Machine Learning Techniques in Nuclear Material Detection, Drug Ranking and Video Tracking

by

Yan Yang

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Industrial Engineering and Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Fall 2013

Machine Learning Techniques in Nuclear Material Detection, Drug Ranking and Video Tracking

Abstract

Machine Learning Techniques in Nuclear Material Detection, Drug Ranking and Video Tracking

by

Yan Yang

Doctor of Philosophy in Engineering – Industrial Engineering and Operations Research

University of California, Berkeley

Professor Dorit S. Hochbaum, Chair

The main focus of this thesis is using machine learning and data mining techniques to solve challenging problems. Three problems from different subject areas are discussed: nuclear material detection, drug ranking and target tracking in video sequences. The techniques of the three problems described are all based on an efficiently solvable variant of normalized cut, Normalized Cut Prime (or NC$'$).

The first problem concerns detecting concealed illicit nuclear material, an important part of strategies preventing and deterring nuclear terrorism. What makes this an extremely difficult task are physical limitations of nuclear radiation detectors (arising from energy resolutions and efficiency) and shielding materials terrorists would presumably use to surround the radioactive nuclear material and absorb some of the radiation, thereby reducing the strength of the detected signal. This means the central data analysis problem is identifying a potentially very weak signal, and distinguishing it from both background noise arising from the detector characteristics and naturally occurring environmental radiation. We aim at enhancing the capabilities of detection with algorithmic methods specifically tailored to nuclear data. A novel graph-theory-based methodology based on NC$'$ is used, called *Supervised Normalized Cut* (SNC). This data mining method classifies measurements obtained from very low resolution plastic scintillation detectors. The accompanying computational study, comparing SNC method with several alternative classification methods shows that in terms of accuracy, the SNC method is on par with alternative approaches, yet SNC is computationally more efficient.

The second subject area is in the field of drug ranking. This problem refers to placing in rank order, according to their effectiveness, several drugs treating the same disease, using data derived from cell images. Current technologies use the recently developed high-throughput drug profiling (high content screening or HCS). Despite the potential of HCS for accurate descriptions of drug profiles, it produces a deluge of data of quantitative and multidimen-

sional nature, posing analytical challenges in the data mining process. Our new framework is designed to alleviate these difficulties, in the way of producing graph theoretic descriptors and automatically ordering the performance of drugs, called fractional adjusted bi-partitional score (FABS), a way of converting classification to scores. We experimented with the FABS framework by implementing different algorithms and assessing the accuracy of results by a comparative study, which includes other four baseline methods. The conclusion is encouraging: FABS implemented with NC$'$ consistently outperforms other implementations of FABS and alternative methods currently used for ranking that are unrelated to FABS.

The third problem is target tracking in video sequences – it can be framed as an unsupervised learning problem: the goal is to delineate a target of interest in a video from background. The tracking task is cast as a graph-cut, incorporating intensity and motion data into the formulation. Tests on real-life benchmark videos show that the developed technique, NC-track, based on NC$'$, is more efficient than many existing techniques, and that it delivers good quality results.

To my parents and my advisor

For their moral and financial support during the time I need the most and for teaching me that the most difficult task can be accomplished if it is done one step at a time.

Contents

List of Figures

List of Tables

Acknowledgments

I want to acknowledge my advisor Dorit S. Hochbaum and colleagues, Barak Fishbain, Chun-Nan Hsu, Eric Norman and Erik Swanberg for their support and collaboration. I also want to acknowledge CBET-0736232 and the US Department of Homeland Security for their support.

Chapter 1

Introduction

1.1. Taxonomy of Machine Learning Problems

Machine learning techniques can be categorized either as supervised (inductive) learning, i.e. inferring a function from supervised training data [17], or as *un*supervised learning, thus the learner is given only unlabeled examples [31].

In supervised learning, one tries to infer a functional relation $y = f(x)$ called classifier from a training set $\mathcal{T} = \{(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)\}$, where $\vec{x}$'s are feature-vectors of data points, and the y's are the known labels assigned to the vectors of the training set and $y_i \in C$ where C is the set of all possible labels in the problem. We seek a classifier f that can accurately map the labels $y_{m+1}, y_{m+2}, ...$ for future input vectors $\vec{x}_{m+1}, \vec{x}_{m+2}, ...$ This paradigm is common to many other problems that appear in different areas, such as computer vision [15], geostatistics [28], credit scoring [106] and biometric identification [112].

