal regression's mean $\mu_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i}$. In the Gamma regression case, our parameters

$$\delta_i = -\ [\ \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i}\ ],$$

discussed in page 5 Item [4] above.

Based on the discussion above, the likelihood for a Gamma regression is

$$\text{Likelihood} = \prod_{i=1}^{n} \{\ [\ \delta_i^{\alpha}\, y_i^{\alpha-1} / \Gamma(\alpha)\ ] \exp\ \{ -\delta_i y_i \}\ \}$$

$$= \prod_{i=1}^{n} \{\ \{\ [-(\ \beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i}\ )]^{\alpha} y_i^{\alpha-1} / \Gamma(\alpha)\ \}$$

$$* \exp[\ (\beta_0 + \beta_1 x_{1i} + \ldots + \beta_{m-1} x_{m-1i}\ ) * y_i\ ]\ \}$$

which involves all data points $y_1, y_2, \ldots, y_n$ and unknown distribution parameter α and regression coefficients, $\beta_0, \beta_1, \ldots, \beta_{m-1}$.

[2] Because every pdf has a value between 0 and 1, multiplication of these pdf's n times will make the value of the likelihood very small. Traditionally, to find the MLEs of these regression coefficients, one

maximizes logarithmic of the likelihood, called log-Likelihood with respect to these *m* unknown regression coefficient.

Step-3: Apply a computing algorithm such as Newton-Raphson method to maximize the Log-Likelihood(α; β_0, β_1, …, β_{m-1}) for locating the MLEs, α_hat, β_0_hat, β_1_hat, …, β_{m-1}_hat.

Step-4: Use the MLEs in a Gamma regression function to make a prediction. Large-Sample CLT theorem can be used to obtain the confidence interval of the prediction and test whether each regression coefficient estimate is significant.

14. List the steps logically in using the linearly weighted sum method for optimizing multiple objectives? [10 points]

Answer:

Step-1: Normalized all objective functions such that their values are compatible, e.g., their values are all in the [0, 1] regions.

A procedure to normalize objective functions is to use the following re-centering and re-scaling formula to make their values into [0, 1] regions.

Normalized Objective = f_i^N**(decision variables)**

= (Objective function – Min) / (Max – Min),

where Min (or Max) = minimum (or maximum) of this objective function evaluated with respect to (w.r.t.) all possible values of decision variables.

Step-2: Decide the weights w_i, $i = 1, 2, \ldots, K$ (= #objectives), for each objective. Note that sum of all weights is one.

Step-3: Create a single objective by adding all normalized objectives multiplied by their corresponding weights.

Overall Objective(decision variables) = $\sum_{i=1}^{K} w_i * f_i^N$(decision variables).

Step-4: Perform the optimization of the above overall objective w.r.t. decision variables.

2020 Fall ISyE 4803/8803 (LUJ-DDA) (V1) Exam #1

[70%]

A) **True and False Questions (5 points for each question).**

(True, False) 1. In logistic regressions the logit-transformation can be replaced by any transformation that maps $\pi_i = \Pr(Y_i = 1)$ with the [0, 1] range to another parameter with $(-\infty, +\infty)$ range.

Answer: True. For example, Probit transformation $\Phi^{-1}(\pi_i)$ can do the same as what logit-transformation does. Please google Probit-transformation for details.

(True, False) 2. In Poisson Regressions (an example of GLM) the mean of Y-data is a nonlinear function of regression parameters $\beta_0, \beta_1, \beta_2, \ldots$

Answer: True. Because the link function in Poisson-regression is $\log(\lambda) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$, the mean of Y-data = $E(Y) = \lambda$ is linked to regression parameters as $\lambda = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots)$, which is nonlinear.

(True, False) 3. To formulate the **likelihood** for the maximum-likelihood-estimation (MLE) of model parameters, the **only information** needed is that the i.i.d. assumption is valid.

Answer: False. Besides the i.i.d. assumption, the Y-data-distribution information is critical in formulating the likelihood.

(True, False) 4. In logistic regressions, p-values for parameter-estimates should be calculated from the student-*t* distribution like those in the typical linear-regressions.

Answer: False. Because MLE is used to estimate parameters in the logistic-regression, normal distribution shall be used to calculate their p-values (assuming that the sample size is large).

(True, False) 5. R-square or Adjusted-R-square commonly seen in typical regression should be useful in logistic regressions.

Answer: False. The assumptions needed to calculate the R-square or Adjust-R-square (e.g., ANOVA decomposition to calculate SSR) do not exist in the logistic-regression.

(True, False) 6. Both stepwise-regression and all-subsets-regression are not useable for selecting important variables in logistic regressions.

Answer: False. All subsets regression is still useful. However, the AIC-metric should be used in comparing models with various variable-combinations.

(True, False) 7. When a logistic regression is used to classify the predicted outcomes into Class-1 or Class-0, one always uses the rule that Prediction of $\Pr(Y_0 = 1) \geq \mathbf{0.5}$, Classify Y_0 into Class-1; otherwise, classify it into Class-0. The percentage 0.5 in the decision-rule *cannot* be changed to another percentage.

Answer: False. The percentage 0.5 is used for convenience only. In practice, if the risk of the mis-classification into Class-1 is higher, one can raise this percentage to 0.6 (or higher) to make its mis-classification rate smaller.

(True, False) 8. In solving multi-objective optimization problems, the ϵ-constraint method only maximizes (or minimizes) one objective-function and leaves the other objective-functions into constraints.

