Stony Brook University

Exploration of statistical learning strategies and their applications on medical image data for computer-aided diagnosis

A Dissertation Presented

by

Yifan Hu

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

Jan 2017

Stony Brook University
The Graduate School

Yifan Hu

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Wei Zhu, PhD – Dissertation Advisor
Professor of Applied Mathematics and Statistics

Song Wu, PhD - Chairperson of Defense
Associate Professor of Applied Mathematics and Statistics

Yi Gao, PhD - Committee Member
Assistant Professor of Biomedical Informatics

Keli Xiao, PhD - Committee Member
Assistant Professor of Finance

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Exploration of statistical learning strategies and their applications on medical image data for computer-aided diagnosis

by

Yifan Hu

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2017

Machine learning addresses the question of how computer make decisions and predictions automatically through existing experiences and data, which has become an increasingly important topic with the advent of modern data science and automated big data analysis. Several algorithms are widely used in machine learning. However, each classifier, inevitably, has certain inadequacy for which we hope to compensate. To address these issues, this study first introduces the necessary theoretical background and principles for machine learning and those typical classifiers. Based on these classifiers, this paper attempts to (1) use bagging/boosting to improve the simple classifier, and, (2) find some combination strategies to make use of the advantage of each classifier. The second part of this paper is to verify the robustness of these innovative ideas via multiple datasets. First, several common datasets are analyzed with the results compared between our new algorithm and those typical classifiers. Overall, we can obtain some gains in terms of the AUC value in virtually every dataset with the new algorithm and significant gains in most dataset. Secondly, we apply these algorithms to a real-life image data classification problem. The pipeline of this project includes 3D texture feature amplification, feature extraction via KL-transform, feature selection and classification. Finally, we gladly report that significant improvements have been achieved through both the new feature selection method and the new classification algorithm.

To my family

Contents

List of Figures

List of Tables

List of Acronyms

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
AUC	area under the curve
C4.5	a software extension
CADe	computer-aided detection
CADx	computer-aided diagnosis
CART	classification and regression tree
CHAID	Chi-square automatic interaction detector
CM	co-occurrence matrix
CRC	colorectal carcinoma
CTC	computed tomography colonography
FN	false negative
FP	false positive
FPR	false positive rate
ID3	iterative dichotomiser 3
KKT condition	Karush–Kuhn–Tucker conditions
KL-transform	Karhunen-Loeve Transform
KNN	K nearest neighbor
LDA	linear discriminant analysis
LI	location index
LIRF	location index embedded random forest
MARS	multivariate adaptive regression splines
OC	optical colonoscopy
OOB error	out-of-bag error
RBF	radial basis function
ROC	receiver operating characteristic
SVM	support vector machine
TN	true negative
TP	true positive
TPR	true positive rate
VOI	volume of interest
wKNN	weighted K nearest neighbor

Acknowledgements

First of all, I would like to thank my dissertation adviser, Prof. Wei Zhu, for his patient guidance, brilliant ideas, and financial support during my PhD study. Indeed, this dissertation would never have been completed without her enthusiasm, dedication and support. I would also like to thank my dissertation committee – Prof. Song Wu, Prof. Yi Gao, and Prof. Keli Xiao for their precious time and valuable suggestions.

I would also like to thank current and previous members of the Laboratory for Imaging Research and Informatics (IRIS lab), especially for Prof. Jerome Liang and his software packages on computer-aided diagnosis; Dr. Bowen Song, Dr. Yan Liu, Dr. Hao Zhang, Dr. Hao Han and Dr. Lihong Li, for their helpful discussions on my research work; Priya Bhattacharji for her assistance on clinical data management. Finally, I would like to thank my parents, other family members and friends, for their love, encouragement and support in my life.

This work was supported in part by the NIH/NCI under grant #CA143111 and #CA082402.

Chapter 1 . Backgrounds and typical classification algorithms

Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, *machine learning* (T. Mitchell 1997, C. Bishop 2006, I. Goodfellow et al. 2016) is defined as a "field of study that gives computers the ability to learn without being explicitly programmed". The main task of machine learning is to develop the algorithms that can learn from and make predictions on data. In order to make data-driven predictions or decisions expressed as outputs, machine learning algorithms will build a model from a sample of input observations generally referred to as the training set.

Based on the nature of the learning "signal" or "feedback" available to a learning system, we can typically divide the machine learning problem into two main categories: *supervised learning* and *unsupervised learning*. The most fundamental difference between these two categories is that the supervised learning is to infer a function that maps features to labels given labeled data while the unsupervised learning is to infer a function to describe hidden structure from unlabeled data. In this thesis, we will focus on the research of the classification problem, which is supervised learning problem.

1.1 Classification and classifier

Based on a set of labeled training examples, the classification problem aims to classify new observations into a given set of categories (labels). An observation/data includes two parts: a *label*/category in $\mathbb{R}^1$ and a *feature vector* in $\mathbb{R}^n$. The algorithm, which can map the input feature vector into the output categories of new observations, is considered as "*classifier*". The classifier could be a mathematical function, or some sophisticated data structure, like the decision tree. Mostly, we can just consider the model (classifier) as $f: \mathbb{R}^n \mapsto \{1, \dots, k\}$ and sometimes the output can also be a distribution over the classes. When k=2, we call the problem binary classification and when k>2, we call the problem multiclass classification. My research mainly focuses on the binary classification.

In many applications, the classification algorithms often achieve a much more accurate result than human rules and experiences because people are incapable of grasping complicated relationships and explaining what they know while the classification models can easily do so. Thus, classification algorithms have become increasingly popular in many fields such as:

- Medical image analysis (e.g., pathology diagnosis of the cancer)
- Text categorization (e.g., spam filtering)
- Computer vision (e.g., face detection)
- Natural-language processing (e.g., spoken language understanding)
- Bioinformatics (e.g., classify proteins according to their function)
- Market segmentation (e.g., predict if customer will respond to promotion)

We can see from the applications above that the classification algorithms help us make a prediction or decision about the new cases/observations/test data. Therefore, the primary goal of a classification algorithm is to achieve a higher and more accurate prediction results on test data. In the beginning, the measures precision and recall are popular metrics used to evaluate the quality of a classification algorithm. But more recently, Receiver operating characteristics (ROC) curves have been widely used to evaluate the tradeoff between true positives (TP) and false positives (FP) rates of classification algorithms. In this thesis, we only consider the ROC curves and the Area Under the Curve (AUC) values as the evaluation measures, which will be briefly introduced in the next chapter.

Another interesting property of machine learning is the so called "no free lunch" theorem (D. Wolpert 1997). This theorem states that there is no one machine learning model that performs best for every problem. Therefore, for each particular data set or problem, we should try multiple models and choose the one works best on it. Furthermore, it is also important to consider the trade-offs between speed, complexity and accuracy of different models and find a proper one for your data set. I list many classical and widely used classification algorithms in the following:

Linear classifiers

  - Logistic regression
  - LDA
  - Na ïve Bayes classifier
- SVM
- KNN
- Decision trees
  - Random forest
- Neural networks
  - Deep learning

Details for these algorithms will be illustrated in Chapter 2.

1.2 Related topics of classification: overfitting and under fitting

The key idea in machine learning and classification is that the model we fit by the training set must perform well on the test set, which should not contribute in building the model. Therefore, this model should have the ability of "generalization". That is to say, in a good machine learning

and classification model, we not only want to make the training error small, but also hope the performance on the test set can be similar to that of the training set. That is, the gap between the training error and the test error is small. Large training and/or test error would lead to "*underfitting*" and "*overfitting*". Underfitting means that your fitted model is not so good that results in a high training error. Overfitting means that although you fit your training set very well, the gap between the training and test errors is too large. Below is an example for underfitting and overfitting in Figure 1.1.

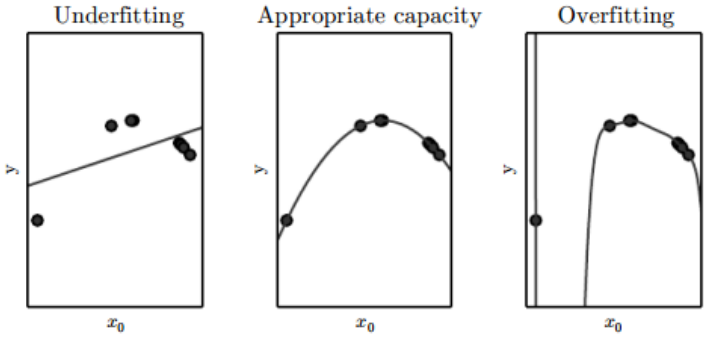


Figure 1.1. Illustration of overfitting and underfitting of a regression example

In this regression example we can see that both the fit by 2-degree and 10-degree function for the training set (9 points) have small training errors, however, the test error for the fit by 10-degree function is huge, where the overfitting happens. Generally, when it comes to the relation between model complexity and errors, the training and test errors behave differently as shown in Figure 1.2:

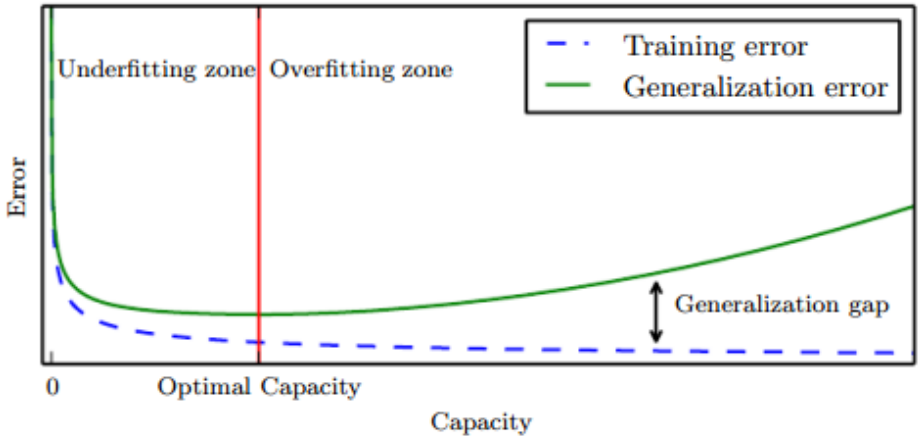


Figure 1.2. The gap between training and test errors as the complexity of the model changes.

Therefore, how to choose a model with proper complexity becomes an important topic in machine learning and classification algorithm. One popular method in classification algorithm is known as *regularization*, which will be discussed in the next section.

1.3 Related topics of classification: cost function, regularization and parameter estimation

Before we talk about the idea of regularization, it is proper to firstly introduce the concept of *cost function* in machine learning and classification algorithms. Cost function, sometimes also called loss function, is to evaluate the difference between your prediction and the truth. Thus, we will always encounter an optimization problem that seeks to minimize a cost function. In the fitting example of Figure 1, we can express the cost function as $\mathrm{J} = \sum_{i=1}^{9}(\hat{f}(x_i) - y_i)^2$. Then when we fit the model by a polynomial function with degree 1, 2 and 9, respectively, the parameters for the polynomial function will be easy to calculate by setting the partial derivatives of J with respect to these parameters to be 0. After developing the models, we can calculate the training and testing errors for each model and find that the overfitting occurs when we fit the model with 9-degree polynomial function. Actually, the Occam's razor reveals that among competing hypotheses, the one with the fewest assumptions should be selected. In the fitting example, we notice that the quadratic fit and the 9-degree fit have the similar performance on the training set. Thus, the Occam's razor indicates that the quadratic model should be chosen as our model. Naturally, we wonder if there is a strategy to help us avoid the overly complex model and choose the simpler model automatically. The answer lies in the method of **regularization**.

Simply, we can interpret the regularization as a way to penalize the complexity of our model. For example, we set the penalized term called regularizer equals to $C * \omega^T\omega$, where ω is the parameter of the polynomial function and C is a constant, and the cost function becomes $\mathrm{J}(\omega) = \sum_{i=1}^{9}\left(\hat{f}(x_i) - y_i\right)^2 + C * \omega^T\omega$. This example only shows us the L-2 norm regularizer, but there are also many other types of regularizers applied in different algorithms such as the L-1 norm and the L-inf norm regularizers. It is obvious that the larger the C we choose to construct the cost function, the simpler the model we hope to develop. Actually, the penalized cost function does not help us to achieve the optimized (lowest) training error, however, the added penalized term provides us a way to find a model with proper complexity that makes its generalization error low and thus avoid the overfitting.

For a given C, we can solve the optimization problem $\min \mathrm{J}(\omega)$ and figure out the parameter ω. So, it is important to decide how to figure out a proper C because a higher C might lead to an underfitting model with higher training error while a lower C will cause the overfitting problem. Since this parameter C helps us adjust the generalization error, *cross-validation* (S. Arlot 2010) is a common and useful strategy to solve this problem. Since we have all the information of the training set, cross-validation could help us estimate how accurately a predictive model will perform and generalize to an independent dataset in the following strategies:

1. Leave-p-out cross-validation: divide the (original) training set into a validation set of p observations and the rest as the (cross-validation) training set. Under this strategy, it requires to learn and validate C(n,p) times, where n is the size of the (original) training set. For each constructed model, you can evaluate the performance and eventually choose the optimal model with lowest prediction error.
   Special case: Leave-one-out cross-validation.
2. K-fold cross-validation: divide the (original) training set into k equal sized subsamples, each time we consider 1 subsample as the validation set and k-1 subsamples as the (cross-validation) training set. Under this strategy, then it takes k times to learn and validate and find the optimal model.

In short, since the most important goal for a good classification algorithm is the prediction accuracy, we need some tools like regularization and cross-validation help us build a model with good generalization property instead of only building a perfect model on the training set.

1.4 Related topics of classification: bias-variance tradeoff (S. Vijayakumar 2007)

As we mentioned above, the true model between the observation and the label can be interpreted as a functional and noisy relation: $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\varepsilon}$, where the noise $\boldsymbol{\varepsilon}$ has mean *0* and variance σ^2. And our task is to fit a machine learning or classification model $\hat{f}(\boldsymbol{x})$ as well as possible. Thus, the mean squared error between the truth and the prediction needs be as small as possible, that is to say, our target is a model that minimizes the following function: $\mathrm{E}[(\boldsymbol{y} - \hat{f}(\boldsymbol{x}))^2]$.

This function can be decomposed as follows:

$$
\begin{aligned}
\mathrm{E}\left[\left(\boldsymbol{y} - \hat{f}(\boldsymbol{x})\right)^2\right] &= \mathrm{E}[\boldsymbol{y}^2] + \mathrm{E}\left[\hat{f}(\boldsymbol{x})^2\right] - 2\mathrm{E}\left[\boldsymbol{y}\hat{f}(\boldsymbol{x})\right] \\
&= (\mathrm{Var}(\boldsymbol{y}) + (\mathrm{E}[\boldsymbol{y}])^2) + \left(\mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(\mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2\right) - 2\mathrm{E}\left[\boldsymbol{y}\hat{f}(\boldsymbol{x})\right] \\
&\triangleq \mathrm{Var}(\boldsymbol{y}) + \mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(f(\boldsymbol{x}) - \mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2 \\
&= \mathrm{E}[(\boldsymbol{y} - \mathrm{E}[\boldsymbol{y}])^2] + \mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(\mathrm{Bias}\left(\hat{f}(\boldsymbol{x})\right)\right)^2 \\
&= \mathrm{E}\left[\left(\boldsymbol{y} - f(\boldsymbol{x})\right)^2\right] + \mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(\mathrm{Bias}\left(\hat{f}(\boldsymbol{x})\right)\right)^2 \\
&= \mathrm{E}[\boldsymbol{\varepsilon}^2] + \mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(\mathrm{Bias}\left(\hat{f}(\boldsymbol{x})\right)\right)^2 \\
&= \mathrm{Var}(\boldsymbol{\varepsilon}) + (E[\boldsymbol{\varepsilon}])^2 + \mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(\mathrm{Bias}\left(\hat{f}(\boldsymbol{x})\right)\right)^2 \\
&= \sigma^2 + \mathrm{Var}\left(\hat{f}(\boldsymbol{x})\right) + \left(\mathrm{Bias}\left(\hat{f}(\boldsymbol{x})\right)\right)^2
\end{aligned}
$$

Where $f(\boldsymbol{x})$ is deterministic so $\mathrm{E}[\boldsymbol{y}] = f(\boldsymbol{x})$ and $\boldsymbol{y}$ and $\hat{f}(\boldsymbol{x})$ are independent, then $\triangleq$ is because

$$(\mathrm{E}[\boldsymbol{y}])^2 + \left(\mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2 - 2\mathrm{E}\left[\boldsymbol{y}\hat{f}(\boldsymbol{x})\right] = (\mathrm{E}[\boldsymbol{y}])^2 + \left(\mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2 - 2\mathrm{E}[\boldsymbol{y}]\mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]$$
$$= \left(\mathrm{E}[\boldsymbol{y}] - \mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2 = \left(f(\boldsymbol{x}) - \mathrm{E}\left[\hat{f}(\boldsymbol{x})\right]\right)^2$$

Thus, the structure of the mean squared error is the combination of bias of the model, variance of the model and the irreducible error. Generally, a complex model might have low bias but high variance on the data set while a simple model has high bias but low variance.

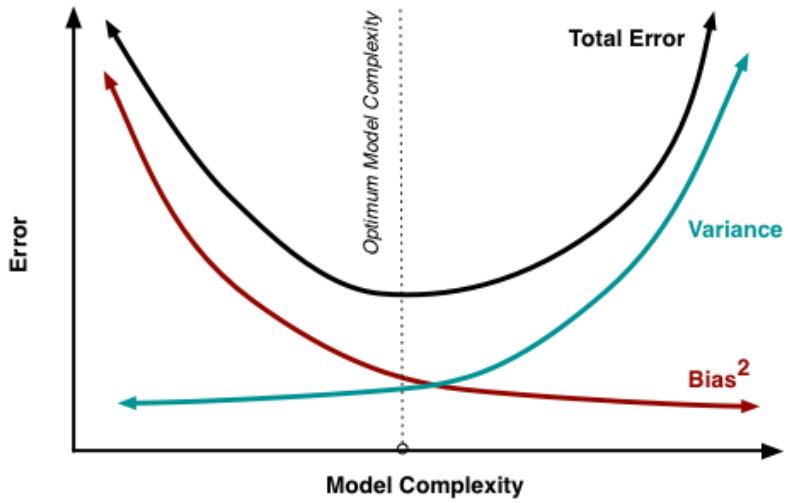


Figure 1.3. The illustration of bias-variance tradeoff when the model complexity changed.

From this figure 1.3 above, it is clear that there is a tradeoff between Bias^2 and variance since we are required to achieve the lowest (Bias^2 + variance).

In this thesis, I will introduce one smart way to reduce the variance without increase the bias, which is the bagging (Bootstrap Aggregation) method proposed by Leo Breiman (L. Breiman 1994) and applied it on his famous work——random forest.

The idea of the bagging method is as follows:

Suppose we fit a model $\hat{f}(\boldsymbol{x})$ on training set and fit another n uncorrelated models: $\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_n(\boldsymbol{x})$ such that $\hat{f}_i(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x})$, then we have $\mathrm{Var}\left(\frac{\sum_{i=1}^{n}\hat{f}_i(\boldsymbol{x})}{n}\right) = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(\hat{f}_i(\boldsymbol{x})) \approx \frac{1}{n}\mathrm{Var}(\hat{f}(\boldsymbol{x}))$.

However, we only have one training set and need built multiple models from it. Breiman provided us the solution that we can use the bootstrap sample to build a model each time. A bootstrap sample of size k is approached by picking one sample from a data set with replacement for k times. In this way, we can build $\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_n(\boldsymbol{x})$ by n bootstrap models and take the average of them to be the final model. Although the reduction factor of variance from $\mathrm{Var}(\hat{f}(\boldsymbol{x}))$ will

definitely larger than $\frac{1}{n}$ because these models are correlated, we can still get a lower variance compare with the origin model $\hat{f}(\boldsymbol{x})$.

Chapter 2 . Brief introduction to some typical classifiers and their ideas

In this section, I will introduce and analyze 3 kinds of classifiers with totally different properties: SVM (V. Vapnik and A. Chervonenkis 1963), RF (L. Breiman 1994) and KNN(T. Cover and P. Hart 1967).