Unsupervised learning groups data points into different clusters, based on some measure of sameness, *without using prior training data*. The general procedure is to map all the data points into different clusters such that some criteria are optimized.

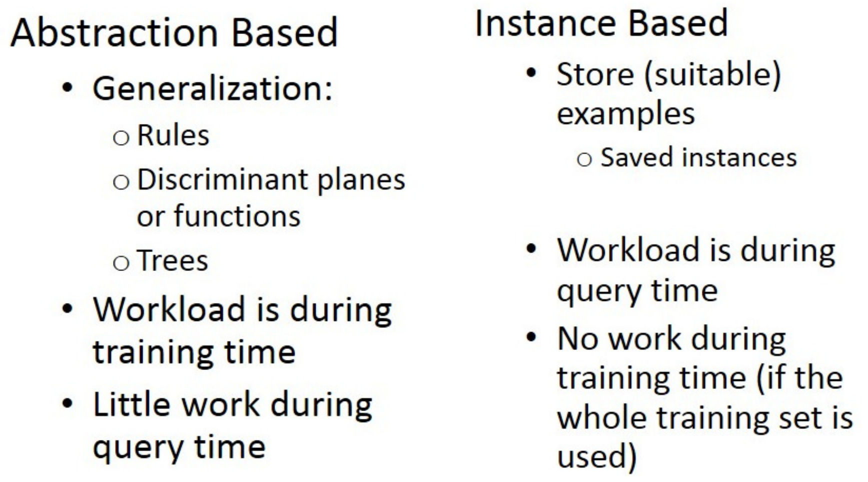
Examples of criteria for binary clustering, also referred to as bi-partitioning, are: (i) minimum-cut [37] (ii) ratio regions, (presented by Cox [23]), (iii) the normalized cut (suggested by Shi and Malik [95]) and (iv) a variant of normalized cut, NC$'$ (studied by Hochbaum [49]).

1.2. Paradigms of Solving Supervised Learning Problems

In the realm of supervised learning problems, among many kinds of solution methods, two diametrically opposite approaches emerge: 1) abstraction-based algorithms and 2) instance-based algorithms [1]. In an abstraction-based algorithm, the training set $\mathcal{T}$ is used implicitly by first generalizing it to a classifier. The classifier is then used to classify new objects without directly relating them to $\mathcal{T}$. In decision trees, for example, the classifier is the tree itself. In support vector machine (SVM), it is the discriminate plane. The key is that $\mathcal{T}$ is discarded after such abstraction is constructed. The goal of the training step is to extract the abstraction.

On the other hand, instance-based algorithms do not perform such an extraction task. Their underlying mechanism is to use a subset $\mathcal{T}^* \subseteq \mathcal{T}$ directly to classify the new objects. To this end, two additional functions are involved: $sim : (\vec{x}_1, \vec{x}_2) \rightarrow R$ is the similarity function - it takes two feature vectors and convert them into a real number proportional to how similar they are. One example of the similarity function is the reciprocal of Euclidean distance between the two feature vectors. Another function is the classification function $class : \mathcal{T}^*, \vec{x}_i, sim \rightarrow y_i \in C$, giving a label to any feature vector $\vec{x}_i$. The classification of $\vec{x}_i$ is done by applying the classification function $class$ to $\vec{x}_i$. One commonly used classification function is to compare similarities between $\vec{x}_i$ to all points in $\mathcal{T}^*$ and give $\vec{x}_i$ the label of the most similar point in $\mathcal{T}^*$ to $\vec{x}_i$ (the nearest neighbor search) [1].

A comparison between the two solution paradigms can be found in Figure 1.1. This thesis is focused on abstraction-based algorithms.


Abstraction Based

- Generalization:
  - Rules
  - Discriminant planes or functions
  - Trees
- Workload is during training time
- Little work during query time

Instance Based

- Store (suitable) examples
  - Saved instances
- Workload is during query time
- No work during training time (if the whole training set is used)

Figure 1.1: Comparison between abstract-based learning and instance-based learning.