Answer: True. This is the definition of the ϵ-constraint method.

(True, False) 9. In applying layer-of-modeling to explore uncertainties in the airplane-defrosting-management-system (ADMS), weather-with-heavy-snow or weather-with-no-snow should be modeled in the top-layer, but source-of-uncertainty such as weather-condition and external-factor (e.g., VIP-plane arrivals and takeoffs) should be modeled in the second layer.

Answer: False. They should be reversed. That is, details of the weather condition (e.g., heavy or no snow) should be modeled in the second layer.

(True, False) 10. The first task in building a simulation system for entertaining what-if decisions in the ADMS is to layout all steps an airplane needs to go through before departing. The second task is then to compile past-data for understanding the distribution of elapse-time in each step.

Answer: True.

B) **Short-Answer Questions:**

11. What are the *three* best properties of the MLEs? [6 points]

Answer: When sample-size is large (e.g., more than 30),

(1) the MLEs have a normal distribution,

(2) consistency, which means that the bias (= E(MLE) – true parameter value) goes to zero,

(3) smallest variance compared variances from all other estimation methods.

12. (a) Present one example briefly for the nonlinear-optimization model. Use generic notation that the objective function $f(x_1, x_2, x_3)$ is a function of three decision-variables x_1, x_2 and x_3. [6 points]

(b) NAME the key-technique addressed in the lecture for solving the nonlinear optimization problem? [4 points]

Answer: (a) $f(x_1, x_2, x_3) = x_1 * x_2 + x_3$.

(b) Linear approximation (based on the Taylor expansion).

13. Consider a multi-objective-optimization problem with two objective-functions.

(a) Use a figure to present the Pareto Frontier, [5 points]

(b) Use the above figure to explain how to select the optimal solution(s). [5 points]

Answer: See lecture notes. Students did quite well for this problem.

C) **Essay Questions:**

14. In the plane-defrosting project suppose that there are three objectives listed in the order of importance as below. Provide steps with just enough details to describe how to implement the

Lexicographic method for this multi-objective optimization problem. [24 points]

(1) Minimize the **percentage of planes, which did not meet the "less-than-40-minute" elapse-time** between door-closed-time to plane-depart-defrosting-station-time.

(2) Minimize the **average elapse-time** defined in (1).

(3) Minimize the **average amount of the any defrosting-station is idled** (i.e., not defrosting any plane)**.**

Answer: See lecture notes and past-exam solutions. Students did quite well with this problem.

ISyE DDA – Agenda for 02/14/23 Lecture (Lec.11)

1. New Materials for Exam-2

i) Decisions in an Uncertain Environment (DiUE) – Data Modeling

ii) Concluding Remarks for DiUE studies

Note that the decision analytics modeling technique is explained in this topic.

Details:

1. Decisions in an Uncertain Environment

Example – Order Quantity Decisions Considering Uncertainty in Supply-Chain Logistics Operations (Kim, Lu, Kvam, Tsao, 2011)

Step-0: Problem Background - Figure 1 in Kim *at el.* (2011)

Our goal in this study is to decide the optimal Order Quantities Q_i (i = 1, 2, …, K) for K (=eg= 3) suppliers such that Retailer's profit is maximized. These suppliers S_i (i = 1, 2, …, k) ship the **same product** through a distribution center ("DC" below) to one Retail ("R" notation). See Figure 1

below for a graphical display of this logistic system. Note that there are uncertainties in the shipping process. The uncertainties include defect occurred during the shipping process and delayed schedules for shipments. They are all quantified in "**defects**" from imperfect shipping in our studies below.

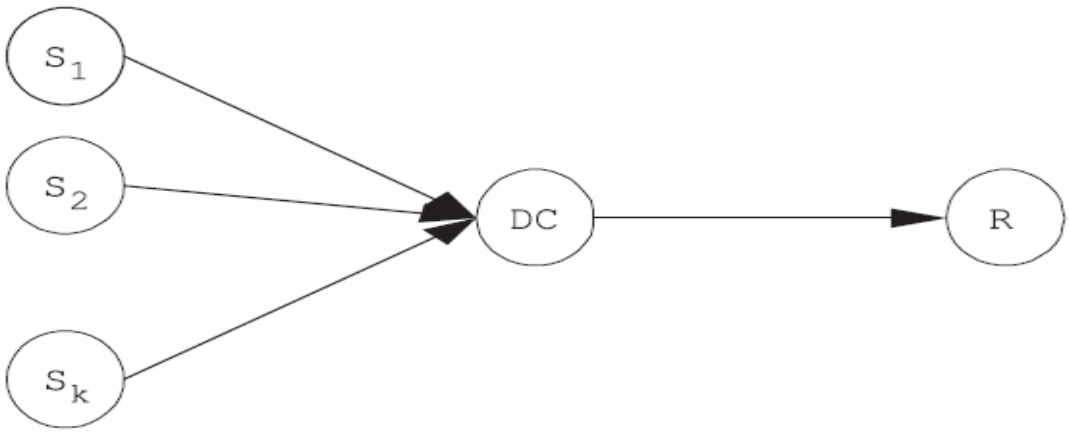


Fig. 1. Supply-chain network with *k* suppliers and one DC.

Step-1: Decision Model

The goal is to locate the optimal *order-quantity* by maximizing the profit. However, due to uncertainty in the regular or contingent situations, the optimization process is much more involved. See Step-3 for details.