2.1 SVM

A. Linear classifier in separable case

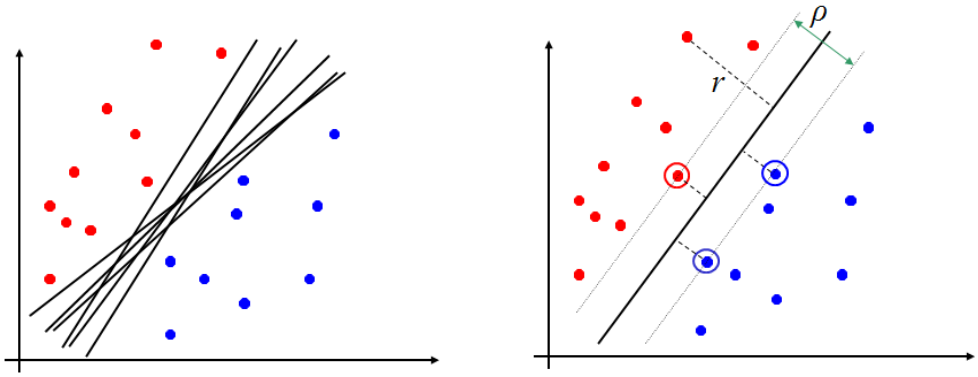
For a binary classification problem, we are asked to learn a classifier such that

$$f(\boldsymbol{x}_i) = \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases} \tag{2.1}$$

for a given training data $(\boldsymbol{x}_i, y_i)$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$, and $f(\boldsymbol{x}) = 0$ is the hyperplane that separates those two classes.

Let us first consider the simplest case: linear separator trained on separable data (and the general case is the nonlinear separator on non-separable data).

In this case, we need to find a linear function $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ to separate the data into two classes: $\boldsymbol{y} = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b)$ which indicates that $\boldsymbol{y} = +\mathbf{1}\ \boldsymbol{or}\ -\mathbf{1}$.



(a) (b)

Figure 2.1. (a) data can be separated by many hyperplanes (b) the optimal hyperplane provided by SVM

However, from figure 2.1(a), it is easy to see that there are many hyperplanes (in 2-D space, the hyperplane is 1-D lines) that can classify those points correctly so we must decide an optimal hyperplane among them.

For a hyperplane $\boldsymbol{w^T x} + b = \mathbf{0}$, the distance between a data point and the hyperplane is $r = \frac{|\boldsymbol{w^T x_i} + b|}{\|\boldsymbol{w}\|}$. Thus, here we define the support vectors to be the data point that is closest to the hyperplane (for example the circled points in figure 2.1(b)) and the margin ρ is the sum of the two distances (two classes) between the support vector to the hyperplane. So the basic idea of SVM is to find a hyperplane that can maximize the margin.

From Figure 2.1(b), suppose all data points satisfy:

$$\begin{cases} \boldsymbol{w^T x_i} + b \leq -1, \ if \ y_i = -1 \\ \boldsymbol{w^T x_i} + b \geq 1, \ if \ y_i = 1 \end{cases} \tag{2.2}$$

which is equivalent to $y_i(\boldsymbol{w^T x_i} + b) - 1 \geq 0$ and all support vectors are located on $\boldsymbol{w^T x_i} + b = \pm 1$. (Note: It supposes to be $y_i(\boldsymbol{w^T x_i} + b) - \frac{\rho}{2} \geq 0$, but both sides of the inequality can be divided by $\frac{\rho}{2}$ and make it to be $y_i(\boldsymbol{w^T x_i} + b) - 1 \geq 0$).

Then the margin equals to $2r = 2\frac{|\boldsymbol{w^T x_i} + b|}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$. Since we hope the margin to be as large as possible, we can build the cost function and the optimization problem becomes:

Find $\boldsymbol{w}, b$ such that $\Psi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w^T w}$ is minimized, given $y_i(\boldsymbol{w^T x_i} + b) - 1 \geq 0$ satisfies on all points $(\boldsymbol{x}_i, y_i)$.

In order to solve this optimization problem, we use the generalized Lagrange multipliers:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w^T w} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i(y_i(\boldsymbol{w^T x_i} + b) - 1) \tag{2.3}$$

and the problem becomes:

$$p^* = \min_{\boldsymbol{w}} \Psi(\boldsymbol{w}) = \min_{\boldsymbol{w},b,\alpha} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha_i} \geq 0} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) \tag{2.4}$$

(notice (2.4), please see the reference (K. Bennett 2000, A. Ng's lecture notes) for more details and here is a corollary that only non-zero α_i correspond to support vector $\boldsymbol{x_i}$, otherwise $\min_{\boldsymbol{w}} \Psi(\boldsymbol{w}) \neq \min_{\boldsymbol{w},b,\alpha} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$)

Now we consider the dual problem of this problem:

$$d^* = \max_{\boldsymbol{\alpha_i} \geq 0} \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$$

and we know that when all of the KKT condition (H. Kuhn and A. Tucker 1951) are satisfied, we can have $p^* = d^*$. Then let's find the dual form of the problem:

$$\begin{cases} \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i \boldsymbol{x_i} = \boldsymbol{0} \\ \nabla_b \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i = 0 \end{cases} \Rightarrow \begin{cases} \boldsymbol{w} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i \boldsymbol{x_i} \\ \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i = 0 \end{cases} \tag{2.5}$$

Plug this result back to (2.3), we get the dual problem:

$$\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i - \frac{1}{2} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j} \tag{2.6}$$

where $\alpha_i \geq 0, i = 1, \dots, n$ and $\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i = 0$

After we test the KKT condition, we can solve this dual problem as our result (with α_i we can get the $\boldsymbol{w}$ by equation (2.5) and $b = y_i - \boldsymbol{w^T x_i}$ for any $\alpha_i > 0$).

B. Model complexity and Kernel function

From equation (2.5), we notice that the solution of α_i will depend on the features ($\boldsymbol{x_i}$) of each data. In the example in last section, we have a lucky case that we achieve a hyperplane to separate the 2D data. Unfortunately, in most problems it is so hard that we cannot find such a hyperplane to separate the data perfectly.

For example: try to find a hyperplane of the following 10 points:

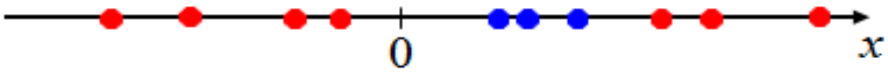


Figure 2.2. A simple 1D case that we cannot find a threshold to classify the data into 2 classes.

The reason why we fail to classify the 10 points is that we only have one feature and then we can only build a very simple model based on it. In the regression example in Chapter 1, we can get a higher accuracy with fitting a higher polynomial function (a more complex model). Likewise, here we introduce the kernel function in SVM to enhance our model complexity.

Let's first consider an example correspond to Figure 2.2:

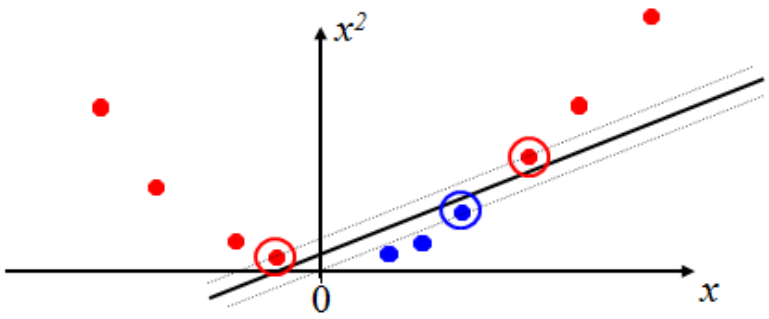


Figure 2.3. A separable case by mapping the 1D data into 2D space.

In this example, we can easily find the hyperplane by mapping the 1D data into a higher feature space with function $\Phi(x) = (x, x^2)$. Therefore, we may want to learn from some features $\Phi(x)$ instead of applying SVM on the original input features. To do so, we can simply replace $\boldsymbol{x}$ in our previous algorithm with $\boldsymbol{\Phi(x)}$ and solve the following problem:

$$\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i - \frac{1}{2} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i \alpha_j y_i y_j \boldsymbol{\Phi(x_i)} \cdot \boldsymbol{\Phi(x_j)} \tag{2.7}$$

where $\alpha_i \geq 0, i = 1, \ldots, n$ and $\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i = 0$

Since in the dual problem we only consider the inner product $\langle \boldsymbol{x_i}, \boldsymbol{x_j} \rangle$ and $\langle \boldsymbol{\Phi(x_i)}, \boldsymbol{\Phi(x_j)} \rangle$, we can define the corresponding Kernel function (B. Boser et al. 1992) to be: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{\Phi(x_i)} \cdot \boldsymbol{\Phi(x_j)}$

In practice, computing the high-dimensional $\boldsymbol{\Phi(x)}$ consume too many time and we usually compute $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ directly instead, which takes only O(n) time. Thus, we hope to find a function $K: \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ which can be written as $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{\Phi(x_i)} \cdot \boldsymbol{\Phi(x_j)}$. Actually, if the function K satisfy the Mercer's condition (J. Mercer 1909), then $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ would be a valid kernel such that $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{\Phi(x_i)} \cdot \boldsymbol{\Phi(x_j)}$.

Here are some famous and widely used kernels in SVM as follows:

- Linear: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i}^T \boldsymbol{x_j}$ where $\boldsymbol{x_i} \longrightarrow \boldsymbol{\Phi(x_i)}$ and $\boldsymbol{\Phi(x_i)}$ is $\boldsymbol{x_i}$ itself.
- Polynomial of power p: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (1 + \boldsymbol{x_i}^T \boldsymbol{x_j})^p$, where $\boldsymbol{\Phi(x_i)}$ has $\binom{n+p}{p}$ dimensions.
- Radial basis function (RBF, Gaussian): $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\gamma \|x_i - x_j\|^2}$, where $\boldsymbol{\Phi(x_i)}$ has infinite dimensions (due to the Taylor's expansion of $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ has infinite terms)
- Other functions: sigmoid: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \tanh(\kappa \boldsymbol{x_i}^T \boldsymbol{x_j} - \delta)$, …

In this thesis, we will use the RBF kernel since the RBF kernel can map the data into infinite dimension and choose a proper complexity by adjusting the parameters, which is able to adapt different data sets.

C. Non-separable cases and regularization——soft margin classification (C. Cortes and V. Vapnik 1995)

Since RBF kernel can map your data into an infinite feature space, you can train a very complex and accurate model on your training data. However, we still need to consider about the overfitting problem here.

For most data set, even after applying the kernel mapping, the constraints for all the data:

$$\begin{cases} \boldsymbol{w^T x_i} + b \le -1, \ if\ y_i = -1 \\ \boldsymbol{w^T x_i} + b \ge 1, \ if\ y_i = 1 \end{cases} \tag{2.8}$$

is still too strict, that is to say, we have to undertake the risk of overfitting if we try to develop a perfect model. It will be sure that a set of "complex" data corresponds to a complex perfect fitting model. Thus, we can relax the constraints by introducing slack variables $\xi_i \ge 0$, $i = 1, \dots, n$, and the constraints become:

$$\begin{cases} \boldsymbol{w^T x_i} + b \le -1 + \xi_i, \ if\ y_i = -1 \\ \boldsymbol{w^T x_i} + b \ge 1 - \xi_i, \ if\ y_i = 1 \end{cases} \tag{2.9}$$

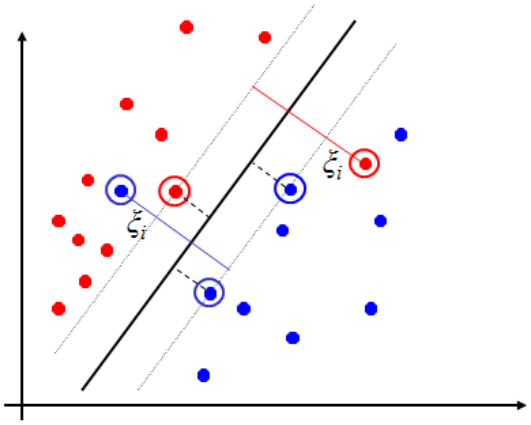


Figure 2.4. Illustration of slack variables and the support vectors.

And then the problem becomes:

Primal problem: Find $\boldsymbol{w}, b$ such that $\Psi(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w^T w} + C\sum_{i=1}^{n}\xi_i$ is minimized, given $y_i(\boldsymbol{w^T x_i} + b) \ge 1 - \xi_i$ satisfies on all points $(\boldsymbol{x}_i, y_i)$.

Dual problem: $\max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i - \frac{1}{2} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j}$ where $0 \leq \alpha_i \leq C, i = 1, \dots, n$ and $\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \alpha_i y_i = 0$

Here C is a parameter to be decided by user. A larger C corresponds to a higher slack penalty to the cost function, in this case, the model cannot bear a large ξ_i, however, oppositely, a small C lead to a large ξ_i and bring in many misclassified examples.

In practice, for example, when using the RBF kernel, we encounter two parameters: C and γ which need to be tuned. Here γ defines how far the influence of a single training point reaches. A low γ will have a far influence (think about the normal distribution, low γ means large variance). It indicates that even the point far from the hyperplane will have relatively high influence to the hyperplane and make the hyperplane smoother if γ is small. On the contrary, a large γ will lead to an irregular hyperplane. Both of these two parameters control the complexity of the model. Therefore, as we told before, the best way to tune these parameters is cross-validation and use grid search to find the optimal parameter.

2.2 Decision tree and random forest

A. Decision tree and Classification and Regression tree (CART)

A decision tree (R. Quinlan 1985) is a tree-like data structure which goal is trying to achieve perfect classification with minimal number of decisions. It contains three components: internal nodes, branches and leaf nodes. Each internal node splits the instance space into two or more sub-spaces according to one feature (attribute). The branches extending from an internal node represent a function or splitting criteria which is depending on the feature in the internal node. Each leaf node, which has exactly one incoming branch and no outgoing branch, represents the classification result (a class label). Thus, when a test point is classified by a decision tree, the path from root to leaf will be the classification rule and the final label will be the classification result. So we just need to build a decision tree with our training set and then apply the tree on the test set to get the result (label).

There are many kinds of decision trees: ID3 (R. Quinlan 1986), C4.5 (R. Quinlan 1993), CHAID (G. Kass 1980), MARS (J. Friedman 1991) and CART (L. Breiman 1984). The main difference between them is the splitting criteria. CART is one kind of decision trees and a widely used algorithm, which is also the elementary of the random forest algorithm. Next I will explain how to build a CART based on an example training set.

Suppose we have the following dataset with label “defaulted (Y)” and “not defaulted (N)” and two features “years at current job” and “number of payment missed”:

years at current job	number of payment missed	Defaulted?
7	0	N
0.75	0	N
3	0	N
9	0	N
4	2	Y
0.25	0	Y
5	1	N
8	4	Y
1	0	N
1.75	0	N
3	3	N

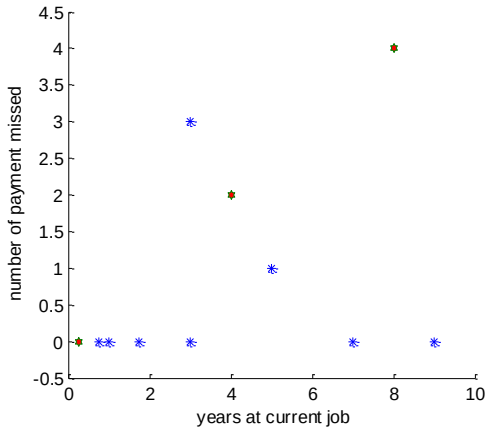


Figure 2.5. A simple data set of 11 observations with 2 features

Now we have the information of our training set $(\boldsymbol{x}_i, y_i)$ and encounter some problems in the construction of the tree:

- Each internal node will be split based on one feature, how do we decide the feature?
- How to split based on one feature and how many branches will be split?
- When should we stop splitting?

In CART, we usually do a binary split on each node. Suppose we want to split the instances based on feature j and these features are $F = \{x_{1j}, x_{2j}, \dots, x_{nj}\}$. For a binary splitting, if this feature is a continuous feature, we will first sort them and get a best split C with $\{x_{ij} \geq C\}$ and $\{x_{ij} < C\}$. If it is a categorical feature, then we will find two feature subsets $F_1, F_2 \in F$ where $F_1 \cup F_2 = F$ and $F_1 \cap F_2 = \emptyset$ with the best split. So the question is how to determine the threshold C and F_1, F_2.

The splitting criteria is called the Gini index/Gini impurity (C. Gini 1912) which is defined as

$$I(t) = \sum_{j=1}^{m} p_j(1 - p_j) = 1 - \sum_{j=1}^{m} {p_j}^2 \tag{2.10}$$

where $I(t)$ is the Gini index at node t and p_j is the proportion of observations with class j in the data set at node t.

From the formula, the Gini index is a measure of misclassification rate if we randomly choose and label an observation. The larger the Gini index is, the higher misclassification rate the dataset will have. Since our goal is to classify all the data correctly, we should consider about a good strategy that can decrease the Gini index from the original Gini index quickly.

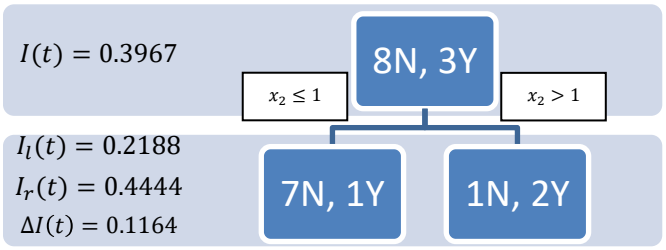
Naturally, let us think about the "best split" for each node. As we discussed above, the best split should have a maximum reduction of the Gini index on the node. Because of the binary

splitting for each node, the change of the Gini index at each splitting which is called **information gain** (T. Mitchell 1997) is:

$$\Delta I(t) = I(t) - p_l I_l(t) - p_r I_r(t) \tag{2.11}$$

and we need compute the split where $\max \Delta I(t)$ reaches. Now we go back to the credit example and build the CART on it.

We begin with the root node which include all samples in training set and compute the Gini index of the root node: $I(t) = 1 - \sum_{j=1}^{m} {p_j}^2 = 1 - \left(\frac{8}{11}\right)^2 - \left(\frac{3}{11}\right)^2 = 0.3967$. (8 N and 3 Y) Since all features are continuous, we consider each feature one by one and rank each feature. For the first feature x_1, we have $0.25 < 0.75 < 1 < 1.75 < 3 < 4 < 5 < 7 < 8 < 9$, so we have 9 splits: $\{x_1 \leq 0.25 \text{ and } x_1 > 0.25\}$,…, $\{x_1 \leq 8 \text{ and } x_1 > 8\}$. For the second feature x_2, we have $0 < 1 < 2 < 3 < 4$, so we have 4 splits: $\{x_2 \leq 0 \text{ and } x_2 > 0\}$,…, $\{x_2 \leq 3 \text{ and } x_2 > 3\}$. In total there could be 13 different splits and now we need find which split provides us the maximum information gain. For one example of $\{x_1 \leq 3 \text{ and } x_1 > 3\}$, $\Delta I(t) = 0.3967 - \frac{6}{11}\left(1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2\right) - \frac{5}{11}\left(1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2\right) = 0.0270$. For the other example of $\{x_2 \leq 1 \text{ and } x_2 > 1\}$, $\Delta I(t) = 0.3967 - \frac{8}{11}\left(1 - \left(\frac{1}{8}\right)^2 - \left(\frac{7}{8}\right)^2\right) - \frac{3}{11}\left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) = 0.1164$. After you compare 13 splits, you can find $\max \Delta I(t) = 0.1164$ and its corresponding split is $\{x_2 \leq 1 \text{ and } x_2 > 1\}$.



Thus we can do this step recursively on each node until once the data in one node only belongs to a single class, where the Gini index of that node is 0. For the example above, the full CART will be:

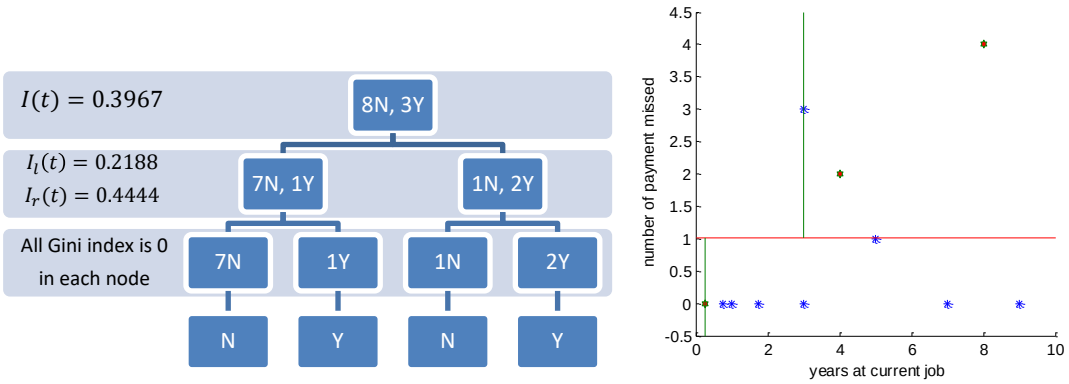


Figure 2.6. The Geometric demonstration of a CART.