1.3. Notations

In this section, we introduce some notations that will facilitate the discussion in this thesis. As mentioned in Section 1.1, $\vec{x}$'s denote feature-vectors of data points. Each feature-vector consists of attribute values associated with the corresponding data point. The dimension of the data points is the length of the feature-vector. In general, vectors are denoted as $\vec{v}$ and a matrix is denoted in bold capital letters such as $\mathbf{M}$.

For drug ranking applications, the data points, i.e. feature-vectors are grouped into k population sets, $\{P_1, ..., P_k\}$. Each population set represents a set of feature vectors corresponding to the same drug. Each feature vector x_i belongs to one of the population sets and it is labeled with l_{x_i}, which is the label of the population set it belongs to. Sometimes, we are also given at least two population sets $\{R_1, R_2\}$, not belonging to $\{P_1, ..., P_k\}$ that can be used as training.

For video tracking applications, each feature-vector is decomposed into two *vectors*: $\vec{I}_{klt}$ and $\vec{m}_{klt}$. $\vec{I}_{klt}$ is the color representation vector of the pixel coordinates (k, l) of frame t. The color representation can be in any form (e.g, R-G-B, Y-Cb-Cr, H-S-V or L-a-b). The vector $\vec{m}_{klt}$ is the motion component which typically contains two components: the horizontal and vertical motions. For the subsequent processing stages, the horizontal and vertical motions are presented instead in polar coordinates, hence magnitude, A_{klt}, and angle, φ_{klt}, of the motion vector.

Each feature-vector is represented as a node in the attribute space. An undirected graph $G = (V, E)$ is constructed, where each node $v \in V$ corresponds to a data point. There is an edge in the graph for each pair of nodes i and j associated with a weight, w_{ij} that corresponds to the similarity between the feature-vectors associated with nodes i and j. Higher similarity is associated with higher weights.

The weight function $w : V \times V \rightarrow \Re^+$ associates with each pair of nodes $\{i, j\}$ (an edge) its similarity strength between the two nodes. For each edge $[i, j]$, the weight w_{ij} and the distance between the two points v_i and v_j have the relationship: one goes up as the other goes down (or vice versa) - this also means that w_{ij} and the similarity between v_i and v_j both go up or down together. Several distance measures can be used for this purpose, among them, Euclidean, city block, and Minkowski distances.

Given a graph $G = (V, E)$, a bi-partition of the graph is called a *cut*, is defined as $(S, \bar{S}) = \{[i, j] | i \in S, j \in \bar{S}\}$, where $\bar{S} = V \setminus S$. The *capacity of a cut* $(S, \bar{S})$, is defined as:

$$C(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}, [i,j] \in E} w_{ij}. \tag{1.1a}$$

The *capacity of a set* $S \subset V$ is denoted by:

$$C(S, S) = \sum_{i,j \in S, [i,j] \in E} w_{ij}. \tag{1.1b}$$

More generally, for any pair of sets $A, B \subseteq V$ the capacity of the cut is denoted by $C(A, B) = \sum_{i \in A, j \in B} w_{ij}$. Similarly, the *capacity of a set*, $D \subset V$, is denoted by $C(D) = C(D, D) = \sum_{i,j \in D, [i,j] \in E} w_{ij}$.

We denote the sum of all weights of edges connected to a node i, by:

$$d_i = \sum_{j \in V, [i,j] \in E} w_{ij} \tag{1.1c}$$

1.4. Thesis Overview

The organization of the thesis is the following:

1 Related Works of Machine Learning Techniques: in Chapter 2, related works are discussed. In particular, a variant of normalized cut - the Normalized Cut Prime of Hochbaum [49] is described. The techniques developed in nuclear material identification, drug ranking and target tracking are all based on Normalized Cut Prime (NC$'$). Other leading machine learning techniques are described as well;

2 Nuclear Material Identification with Supervised Normalized Cut (SNC): Chapter 3 describes the problem of detecting illicit nuclear materials. This chapter presents a graph-theory-based methodology, called *Supervised Normalized Cut* (SNC), based on NC$'$, for data mining and classification of measurements obtained from plastic scintillation detectors which are of particularly low resolution. We also include here a computational study, comparing the supervised normalized cut method with alternative current classification methods. It shows that SNC is superior to alternative methods at classifying nuclear data;