A. Decision Variable: Q_T = Total Order-Quantity for all suppliers

B. Objective Function:

Retailer's Profit =

$\Pi(Q)$ = sales revenue – procurement cost – inventory cost – "stock-out" cost

$$= r\,\text{Min}[\,\xi, (1 - Y)Q\,] - c(1 - Y)Q - h[(1 - Y)Q - \xi]^+ - \pi[\xi - (1 - Y)Q]^+, \qquad \text{Eq. (2.2)}$$

where $[A]^+$ represents max[A, 0].

Notations:

[1] r = unit-profit for products sold, ξ = amount of products sold;

[2] Y is the average (random) proportion of defects for all products arrived at the retail store; Thus, $(1 - Y)Q_T$ = #products (ordered and arrived at the store) without defects;

[3] c = unit-procurement cost, h = unit-holding-cost, and π = unit-stockout-cost, where stock-out means that no product to sell.

Step-2: Data Models for Y:

[1] Background:

A. Defect-Data (Y) Modeling includes the following **four stages**:

(1) #defects occurred during shipping from a Supplier S_j (j = 1, 2, …, K) to DC;

(2) Aggregation of products at the DC;

(3) #defects occurred during shipping from DC to Store (R);

(4) Total #defects occurred during (1) – (3) above. See detailed discussion of this stage in Layer-2 modeling in file 1.2) Details_Supplier_to-DC-to-Store…..

B. "Contingent" versus "Typical" Situations:

In this problem our focus of the *uncertainty* is that there are two situations for the shipping process: (i) contingent situation and (ii) typical situation. In the contingent situation, the proportion of #defects in the shipped products is large, e.g., 40%, but the chance to face this situation is small, e.g., 0 – 5%. On the other hand, in the typical situation, the

proportion of #defects is small, e.g., 5%, but the chance for this to happen is large, e.g., 95%.

The following provides an example of modeling the #defects in the process of **shipping products from a supplier S_j (j = 1, 2, …, K) to DC**. Denoted by X_{jC} and X_{jN} (for j = 1, 2, …, K) the (random) proportion of defective products in a unit-product shipment during the contingent and typical situations, respectively. Define an indicator function I_j as $I_j = 1$ if it is a contingent situation; $I_j = 1$, otherwise. For a supplier S_j the proportion of defects for a unit of products is then

$$P_{jDC} = (1 - I_j)*X_{jN} + I_j*X_{jC}, \quad j = 1, 2, \ldots, K, \qquad (1)$$

which is a random variable consisting of two cases with their own corresponding proportions of defects. Discussion of the distributions of these *three* random variables are skipped here. See file 1.2) Example_Supplier_to-DC-to-Store….. for details.

Remark: There could be more uncertain situations than our example above with two situations (one is the contingent situation and the other is the typical situation). One can assign a probability to each case, respectively, to formulate a "mixture" data-model similar to Eq. (1) above.

---

[2] Details of Defect-Data (Y) Modeling:

The following provide details of probability-based Y-data modeling for the *four stages* (discussed in page 4 above) from suppliers to the retail store.

1> Stage-1: Shipping from Supplier S_j ($j = 1, 2, \ldots, K$) to DC:

Denoted by X_{jC} and X_{jN} (for $j = 1, 2, \ldots, K$) the proportion of defective products in a unit-product shipment during the *contingent* and *typical situations*, respectively. They are both random-variables with distributions G_C and G_N, respectively.

Define an indicator function I_j as $I_j = 1$ if it is a contingent situation; $I_j = 0$, otherwise. The distribution of I_j is a Bernoulli with $\Pr(I_j = 1) = p$ and $\Pr(I_j = 0) = 1 - p$. Assume that the three random-variables, X_{jC}, X_{jN} and I_j are mutually independent. For a supplier S_j **the proportion of defects** for a unit of products is then

$$P_{jDC} = (1 - I_j)*X_{jN} + I_j*X_{jC}, \quad j = 1, 2, \ldots, K, \qquad (1)$$

which is a random variable with a mixture distribution of two cases with their own corresponding proportions of defects.

2> **Stage-2: Aggregation of Products at DC - "Average-Aggregation":**

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

Because for each supplier there are $Q_{Sj}*P_{jDC}$ #defects among Q_{Sj} #ordered-products, the **average-aggregated proportion of defects from all suppliers** is

$$P_{au,DC} = \Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC}) / Q_T, \qquad (2)$$

where $Q_T = \Sigma_{j=1}^{K} Q_{Sj}$ is the total number of ordered-products. Note that the numerator in Eq.(2) is the total number of defects from all suppliers' products.

(2) Evenly-split Ordered-Quantity $Q_{Sj} = Q_T / K$ for $j = 1, 2, \ldots, K$ – this is our focus.

This option is a special case of (1) unevenly-split ordered-quantity above. Apply the same formula with $Q_{Sj} = Q_T / K$ to get

$$P_{ae,DC} = \Sigma_{j=1}^{K} [(Q_T / K) * P_{jDC})] / Q_T = (\Sigma_{j=1}^{K} P_{jDC}) / K,$$

which is a simple average of proportion of defects from all suppliers'. Notice the index "ae" in the Proportion "*P*" for this case.

3> **Stage-3: Shipping from DC to Store:**

Our goal is to formulate **Y* ≡ the *average proportion of defects* for one unit of products shipped from suppliers through DC and to the store,** which was the goal of the defect-data-modeling studies.