Clearly, a CART will split the space into several rectangular-like areas and do classification by the training points inside each area and the Gini index (or other splitting criteria) gives you the optimal strategy to split. However, this method is also easy to have the overfitting problem. For the following instance in Figure 2.6,

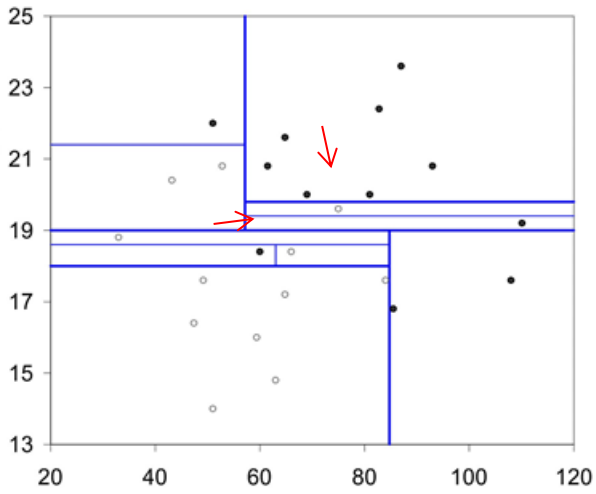


Figure 2.7. Illustration of an example of overfitting in the construction of CART

These two noisy points in training set will have impact on the whole rectangular-like area though that area should have the same class as their neighbor rectangular areas. Usually, you cannot construct a decision tree until data in every leaf node only belongs to one class. It will not only have the overfitting problem but also cost too many time to do it, though the accuracy of training set will improve under a large tree.

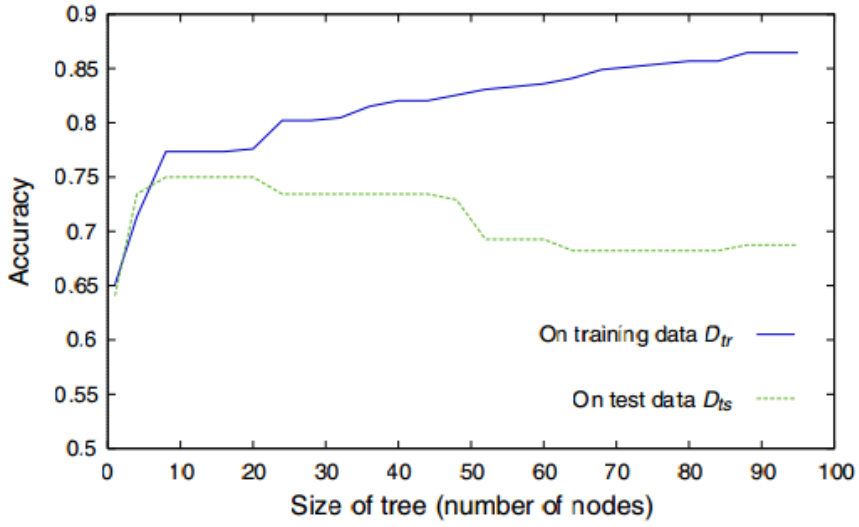


Figure 2.8. Illustration of the overfitting problem when the size of CART becomes large

To avoid this overfitting problem, many strategies have been developed as follows:

- Add a regularizer on the cost function $J = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2$, like the number of nodes or depths of CART.
- Stopping criteria, like $depth < C$ or $\Delta I(t) < C$
- Pruning decision trees, like reduced error pruning and minimal cost complexity pruning
- Ensemble methods with bagging strategy, like random forest algorithm.

With these tools applied to CART, it becomes more robust and has many advantages:

- Simpler to build and interpret.
- It is a non-parametric algorithm that means you do not need to train any parameters of the tree like C in SVM.
- This algorithm can handle both continuous and categorical data.

However, there are still many limitations of the CART algorithm, like:

- It is instable because sometimes the structure of the tree will be totally different even we perturb one data a little bit, which also make the variance large.
- The decision boundaries are not smooth. CART only provides us the rectangular-like boundary while the boundaries of continuous data are usually curves.
- Some nearby data, which should have similar properties, could be labeled different under CART.

Therefore, we need some strategies that can overcome these limitations and then the random forest, an ensemble method with bagging and random selecting features idea, is design based on CART to solve the problems.

B. Random forest

In order to overcome the large variance and unsmooth boundary, a genius "bagging" algorithm was proposed by Breiman -- Bagging (L. Breiman 1994), which is the abbreviation of bootstrap aggregating. It is a powerful idea that makes use of several subsets to build the model and then takes the majority vote as the final result.

Bagging algorithm: Suppose we have a binary classification training set T with N samples

1. Construct a bootstrap sample: pick one sample from T with replacement and repeat this for m times (usually $m = N$), pick out the not repeating samples together to be the bootstrap sample T'.
2. Construct a CART without pruning based on T'.
3. Repeat 1-2 steps for a large number of times M and get M CARTs.
4. For each observation in test set, we compute M results of the label from M CARTs.
5. Regard the majority voting of the M results as the final results.

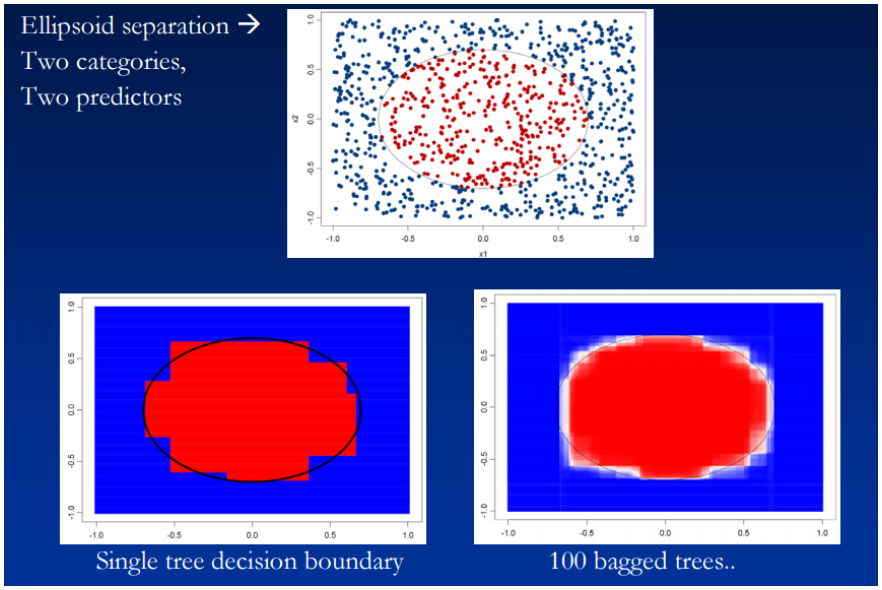


Figure 2.9. An comparison of the boundary between single CART and Random Forest

Meanwhile, since we average (majority voting) the result, we not only get rid of some random event, but also make the boundary to be smoother and this boundary will be more reasonable.

Although bagging is an algorithm with so many good properties, random forest algorithm further expands the bagging idea into a more robust algorithm. Compared with bagging

algorithm, RF is a more "random" algorithm on features (variables) and further makes use of the out-of-bag sample (the data not included in bootstrap sample) as the validation data to test how good the random features you choose.

Random forest algorithm (A. Liaw and M. Wiener 2002): Suppose we have a binary classification training set T with N samples and M features.

1. Construct a bootstrap sample T'.
2. Build a CART with T' without pruning. When constructing the CART and splitting each node, randomly pick $m \ll M$ features and find a best splitting based on these m features (these features can be sampled with or without replacement).
3. For a constructed tree, compute the out-of-bag error. It is achieved by testing each out-of-bag sample with the CART and figuring out the misclassification rate, which is the OOB error.
4. Repeat step 1-3 for a large number of times like 500 or 1000.
5. Choose a proper m, usually the initial $m_0 = \sqrt{M}$. For each m, we can compute a corresponding OOB error. Then do right test for the OOB error of $m_{i+1} = 2 * m_i$ until $OOB_{m_{i+1}} > OOB_{m_i}$ and left test for the OOB error of $m_{j+1} = \frac{1}{2} * m_j$ until $OOB_{m_{j+1}} > OOB_{m_j}$. Finally, the optimal $m = \arg_{\{m_i, m_j\}} \min(OOB_{m_i}, OOB_{m_j})$.
6. For each test point, we compute a label for each tree (CART) and regard the majority voting as the final classification result.

Random forest has many advantages. Firstly, compared with the bagging method, the trees in the RF have less correlation because of the randomness on features. Thus, the reduction of the variance from the CART becomes larger. Secondly, overall, random forest can make use of all information of the training set, meanwhile, it decrease the complexity of the model comparing with CART. Thirdly, RF did not simply randomly choose and use a feature but still consider the best splitting, which is a feature selection idea. Simply speaking, RF not only utilizes all features but also considers the importance of all features.

2.3 K-Nearest Neighbor (KNN)

KNN is one of the simplest machine learning algorithms, which is a type of instance-based learning. The KNN model is just the training dataset itself and the classification result is computed by a distance based measure from the dataset.

Since we have the data of the training set and its distribution in the feature space, the KNN idea is to decide the class of a test point by comparing the similarity between it and its neighbors in the feature space. Here K is a user-defined hyper-parameter. With a given K and a test point, we are easy to find the K-nearest neighbors of the test point. Then we define the similarity as that

the class of the test point is the same as the most frequent class among its K neighbors. (usually we use the Euclidean distance here, sometimes we can also try Manhattan or Minkowski distance) For example in the following figure, we can easily see that the class of the test point only depends on its K neighbors:

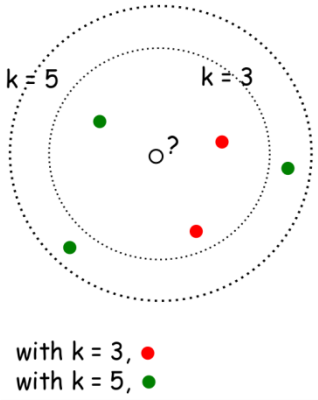


Figure 2.10. An illustration of how KNN works

However, there is still a problem that how to determine the optimal "K" because different K show the different properties in the classifier. A small "K" like K=1 often make the boundary too fine and complex, which lead to the overfitting problem while a large K can achieve a smooth boundary but may include too many training points from other class which is harmful to predict. In practice, we usually play a cross-validation on the training set in order to obtain an optimal K.

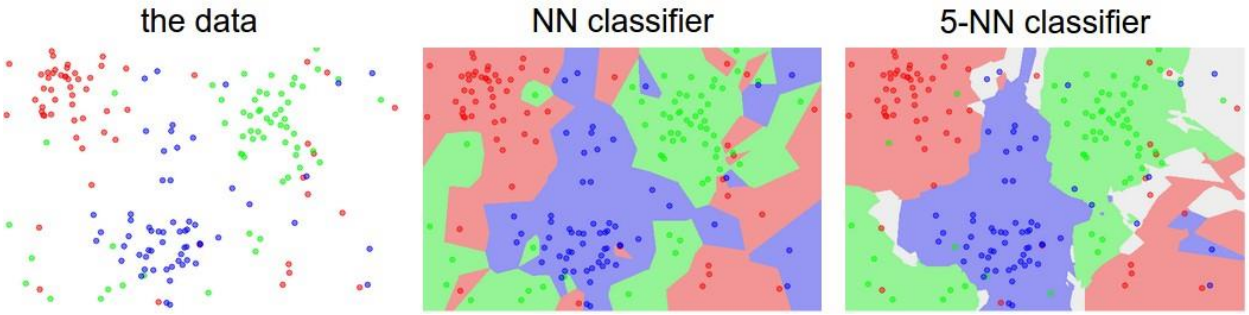


Figure 2.11. The impact of different K in KNN algorithm

Chapter 3 . Introduction to the evaluation method: ROC curve

Given a dataset and several classifiers, since we have the prediction labels and the actual labels of the test set, we might want to know which classifier has the best performance on the dataset. Therefore, an evaluation method and an evaluation score are required for us to compare the performance of each classifier.

3.1 Confusion matrix

Based on the information of predication and actual labels, we can build a confusion matrix (S. Stehman 1997) like this (positive and negative can be interpreted as class 1 and class 2):

		Total Prediction Information	
		Predicted Positve	Predicted Negative
Total Actual Information	Actual Positve	True Positive (TP)	False Negative (FN)
	Actual Negative	False Positive (FP)	True Negative (TN)

Table 3.1. The definition of TP, FP, TN and FN in confusion matrix

From this confusion matrix, a lot of measures can be calculated like the accuracy:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \tag{3.1}$$

Accuracy is a simple measure for the performance of a classifier. Intuitively, we might think the higher the accuracy is, the better the classifier will be. However, this accuracy measure has some limitations.

Firstly, accuracy does not work well on imbalanced data. Let us consider this medical diagnosis classification problem. Suppose we can 5 people with cancer and 995 healthy people, classifier 1 classifies 5 people with cancer correctly but misclassifies 10 healthy people as the cancer people and classifier 2 only misclassifies 1 cancer people as healthy people. Then we find the accuracy of classifier 2 is 999/1000=0.999 which is far larger than the accuracy of classifier 1 (990/1000=0.99). Although the accuracy of classifier 2 is larger, the only one mistake is fatal. In

another aspect, classifier 1 has the accuracy on cancer people is 100% and healthy people is 98.99% while classifier 2 only has the accuracy on cancer people is 80% and healthy people is 100%.

Thus, we need consider more measures rather than simply use the accuracy and then we try the following measures: sensitivity, specificity, precision, F1 score, Matthews correlation coefficient and so on. Each of them will compute the accuracy based on certain focus in your prediction. For the cancer example, you might care more about the accuracy on cancer prediction so that you prefer the $\text{sensitivity} = \frac{TP}{TP+FN}$ as your evaluation method. However, most of the measures still cannot reflect the trade-off between the performances on two classes.

The second limitation comes from the output of a classifier. Before we only discussed the binary output 0 or 1 as the decisions, however, the output for each data point can also be a probability/score between 0 and 1. For example, we can define this score in random forest as $p_{x=1} = \frac{Total\ number\ of\ trees\ vote\ to\ 1}{Total\ number\ of\ trees}$, and in KNN as $p_{x=1} = \frac{Total\ neighbors\ with\ class\ 1}{Total\ neighbors}$. In both cases, a default threshold equals 0.5 is set to determine the binary output that the output is 1 when $p_{x=1} \geq 0.5$ and 0 when $p_{x=1} < 0.5$. However, whether the default threshold is proper is a big question. Considering a very imbalanced data set and the KNN model, since most training samples belong to one class, a majority vote of the K neighbors will tend to be that class. Moreover, the outputs of some classifiers like the logistic regression are values between 0 and 1, where you have to choose a threshold value and this value will have a large impact on the measures we mentioned before. Actually, an output of probabilities has more information than the binary results. For example, suppose we have two test points in a RF algorithm with $p_1 = 1$ and $p_2 = 0.51$, in a binary output strategy we may simply classify them into the same class, however, we must notice that the difference of the score at least reflects the properties of point 2 is closer to the other class than point 1.

3.2 ROC curve and AUC value

The Receiver operating characteristics (ROC) curve and the Area Under the Curve (AUC) provide us the measure that solve these two limitations (T. Fawcett 2006, B. Song 2014). Differ ing from those single measures, ROC curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied, that is to say, this curve includes the performance of the classifier at each threshold. In a ROC curve, the true positive rate (sensitivity, $TPR = \frac{TP}{TP+FN}$) is plotted in function of the false positive rate (1-specificity, $FPR = \frac{FP}{FP+TN}$) for different threshold settings. In order to construct the ROC curve, we first rank the scores of the test set and since we know the corresponding score for each point, we can

compute the TPR and FPR as the coordinate for each point one by one as the threshold and finally connect all the coordinates into a curve.

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

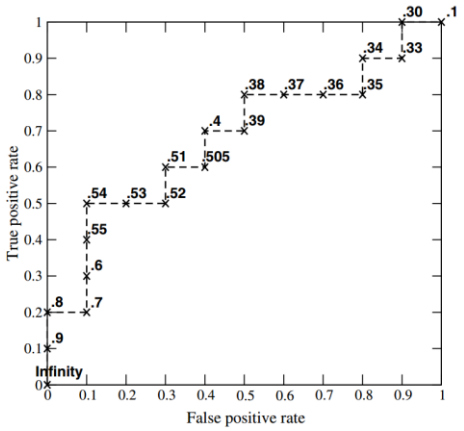


Figure 3.1. The construction of the ROC curve

When you compare two models based on the performance on a data set, you can plot two ROC curves like this:

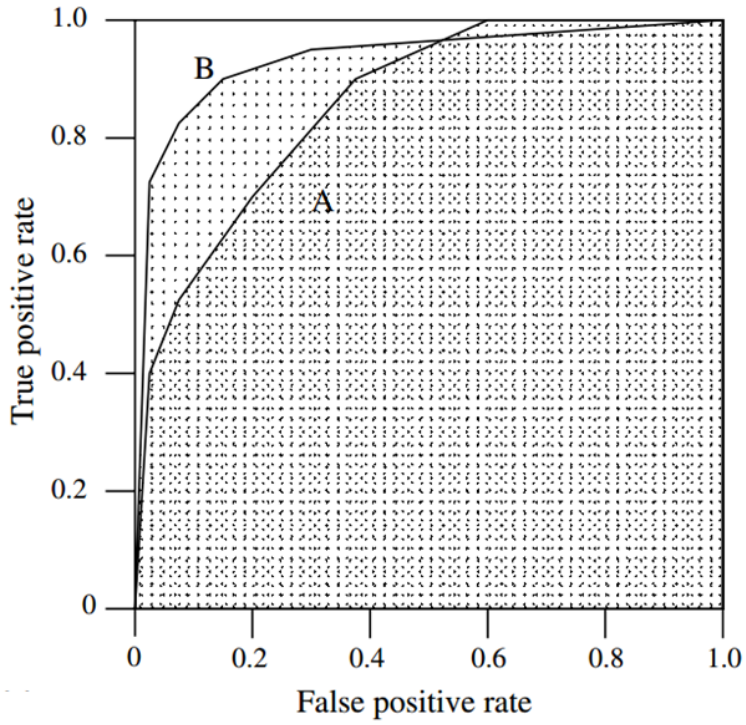


Figure 3.2. A comparison of two ROC curves from two models under the same coordinates

The ideal case is that the curve of model B is above the curve of model A, which means that given the same FPR level by two models, we can always get a higher TPR by model B than model A. That is to say, model B has a better performance than model A in this data set.

However we have to notice that most cases we only get two curves like those in the figure that would cross each other rather than one curve always top the other. Thus, the AUC value is designed here to compare the overall performance between models. AUC value is not only equal to the area under the curve but can also be considered as the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. (T. Fawcett 2006, Hanley and McNeil 1982) Usually we will regard the model with higher AUC as the better model. Furthermore, another advantage of the ROC curve is that in some special problem like the cancer detection problem, the doctor might have a strict requirement on the FPR. Thus, we can choose the best model and the threshold according to the requirement value with the ROC curve. In this paper, all evaluation and comparison for the classifiers will be based on the AUC value.

In order to apply the ROC analysis on the classifier evaluation, each data point will be assigned a score/probability by a classifier, rather than only be assigned a binary label. In RF and KNN, the probabilities for class i are easily defined as $\mathrm{P}(y = 1|x) = \frac{votings\ for\ class\ 1}{total\ votings}$ and $\mathrm{P}(y = 1|x) = \frac{\#\ with\ class\ 1\ of\ K\ neighbors}{K\ neighbors}$. For the SVM classifier, it is not that easy to produce a calibrated value because only the hyperplane is calculated from the SVM. In such cases, Platt (J. Platt 1999) proposed a strategy that can produce probabilities of each sample according to the signed distance between the sample and the hyperplane:

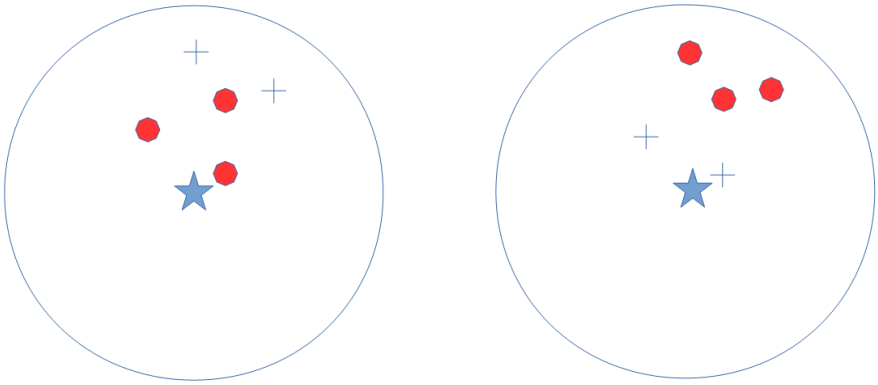
$$\mathrm{P}(y = 1|x) = \frac{1}{1+\exp(A*f(x)+B)} \tag{3.2}$$

where $f(x)$ is the distance between point x and the hyperplane, A and B are two parameters estimated by the maximum likelihood method that optimizes on the same training set.