3 Drug Ranking with Fractional Adjusted Bi-partitional Score (FABS): in Chapter 4, we discuss a framework for ranking the effectiveness of several drugs using multidimensional data obtained through high-throughput drug profiling (high content screening or HCS). This framework produces graph theoretic descriptors, automatically ordering the performance of drugs, called fractional adjusted bi-partitional score (FABS). Computational experiments show that FABS framework implemented with NC$'$ (FABS-NC$'$) outperforms other implementations of FABS and alternative methods currently used for ranking that are unrelated to FABS.

4 Target Tracking in Video Sequences: in Chapter 5, we explore the problem of tracking a target in a sequence of videos – an important application in surveillance and security. This problem is formulated as a clustering problem on a graph. NC$'$, incorporating intensity and motion data is presented here. Tests on real-life benchmark videos show that the presented technique is more efficient than many existing techniques, and that it delivers good quality results;

5 Conclusion: we conclude this thesis in Chapter 6 by summarizing the findings in our studies. In addition, future research for possible expansions of SNC and FABS is proposed.

Chapter 2

Related Works

This chapter provides the necessary background on several machine learning methods and that will make this thesis relatively self-contained. The major part of the chapter is the introduction of a variant of normalized cut - Normalized Cut Prime (NC$'$) by [49]. It is a type of machine learning formulation designed to solve clustering problems as discussed in Chapter 1. We will describe its formulation and solution methods. Moreover, the relationship between supervised normalized cut (SNC) and NC$'$ is described. In addition, principal component analysis (PCA), will be discussed. The rest of the chapter is devoted to several supervised learning techniques: linear discriminant analysis (LDA), support vector machine (SVM), SVM-based algorithms, artificial neural networks (ANN), regression methods (LIN REG and LOG REG) and ensemble methods (ENSEM).

2.1. The Variant of Normalized Cut

We use notations introduced in Chapter 1 to discuss a variant of normalized cut - Normalized Cut Prime (NC$'$). Given an undirected graph $G = (V, E)$, the goal of NC$'$ is to find a partition to two disjoint sets minimizing the ratio of two criteria. One is to maximize the similarity of the feature-vectors within each group. The second criterion is to minimize the similarity between S^* and its complement. The solution set S^* is called a *source* set, and its complement is called a *sink* set. The mathematical objective of the above goal can be written as [49, 94]:

$$NC'(S^*) = \min_{S \subset V} \frac{C(S, \bar{S})}{C(S, S)}. \tag{2.1}$$

$C(S, \bar{S})$ represents the similarity between the two parts of the bipartition, while the similarity between the elements of S can be written as $C(S, S)$.

Hochbaum showed in [49] that (2.1) is equivalent to minimizing one term in Shi's and Malik's *Normalized Cut* (NC) optimization criterion, which was defined in [95]. Hochbaum [49] used this name and notation. Hochbaum has also showed, in [49], that (2.1) is solvable in polynomial time and that this optimization criterion is efficient and extremely robust for image segmentation.

Since NC$'$ does not require any class information on the data, it is an unsupervised problem. It provides better quality solutions for image clustering and pattern recognition applications than other, commonly used, techniques [49, 94]. In addition, as mentioned below, NC$'$ was shown to be efficiently solvable [49] and as such it is a good candidate for solving clustering problems in short running times.

The integer programming formulation of (2.1), is provided in [49] and in [51], where the NC$'$ problem is also shown to be equivalent to the "Weighted Ratio Region" problem, or WRR. Thus, any optimal solution to (2.1) is an optimal solution to respective WRR,

$$(WRR)\ \min\ \frac{C(S,\bar{S})}{\sum_{i\in S} d_i} \tag{2.2}$$

for d_i the sum of all weights of edges connected to i.