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

Similar to the definitions of the three random variables (1) above, for **shipping from DC to store** we have X_C^* and X_N^* (for $j = 1, 2, \ldots, K$) the proportion of defective products in a unit-product shipment during the contingent and typical situations, respectively. The indicator function I_0 is defined as $I_0 = 1$ if it is a contingent situation; $I_0 = 1$, otherwise.

Then, the proportion (P_R) of defects from **shipping between DC and store** is

$$P_R = (1 - I_0)*X_N^* + I_0*X_C^*. \quad (3)$$

Note that there are $\Sigma_{j=1}^{K}(Q_{Sj}*P_{jDC})$ amount of defects out of Q_T total number of products-ordered (and shipped from suppliers). Assume that there is NO INSPECTION at the DC for taking out defective products. There are

$$Q_T - \Sigma_{j=1}^{K}(Q_{Sj}*P_{jDC})$$

$$= Q_T - Q_T *P_{au,DC} = Q_T *(1 - P_{au,DC}) \quad (4)$$

amount of non-defective products shipped from DC to store. See Eq. (2) for understanding the first equality above.

Since the $Q_T *(1 - P_{au,DC})$ amount of non-defective products shipped from DC to store will subject to further shipping damage (with the probability P_R defined in Eq.(3)), the #defects for non-defective products shipped from DC and arrived at the store is

$$Q_T *(1 - P_{au,DC}) * P_R.$$

Note that there are only $Q_T *(1 - P_{au,DC}) * (1 - P_R)$ amount of non-defective products arrived at the store eventually.

(2) Evenly-split Ordered-Quantity $Q_{Sj} = Q_T / K$ for $j = 1, 2, \ldots, K$:

This option is a special case of (1) unevenly-split ordered-quantity above. Applying the same ideas as given above by replacing $P_{au,DC}$ with $P_{ae,DC}$ for this evenly-split case, we have the following #defects for non-defective products shipped from DC and arrived at the store:

$$Q_T * (1 - P_{ae,DC}) * P_R.$$

4> Stage-4: Calculate total #defects occurred at all stages.

From supplier's (or logistics service provider's) point of view, the total #defects (and its proportion) occurred at Stage (1) and (3)'s shipping processes is important for calculating the cost of product-supply. From retailer's (store manager's) point of view, the #good-products arrived at the store and also the proportion (#good-products arrived at the store /

#total-ordered-products) is important for supporting their sales (to meet customer demands).

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

There are $\Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC})$ amount of defects in Stage-1. Among $Q_T * (1 - P_{au,DC})$ good-products (see Eq.(4)) shipped in Stage-3, there are $Q_T * (1 - P_{au,DC}) * P_R$ amount of defects. Thus, the total #defects is

$$\Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC}) + Q_T * (1 - P_{au,DC}) * P_R.$$

If one is interested in the **average-proportion of defects in all stages**, it is then

$$Y = [\ \Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC}) + Q_T * (1 - P_{au,DC}) * P_R\ ] / Q_T\ .$$

(2) Evenly-split Ordered-Quantity $Q_{Sj} = Q_T / K$ for $j = 1, 2, \ldots, K$:

Again, with $Q_{Sj} = Q_T / K$ the above results in (1) becomes

$$Y = [\ \Sigma_{j=1}^{K}[(Q/K)*P_{jDC}] + Q_T *(1 - P_{au,DC}) * P_R\ ] / Q_T$$

$$= (\Sigma_{j=1}^{K} P_{jDC})/K + (1 - P_{ae,DC}) * P_R.$$

Since $P_{ae,DC} = (\ \Sigma_{j=1}^{K} P_{jDC}\ ) / K$, the above formula becomes

$$Y = P_{ae,DC} + (1 - P_{ae,DC}) * P_R$$

$$= P_{ae,DC}\,(1 - P_R) + P_R$$

$$= [(\ \Sigma_{j=1}^{K} P_{jDC}\ ) / K\ ] * (1 - P_R) + P_R,$$

which is the average of proportion of defects for all products shipped from K-suppliers to store. This is the needed Y in the objective function (2.2).

Step-3: Optimization-Solution Procedures

1) **Risk Neutral: Find $Q \equiv Q_{T,opt}$ = total order-quantity to maximize Expected Profit.**
2) **Risk Averse:**
   i) **Constrained Optimization I – Profit Constraint**

   Maximize the Expected Profit

subject to that the expected profit in the contingent situation is over a certain threshold. That is,

$$\max_{Q \geq 0} \; E_G[\Pi(Q,Y)]$$

$$\text{s.t.} \quad E_{G_c}[\Pi(Q,Y)] \equiv E_G[\Pi(Q,Y)|I=1] \geq \Pi_0, \qquad (3.4)$$

where I is the indicator function for a contingency.

Notations:

[1] G is the distribution for the random variable Y (= average proportion of defects for products arrived at the store). G has the mixture distribution mixing from distributions in both contingent and typical situations; The expectation in the main objective is taken with respect to (w.r.t.) the distribution G; See File 1.2) Example_Supplier…. for details of G.