Chapter 4 . Innovative ideas and new classification models

4.1 An improvement of KNN: location index (Y. Hu et al. 2014, Y. Hu et al. 2015)

KNN is a simple and smart idea but not a robust classifier. It has several shortcomings. First and foremost, KNN classifier only considers the impact of K neighbors for each test point but totally ignore the impact of distance between the neighboring point and the test point. Intuitively, a nearer neighbor should have larger impact on the test point than a farther neighbor. Moreover, the KNN algorithm also suffers from the scale of the dataset and the curse of dimension, whose time cost is O(nd+kn).



(a) (b)

Fig. 4.1. Intuitively, the two sets of test points in (a) and (b) should belongs to different classes (red in (a) and blue in (b)). However, they will be classified to red under kNN scheme when K=5.

A better model should reflect the information of the distance from the test point to each neighbor. Thus, the weighted KNN (T. Mitchell 1997) is designed to solve this problem and the weight for each neighbor is the reciprocal of the distance between this neighbor and the test point, which can be interpreted as a locally regression based on its K neighbors. However, this algorithm also features two limitations: 1. It needs to train a proper “K”; 2. It is a very time-consuming algorithm just like KNN. As illustrated in section 2.3, a small “K will lead to a fine and complex boundary which often accompany with the overfitting problem while a large “K”

make the boundary too smooth to underfit the model. In a highly imbalanced dataset, it could even encounter the case that the large class will always dominate whatever K you choose. Then no K will be proper and KNN will not work.

Therefore, the location index or LI in our work is defined as follows, based on mainly the idea of "weighted KNN" classifier (wKNN) (T. Mitchell 1997) and the idea of the Newton's law of universal gravitation. Similar to the law of universal gravitation, we assume that each training point, instead of the K neighbors, will have impact on the test point and this impact will be proportional to the label of the training point and inversely proportional to the square of the distance between them. Then given the distribution of the training set, it is easy to compute a location index for everywhere. That is to say, we can imagine that the training set determines a distribution for the space and each point in the space has a density score that is the location index.

Before computing the location index, we should first normalize each feature to the same scale like [0,1]. Otherwise, it means that you put some weights on each features because the distance depends on each of your features and the feature with large scale will dominate the final output.

Definition 1: Location index

For each test point x, the LI of x is a score or posterior probability $p_x = \frac{\sum_j \frac{y_j}{d(x_j,x)^2}}{\sum_j \frac{1}{d(x_j,x)^2}}$.

In the formula above, y_j (0 or 1) is the label of each training point, x_j is the feature vector of each training point and $d(.)$ is the Euclidean distance between the test point x and training point x_j.

Actually, this score is a linear combination of the labels of the training set and the weight for each label y_j is the $\frac{\frac{1}{d(x_j,x)^2}}{\sum_j \frac{1}{d(x_j,x)^2}}$, which is the same as one of the weighted KNN classifiers.

Many obvious properties can be seen from the LI by the following, see also Figure 4.1:

1. If the label of all training point is 0, the score for this test point will be 0. If the label of all training point is 1, the score for this test point will be 1. Then the score for each test point will between 0 and 1.
2. The nearer the training point is to the test point, the larger the weight of that training point is. The further the training point is from the test point, the smaller the weight of that training point is.
3. Reflect the location of the test point, whether it is located at a "boundary" area (the score is far from 0 or 1) or "non-boundary" area. (the score is near 0 or 1) .
4. This index is continuous in the space, then it will have less variance than kNN/wKNN model.

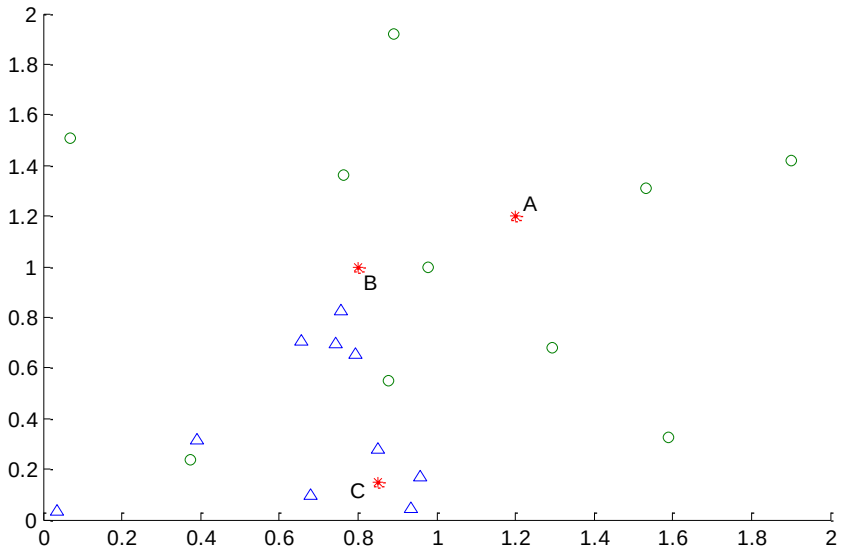


Fig. 4.2. The red points are three test points, the green points (label=1) and the blue points (label=0) are training points.

For a simple example as shown above by Figure 4.2, the location index $p_A = 0.6046$, the location index $p_B = 0.4652$, and the location index $p_c = 0.2281$. This result shows that the larger the location index for a test point is, the more points with the class 1 label are surrounding this test point. Actually, similar to the logistic regression algorithm, the location index classifier is a continuous mapping f: $\mathbb{R}^{n+1} \to [0,1]$. Thus, unlike the KNN or wKNN that are discontinuous, the LI model will have less variance while not raise the bias of the prediction. The boundary of two classes: $f(x_0) = p_{x_0}$ is smooth and accuracy which divide the space into two classes: class=1 where $f(x) > p_{x_0}$ and class=0 where $f(x) < p_{x_0}$. Furthermore, the most important property of this classifier is that this location index can better reflect the surrounding information of each point because KNN and wKNN only focus on the nearest neighbors but ignore the distances to the farther neighbors.

Furthermore, the LI itself can serve as a good classifier, since the input is just the training set and the output score is the index value for each test point: $p(x|y = 1) = p_x$ and $p(x|y = 0) = 1 - p_x$. After obtaining all the scores, it could be evaluated by the ROC (receiver operating characteristics) analysis and the AUC (area under the ROC curve) value. Compared to the wKNN classifier, the LI model does not need to find the optimal "K", nor the K nearest neighbors, which transpires into computational efficiency that cut down the time cost from O(nd+kn) to O(nd). For a large dataset, we can apply the parallel computing or Map-Reduce on the dataset to compute the location index easily. Moreover, location index can be naturally generalized to multiclass classification problem where $p_{x \in C_k} = \frac{\sum_j \frac{\delta(y_j \in C_k)}{d(x_j,x)^2}}{\sum_j \frac{1}{d(x_j,x)^2}}$. (if $y_j \in C_k$, $\delta(y_j \in C_k) = 1$, otherwise, $\delta(y_j \in C_k) = 0$)

Based on the results of the experiments on several data sets, we can argue that this LI model is an algorithm as robust as the SVM and the RF.

4.2 Introduction to classifier combination (J. Kittler et al. 1998, S.Tulyakov et al. 2008)

The classifier combinations problem try to construct models and derive combination rules via logic and statistics. Not only the combined experts are classifiers, the result of the combination is also a classifier. As we mentioned before, the outputs of classifiers can be represented as vectors of scores where the dimension of vectors is equal to the number of classes. Similar to the neural network as Fig. 4.3 shows, we can consider the outputs of each individual classifier as a set of new features waiting for further rules to classify them, which is the classifier combination. As a result, the combination problem can be defined as a problem of optimizing the combination function accepting N-dimensional score vectors from M classifiers and outputting N final classification scores.

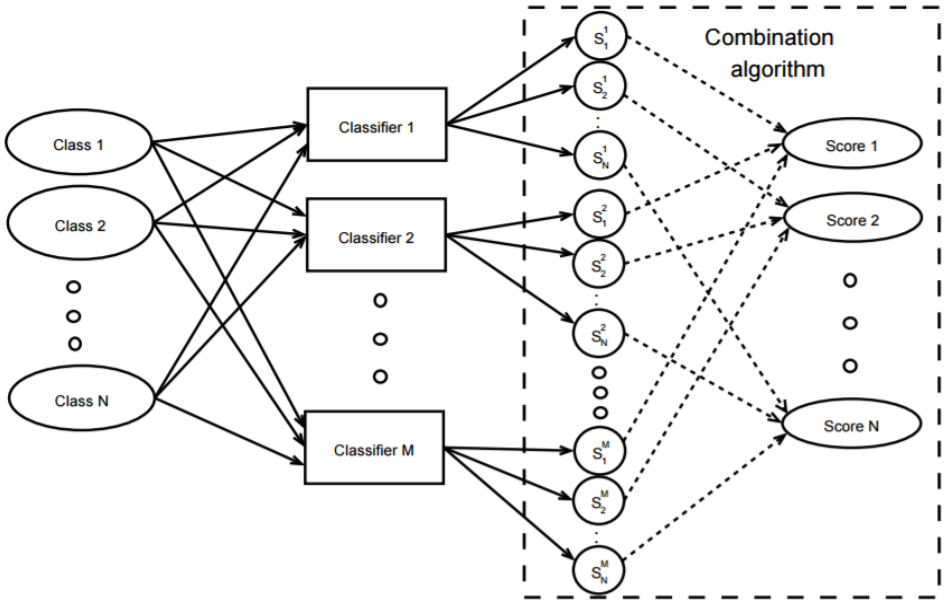


Fig. 4.3. The diagram structure about the classifier combination.

Usually a lot of information can be obtained from each individual classifier, such as predicted labels, rank of the confidences (score) and the predicted measures (score). Derived from these information, many combination strategies have been explored by scientists:

- Elementary combination schemes on measurement level: sum-rule, product-rule, …
- Ensemble methods: bagging and boosting.
- Locality based combination methods.
- Other methods: majority voting, combination schemes on rank level, Dempster-Shafer theory of evidence, …

In this chapter, I will explore three innovative classifier combination methods based on the first three ideas above, respectively. In section 4.3, I will introduce the RF embedded location

index which is the locality base combination method. In section 4.4, I will apply the ensemble method on LI and finally in section 4.5, I will explore a new combination rule on measurement level.

4.3 An embedded strategy: RF embedded location index (Y. Hu et al. 2015)

From the example of Figure 4.4, it can be seen that there are two types of points:

1. Points located inside one class (like points inside green and blue areas), which should be classified correctly.
2. Points located on the boundary between two classes (like points inside red area) are pretty hard to classify.

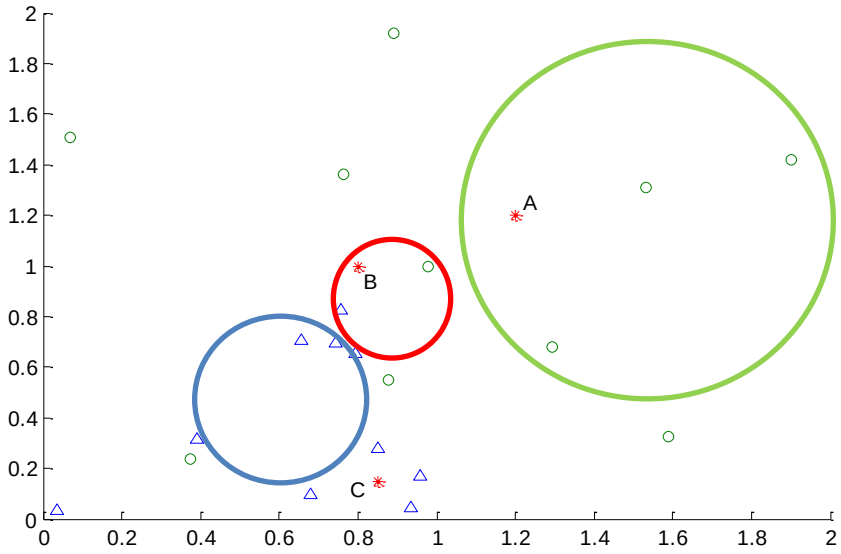


Fig. 4.4. Different types of points in a dataset.

Thus, we are wondering whether it is helpful to conduct with these two types of points differently and then the RF embedded location index (LIRF) is designed under this divide and conquer idea.

According to the property of the location index, the points in the green area will have the output near 1 and in the blue area near 0 and points inside these areas should be classified correctly. Therefore, the location index is a proper and accurate classifier to express these points due to the geometric sense. However, it is still hard to determine the class of "boundary" points (like points in red area) by LI because of some reasons like the irregular distribution of the classes or the noisy points.

Therefore, our idea is to classify those two types of points differently. For each test point, we first decide which type the point is. This step could be performed by computing its LI value. As we mentioned before, a threshold value α can help us to divide the space into two parts: $f(x) > \alpha$ and $f(x) < \alpha$. So, we can tune a pair of parameters (α, β) and express those 2 types of points in the space. If the LI is very large ($\beta \leq f(x) \leq 1$) or small ($0 \leq f(x) \leq \alpha$), the point belongs to type-1; else if ($\alpha < f(x) < \beta$) the point belongs to type-2 points set. Then we can directly

classify type-1 points by LI and classify type-2 points by another classifier. This method will work because for the same dataset, different classifiers will have different boundaries and boundary points, where the most errors come from. That is to say, the boundary points with LI are not really to be boundary points with other classifiers. Therefore, I use another robust classifier RF as the second component of this combination strategy.

Since LI will be more robust on classifying type-1 data and RF is good at conducting with type-2 data, based on the all the dataset, we propose to embed the LI into the RF algorithm to improve the RF performance on classifying the type-1 points, where RF might sometimes have unsatisfactory performance. The proposed location index embedded Random Forest (LIRF) algorithm is presented below:

Location index embedded Random Forest classifier

1. Divide our dataset H into training set H_1 and test set H_2.
2. Build a RF model by the training set.
3. Decide two threshold parameters: α and β. ($\alpha > \beta$)

For each test point
{
Calculate the LI of the test point.
If $LI > \alpha$ or $LI < \beta$
Final score of the test point = location index of the test point
Else
Calculate a score p of the test point with the RF model.
Final score of the test point = $(\alpha - \beta) * p + \beta$.
End
}

4. Perform the ROC analysis with all the scores and compute the AUC value.

From the algorithm, it is easy to conclude that the worst case of LIRF classifier will be the maximum of LI and RF, because when $\alpha = \beta$, LIRF will be degenerate to LI and when $\alpha = 1, \beta = 0$, LIRF will be degenerate to RF. The crucial step of this algorithm is the "embedding", where I keep the LI score unchanged and map the RF score from [0,1] to (α, β). This step indicates that we firstly decide how much we trust the LI classifier and then use the RF to modify and reorder the result of the boundary points by LI. Thus, a good pair of parameters leads to a good result. Then the most important task of this classifier is to find a proper pair of the parameters (α, β). Since different datasets will have different class distributions, the optimal pair of parameters will be different. In order to obtain this optimal value, we can use the cross-validation strategy and a grid-search (because you need to find two parameters) and choose the parameters that make the average AUC the highest.

This is also a good classifier on the imbalanced problem because most imbalanced parts of the test set are cut off and classified correctly by LI and a more balanced dataset will be taken into consider by the RF, where less errors exist.

4.4 An ensemble exploration: LI forest (Y. Hu et al. 2016)

When we solve a problem with a large dataset and high-dimensional feature space, similar as the decision tree, the LI classifier also encounters the problems that the model might be too complex or too time consuming to build (O(nd)). Beside the speed issue, the variance can also be decreased by the ensemble methods. Therefore, I also explore an ensemble algorithm based on the LI classifier, which is the LI forest.

We can apply the ensemble strategy here because of three reasons: 1. LI is a non-parametric classifier, which means we do not need to tune any parameters before we construct the LI model. (So that it is hard to apply a likewise strategy on SVM.) 2. LI is such a simple idea that the parallel computing is easily applied on the ensemble algorithm. 3. The running time decreases a lot when we use the ensemble algorithm.

Thus, similar to the RF strategy based on the CART, the idea of the LI forest is firstly take "random" on both data and features on building small LI model, and then ensemble them together to be a "forest". We will see how this ensemble method is designed:

Location index forest classifier

Part I: construct a series of LI model with the training set
(N observations, M features)

1. Conduct a feature selection (ranking) with RF and rank the importance of all features by training set.
2. Divide the training set into a bootstrap sample and validation sample for K times and then combine them as the LI forest.
3. Decide how many features m should we use in the LI forest:
   a. For each LI model, we use the initial value $m_0 = \frac{M}{2}$
   b. Each time we choose a subset M^* features (In the experiment, $M^* = \frac{M'}{3}$) from the total M' (M') features and choose the optimal feature without replacement according to the feature ranking. This step will be done m times to construct the feature set for each LI model.
   c. Then obtain the average $AUC = f(m_i)$ from the validation set. (Since we can compute an AUC from each validation sample.)
   d. Perform a series of left tests: $m_{i+1} = \frac{1}{2}m_i$ and choose the feature set based on step a and b. Once $f(m_{i+1}) < f(m_i)$, $m_{left} = m_i$
   e. Perform a series of right tests: $m_{i+1} = \frac{1}{2}(m_i + M)$ and choose the feature set based on step a and b. Once $f(m_{i+1}) < f(m_i)$, $m_{right} = m_i$
   f. Let $m = \text{argmax}_{m_{left}, m_{right}}(f(m_{left}), f(m_{right}))$
4. The LI forest model is constructed by K bootstrap samples and for each sample, we use m features. (Note: m features between different LI model are different.)

Part II: Compute the result by constructing the model with tuned parameter

1. Construct a new LI forest with K LI models where each LI model include all training points and m features, the step for these features are the same as Part I.

2. For each test point, one probability will be computed by each LI model and the averaged probability will be the final score for this test point.

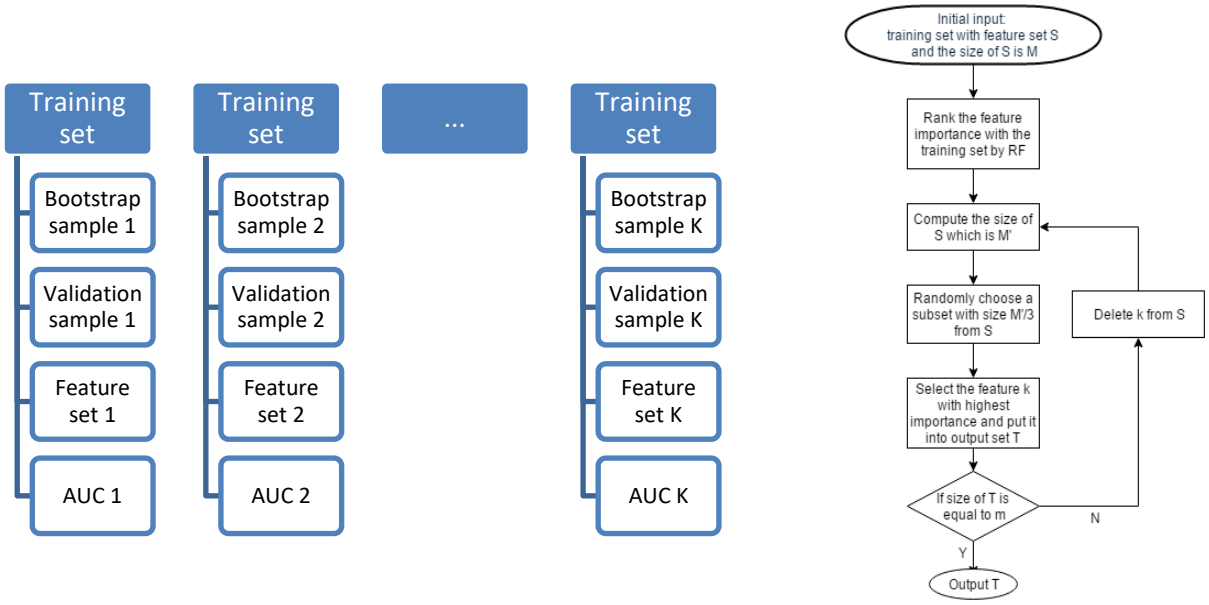


Fig. 4.5. The flowchart and structure about the LI forest classifier.

This Location index forest classifier has several improvements based on the location index classifier. Firstly, it has a feature selection step and the high-importance feature will have a high-usage, which means we put different weights on different features. In a single LI classifier, different training points are weighted while it is very hard to give some meaningful weights on different features. If we just make use of some features with high importance, the loss of information will be unavoidable. Secondly, as we discussed before about the ensemble method, similar to the RF model, this LI forest model can also have a lower bias and variance than the LI model, though the bias and variance is small in LI model. Thirdly, when it comes to a big dataset, this classifier can still have good performance. On the one hand, we can easily apply the parallel computing algorithms not only to build a large amount of trees but also to compute the location index. Since we do not need to consider about the neighbors, we can use "the stochastic ideas" that choose a small sized training and validation sets randomly instead of using the bootstrap samples, and then construct more small single LI models by parallel algorithms in order to keep the accuracy.