The problem formulation uses x_i and y_{ij} binary variables:

$$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \in \bar{S}, \end{cases} \tag{2.3a}$$

$$y_{ij} = \begin{cases} 1 & \text{if } i \in S, j \in \bar{S}, \text{ or } i \in \bar{S}, j \in S \\ 0 & \text{if } i,j \in S \text{ or } i,j \in \bar{S}. \end{cases} \tag{2.3b}$$

$$\begin{array}{lll} (WRR)\ \min & \frac{\sum_{[i,j]\in E} w_{ij} y_{ij}}{\sum_{j\in V} d_j x_j} & \\ \text{subject to} & x_i - x_j \le y_{ij} & \text{for all } [i,j]\in E \\ & x_j - x_i \le y_{ij} & \text{for all } [i,j]\in E \\ & x_j \ \texttt{binary}\ j\in V & \\ & y_{ij}\ \texttt{binary}\ [i,j]\in E. & \end{array} \tag{2.3c}$$

where, as noted above $d_j = \sum_{k:[j,k]\in E} w_{jk}$.

This formulation of the problem was shown to be an integer program in monotone inequalities [49]. It implies that the linearized version of the above is solvable as a minimum s,t-cut problem on an associated graph [50].

To elaborate, Problem (2.3) is a ratio problem which is "linearized". The linearization introduces a parameter λ that affects the capacities in the associated graph. The linearized problem is solved as a *parametric* s,t-cut problem in the same complexity (and roughly the same run-time) as a single application of s,t-cut [49]. From this parametric solution one can extract the optimal solution to the ratio problem - Equation (2.1), as well as all possible weighted combinations of the objectives.

The associated graph G'_{st} constructed for this problem is of the same size as the original graph G. Each node representing a variable x_j has an arc going to a sink node with capacity λd_j. The arc from x_i to x_j has capacity w_{ij} and so does the arc from x_j to x_i as in our problem $w_{ij} = w_{ji}$. Arcs of infinite capacity are drawn from the source node, s, to all *seed* *nodes* that a priori have been tagged as being in S and from all *seed nodes* that are tagged as being in $\bar{S}$ to t. The algorithm solving the problem is then a simple parametric cut algorithm in the graph G'_{st}. The graph is presented in Figure 2.1. For more details on the graph construction and its validity see [49].

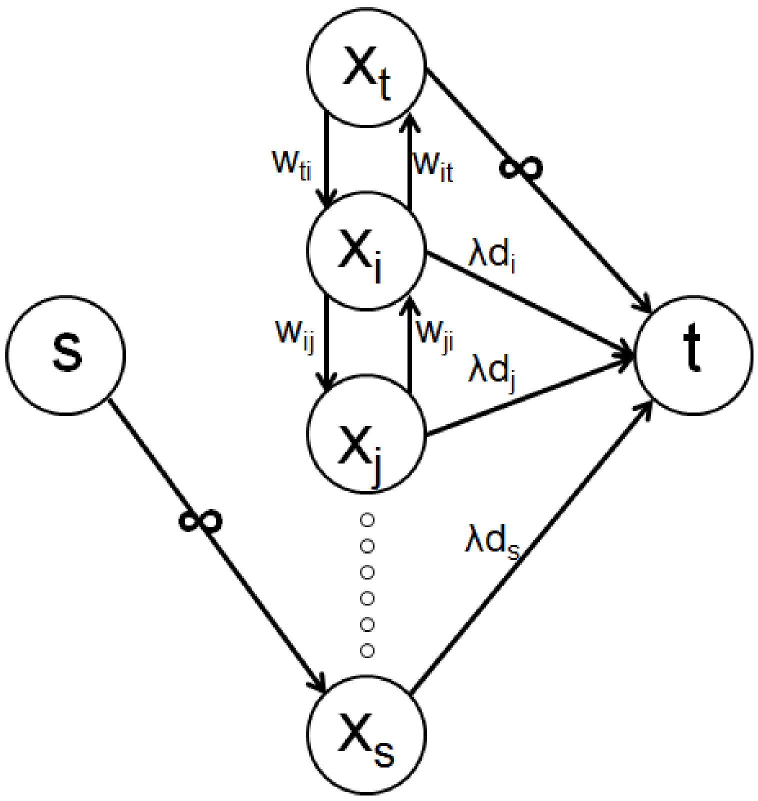


Figure 2.1: The graph G'_{st} for the normalized-cut' problem with node x_s serving as source seed.