[2] G_c is the distribution for the random variable Y in the contingent situation; The expectation in the constraint is taken w.r.t. G_c.

ii) Constrained Optimization II – Probability Constraint

Maximize Expected Profit

subject to that in the contingent situation the probability of (profit being less than a given amount) is less than or equal to a certain threshold.

$$\max_{Q \geq 0} \quad E_g[\Pi(Q,Y)]$$

$$\text{s.t.} \quad P_g(\Pi(Q,Y) \leq \Pi_1) \leq \gamma, \tag{3.5}$$

where

$$\Pi(Q,Y) = \begin{cases} r\xi - c(1-Y)Q - h((1-Y)Q - \xi), & (1-Y)Q \geq \xi, \\ r\xi - c(1-Y)Q - (r+\pi)(\xi - (1-Y)Q), & (1-Y)Q \leq \xi. \end{cases}$$

Remarks:

[1] The expectation for the objective should be taken w.r.t. to G defined in (i) above.

[2] The probability evaluation for the constraint should be taken w.r.t. to G_c for the contingent situation.

iii) Mean-Variance Solution

$$\max_{Q \geq 0} E_G[\Pi(Q,Y)] - \alpha Var_G[\Pi(Q,Y)]$$

or

$$\max_{Q \geq 0} (E_{G_N}[\Pi(Q,Y)] - \alpha Var_{G_N}[\Pi(Q,Y)])P[I=0] + (E_{G_C}[\Pi(Q,Y)] - \alpha Var_{G_C}[\Pi(Q,Y)])P[I=1].$$

where G_N is the distribution for the random variable Y in the typical situation;

iv) Max-Min Solution

$$\max_{Q} \min_{G} E_G[\Pi(Q,Y)].$$

The Max-Min procedure provides a solution which maximizes the expected profit under the worst case scenario. Figures

Remarks: The worse-case of the expected-profit occurs in the contingent situation, where 40% of products shipped are defective. Thus, the optimal total order-quantity $Q_{T,opt}$ is located to maximize this worse-case expected-profit.

Final Remarks:

When a procedure is developed, one needs to explore properties of this procedure and compare them with respect to a few alternative solutions (i.e., optimization models/methods). In exploring these properties, data models' distribution parameter values should be changed to represent various real-world situations. Moreover, several constants involved in the decision-model (e.g., unit-profit, and unit-procurement, - holding, - stockout costs) need to be varied for exploring different situations possibly occurred in real-life supply-chain practices.

The original studies Kim, Lu, Kvam and Tsao (2011) have applied analytical methods and numerical calculations to explore properties of the developed models and optimization methods. The following provides an example of a brief summary of these properties.

[1] The traditional expected-value approaches for product-ordering decisions fail to provide the retailer with sufficient means of protection against the effects of contingency.

[2] In the *first* procedure, by *constraining the maximization problem with respect to conditional expected profit*, a more stable and risk-averse solution can be found. We consider two cases, where contingency either increases the mean of the proportion of damage distribution or increases the variance of the distribution. An increase in variance causes a rapid drop in expected profit, leaving no other alternative to compensate for the profit loss. On the other hand, the more robust methods introduced here compensate for a mean shift of the profit curve, resulting in an increased quantity of the order.

In practice, it is recommended that the retailer investigate the characteristic of potential contingency to see how and if it affects the mean or the variance of Y. If the model implies that the contingency changes only the mean, then the retailer can benefit from the *first* constrained optimization solution. However, if the contingency adversely affects the variance, the decision maker should find a way to reduce the variance, perhaps by using multiple sourcing or purchasing options by which the damage distribution can be truncated.

[3] In the *second* procedure, we utilize the *probability constraint* to restrict possible solutions. With this procedure, the decision-maker has more options to include risk preferences in the solution; one can change either the target profit level or the target probability level to derive risk-averse solutions. This procedure illustrates the *risk-pooling effects* of integrated logistics operations under certain conditions.

Whenever the inventory holding cost, the **shortage cost** and retail prices of products from *different suppliers* are *identical*, **separated logistics** operations between the distribution center and the retailer generate solutions with higher expected profits compared to those of the integrated logistics operation case. But in our examples, we show that the resulting expected profits may be **higher under the integrated logistics operation strategy** than the expected profits under separated logistics operations when ***shortage costs are significantly different***.

ISyE DDA – Agenda for 02/09/23 Lecture (Lec.10)

1. New Materials for Exam-2

i) Decisions in an Uncertain Environment

Note that the decision analytics modeling technique is explained in this topic.

---

1. Decisions in an Uncertain Environment

Details:

This note uses one case study posted in the following directory to summarize decisions and data and models for supporting decision-making in an uncertain situation. Because the note here is only a summary, for more details of the study, please see presentation in the following files posted on Canvas.

Directory: Files> 2.2) Subject-Focused Lecture Notes> Decision 2. Uncertainty Models and Dynamic Optimizations>

Example – Order Quantity Decisions Considering Uncertainty in Supply-Chain Logistics Operations (Kim, Lu, Kvam, Tsao, 2011)

References: Same Directory

i) Summary of Data and Decision Analytics – see 1.2)

Example_Supplier_to-DC-to-Store_Defects.docx

ii) The original publication – see 1.3)

Contract_Logistics_Decision_Uncertainty_IJPE_2011.pdf

Step-0: Problem Background - Figure 1 in Kim *at el.* (2011)

Our goal in this study is to decide the optimal Order Quantities Q_i (i = 1, 2, …, K) for K (=eg= 3) suppliers such that Retailer's profit is maximized. These suppliers S_i (i = 1, 2, …, k) ship the **same product** through a distribution center ("DC" below) to one Retail ("R" notation). See Figure 1 below for a graphical display of this logistic system. Note that there are uncertainties in the shipping process. The uncertainties include defect occurred during the shipping process and delayed schedules for shipments. They are all quantified in "**defects**" from imperfect shipping in our studies below.