4.5 Mixture classifier with classifier selection based on the AUC value (Y. Hu et al. 2014, M. Ma et al. 2014)

We have learned and interpreted several classifiers by now, and some of which are similar, like the tree based algorithms, which share a similar data structure and the main difference between them is the splitting strategies. However, most of them show us many different

properties (B. Song, etc. 2011). In this section, we will explore a mixture strategy based on many selected classifiers by the AUC value.

Actually, the whole dataset can be divided into two parts by every classifier according to the output value: 1. boundary points (output far from 0 or 1) and 2. points far from the boundary (output near 0 or 1). And we also discuss the theoretical background about each classifier in chapter 2: SVM is a kernel based algorithm, RF is a tree based data structure and LI is a distance-based algorithm. Furthermore, the cost functions of them are also different from each other. The cost function of SVM is an optimization problem while the RF is about the GINI index. Therefore, the boundaries and output schemes are totally different for different classifiers. Then it leads to that the "boundary" points are different for each classifier. However, in some easy cases like points in green area in Fig 4.3, a robust classifier will not misclassify them (with high score for every classifier), like SVM, RF, LI, ANN (P. Werbos 1975), etc, otherwise there will be a huge error for the classification result. Then the mainly differences will occur when it comes to the boundary points from each classifier.

As we know, we can interpret the output score of a classifier as the confidence of a point be classified to class 1/0. Based on this, we should attempt to design a combination strategy, which could not only promise the score of the easy case still be with high confidence after combination, but also a reasonable measurement could be obtained from this combination strategy. Before a lot of methods are widely used by scientists like sum-rule, product-rule and maximum-rule, which means we take the sum/product/maximum of all the output scores for each individual classifier. However, those outputs have, in general, neither the same range and scale nor the same distribution, which the classifier with a larger scale may dominate in the combination score. For example, one output is (0.9, 0.4, 0.3) and the other is (0.3, 0.5, 0.4). Two classifiers have more confidence that the latter one is more likely to be class 1 than the former case while the result of both the sum rule and the product rule will be dominated by the first classifier. In order to solve this problem, I modify the product-rule and add a penalty term that helpd to balance the "domination".

In product-rule, $p_{x=1} = p_1 p_2 p_3$. Without loss of generality, suppose p_1 is the dominated classifier, then I balanced it by adding a term includes $(1 - p_1)$, which is $(1 - p_1)p_2 p_3$, because if p_1 is an extreme value, $(1 - p_1)$ will also be an extreme value. And in order to keep the symmetry for each individual, the total penalty term is $(1 - p_1)p_2 p_3 + p_1(1 - p_2)p_3 + p_1 p_2(1 - p_3)$. Finally, the output score is $p_{x=1} = p_1 p_2 p_3 + (1 - p_1)p_2 p_3 + p_1(1 - p_2)p_3 + p_1 p_2(1 - p_3)$.

This method also make use of the "majority voting" idea. For a test point, if at least two of three classifiers hint this point to be 1 with high confidence, then this point should have a high score after combination and we hope the final score can reflect this. Thus, $p_1 p_2 p_3$ means the score that 3 of them vote to class 1, and $(1 - p_1)p_2 p_3$ means method 1 votes to class 0 and the other 2 vote to class 1, and so on. Even though those outputs are not independent with each other,

we still use this formula to estimate the final score. This "majority voting" idea also indicates that we should put odd number of classifiers into combination in order to obtain the majority.

Another issue is that sometimes too many less-predictive component classifiers in the combination may bring too many errors for the combination. For example, classifier 1 is robust and the other two are not, where all the misclassified points from classifier 1 are also misclassified by the other two. Then the other two classifiers will only take into some errors into the combination classifier. In this paper, we only use 3 classifiers to obtain the combination and compare the result with the sum and product rule because we don't have the classifier selection step in these two rules.

Thus, in this algorithm, I firstly do a classifier selection by the training set, which means I evaluate each classifier by a cross-validation strategy and choose two or three robust classifiers by AUC from training set into the combination step. Based on the 3 AUCs from the training set, we think that if the second largest value is larger than the mean of them, then these 3 value include "2 good" and "one bad". (like 0.9, 0.89, 0.85), otherwise (0.9, 0.86, 0.85), we are going to use the best one twice and the second largest one as the hedge classifier. (0.9, 0.9, 0.86).

Mix classifier with classifier selection via AUC

Part I: Select the classifier

1. Perform a leave-one-out cross validation on training set with SVM, RF and LI, respectively.
2. From step 1, we obtain 3 scores, which is the probability that the data is classified as 1, for each data in training set: p_{svm}, p_{rf} and p_{li}
3. With scores of all data obtained in step 2, we can compute AUC_{svm}, AUC_{rf} and AUC_{li}
4. Rank the scores and the second largest of the scores is A, and the mean of the scores is B.
5. If A>B, we choose all the classifiers to build our mixture classifier, else we choose two times of classifier with largest score and one time of classifier with second largest score.

Part II: mixture classifier

1. Use 3 classifiers we choose from Part I. (Maybe two of them are same classifiers)
2. For each data in test set, compute 3 scores p_1, p_2 and p_3 with classifiers in step 1.
3. The final score of the data is $p_{x=1} = p_1p_2p_3 + (1-p_1)p_2p_3 + p_1(1-p_2)p_3 + p_1p_2(1-p_3)$.

In part I of this classifier, we apply the leave-one-out cross validation on the training set to evaluate the performance on the training set for each classifier. Therefore, these results can be compared to show which classifier has a good performance on a specific dataset. However, a higher performance of AUC in training set does not mean everything. On the one hand, the

performance in training set might be different from the performance in test set. On the other hand, even the classifier with a bad performance of AUC might perform better in some data point than other classifiers do. Thus, we design this mixture algorithm that can both complement each other and avoid the bad components/classifiers. The core idea of the mixture is the majority voting, which computes the probability that most components of the mixture classifier vote to class 1. In the meantime, we hope at least 2 of 3 classifiers are "good", which is judged by comparing the second largest AUC and the mean AUC of 3 classifiers. Otherwise, we will give the better classifier a higher weight. Then when selecting 3 different classifiers, since they have such different properties, we can regard their results as "nearly independent" to each other. So it is easy to have $p_{x=1} = p_1p_2p_3 + (1-p_1)p_2p_3 + p_1(1-p_2)p_3 + p_1p_2(1-p_3$, which means the probability of at least 2 of 3 classifiers vote to class 1. When only selecting 2 classifiers, we need consider different weights on them. Firstly, $p_{x=1}{}' = {p_1}^2p_2 + {p_1}^2(1-p_2) = {p_1}^2$ and $p_{x=0}{}' = (1-p_1)^2p_2 + (1-p_1)^2(1-p_2) = (1-p_1)^2$, here we put more weight on the more important classifier. Since we hope $p_{x=1} + p_{x=0} = 1$ and the score is balanced by p_2, we let $p_{x=1} = p_{x=1}{}' + 2p_1(1-p_1)p_2$ and $p_{x=0} = p_{x=0}{}' + 2p_1(1-p_1)(1-p_2)$, which indicate that the second term and the final score are also weighted by p_2.

Finally, we have the expression for the mixture classifier with 3 components: $p_{x=1} = p_1p_2p_3 + p_1(1-p_2)(1-p_3) + (1-p_1)p_2(1-p_3) + (1-p_1)(1-p_2)p_3$. (In two components case, we just let $p_1 = p_2$)

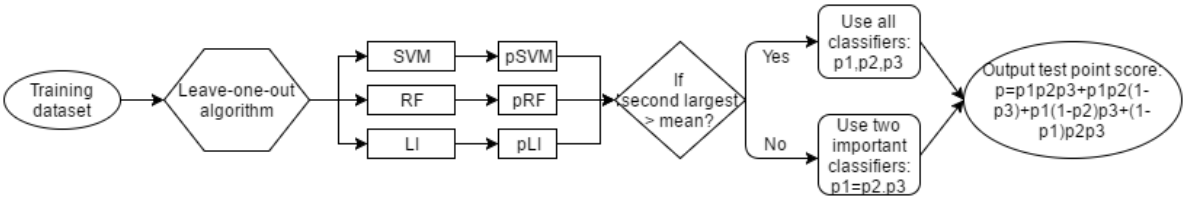


Fig. 4.6. The flowchart of the mixture classifier with classifier selection.

This algorithm also shares some ideas from neural network classifier. Our input are a large amount of features, then we serve the classifiers as the mapping function between the first and second layers and map the features into two or three features in the second layer. Finally, we make use of the property of the features in second layer and the majority voting idea to obtain the final score.

Chapter 5 Application I: Experiments and discussion on some open-source data

In this chapter, I applied the algorithms introduced in chapter 4 on some open-source data (M. Lichman 2013) and compared the performance of those methods with widely used classifiers SVM and RF.

5.1 Database and experiment design for experimental studies

All databases used in this chapter are from the UCI Machine Learning Repository as follows:

Dataset name	Area	Dataset size	Size of class 1	Size of class 2	Imbalance ratio	Size of feature vector
Climate Model Simulation Crashes	Physical	540	494	46	10.739	18
Ecoli	Life	336	143	193	1.350	8
Fertility	Life	100	88	12	7.333	10
Glass Identification	Physical	214	163	51	3.196	10
Heart Disease	Life	270	120	150	1.25	14
Hill-Valley	Physical	1212	612	600	1.02	100
Ionosphere	Physical	351	225	126	2.024	34
Connectionist Bench	Physical	208	97	111	1.144	60
Breast Cancer Wisconsin (Diagnostic)	Life	569	212	357	1.684	32
Wholesale customers	Business	440	142	298	2.099	8
LSVT Voice Rehabilitation	Life	126	84	42	2	309
Steel Plates Faults	Physical	1941	1268	673	1.884	27
Parkinsons	Life	195	147	48	3.063	23
Spect_heart	Life	267	212	55	3.855	22

Table 5.1. Databases for experiments of testing the classification algorithms

I selected almost all the dataset (in total 14) in UCI which satisfy the following conditions:

- The dataset are used for a binary classification algorithm.

- The dataset do not include missing values.
- Most features of the dataset are integer and real features.
- The number of instances is less than 2000.

The databases are from many areas like physics, geography, medicine, material science, agronomy, business, etc and all of them have been tested and published by scholars and scientists. Moreover, there are also some imbalanced dataset and dataset with more features than instances. Thus, in this chapter we will attempt to argue the robustness of classifier based on that it has a better classification performance on most dataset.

In the classification task, we use all the data and features in each dataset. Then for each experiment on one dataset, the division of training and testing datasets was randomized 100 times equally for the purpose of increasing statistical confidence or minimizing the "random" (or statistical variation). In the division, the two classes were equally distributed in training and testing sets, which means that the number of class 1 observations and class 2 observations were the same in both datasets, respectively. Finally, The 100 classification outcomes were averaged for the final result.

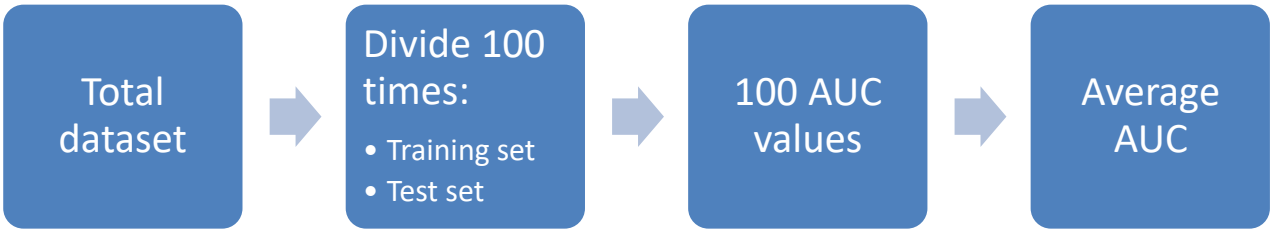


Fig. 5.1. How data are separated and the experimental design.

5.2 Results of experiments on LI classifier and the mixture classifier

Table 5.2(a) and 5.2(b) illustrates the classification results of 3 different individual algorithms: LI, SVM, RF and 4 different mixture models over 100 runs of experiments based on the imbalanced and balanced data, respectively.

We can first observe that the LI classifier is quiet robust that it is hard to argue whether SVM or RF is better than LI classifier because each of them has a better performance on some dataset but fail on the other. With a not bad performance, the other advantage of LI is that this algorithm takes less time than SVM and RF, which need tune some parameters before constructing the model, especially for the large dataset. We can further apply some parallel computing or map-reduce algorithms on it, which are hard to apply on SVM and RF.

Dataset name	LI result	SVM result	RF result	Sum-rule	Product-rule	Mixture without classifier selection	Mixture

Climate Model Simulation Crashes	0.9323	0.9449	0.8847	0.9414	0.9413	0.9422	0.9451
Fertility	0.6749	0.5973	0.6429	0.6457	0.6441	0.6523	0.6468
Glass Identification	0.9795	0.9547	0.9810	0.9819	0.9819	0.9822	0.9822
Parkinsons	0.9603	0.9182	0.9534	0.9560	0.9557	0.9561	0.9561
Ionosphere	0.9516	0.9739	0.9728	0.9774	0.9795	0.9777	0.9777
Wholesale customers	0.9579	0.9499	0.9548	0.9591	0.9557	0.9585	0.9593
LSVT Voice Rehabilitation	0.84	0.8815	0.8633	0.8883	0.8871	0.8888	0.8897
Spect_heart	0.8490	0.7986	0.8381	0.8384	0.8385	0.8379	0.8379

Table 5.2(a). Averaged AUC information of LI, RF, SVM and mixture classifier algorithms for imbalanced data.

From table 5.2, 5 of 8 imbalanced datasets show some gains after the mixture combination method while all the balanced datasets obtain gains with the mixture combination. Moreover, compared with two typical combination rules, the mixture classifier strategy (without the classifier selection) is better than the sum-rule and product-rule for most cases, and further the selection step help to choose more important classifiers into the mixture strategy, which is also indicated by the results.

Dataset name	**LI result**	**SVM result**	**RF result**	**Sum-rule**	**Product-rule**	**Mixture without classifier selection**	**Mixture**
Ecoli	0.9860	0.9889	0.9889	0.9892	0.9895	0.9897	0.9897
Heart Disease	0.8865	0.8998	0.8976	0.9046	0.9050	0.9049	0.9049
Connectionist Bench	0.8989	0.9127	0.9067	0.9213	0.9210	0.9212	0.9220
Breast Cancer Wisconsin (Diagnostic)	0.9883	0.9947	0.9898	0.9942	0.9940	0.9946	0.9949

Table 5.2(b). Averaged AUC information of LI, RF, SVM and mixture classifier algorithms for balanced data.

Thus, regarded as 3 strong classifiers, LI, SVM and RF are selected to construct the mixture classifier. Compared with its 3 components, the mixture classifier achieves the largest AUC value in most datasets (9 in 12, 2 of them take too much times). Furthermore, we also play a Wilcoxon signed-rank test between mixture classifier and its components. The results are illustrated in Table 5.3 as follows:

Mixture classifier	LI	SVM	RF
Climate Model Simulation Crashes	Yes	No	Yes
Ecoli	Yes	No	No
Fertility	No	Yes	No
Glass Identification	Yes	Yes	No
Heart Disease	Yes	No	Yes
Parkinsons	No	Yes	No
Ionosphere	Yes	Yes	Yes
Connectionist Bench	Yes	Yes	Yes
Breast Cancer Wisconsin (Diagnostic)	Yes	No	Yes
Wholesale customers	No	Yes	Yes
LSVT Voice Rehabilitation	Yes	Yes	Yes
Spect_heart	No	Yes	Yes

Table 5.3. Wilcoxon signed-rank test between mixture classifier and its 3 components. (Yes: the result of mixture classifier is significantly larger (P-value<0.1). No: the result of mixture classifier is not significantly larger.) Dataset with blue is the imbalanced data and yellow is the balanced data.

Table 5.3 shows us that, in most dataset, the AUC of mixture classifier is at least significantly larger than the AUC values of two of its components. Therefore, it indicates that the mixture classifier has a stable and consistent performance on most datasets. Because of the no free lunch theorem, we have no idea on which is the best algorithm for a specific task. However, the mixture classifier will first do a rough selection on single classifiers to delete some bad classifiers on the dataset and then combine the selected classifiers as a more robust classifier, which can beat its component.

5.3 Results of experiments on RF embedded location index

It is shown from table 5.4(a) and 5.4(b) that some AUC gains can be obtained from the LIRF classifier for all the dataset except the climate dataset. This result also hints that the LI classifier can classify the type 1 points introduced in chapter 4.2 perfectly, and then the divide and conquer strategy works. Moreover, the speed of this algorithm is as fast as RF, so that we can make use of it on

some large dataset.

Dataset name	LI result	RF result	LIRF
Climate Model Simulation Crashes	0.9323	0.8847	0.9323
Fertility	0.6749	0.6398	0.6755
Glass Identification	0.9795	0.9807	0.9874
Ionosphere	0.9516	0.9725	0.9774
LSVT Voice Rehabilitation	0.8400	0.8643	0.8745
Parkinsons	0.9549	0.9410	0.9551
Connectionist Bench	0.8989	0.9054	0.9077
Spect_heart	0.8490	0.8367	0.8541
Wholesale Customers	0.9579	0.9549	0.9597

Table 5.4(a). Averaged AUC information of 100 runs under LI, RF and LIRF algorithm on imbalanced data.

Dataset name	LI result	RF result	LIRF
Ecoli	0.9860	0.9889	0.9890
Heart_Disease	0.8865	0.8986	0.9019
Hill_valley	0.6144	0.5949	0.6152
Connectionist Bench	0.8989	0.9054	0.9077
Steel_plates	0.8119	0.8747	0.8761
Breast Cancer Wisconsin (Diagnostic)	0.9883	0.9898	0.9907

Table 5.4(b). Averaged AUC information of 100 runs under LI, RF and LIRF algorithm on balanced data.

In addition to the improvement on the AUC value, we can also notice that the result of LIRF is significantly larger than the result of its two components in 6 dataset of 14 from table 5.5. A more interesting fact is that the result of 5 of 8 datasets with LIRF show a significant gain both from LI and RF. Thus, I will argue that this embedded classifier is robust when dealing with the imbalanced data. For other datasets, the results of LIRF are still significantly larger than the result of RF. The only insufficiency of LIRF is that we need a grid search to find optimal parameters for different dataset and it consume some times.

LIRF	LI	RF
Climate Model Simulation Crashes	No	Yes
Ecoli	Yes	No
Fertility	No	Yes
Glass Identification	Yes	Yes
Heart_Disease	Yes	No
Hill_valley	No	Yes
Ionosphere	Yes	Yes

LSVT Voice Rehabilitation	Yes	Yes
Parkinsons	No	Yes
Connectionist Bench	Yes	No
Spect_heart	Yes	Yes
Steel_plates	Yes	No
Breast Cancer Wisconsin (Diagnostic)	Yes	Yes
Wholesale Customers	Yes	Yes

Table 5.5. Wilcoxon signed-rank test between LIRF classifier and its 2 components. (Yes: the result of LIRF is significantly larger (P-value<0.1). No: the result of LIRF is not significantly larger.) Dataset with blue is the imbalanced data and yellow is the balanced data

5.4 Results of experiments on LI forest

Table 5.6 shows that the average classification results over LI and LI forest, respectively. An obvious improvement of classification accuracy is obtained in 12 of 14 dataset after we use the ensemble method on LI classifier. Then a Wilcoxon signed-rank test was performed and the result is shown in Table 5.7 by comparing the AUC values before and after we use the ensemble method, where the P-value of most dataset <0.1, indicating the LI forest classifier is more robust in classification task than the LI itself. Especially in the balanced dataset, we can always observe a significant improvement after we use the ensemble method, which means the variance of the model brings more errors into the classification than the imbalance of the dataset does. Moreover, when applying the parallel computing on the LI forest algorithm, the speed is much faster than the LI, which is another advantage of the LI forest algorithm.

Dataset name	**LI result**	**LI forest result**
Climate Model Simulation Crashes	0.9323	0.9480
Glass	0.9795	0.9786
Ionosphere	0.9516	0.9643
LSVT	0.84	0.9152
Parkinsons	0.9549	0.9570
Steel_plates	0.8119	0.8167
Wholesale	0.9579	0.9591

Table 5.6(a). Averaged AUC information of 100 runs under LI and LI forest algorithm with imbalanced data.

Dataset name	**LI result**	**LI forest result**
Ecoli	0.9860	0.9861
Heart	0.8865	0.8947
Hill_valley	0.6144	0.6257
Connectionist Bench	0.8989	0.9080
Breast Cancer Wisconsin (Diagnostic)	0.9883	0.9912

Table 5.6(b). Averaged AUC information of 100 runs under LI and LI forest algorithm with balanced data.

LI forest	LI
Climate Model Simulation Crashes	Yes
Ecoli	No
Glass	No
Heart	Yes
Hill_valley	Yes
Ionosphere	Yes
LSVT	Yes
Parkinsons	No
Connectionist Bench	Yes
Steel_plates	Yes
Breast Cancer Wisconsin (Diagnostic)	Yes
Wholesale	No

Table 5.7. Wilcoxon signed-rank test between LI and LI forest classifier. (Yes: the result of LI forest is significantly larger (P-value<0.1). No: the result of LI forest is not significantly larger.) Dataset with blue is the imbalanced data and yellow is the balanced data.