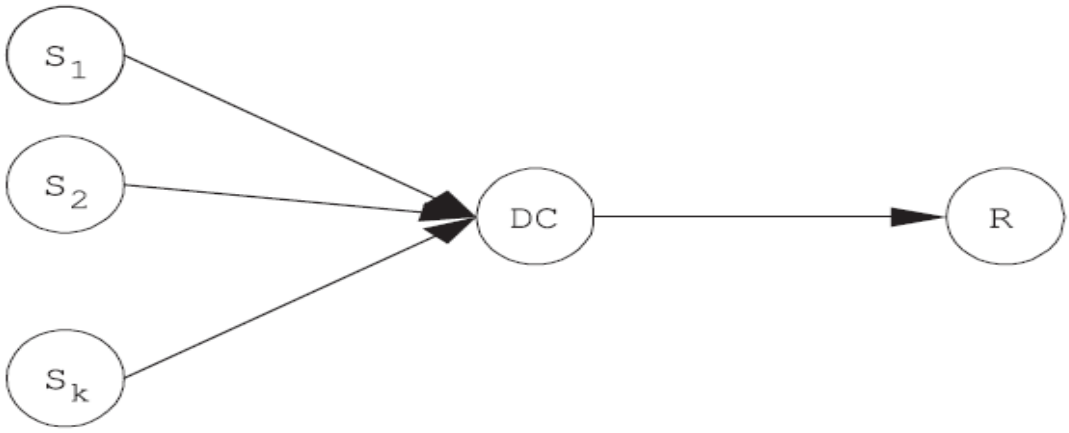


Fig. 1. Supply-chain network with *k* suppliers and one DC.

Step-1: Decision Model

The goal is to locate the optimal order-quantity by maximizing the profit. However, due to uncertainty in the regular or contingent situations, the optimization process is much more involved. See Step-3 for details.

A. Decision Variable: Q = order-quantity

To keep the study simple, we assume that all suppliers have an equal order quantity $Q_i = Q$ for $i = 1, 2, \ldots, K$.

B. Objective Function:

Retailer's Profit =

$\Pi(Q)$ = sales revenue – procurement cost – inventory cost

– "stock-out" cost

$= r$ Min[ξ, $(1 - Y)Q$] – $c(1 - Y)Q$ – $h[(1 - Y)Q - \xi]^+$

– $\pi[\xi - (1 - Y)Q]^+$, **Eq. (2.2)**

where $[A]^+$ represents max[A, 0].

See file Canvas> Files> 2.2) Subject-Focused Lecture Notes> Decision 2. …> 1.3) Contract_Logistics_Decision_Uncertainty_IJPE_2011.pdf for details.

Notations:

[1] r = unit-profit for products sold, ξ = amount of products sold;

[2] Y is the average (random) proportion of defects for all products arrived at the retail store; Thus, $(1 - Y)Q$ = #products (ordered and arrived at the store) without defects;

[3] c = unit-procurement cost, h = unit-holding-cost, and π = unit-stockout-cost, where stock-out means that no product to sell.

Explanations of the Profit (2.2):

[1] The first term is "sales revenue", where r = unit-profit for products sold and ξ = amount of products sold. ***Y* is the proportion of defects for one unit of products shipped from suppliers through DC and to the store.** Thus, $(1 - Y)Q$ = #products-ordered without defects. Sales revenue is unit-profit multiplied by #products sold. But, if $(1 - Y)Q$ is less than #products sold (ξ) due to many shipping defects, unit-profit (r) should be multiplied to $(1 - Y)Q$ instead of ξ.

[2] The second term is the procurement cost, where c = unit-procurement cost counted at the store. Because store manager only pays procurement cost for products arrived at store without defects, the total procurement cost is $c\,(1 - Y)Q$. One could add another cost to incur the transportation cost for defective products from supplier to DC and DC to the store. But, since the store manager will not accept defective products, this additional transportation cost should be charged to the logistics company or to the supplier, but not to the retailer.

[3] Inventory cost will incur for the case that $(1 - Y)Q$ = #products-ordered without defects is larger than or equal to

amount of products sold (ξ). If there is not enough good-products to sell, the last-term, cost of "stock-out", will be incurred. The notations h = unit-holding-cost, and π = unit-stockout-cost.

Step-2: Data Models for Y – "First-Path Presentation"

A. Defect-Data Modeling includes the following **four stages**:

(1) #defects occurred during shipping from a Supplier S_j (j = 1, 2, …, K) to DC;

(2) Aggregation of products at the DC;

(3) #defects occurred during shipping from DC to Store (R);

(4) Total #defects occurred during (1) – (3) above. See detailed discussion of this stage in Layer-2 modeling in file 1.2) Details_Supplier_to-DC-to-Store…..

B. "Contingent" versus "Typical" Situations:

In this problem our focus of the *uncertainty* is that there are two situations for the shipping process: (i) contingent situation and (ii) typical situation. In the contingent situation, the proportion of #defects in the shipped products is large, e.g., 40%, but the chance to face this situation is small, e.g., 0

– 5%. On the other hand, in the typical situation, the proportion of #defects is small, e.g., 5%, but the chance for this to happen is large, e.g., 95%.

The following provides an example of modeling the #defects in the process of **shipping products from a supplier S_j (j = 1, 2, …, K) to DC**. Denoted by X_{jC} and X_{jN} (for j = 1, 2, …, K) the (random) proportion of defective products in a unit-product shipment during the contingent and typical situations, respectively. Define an indicator function I_j as $I_j = 1$ if it is a contingent situation; $I_j = 1$, otherwise. For a supplier S_j the proportion of defects for a unit of products is then

$$P_{jDC} = (1 - I_j)*X_{jN} + I_j*X_{jC}, \quad j = 1, 2, \ldots, K, \qquad (1)$$

which is a random variable consisting of two cases with their own corresponding proportions of defects. Discussion of the distributions of these *three* random variables are skipped here. See file 1.2) Example_Supplier_to-DC-to-Store….. for details.

Remark: There could be more uncertain situations than our example above with two situations (one is the contingent situation and the other is the typical situation). For example, in our defrosting projects, one can see that when

the weather is "typically" nice, the proportion of planes met the "40-minute-rule" is above 90%. On the other hand, when the weather is not that bad, but still not the best, the proportion of planes met the "40-minute-rule" could be between 80 to 90%. Finally, when the weather is very bad, the proportion of planes met the "40-minute-rule" is below 80%. Of course, one can look into more details about the "very bad weather" situations to classify it into more "contingent situations".

The following provide details of probability-based Y-data modeling for the *four stages* (discussed in page 6 above) from suppliers to the retail store.

1> Stage-1: Shipping from Supplier S_j (j = 1, 2, …, K) to DC:

Denoted by X_{jC} and X_{jN} (for j = 1, 2, …, K) the proportion of defective products in a unit-product shipment during the *contingent* and *typical situations*, respectively. They are both random-variables with distributions G_C and G_N, respectively.

Define an indicator function I_j as $I_j = 1$ if it is a contingent situation; $I_j = 0$, otherwise. The distribution of I_j is a Bernoulli with $\Pr(I_j = 1) = p$ and $\Pr(I_j = 0) = 1 - p$. Assume that the three random-variables, X_{jC}, X_{jN} and I_j are mutually independent. For a supplier S_j **the proportion of defects** for a unit of products is then

$$P_{jDC} = (1 - I_j)*X_{jN} + I_j*X_{jC}, \quad j = 1, 2, \ldots, K, \qquad (1)$$

which is a random variable with a mixture distribution of two cases with their own corresponding proportions of defects.

2> Stage-2: Aggregation of Products at DC - "Average-Aggregation":

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

Because for each supplier there are $Q_{Sj}*P_{jDC}$ #defects among Q_{Sj} #ordered-products, the **average-aggregated proportion of defects from all suppliers** is

$$P_{au,DC} = \Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC}) / Q, \text{ where } Q = \Sigma_{j=1}^{K} Q_{Sj}. \quad (2)$$

Here, the numerator is the total number of defects from all suppliers' products, and the denominator is the total number of ordered-products.

(2) Evenly-split Ordered-Quantity $Q_{Sj} = Q/K$ for $j = 1, 2, \ldots, K$ – this is our focus.

This option is a special case of (1) unevenly-split ordered-quantity above. Apply the same formula with $Q_{Sj} = Q/K$ to get

$$P_{ae,DC} = \Sigma_{j=1}^{K} [(Q/K)*P_{jDC})] / Q = (\Sigma_{j=1}^{K} P_{jDC}) / K,$$

which is a simple average of proportion of defects from all suppliers'. Notice the index "ae" in the Proportion "*P*" for this case.

3> **Stage-3: Shipping from DC to Store:**

Our goal is to formulate **Y* ≡ the *average proportion of defects* for one unit of products shipped from suppliers through DC**

and to the store, which was the goal of the defect-data-modeling studies.

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

Similar to the definitions of the three random variables (1) above, for **shipping from DC to store** we have X_C^* and X_N^* (for $j = 1, 2, \ldots, K$) the proportion of defective products in a unit-product shipment during the contingent and typical situations, respectively. The indicator function I_0 is defined as $I_0 = 1$ if it is a contingent situation; $I_0 = 1$, otherwise. Then, the proportion (P_R) of defects from **shipping between DC and store** is

$$P_R = (1 - I_0)*X_N^* + I_0*X_C^*. \qquad (3)$$

Note that there are $\Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC})$ amount of defects out of Q total number of products-ordered (and shipped from suppliers). Assume that there is NO INSPECTION at the DC for taking out defective products. There are

$$Q - \Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC})$$

$$= Q - Q*P_{au,DC} = Q*(1 - P_{au,DC}) \qquad (4)$$

amount of non-defective products shipped from DC to store. See Eq. (2) for understanding the first equality above.

Since the Q*(1 − $P_{au,DC}$) amount of non-defective products shipped from DC to store will subject to further shipping damage (with the probability P_R defined in Eq.(3)), the #defects for non-defective products shipped from DC and arrived at the store is

Q*(1 − $P_{au,DC}$) * P_R.

Note that there are only Q*(1 − $P_{au,DC}$) * (1 − P_R) amount of non-defective products arrived at the store eventually.

(2) Evenly-split Ordered-Quantity Q_{Sj} = Q/*K* for *j* = 1, 2, …, *K*:

This option is a special case of (1) unevenly-split ordered-quantity above. Applying the same ideas as given above by replacing $P_{au,DC}$ with $P_{ae,DC}$ for this evenly-split case, we have the following #defects for

non-defective products shipped from DC and arrived at the store:

$Q^*(1 - P_{ae,DC}) * P_R$.