Chapter 6 Application II: Diagnosis of Colon Cancer

6.1 INTRODUCTION

According to the recent statistics from American Cancer Society (ACS), colorectal carcinoma (CRC) is the third most commonly diagnosed cancer and the second leading cause of cancer-related death in the United States (ACS 2014). It was estimated that 142,820 new cases would be diagnosed with 50,830 dying from the disease in 2014. Fortunately, most CRC arises from colorectal polyps, and the process could take 5-15 years for malignant transformation into cancer. Thus, early detection and removal of the polyps before or during the malignant transformation will effectively decrease the incidence rate of CRC (B. Levin et al. 2008).

Clinical optical colonoscopy (OC) is currently the gold standard for detection and removal of the polyps. Because of its invasive nature, OC would demand a prohibitory resource to screen the large population with age over 50 (J. Liang and R. Richards 2010). Computed tomography colonography (CTC) has been under development over the past decades to relieve the burden of OC for the screening purpose and has shown comparable performance to OC with computer-aided detection (CADe) of the polyps sized 8mm and larger (C. Johnson et al. 2008).

With an increasing number of polyps as detected by the CTC screening, the need for removal of the detected polyps will increase and eventually would also demand a great resource to reduce the incidence rate of CRC. Fortunately, in the screening population of age 50 and older, a significant amount of the polyps are non-neoplastic, named hyperplastic (H) (D. Lieberman et al. 2005, P. Pickhardt and D. Kim 2009), which are abnormal growths with no risk. Removal of these growths would gain nothing, but will consume a great resource.

Efforts have been devoted to differentiate H from adenomatous (A) (or neoplastic) polyps in both OC and CTC fields by the measures of polyp size and surface characteristics, and the gain is limited, e.g. in the studies (P. Pickhardt and D. Kim 2009, P. Pickhardt et al. 2013), where CTC was performed for polyp screening and followed by OC to resect the found polyps, more than 20%

resections were hyperplastic. Fig. 6.1 illustrates five typical examples of polyps, of which two are H and three are A polyps. From the screenshots of endoscopic views of these polyps, it is clear the differentiation task is quite challenging if only the shape, surface property and size of the polyps are considered. More information is needed.

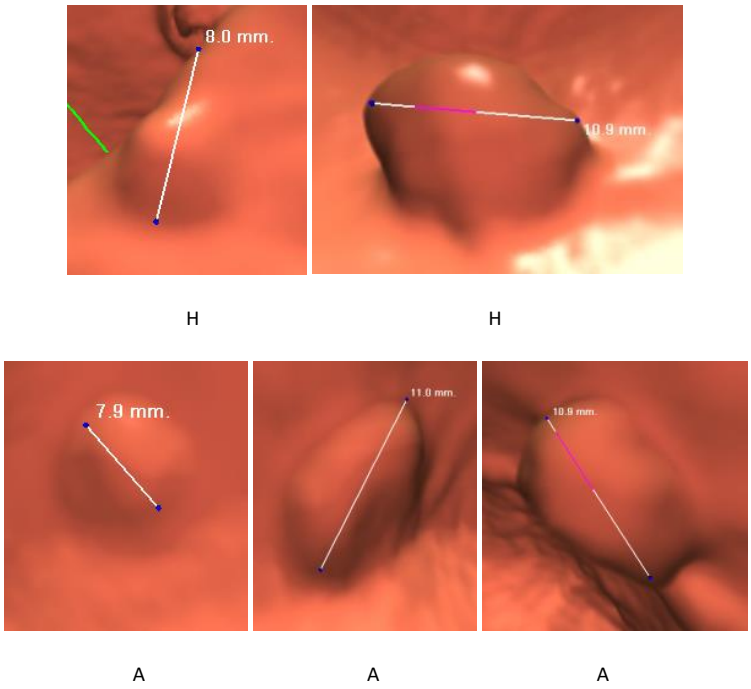


Fig. 6.1 An illustration of five typical polyps, which were randomly selected from the database in this work. (H) – hyperplastic and (A) – adenomatous.

By the CTC screening, fully three-dimensional (3D) volume image data (also including the size and surface characteristics) are readily available not only for the purpose of polyp detection (by either human observer or computer observer–CADe) but also for the possibility of polyp differentiation and other clinical tasks beyond the detection purpose. This study aims to explore the feasibility of differentiating H from A polyps (by a computer-aided diagnosis–CADx strategy) using texture features derived from the 3D volumetric data.

While many texture features have been extracted and applied for various clinical purposes, e.g. (G. Castellano et al. 2004, C. Showalter et al. 2006, J. Yao et al. 2011), the feature extraction method of Haralick *et al.* (R. Haralick et al. 1973) is attractive, because it gives a series of texture measures about the image intensity correlations among the image pixels on an image slice. Because of its attractiveness, efforts have been devoted to expand the Haralick's method from 2D domain into 3D space to compute the texture measures among the image voxels and apply the 3D models for the CADe and CADx tasks (C. Philips et al. 2008, B. Song et al. 2014). An important issue in the expansion is how to handle the spatial variation of computing the texture measures from the 2D domain to the 3D space where the shapes and orientations of the

polyp volumes can change dramatically. This study presents a simple idea to handle this spatial variation.

To our knowledge, most (if not all) texture features are derived from intensity images. In producing the intensity images, various efforts have been devoted to smooth the image except at the objects' borders in the image, because of inconsistence in acquired data due to noise and other measurement errors. During the piecewise smoothing, texture features would be sacrificed. To compensate for this loss, we have proposed a way to amplify the textures, similar to the spatial scale magnification in microscopy, by performing derivative operations on the intensity image (B. Song et al. 2014). This study will incorporate the simple idea of derivative amplification operations with the simple idea of handling spatial variation as an integrated adaptive approach to extract the volumetric texture features for the ultimate goal of differentiating H from A polyps.

The remainder of this paper is organized as follows. In Section II, a review of the Haralick method and its expansion from 2D to 3D space is given, followed by a presentation of our strategy of handling the 3D spatial variation. Then, a description of incorporating our texture amplification strategy to extract texture features in the derivative space is detailed. In Section III, experimental design for evaluating the extracted volumetric texture features is outlined and the results are reported with comparison to the previous method. Finally, discussion and conclusions are given in Section IV.

6.2 Methods

II.A. Review of the 2D Haralick Method for Texture Feature Extraction

In 1973, Haralick et al. introduced a method for texture feature extraction from 2D intensity or gray-level image (R. Haralick et al. 1973). By this method, a co-occurrence matrix (CM) is first defined and then applied to capture the gray-level correlations among resolution cells or image pixels in a 2D image slice. In implementation, a total of 14 texture measures along a direction through the image slice are calculated from the CM. The 14 texture measures are listed in (R. Haralick et al. 1973). A total of four directions (0, 45, 90 and 135 degrees) are defined on the image plane which are sufficient to span over the image slice, see Fig 6.2. Assuming a similarity among the four directions, an average of each of the 14 measures over the four directions is computed as the corresponding texture feature, resulting in a total of 14 mean features. Additionally, the range of each of the 14 measures is also computed as another texture feature to reflect spatial variation, resulting in a total of 14 range features. Thus, a total of 28 texture features (14 means and 14 ranges) are obtained, which are usually called Haralick features in the literature. The definition of the CM and the computation of the average and range over the directions together reflect the core idea of the Haralick method. The core idea is called Haralick model hereafter.

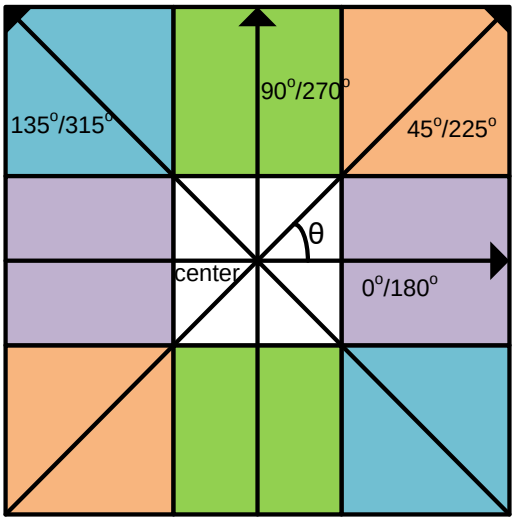
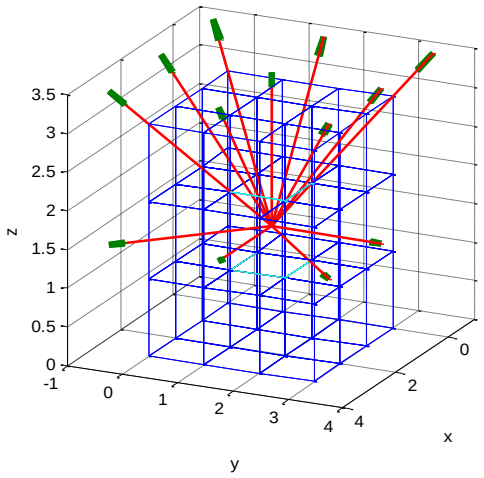


Fig. 6.2 Illustration of the 2D Haralick method for extraction of texture features with image pixel size unit of d = 1 and four directions in an image slice.

II.B. Expansion of the Haralick Model from 2D to 3D Space

As mentioned above, because of the attractive nature in using the CM to capture the gray-level correlations, a great effort has been devoted to expand the Haralick model from 2D plane domain to 3D volume space (B. Song et al. 2014). Similar to the selection of the four directions (from the 8 neighbors) in the 2D case of Fig. 6.2, a total of 13 directions (from the 26 neighbors) in the 3D space can be selected as shown by Fig. 6.3. Along each direction, 14 texture measures can be computed using the CM as defined by the Haralick model. Using the same philosophy as the Haralick model did, the average and range values of each of the 14 measures over the 13 directions can be computed as the 3D texture features, resulting in a total of 28 features (G. Zhang et al. 2012). The computed mean and range features are called 3D-Haralick features hereafter (in contrast to the 2D-Haralick features from a 2D image slice) or simply intensity features. While the mean and range can reflect some degree of the spatial variation of the texture measures along the 13 directions, a more adaptive strategy is desirable, e.g. Philips *et al*. (C. Philips et al. 2008) analyzed the directional variation on the CM measures within the liver. In this study, we explore an adaptive approach to extract and select the 3D texture features as detailed in the section of **Feature Extraction** below, instead of taking the mean and range features as the Haralick model did.



$A_1 = (1, 0, 0)$; $A_2 = (0, 1, 0)$; $A_3 = (0, 0, 1)$; $A_4 = (1, 1, 0)$;

$A_5 = (1, -1, 0)$; $A_6 = (1, 1, 1)$; $A_7 = (1, 0, 1)$; $A_8 = (1, -1, 1)$;

$A_9 = (0, -1, 1)$; $A_{10} = (-1, -1, 1)$; $A_{11} = (-1, 0, 1)$; $A_{12} = (-1, 1, 1)$;

$A_{13} = (0, 1, 1)$.

Fig. 6.3 The 3D resolution cells or image voxels for one center voxel in 13 directions. The direction Ai (i=1, 2, …, 13) is equivalent to (x, y, z), which is a direction in the 3D coordinates. The center point is A0 = (1.5, 1.5, 1.5) and the 13 arrows beginning from this point is represented as A0+k* Ai, where k does not equals to 0.

II.C. Expansion of Volumetric Texture Measures

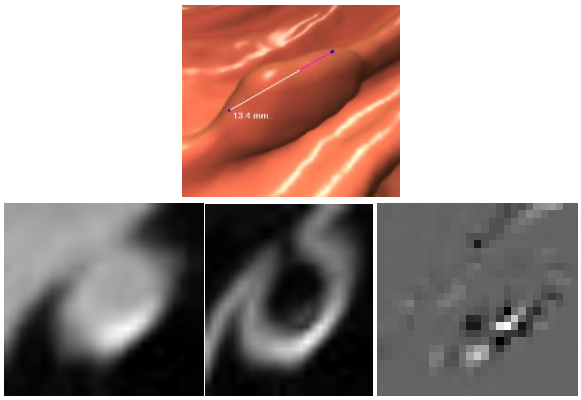
While the 14 texture measures of the Haralick model were designed to reflect some statistics or information about the pixel and pixel correlation, more measures can be designed to reflect a complete picture about the correlation. In this study, we introduce the following 16 new texture measures, which can be computed from each CM in the 3D space. (It is noted that the definitions of the notations in these new measures are the same as that of the 14 measures :

Feature number	**Description**
$f_{15} = \max_{i,j} p(i, j)$	15. Maximum probability.
$f_{16} = \text{median}_{i,j}\, p(i, j)$	16. Median probability.
$f_{17} = \text{firstquantile}_{i,j}\, p(i, j)$	17. First quantile probability
$f_{18} = \text{thirdquantile}_{i,j}\, p(i, j)$	18. Third quantile probability
$f_{19} = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\{p(i, j)\}^2}{N_g^{\,2}} - \left(\frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i, j)}{N_g^{\,2}}\right)^2$	19. Deviation of probability.
$f_{20} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i * j)\, p(i, j)$	20. Autocorrelation.

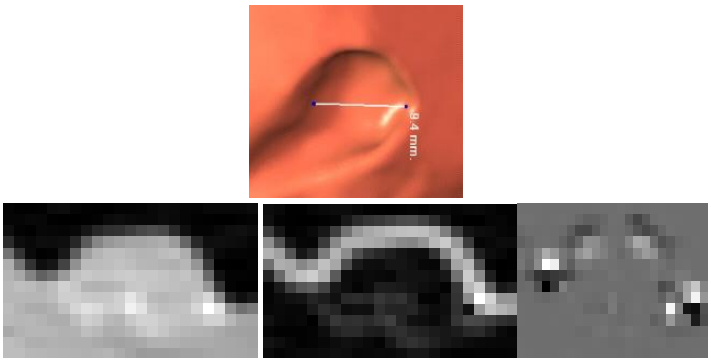
$f_{21} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)\,p(i,j)$	21. Cluster average.
$f_{22} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)^2\,p(i,j)$	22. Cluster variance.
$f_{23} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)^3\,p(i,j)$	23. Cluster shade.
$f_{24} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i+j-\mu_x-\mu_y)^4\,p(i,j)$	24. Cluster prominence.
$f_{25} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\lvert i-j\rvert\,p(i,j)$	25.Dissimilarity.
$f_{26} = \frac{p(i,j)}{1+\lvert i-j\rvert / N_g}$	26. Inverse difference.
$f_{27} = \frac{p(i,j)}{1+\lvert i-j\rvert}$	27.Homogeneity II.
$f_{28} = \frac{HXY-HXY2}{\max\{HX,HY\}}$	28. Information measures of correlation III.
$f_{29} = (1-\exp[-2.0(HXY1-HXY)])^{\frac{1}{2}}$	29. Information measures of correlation IV.
$f_{30} = \sum_{i=0}^{N_g-1} i p_{x-y}(i)$	30. Difference average.

II.D. Texture Amplification

As mentioned above in the section of **Introduction**, when reconstructing the intensity images from acquired noise data, especially low-dose CT data, efforts have been devoted to ensure image smoothness except on the borders of objects in the image, sacrificing more or less textures in the reconstructed images. Performing derivative operations across the intensity image, similar to an amplification process, is a simple way to enhance or recover the textures. For example, in the 1st derivative or gradient image, the voxels on the border of an object will have maximal values while the voxels within the object (or a very flat area) will have nearly zero values; and other voxels will have values ranging from nearly zero to the maximum value. Intuitively, in the 2nd and higher derivative images, those voxels on the gradient regions in the intensity image will have non-zero values while others will have zero value. In this paper, we adopted our previous study (B. Song et al. 2014) to include only the 1st- and 2nd- order derivative images to test our adaptive approach to handle the spatial variation in extracting the 3D texture features. The details of computing the derivative images are given in the reports (K. Engal et al. 2006, O. Monga and S. Benayoun 1991). Fig. 6.4 shows the different texture pattern characteristics in the three images of the original intensity, 1st- and 2nd- order derivatives about H and A polyps.



(H)



(A)

Fig. 6.4. An illustration of two different polyp types (of H and A) and their corresponding endoscopic views and image slices, where the image slices are crossing the lines in the endoscopic views, respectively. The sizes of the two polyps are around 10mm. The three slices from left to right are intensity, gradient and curvature images.

Given the acquired 3D intensity image I, and the computed 1st-order derivative (or gradient) image Ig and the 2nd-order derivative (or curvature) image Ic, the 3D-expanded texture feature extraction can be performed as follows.

II.E. Feature Extraction

Given the 3D gradient and curvature images, we can compute the 30 texture measures (14 from Haralick et al. and 16 new ones from section **II.C** above) from the defined CM along each direction in each image as we did for the intensity image (in section **II.B** above), resulting in a vector of dimension of $2 \times 30 = 60$. By including the texture measures from the intensity image, we obtain a vector of dimension of $3 \times 30 = 90$. Repeating the calculation along all the 13 directions, we obtain a total of 13 vectors with a dimension of 90. By adopting the Haralick model of taking the average and range values of each of the 90 measures over the 13 directions as the texture features, we obtain total of 180 3D-texture features. These 180 3D-texture features (including 60 3D-Haralick features from the intensity image, 60 3D-features from the gradient

image -- named gradient features, and 60 3D-features from the curvature image, -- named curvature features) are treated as the reference or baseline for comparison purpose in order to show the gain by the proposed adaptive approach in this study.

As we mentioned above that because of the spatial variation of polyp shapes and orientations in the 3D space, the average and range may not adequately reflect the entire volumetric texture features if the polyp shape is deviated from a sphere (such deviation is common in reality). Instead of computing the average and range of each texture measure over the 13 directions as the 3D-texture features, we would rather take an adaptive approach to address the spatial variation of the calculated CM measures over the 13 directions for the volumetric texture features of each polyp. In this exploratory study, we take the well-known principal component analysis, as an example, to address the spatial variation. Specifically, we take the Karhunen–Loève (KL) transform (R. Dony 2001) on the 13 vectors along the angular axis of the 13 directions. In the KL domain, a new set of 13 directions (or 13 ordered eigenvectors) are obtained which are less dependent on the 3D shape orientation of the polyp in the original 3D patent space. We hypothesize that such adaptive approach will improve the feature extraction and analysis and, therefore, improve the classification or differentiation performance. The experiments of this study will be designed to test this hypothesis. In the KL domain, the volumetric texture features are selected along the 13 ordered eigenvectors. More details on the KL transform procedure are given below, followed by feature selection and classification.

The key procedure in the KL transform is to use an orthogonal transformation to convert a set of observations into a new coordinate system with uncorrelated variables. In the new coordinate system, the mean squared error between the given set of observations and their projections on the new coordinates is minimized. Moreover, a high degree of redundant data is compressed into a more compact form after the removal of the correlation between the observations. Suppose we have M variables and each variable can be described by N observations. Let the observations be represented as N column vectors, $x_1, x_2, \ldots, x_N$, each of which has M elements or variables, making up the M row vectors. For this $M \times N$ matrix, named X, its covariance matrix, named T, can be computed. Since T is a symmetric matrix, an eigen decomposition can be perform on the matrix: $T = VDV^{\tau}$, where τ is the transpose operator, D is a diagonal matrix with all the eigenvalues of T and each column in V is the eigenvector correspond to the eigenvalue in D. After the eigen decomposition, a new $M \times N$ matrix can be calculated, $Y = V^{\tau} X$, which includes N new observations $y_1, y_2, \ldots, y_N$ described by M new variables.

For each polyp, the observations or the CM texture measures in our case are obtained on the 13 directions or variables, so $M = 13$. Along each direction, 30 texture measures are computed from the intensity, gradient image and curvature images, respectively, so $N = 3 \times 30 = 90$. If considering only one image for feature selection and analysis, e.g. intensity image, gradient image or curvature image, respectively, $N = 30$ (Int, Gra, or Cur). If considering two images, e.g. intensity image+gradient image, or intensity image+curvature image, or gradient

image+curvature image, N = 60 (combinations of Int_Gra, Int_Cur, and Gra_Cur). If all the three images are considered together, N = 90 (Int_Gra_Cur).

Since a main goal in this study is to explore adaptive approach to address the spatial variation, instead of taking the average and range on the CM texture measures, therefore, our focus now is on the KL-transformed CM texture measures. The gain by the KL transform is that the KL operation relieves the correlation of the CM measures along the 13 directions. Without any *a priori* knowledge on the spatial variation of the CM measures, we take the KL transformed CM measures as our new volumetric texture features, and then develop a suitable feature selection and classification strategy to analyze the more compact-formatted texture features for the ultimate goal of differentiating hyperplastic from adenomatous polyps.

Finally, all the algorithms introduced in Chapter 4 will be tested here to show the improvement based on each feature dataset.