4> Stage-4: Calculate total #defects occurred at all stages.

From supplier's (or logistics service provider's) point of view, the total #defects (and its proportion) occurred at Stage (1) and (3)'s shipping processes is important for calculating the cost of product-supply. From retailer's (store manager's) point of view, the #good-products arrived at the store and also the proportion (#good-products arrived at the store / #total-ordered-products) is important for supporting their sales (to meet customer demands).

(1) Unevenly-split Ordered-Quantity Q_{Sj} for $j = 1, 2, \ldots, K$:

There are $\Sigma_{j=1}^{K} (Q_{Sj}*P_{jDC})$ amount of defects in Stage-1. Among $Q^*(1 - P_{au,DC})$ good-products (see Eq.(4))

shipped in Stage-3, there are Q*(1 − $P_{au,DC}$) * P_R amount of defects. Thus, the total #defects is

$$\Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC}) + Q*(1 - P_{au,DC}) * P_R.$$

If one is interested in the **average-proportion of defects in all stages**, it is then

$$Y = [\ \Sigma_{j=1}^{K} (Q_{Sj} * P_{jDC}) + Q*(1 - P_{au,DC}) * P_R\ ] / Q.$$

(2) Evenly-split Ordered-Quantity Q_{Sj} = Q/*K* for *j* = 1, 2, …, *K*:

Again, with Q_{Sj} = Q/*K* the above results in (1) becomes

$$Y = [\ \Sigma_{j=1}^{K} [(Q/K) * P_{jDC}] + Q*(1 - P_{au,DC}) * P_R\ ] / Q$$

$$= (\Sigma_{j=1}^{K} P_{jDC})/K + (1 - P_{ae,DC}) * P_R.$$

Since $P_{ae,DC} = (\ \Sigma_{j=1}^{K} P_{jDC}\ ) / K$, the above formula becomes

$$Y = P_{ae,DC} + (1 - P_{ae,DC}) * P_R$$

$$= P_{ae,DC}\ (1 - P_R) + P_R$$

$$= [(\ \Sigma_{j=1}^{K} P_{jDC}\ ) / K\ ] * (1 - P_R) + P_R,$$

which is the average of proportion of defects for all products shipped from *K*-suppliers to store. This is the needed Y in the objective function (2.2).

Step-3: Optimization-Solution Procedures

1) **Risk Neutral: Find Q_{opt} to maximize Expected Profit.**

2) **Risk Averse:**

   i) **Constrained Optimization I – Profit Constraint**

   Maximize the Expected Profit

   subject to that the expected profit in the contingent situation is over a certain threshold. That is,

$$\max_{Q \geq 0} \; E_G[\Pi(Q,Y)]$$

$$\text{s.t.} \quad E_{G_C}[\Pi(Q,Y)] \equiv E_G[\Pi(Q,Y)|I=1] \geq \Pi_0, \qquad (3.4)$$

   where I is the indicator function for a contingency.

Notations:

[1] G is the distribution for the random variable Y (= average proportion of defects for products arrived at the store). G has the mixture distribution mixing from distributions in both contingent and typical situations; The expectation in the main objective is taken with respect to (w.r.t.) the distribution G; See File 1.2) Example_Supplier…. for details of G.

[2] G_c is the distribution for the random variable Y in the contingent situation; The expectation in the constraint is taken w.r.t. G_c.

ii) Constrained Optimization II – Probability Constraint

Maximize Expected Profit

subject to that in the contingent situation the probability of (profit being less than a given amount) is less than or equal to a certain threshold.

$$\max_{Q \geq 0} \; E_g[\Pi(Q,Y)]$$

$$\text{s.t.} \;\; P_g(\Pi(Q,Y) \leq \Pi_1) \leq \gamma, \tag{3.5}$$

where

$$\Pi(Q,Y) = \begin{cases} r\xi - c(1-Y)Q - h((1-Y)Q - \xi), & (1-Y)Q \geq \xi, \\ r\xi - c(1-Y)Q - (r+\pi)(\xi - (1-Y)Q), & (1-Y)Q \leq \xi. \end{cases}$$

Remarks:

[1] The expectation for the objective should be taken w.r.t. to G defined in (i) above.

[2] The probability evaluation for the constraint should be taken w.r.t. to G_c for the contingent situation.

iii) Mean-Variance Solution

$$\max_{Q \geq 0} E_G[\Pi(Q,Y)] - \alpha Var_G[\Pi(Q,Y)]$$

or

$$\max_{Q \geq 0} (E_{G_N}[\Pi(Q,Y)] - \alpha Var_{G_N}[\Pi(Q,Y)])P[I=0] + (E_{G_C}[\Pi(Q,Y)] - \alpha Var_{G_C}[\Pi(Q,Y)])P[I=1].$$

where G_N is the distribution for the random variable Y in the typical situation;

iv) Max-Min Solution

$$\max_{Q} \min_{G} E_G[\Pi(Q,Y)].$$

The Max-Min procedure provides a solution which maximizes the expected profit under the worst case scenario. Figures

Remarks: The worse-case of the expected-profit occurs in the contingent situation, where 40% of products shipped are defective. Thus, the optimal order-quantity Q_{opt} is located to maximize this worse-case expected-profit.