II.F. Feature Selection and Classification

After the KL operation, we obtain a new set of features with dimension of $13 \times N$. For single image scenario N = 30, we have 390 features in 13 groups, and each group has 30 features. The 13 groups are called eigenvectors and are orderly arranged according to their eigenvalues in a decreasing manner. For the scenario of two image combinations N = 60, we have 780 features in 13 groups, and each group or each eigenvector has 60 features. For the scenario of all three image combined N = 90, we have 1,170 features in 13 groups, and each group or each eigenvector has 90 features. For each scenario, we adopt the Random Forest (RF) strategy (L. Breiman et al. 2001), which has the advantage in solving the problems without any *a priori* knowledge on the problem, to select and classify the de-correlated and more compact-formatted features.

II.F.1 Feature Selection

RF is a popular and efficient algorithm for classification and regression problems as described by Breiman. Since the key problem in feature selection is the computation of the importance of the features, the RF algorithm provides us a model which is not only efficient in computation but also low over-fitting errors in accuracy.

In this application, we employ a function of the R-package "randomForest" [50] to construct the "forest" and select the importance order on the tree notes. An average of the total decrease in node impurities over all trees is computed, which is measured by the Gini impurity. According to the CART algorithm in (L. Breiman et al. 1984), Gini impurity is a measure that reflects the mislabeled rate of a random element in the set:

$$GINI(t) = \sum_{i=1}^{m} p_i(1-p_i) = 1 - \sum_{i=1}^{m} p_i^{\,2} \tag{6.1}$$

where p_i is the probability that element t is correctly labeled as i. Each time, the Gini impurity of a node is greater than or equal to the sum of the Gini impurity of its two descendent nodes. With this property, the importance of the node t is calculated as:

$$\mathrm{Im}(t) = \frac{\sum_{i=1}^{n} GINI(t,i) - \sum_{i=1}^{n} GINI(t_{des},i)}{n} \tag{6.2}$$

where $GINI(t,i)$ is the Gini impurity for node t of i-th tree, and t_{des} is the descendent nodes of t.

By the above RF-embedded feature selection, we rank the importance of each feature in a decreasing order. The first feature in the order is most importance and the last one is least importance by the importance measure of Eq.(6.2).

Starting from the first feature on the order, we add the next ranked important feature to have a feature vector of dimension 2 and then perform classification on the feature vector to generate a measure of area under the curve (AUC) of the receiver operating characteristic (ROC). By repeating the two steps of (i) adding next ranked feature on the order into the current feature vector and (ii) performing classification on the new feature vector of increased dimension (by 1) until reaching the maximum dimension (i.e. all features in the order have been added together), we obtain a plot of feature dimension vs. its corresponding AUC measure. From the plot, the feature vector with largest AUC value is the best feature vector and its dimension is called intrinsic dimension. The classification operation is detailed below.

II.F.2 Feature Classification

Similar as feature selection, the RF strategy can be adopted for feature classification. In this study, we employ the R-package "randomForest" (A. Liaw and M. Wiener 2002) again to serve the purpose of feature classification, called RF-embedded feature classification (M. Ma et al. 2014, B. Song et al. 2012). For each classification experiment, since we have divided the data into training and testing datasets, we could build a classifier model with the training dataset information and then evaluate the model by the testing dataset. More details are given below.

As mentioned above in the **Abstract**, in total we have 384 polyp samples (half of them will be selected to be training set, i.e. 192 polyp samples of 26 H and 166 A), 390 features (in the scenario of individual images: Int / Gra / Cur), 780 features (in the scenario of two image combinations: Int_Gra / Int_Cur / Gra_Cur), and 1,170 features (in the scenario of all image combined: Int_Gra_Cur) for each polyp. By dividing the 384 polyp samples as training (192 polyp samples) and testing (192 polyp samples) datasets (the corresponding features of these samples are also divided), we first build up a RF model with the following parameters (1,000 trees and *m* features used in each tree):

$$\begin{cases} n\_trees = 1000 \\ m = \sqrt{M} \end{cases} \tag{6.3}$$

where M is the total number of features in a feature set, and then adjust the parameter *mtry* in (A. Liaw and M. Wiener 2002) until the Out-of-bag error (OOB error) stabilize to a low value. Here for each tree we only use a bootstrap sample set of the whole training dataset and the OOB error could be calculated via the other part of the training dataset. The feature set is a feature vector, selected from the features in the KL domain by the above described RF-embedded feature selection.

After building the RF model of Eq.(6.3) with training dataset, we could generate a single score (or posterior probability) for each test point based on the same RF model of Eq.(6.3) where the feature set is the testing dataset. The final classification decision is obtained by a majority vote law on all the classification trees (and an estimation of the probability of each class can also be deduced by calculating the proportion of each decision on all the classification trees). It is known that RF is an ensemble method and is constructed by a multitude of decision trees. Since each single decision tree can generate a result (0 or 1) during the classification, the final score of the test point will be decided by the votes (over all the trees in the forest).

Given the obtained scores, the ROC analysis can be used to obtain the AUC values for quantitative evaluation of the classification. By performing the ROC analysis on the scores, we can obtain the information about the best selection of feature subset and the highest AUC value of classification in all the three scenarios. The experimental design and outcomes are reported below.

6.3 Experimental Design and Results

III.A. Revision of the Original 14 Haralick Texture Measures

By examining the list of the original 14 Haralick texture measures, some discrepancies were found. The corresponding corrections and modifications are then made as follows.

Firstly, two typographical errors may have occurred in the 7th and 14th measures. In the 7th measure of Sum of Variance, $f_7^* = \sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$ was given in (R. Haralick et al. 1973). It is obvious that the expectation of f_6 in (R. Haralick et al. 1973) should be used for $f_7^* = \sum_{i=2}^{2N_g} (i - f_6)^2 p_{x+y}(i)$, instead of the entropy measure of f_8 in (R. Haralick et al. 1973). By the 14th measure of Maximum Correlation Coefficient, $f_{14}^* = (\text{second largest eigenvalue of } Q)^{\frac{1}{2}}$ where $Q(i,j) = \sum_{k=1}^{N_g} \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)}$, in (R. Haralick et al. 1973), it would reflect the correlation between the i-th row and the j-th column of the CM. Since the CM is symmetric, the correlation

coefficient matrix $Q(.)$ should be a symmetric matrix. So, a correction is made such that $Q(i,j) = \sum_{k=1}^{N_g} \frac{p(i,k)p(j,k)}{p_x(i)p_y(j)}$.

Secondly, for the 4th measure of Sum of Squares or Variance, $f_4 = \sum_{i,j=1}^{N_g,N_g} (i-\mu)^2 p(i,j)$, in (R. Haralick et al. 1973), the definition of μ is missing, which poses an ambiguity in the interpretation of the "Variance". Therefore, this measure is modified into a covariance measure in (R. Haralick et al. 1973): $f_4^* = \sum_{i,j=1}^{N_g N_g} (i-\mu_x)(j-\mu_y)p(i,j)$.

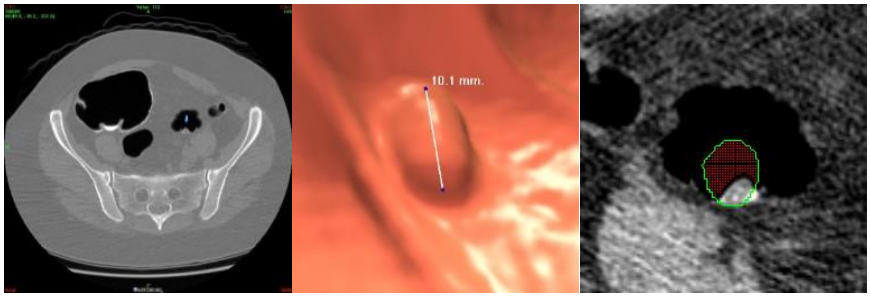
III.B. Database for Experimental Studies

The above presented 3D volumetric texture features are extracted from the original intensity and high-order (gradient and curvature) images of a CTC database of 352 scans from 176 patients. The patient studies were performed during the time period from 2009 to 2013 by a standard CTC protocol (ACR 2005), i.e. a low-volume cathartic bowel preparation, oral fecal tagging, without IV contrast, and multi-detector CT scanners in adherence. The image data were acquired in helical mode with collimations of 1.0–3.0mm, pitch of 1–2, reconstruction intervals of 1.0–1.5mm, and modulated tube current–time products of 50-200mAs and tube voltages of 80-120kVp. The indication for CTC was screening for CRC in all individuals. The protocol was approved by appropriate ethical committee, and the studies were performed in accordance with the ethical standards laid down in the 1964 Declaration of Helsinki and its later amendments. All patients gave their informed consent prior to their inclusion in the study and their identities were removed before the images were processed by the proposed texture extraction algorithms. Each patient was scanned at two positions, e.g. supine and prone, resulting in total of 352 scans. Due to some factors like the gravity, the two scans at supine and prone positions from the same patient might incur some changes in polyp shape and size, and thus these two scans were considered as two different datasets. The 352 scans include a total of 384 polyp datasets (polyp sizes ≥ 8mm: 52 are H and the rest 332 are A polyps according to their path reports, where the group A includes all types of adenomas: 32 serrated adenomas (SA), 200 tubular adenomas (TA), 67 tubulovillous adenomas (VA), 30 asenocarcinomas (AC) (C. Do et al. 2012). The clinical task here is to differentiate the 52 H from the 332 A polyps.

III.C. Semi-automatic Operation for Volume of Interests

Before performing the CADx task of differentiating a polyp's subtypes, that polyp should have been detected by a radiologist expert or a CADe pipeline with labeled coordinate (*x*,*y*,*z*) of that polyp in the CTC volume image data. For each detection with the labeled location (*x*,*y*,*z*), a volume of interest (VOI) for that polyp was first obtained so that texture features can be extracted from the VOI to determine its subtype. For that purpose, a semiautomatic technique, similar to those reported procedures (P. Pickhardt et al. 2013), was applied to extract the VOI.

The semiautomatic technique can be outlined as follows. Firstly the detected polyp is roughly outlined manually on the 2D image slices according to the reported detection location (x,y,z) using a software, e.g. the CTC software (V3D Colon, Viatronix Inc., Stony Brook, NY, USA), and then an automatic air-cleaning algorithm, which is based on the segmentation results, is applied to the outlined volume to remove air voxels for an air-free 3D polyp VOI. Fig. 6.5 illustrates an example where the steps for the VOI extraction are shown. From the obtained VOI, texture features are extracted as described above.



a b c

Fig. 6.5. Steps for semi-automatic extraction of VOI: (a): A report of polyp detected by a radiologist or a CADe algorithm (indicated by an arrow). (b): An endoscopic view of the polyp illustrated using the CTC software (V3D Colon, Viatronix Inc., Stony Brook, NY, USA). (c): A manual outline of the polyp on a 2D image slice (green circle), where the air voxels (red part within the outlines circle) is removed by our automatic air-cleaning algorithm.

III.D. Experimental Outcome

In this section, we will first evaluate the RF-embedded feature selection and then perform the RF-embedded feature classification on the newly extracted volumetric texture features with comparison to the baseline (or reference) volumetric texture features.

III.D.1. Performance of Feature Selection

From the Haralick's original 14 and the newly presented 16 texture measures in section **II.C** above, we have the baseline texture measures or pre-KL texture measures:

(1) 13×30 = 390 texture measures from the intensity image (Int); 390 measures from the gradient image (Gra), and 390 measures from the curvature image (Cur);

(2) 2×390 = 780 measures from each combinations of Int_Gra, or Int_Cur, or Gra_Cur;

(3) 3×390 = 1,170 measures from the combination of all the three images (Int_Gra_Cur).

The above pre-KL texture measures are also called pre-KL texture features hereafter. By computing the mean and range values of the above baseline measures over the 13 directions as the Haralick texture features, we have:

(1) 60 Haralick texture features from each of the images, Int, Gra, or Cur;

(2) 120 Haralick texture features from each combination of Int_Gra, or Int_Cur, or Gra_Cur;

(3) 180 Haralick texture features from the combination of all the three images (Int_Gra_Cur).

By applying the KL transform on the above baseline measures (or pre-KL texture features) along the 13 directions, we have the following corresponding features in the KL domain or post-KL texture features:

(1) 390 post-KL texture features from each of the images, Int, Gra, or Cur;

(2) 780 post-KL texture features from the combinations of Int_Gra, or Int_Cur, or Gra_Cur;

(3) 1,170 post-KL features from the combination of all the three images (Int_Gra_Cur).

Since there is no prior information on the ordering of the pre-KL texture features, the RF-embedded feature selection was performed randomly without any preference on any feature. After performing the selection, the features are ranked by their importance in a decreasing order for the three scenarios above: (1) 390 features for each individual image, (2) 780 measures for each two-image combination; and (3) 1,170 measures for all three image combination. For the post-KL features, the above presented RF-embedded feature selection was performed in the same way as in the selection of the pre-KL features.

To show the performance of the ranked features, different feature sets or feature vectors were selected along the ordering, as described by the last paragraph of section **II.F.1**. The division of training and testing datasets was randomized 100 times for the purpose of increasing statistical confidence or minimizing the "random" (or statistical variation). In the division, the H and A polyps were equally distributed in training and testing sets, which means that the number of H and A polyps were the same in both datasets, respectively. Moreover, for each randomized case, which is one of 100 independent experiments, the training dataset was only used for feature selection and modeling, and the testing dataset was used for evaluation. Then the above presented RF-embedded classification in section **II.F.2** was applied to each randomized case. The 100 classification outcomes were averaged for the final result. The final results can be plotted as a ROC curve for that selected feature set. The AUC value under that ROC curve is usually taken as a quantitative measure on the classification performance and, therefore, was used to indicate the quantitative measure on the repeated experimental outcomes. After all feature sets are processed, a plot can be drawn for the relationship between the AUC values and the selected feature sets. The plot is expected to increase from the feature set of smallest number of features (usually 1) up to reaching a peak at an optimal number of features (called intrinsic feature dimension), and then the plot generally drops down until the total feature set (including all features) was used. The higher the plot peak is, the richer the information embedded inside the features.

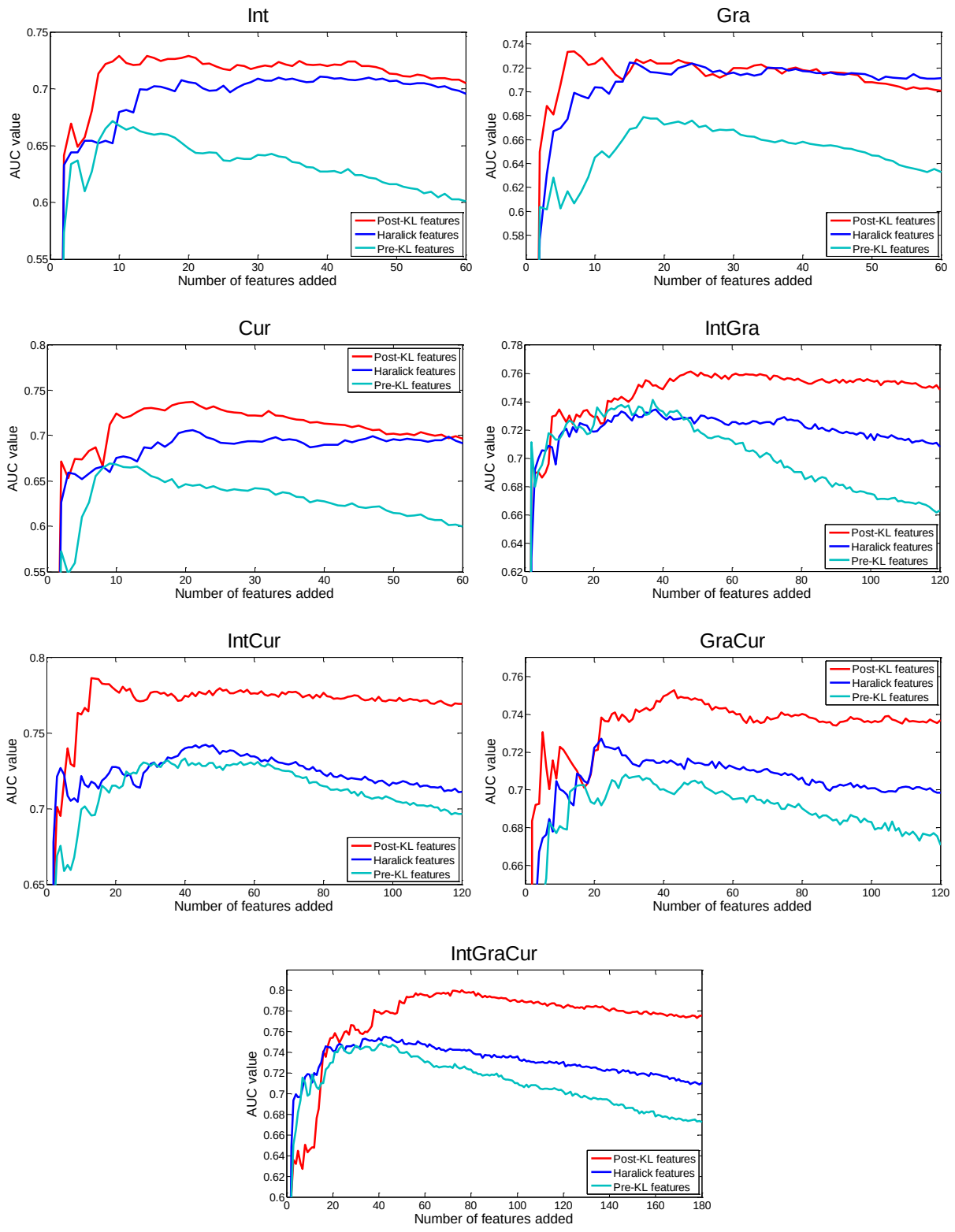


Fig. 6.6. Plots of AUC values vs. selected number of features, which were ordered by the RF-embedded feature selection.

Fig. 6.6 shows the RF-embedded feature selection performance on the pre-KL features, the Haralick features (i.e. the mean and range over the 13 directions), and the post-KL features for three scenarios: (1) individual images of Int, Gra or Cur; (2) two image combinations of IntGra, IntCur, or GraCur; and (3) all three images together of IntGraCur. For each scenario, the performances of the three feature extraction methods were compared as shown by Fig. 6.6. It is observed that the performances of the three methods diversified quickly along the ordered feature sets. All the curves started by increasing the AUC value and then decreased the AUC value after reaching their maximum at the peak. This outcome indicates that the RF selection is effective. Except for the Gra feature set, the curves of the post-KL features are always on the top. Furthermore, the post-KL features reached higher AUC values than the other two types of the pre-KL features and the Haralick features in all the three scenarios of individual images and combinations of multiple images. This experimental outcome indicates the gain by the proposed adaptive approach to addressing the spatial variation of polyp volume orientation in the patient space and the texture amplification of CT images. The pre-KL features did not perform better than the Haralick features for a possible reason that the latter have much small number of features for classification (a gain in curse of dimensionality and computing efficiency). In other words, the texture measures from the 13 directions are correlated, and the selection of the mean and range of the texture measures over the 13 direction as the features is a reasonable choice. However, the simple KL operation is shown to be much better than the selection of the mean and range.

III.D.2. Performance of Feature Classification

The procedure of feature classification was described in the section **II.F.2** above.

Table 6.1 shows the average classification results (AUC values) over 100 runs of the features extracted from the three methods. An obvious improvement of classification accuracy after considerations of the polyp orientation variation by the KL transform and the texture amplification by the derivative operation is seen for all the three scenarios of individual image and combinations of images. Moreover, a significant test was performed as shown in Table 6.2 by comparing the AUC values with and without considerations of the polyp orientation variation and texture amplification, where all of the P-values are <0.05, indicating that the proposed feature extraction model is significantly better than the Haralick feature extraction model. From Table 6.1, it can be seen that the gain by the use of the KL transform for the polyp orientation variation is $(0.8016-0.7553)/0.7553=6\%$ over the Haralick's average method. The gain by the use of the derivative operations for texture amplification is $(0.8016-0.7288)/0.7288=10\%$ for the KL transformed features. The performance in differentiating non-risk group (H) from the risk group (A) reached an AUC value of 0.8016 by the proposed adaptive approach.

TABLE 6.1. Averaged AUC information of the 100 runs before and after KL-transform

Group	AUC information		
	Pre-KL Features	Haralick Features	Post-KL Features
Intensity	0.6716±0.0399	0.7105±0.0334	**0.7288±0.0404**
Gradient	0.6789±0.0384	0.7244±0.0348	0.7339±0.0368
Curvature	0.7057±0.0421	0.6693±0.0394	0.7369±0.0377
Int_Gra	0.7346±0.0369	0.7414±0.0443	0.7613±0.0393
Int_Cur	0.7330±0.0369	0.7421±0.0379	0.7862±0.0399
Gra_Cur	0.7082±0.0351	0.7270±0.0343	0.7528±0.0401
Int_Gra_Cur	**0.7487±0.0436**	**0.7553±0.0377**	**0.8016±0.0352**

Format: mean ± standard deviation

TABLE 6.2: Wilcoxon signed-rank test between the AUC of the post-KL features and the corresponding pre-KL and Haralick features

Post-KL features	Int	Gra	Cur	Int_Gra	Int_Cur	Gra_Cur	Int_Gra_Cur
Haralick features	<<0.05	0.0176	<<0.05	<<0.05	<<0.05	<<0.05	<<0.05
Pre-KL features	<<0.05	<<0.05	<<0.05	<<0.05	<<0.05	<<0.05	<<0.05

The p-value is the result of the Wilcoxon signed-rank test because the normality assumption for t-test is not hold.

Since all the three feature extraction methods can reach their peak AUC values, their corresponding averaged ROC curves are plotted as shown by Fig. 6.7. The threshold averaging or operating point selection strategy was used to obtain the curves (T. Fawcett 2006, B. Song et al. 2014). The curves are consistent with the AUC values in Table 6.1. From these curves, it is seen that the post-KL features reached a higher sensitivity value, under the same specificity, than the other two methods. That is to say, the post-KL features could provide a better classification result than the other two methods.

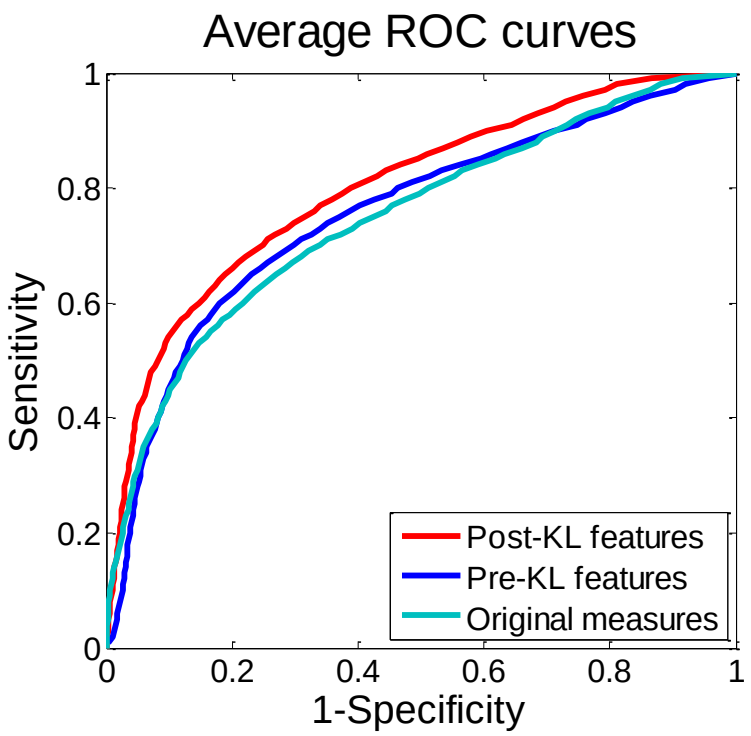


Fig. 6.7. The averaged ROC curves of the results from the three methods corresponding to their highest AUC values.

From the average ROC curves, different sensitivity-specificity paired values can be generated. Table 6.3 shows the average specificity levels for different fixed sensitivity levels and feature groups, based on the 52 H polyps and the 332 A polyps. For example, if we choose 0.75 sensitivity level for the post-KL feature group, its corresponding specificity will reach 0.6859. As a result, by employing a simple RF classifier, the number of correctly classified A polyps is 249 (of 332) and the number of H polyps is 35.67 (of 52) on an average. It is obvious that a higher specificity with the post-KL features can be always achieved for each fixed sensitivity value in the table, which indicates that the post-KL features do perform better than the other two feature extraction methods.

TABLE 6.3. Sensitivity-specificity pairs corresponding to the averaged ROC curves of the three methods in Fig. 6.7.

Group	Corresponding specificity with different sensitivity level			
	Sen=0.6	Sen=0.7	Sen=0.8	Sen=0.9
Post-KL features	0.8533	0.7509	0.6123	0.3955
Haralick features	0.8196	0.7009	0.5375	0.2884
Pre-KL features	0.7839	0.6594	0.4891	0.2865

TABLE 6.4: Averaged AUC information of the 100 runs before and after KL-transform

Group	AUC information		
	Pre-KL Features	Haralick Features	Post-KL Features
Intensity	0.6295±0.0355	0.6855±0.0387	0.6998±0.0443
Gradient	0.6664±0.0329	0.7188±0.0376	0.7354±0.0421
Curvature	0.6544±0.0430	0.6376±0.0456	0.6801±0.0384
Int_Gra	0.7358±0.0388	0.7442±0.0410	0.7454±0.0379
Int_Cur	0.7300±0.0377	0.7389±0.0360	0.7601±0.0436
Gra_Cur	0.7021±0.0391	0.7005±0.0368	0.7436±0.0421
Int_Gra_Cur	0.7415±0.0351	0.7538±0.0383	0.7723±0.0413

Format: mean ± standard deviation

In addition to the above investigations on the variation of polyp orientations and the texture amplification for the low tissue contrast CT images, we further performed experiments to show the gain by the 16 new texture measures of section **II.C**. The results are illustrated in Table 6.4, which have the similar notations as Table 6.1. The highest AUC values are shown in Table 6.5.

TABLE 6.5. Highest averaged AUC information of the 100 runs

	Post-KL features	Haralick features	Pre-KL features
30 measures	0.8016	0.7553	0.7487
14 measures	0.7723	0.7538	0.7415

By comparing Table 6.1 and Table 6.4, we can see that most of the feature sets have some gains by adding the 16 new texture measures into the corresponding feature sets. Using the post-KL feature extraction method as an example, the gain is $(0.8016-0.7723)/0.7723=3.8\%$. The p-value under the Wilcoxon signed-rank test is less than 0.05, which indicates that adding the 16 new texture measures could provide more information than the original 14 texture measures.

TABLE 6.6. Algorithms comparison about the averaged AUC information of the 100 runs

	SVM	RF	LI	LIRF	LI forest
KL	0.7345	0.8016	0.7821	0.8087	0.7904
Haralick	0.6584	0.7553	0.7582	0.7689	0.7628

The results in Table 6.6 shows that we obtain the best classification result with the LIRF. In the meantime, both of the LI forest results are better than the LI result, as well as the time-consuming.

6.4 Discussion and Future Work

The spatial variation of polyp volume in the patient space is a common situation, thus an adaptive approach to address the variation is desired. The use of KL transform to address the variation in this exploratory study is just a simple example. Similarly, image reconstruction usually takes some penalties to smooth data noise, resulting in some loss of textures. The use of the derivative operation to amplify the textures is another simple example. Extracting more texture measures from the polyp volume is always desired. The addition of the 16 new texture measures in this study has shown a noticeable gain. Exploring other strategies for the spatial variation, texture amplification and extraction of more new texture measures are our future research interests.

Since screening the large population is the main purpose of developing CTC, the newly proposed technologies above remain the same screening purpose while aiming to advance the current CTC paradigm of detection-only capability to a new paradigm of not only detection but also characterization of the detections. By deviating from the screening purpose, efforts have been devoted to differentiating neoplastic from non-neoplastic lesions (polyps and masses) by the use of intravenous (IV) contrast-enhanced CTC protocol (F. Ng et al. 2013). The gain by the IV-contrast-enhanced CT image textures is at the cost of the complication of IV related procedure, which would compromise the screening purpose. An alternative attempt of gaining more CT image texture information is to use energy spectral CT (EsCT) (B. Schaeffer et al. 2014) at the cost of increased radiation dose to the patient. Reducing the dose while retaining the EsCT image textures has been a topic of our research interests (Y. Liu 2014).

The database in this study includes polyps of size 8mm and larger. The size threshold of 8mm was chosen because it is currently believed to be clinically desired (P. Pickhardt and D. Kim 2009). Ideally, we would like to perform the classification on polyps with different size ranges, such as from (a) 5mm to 10mm, (b) 10mm to 15mm, (c) 15mm to 20mm, (d) 20mm to 30mm, and (e) 30mm and larger, where the size could be included as a feature in the feature set. Increasing the number of polyps and performing the classification on different polyp size ranges are another research interest of our future research effort.

By current CTC protocol, a patient is usually scanned at two positions of supine and prone, resulting in two sets of image data. Because the body turns over from supine to prone position, the entire colon changes significantly in shape, orientation and size due to mainly the gravity. Since the two image datasets are two statistically independent observations from a source, which changes significantly, researchers in the CTC field usually treat the varying source in the two datasets as two different sources, particularly when the number of sources is small (i.e. small sample size). In theory, there may be some bias in treating the same source as two different ones in the situation. However, for bi-classification, this concern would be relieved because the sample numbers of both hyperplastic and adenomatous polyps are doubled and the relative bias

would be small. This hypothesis would be tested when the number of sources (or sample size) is large. This is one of our future research topics.

The simplicity and effectiveness of RF decision are attractive in theory and applications. In this study, RF was used for feature selection and classification separately. Integrating RF with ROC analysis for simultaneous feature selection and classification is another research interest of our future research effort.

6.5 Conclusion

In this chapter, we first introduced some new texture measures according to the Haralick's 3D model, and then took the well-known principal component analysis or KL transform, as an example, to explore an adaptive idea to address the spatial variation of polyp volume orientation in the patient space, and further integrated a mathematical derivative operation, as an example, for texture amplification to address the compromise of texture loss due to noise smoothing in many state-of-the-art CT image reconstruction algorithms. While the adaptive ideas and the tools (of mathematical derivatives, KL transform and two innovative classification methods) used to realize the ideas for enhancing textures and stabilizing spatial variations are simple, their impacts to the clinical task of differentiating hyperplastic from adenomatous polyps are significant as evidenced by the above reported experiments, which rendered a gain in AUC value (i) from 0.7723 to 0.8087 by addition of 16 new texture measures; (ii) from 0.7553 to 0.8087 by the variation stabilization operation; (iii) from 0.7288 to 0.8087 by the texture amplification operation; (iv) from 0.8016 to 0.8087 by LIRF classifier. The differentiation capability of AUC = 0.8087 indicates quantitatively the feasibility of advancing CTC toward personal healthcare for preventing colorectal cancer. The ideas can be applied to other applications, such as differentiation of lung nodule malignancy.

Selected Publications

- **Hu Y**, Liang Z, Song B, Han H, Pickhardt P, Zhu W, Zhang H Barish M and Lascarides C (2016). "Texture feature extraction and analysis for polyp differentiation via computed tomography colonography." IEEE Transactions on Medical Imaging, DOI: 10.1109/TMI.2016.2518958, in press.

- **Hu Y**, Han H, Pickhardt P, Zhu W, and Liang Z (2015). " New Texture Features for Improved Differentiation of Hyperplastic Polyps from Adenomas via Computed Tomography Colonoscopy." IEEE Nuclear Science Symposium and Medical Imaging Conference Record.

- Zhang H, Han H, Liang Z, **Hu Y**, Liu Y, Moore W, Ma J, and Lu H (2015). "Extracting information from previous full-dose CT scan for knowledge-based Bayesian reconstruction of current low-dose CT images." IEEE Transactions on Medical Imaging, DOI: 10.1109/TMI.2015.2498148, in press.

- Ma M, Li L, Han H, **Hu Y**, Gu D and Liang Z (2015), "Adaptive kernel based multiple kernel learning for computer-aided polyp detection in CT colonography," Geometry, Imaging and Computing, vol. 2, no. 1, pp. 23-45.

- **Hu Y**, Song B, Zhu W, and Liang Z (2015). "An integrated classifier for computer-aided diagnosis of colorectal polyps based on random forest and location index strategies" Proc. SPIE Medical Imaging, in press. (oral)

- **Hu Y**, Song B, Pickhardt P, and Liang Z (2014). "Distance weighted 'inside disc' classifier for computer-aided diagnosis of colonic polyps" Proc. SPIE Medical Imaging, vol.9414. (oral)

Bibliography

American College of Radiology (ACR) (2005), “ACR practice guideline for the performance of CTC in adults”, *ACR Practical Guideline*, **29**: 295-298

American Cancer Society (ACS) (2014), “Cancer facts & figures 2014”, *ACS*, Atlanta, 2014.

S. Arlot and A. Celisse (2010). "A Survey of Cross-validation Procedures for Model Selection." *Statistics Surveys*, **4**: 40-79.

K. Bennett and E. Bredensteiner (2000), “Duality and geometry in svm classifiers”, *Proceedings of the 17th International Conference on Machine Learning*, 57-64.

C. Bishop (2006), “Pattern Recognition and Machine Learning”, *New York: Springer*

B. Boser, I. Guyon and V. Vapnik (1992), “A training algorithm for optimal margin classifiers”, *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*, **5**: 144–152

L. Breiman, J. Friedman, R. Olshen and C. Stone (1984). “Classification and regression trees”, *Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software*

L. Breiman (1996), "Bagging Predictors", Machine Learning, **24(**2): 123-40.

G. Castellano, L. Bonilha, L. Li, and F. Cendes (2004), “Texture analysis of medical images”, *Clinical Radiology*, **59**(12): 1061-1069.

C. Cortes and V. Vapnik (1995), “Support vector networks”, *Machine Learning*, **20**:273–297

T. Cover and P. Hart (1967), “Nearest neighbor pattern classification”, *IEEE transactions on Information Theory*, **13.**

C. Do, C. Bertrand, J. Palasse, et al. (2012), “A new biomarker that predicts colonic neoplasia outcome in patients with hyperplastic colonic polyps,” *Cancer Prevention Research*, **5**: 675-684.

K. Engel, M. Hadwiger, J. Kniss, *et al.* (2006), *Real-Time Volume Graphics*, A K Peters, Ltd, Wellesley, MA.

T. Fawcett (2006), “An introduction to ROC analysis*”, Pattern Recognition Letters*, **27**: 861-874.

J. Friedman (1991), “Multivariate adaptive regression splines (with discussion)”, *Annals of Statistics*, **19**: 1–141.

C. Gini (1912), “Variabilita e mutabilita”, *Studi Economico-Giuridici Fac. Giuris- prudenza Univ.* Cagliari, A. III, parte II.

I. Goodfellow, Y. Bengio, and A. Courville (2015), “Deep Learning”.

R. Haralick, K. Shanmugam, and I. Dinstein (1973), “Textural features for image classification”, *IEEE Transactions on Systems Man and Cybernetics*, **3**(6): 610-621.

Y. Hu, B. Song, P. Pickhardt and Z. Liang (2014). "Distance weighted 'inside disc' classifier for computer-aided diagnosis of colonicpolyps" *Proc. SPIE Medical Imaging*, vol.9414.

Y. Hu, B. Song, M. Ma and Z. Liang (2014). "A mixture classifier for computer-aided diagnosis of polyp malignancy for CT colonography." *IEEE Nuclear Science Symposiumand Medical Imaging Conference Record.*

Y. Hu, B. Song, W. Zhu and Z. Liang (2015). " An integrated classifier for computer-aided diagnosis of colorectal polypsbasedon random forest and location index strategies" *Proc. SPIE Medical Imaging*.

Y. Hu, H. Han, P. Pickhardt, W. Zhu and Z. Liang (2015). " New Texture Features for Improved Differentiation of Hyperplastic Polyps from Adenomas via Computed Tomography Colonoscopy." *IEEE Nuclear Science Symposium and Medical Imaging Conference Record.*

C. Johnson, M. Chen, A. Toledano, *et al*. (2008), “Accuracy of CTC for detection of large adenomas and cancers”, *New England Journal of Medicine*, **359**(12): 1207-1217.

G. Kass (1980), “An exploratory technique for investigating large quantities of categorical data”, *Applied Statistics*, **29**(2):119-127.

H. KUHN and A. TUCKER (1951), "Nonlinear Programming", *Proceedings of the Second Berkeley Symposium on Mathematical Sta- tistics and Probability*, 481-492

Z. Liang and R. Richards (2010), “Virtual colonoscopy v.s. optical colonoscopy”, *Expert Opinion on Medical Diagnostics Journal*, **4**(2): 149-158 (http://informahealthcare.com/toc/edg/4/2).

A. Liaw and M. Wiener (2002), “Classification and regression by randomForest”, *R News*, **2**: 18–22.

D. Lieberman, J. Holub, G. Eisen, *et al*. (2005), "Prevalence of polyps greater than 9 mm in a consortium of diverse clinical practice settings in the United States", *Clinical Gastroenterology and Hepatology*, **3**(8): 798-805.

Y. Liu (2014), "Image reconstruction theory and implementation for low-dose CT", Ph.D Dissertation, Stony Brook University, NY, USA.

M. Ma, B. Song, Y. Hu, X. Gu, and Z. Liang (2014), "Random forest based computer-aided detection of polyps in CTC", *Conference Record of IEEE NSS-MIC*, in CD-ROM.

J. Mercer (1909), "Functions of positive and negative type and their connection with the theory of integral equations", *Philosophical Transactions of the Royal Society*, London A **209**: 415-446.

T. Mitchell (1997), "Machine Learning", *New York: McGraw-Hill.*

O. Monga and S. Benayoun (1991). "Using partial derivatives of 3D images to extract typical surface features", *Computer vision and image understanding*, pp. 171-189.

F. Ng, B. Ganeshan, R. Kozarski, *et al*. (2013), "Assessment of primary colorectal cancer heterogeneity by using whole-tumor texture analysis: contrast-enhanced CT texture as a biomarker of 5-year survival", *Radiology*, **266**(1): 177-184.

C. Philips, D. Li, D. Raicu, J. Furst (2008), "Directional invariance of co-occurrence matrices within the liver", *Intl Conf on Biocomputation, Bioinformatics, and Biomedical Technologies*, in CD-ROM.

P. Pickhardt, B. Levin, and J. Bond (2008), "Screening for nonpolypoid colorectal neoplasma", *Journal of the American Medical Association*, **299**(23): 2743-2744.

P. Pickhardt and D. Kim (2009), "CRC screening with CTC: key concepts regarding polyp prevalence, size, histology, morphology, and natural history", *American Journal of Roentgenology*, **193**(1): 40-46. doi: 10.2214/AJR.08.1709.

P. Pickhardt, D. Kim, B. Pooler, *et al*. (2013), "Assessment of volumetric growth rates of small colorectal polyps with CTC: a longitudinal study of natural history", *Lancet Oncology*, **14**(8): 711-720.

R. Quinlan (1993), "C4.5: Programs for Machine Learning", *Morgan Kaufmann*

R. Quinlan (1986), "Induction of Decision Trees", *Machine Learning*, **1**(1): 81-106.

B. Schaeffer, T. Johnson, T. Mang, et al. (2014), "Dual-energy CTC for preoperative 'one-stop' staging in patients with colonic neoplasia", Academic Radiology, **21**(12): 1567-1572.

C. Showalter, B. Clymer, B. Richmond, and K. Powell (2006), “Three-dimensional texture analysis of cancellous bone cores evaluated at clinical CT resolutions”, *Osteoporos Int*., **17**: 259-266.

B. Song, G. Zhang, H. Lu, *et al*. (2014), “Volumetric texture features from higher-order images for diagnosis of colon lesions via CTC”, *International Journal of Computer Assisted Radiology and Surgery*, **9**: 1021-1032.

B. Song, G. Zhang, W. Zhu, and Z. Liang (2012), “A study on random forests for computer-aided detection in CTC”, *The 26th Intl Congress and Exhibition on Computer Assisted Radiology and Surgery (CARS), Intl. J. CARS*, vol. 7, (Suppl): pp. S273.

B. Song, G. Zhang, W. Zhu, Z. Liang (2014), “ROC operating point selection for classification of imbalanced data with application to computer-aided polyp detection in CTC”, *International Journal of Computer Assisted Radiology and Surgery*, **9**: 79-89.

S. Stehman (1997). "Selecting and interpreting measures of thematic classification accuracy", *Remote Sensing of Environment*, **62** (1): 77–89.

SVM lecture notes: http://cs229.stanford.edu/notes/cs229-notes3.pdf

V. VAPNIK and A. CHERVONENKIS (1964), “A note on one class of perceptrons”, *Automation and Remote Control*, **25**.

S. Vijayakumar (2007). "The Bias–Variance Tradeoff", *University Edinburgh Lecture notes*.

D. Wolpert and W. Macready (1997). "No Free Lunch Theorems for Optimization", *IEEE Transactions on Evolutionary Computation*, **1**(1): 67-82.

J. Yao, A. Dwyer, and D. Mollura (2011), “Computer-aided diagnosis of pulmonary infections using texture analysis and support vector machine classification”, *Academic Radiology*, **18**(3): 306-314.

G. Zhang, B. Song, H. Zhu, Z. Liang (2012), “Computer-aided diagnosis in CTC based on bi-labeled classifier”, *The 26th Intl Congress and Exhibition on Computer Assisted Radiology and Surgery, Intl. J. CARS*, vol. 7 (Suppl): pp. S274.