NC$'$ is the basis of the algorithms described in this thesis. One example is Supervised Normalized Cut (SNC) in Chapter 3. As noted before, the NC$'$ solution procedure requires to assign, in advance, a single node which will be included in the source S (or sink $\bar{S}$) set. This node is referred to as a *seed node*. SNC exploits the seed node mechanism in order to force a-priori the training data to be in either in the source S or in the sink $\bar{S}$, based on the class label of the training data. Specifically, the input consists of three sets: two sets of nodes, A and B, which are associated with feature-vectors acquired from two different class labels, M^1 and M^2, and a third set, I, corresponding to feature-vectors of an unknown data points or point. The goal of SNC is to associate each feature-vector in I with either M^1 or M^2.

2.2. Dimension Reduction

Dimension Reduction approaches try to find a subset of the original variables (also called features or attributes) that can explain the data best. The underlying idea is that other variables do not contribute to the understanding of the data and introduce noise. Therefore classification done in the reduced space may be more accurate than in the original space.

PCA is a routinely used tool for reducing the data space's dimension. The underlying paradigm of PCA is that an orthogonal linear transformation is performed on the data to a new coordinate system with the properties that the first coordinate, the so-called first principal component, contains the greatest variance by any projection of the data; the second coordinate contains the second greatest variance and so on. PCA essentially rotates the data points around their mean and moves variance into the first few dimensions as much as possible. The remaining dimensions contain negligible amounts of variance and are omitted, with a relatively small loss of information. Thus PCA is often used as dimensionality reduction.

The principal components can be calculated as the eigenvectors of the covariance matrix. Consider a data matrix: Each of the n rows represents a different data point, and each of the p columns gives an attribute of the feature space. The data matrix can be transformed to a *standardized data matrix*, $\mathbf{X}$. The standardization is done by subtracting the average of each attribute column from the data matrix. We compute the matrix $\mathbf{V}$ of eigenvectors which diagonalizes the covariance matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$:

$$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$$

where, $\mathbf{D}$ is the diagonal matrix consisting of the set of all eigenvalues of $\mathbf{C}$ along its principal diagonal. If we rearrange the eigenvalues in $\mathbf{C}$ along diagonal in decreasing order, then the eigenvector corresponding to the largest eigenvalue is the first principal component. The second principal component corresponds to the second largest, and so on.

The rotation of the original data into the new principal space can be performed by

$$\mathbf{X}\mathbf{V}.$$

PCA constructs the spanning space (i.e., the principal components) so the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components.

To reduce the dimension, we can delete columns in $\mathbf{V}$ corresponding to small eigenvalues to obtain $\mathbf{V}_d$, then transform the original data into this smaller space by

$$\mathbf{X}\mathbf{V}_d.$$

2.3. Linear discriminant analysis

In terms of supervised learning paradigm discussed earlier, we consider a training set

$$\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\, y_i \in \{0,1\}, i = 1...n\}\,.$$

In the training set, the two classes of training data points have means $\vec{\mu}_{y=0}$ and $\vec{\mu}_{y=1}$ and covariances $\Sigma_{y=0}$ and $\Sigma_{y=1}$.

Linear Discriminant Analysis (LDA) [70, 72] assigns the class label of an arbitrary feature vector $\vec{x}$ by using a *discriminant hyperplane* in the form of

$$f(\vec{x}) = \vec{m} \cdot \vec{x} - c \tag{2.4}$$

where $\vec{m}$ is obtained by maximizing the ratio of the variance between the classes, $\sigma^2_{between} = (\vec{m} \cdot \vec{\mu}_{y=1} - \vec{m} \cdot \vec{\mu}_{y=0})^2$ to the variance within the classes, $\sigma^2_{within} = \vec{m}^T \Sigma_{y=1} \vec{m} + \vec{m}^T \Sigma_{y=0} \vec{m}$ in the training set $\mathcal{T}$,

$$\begin{aligned}
\vec{m}(\mathcal{T}) &= \underset{\vec{m}'}{\operatorname{argmax}} \frac{\sigma^2_{between}}{\sigma^2_{within}} \\
&= \underset{\vec{m}'}{\operatorname{argmax}} \frac{(\vec{m}' \cdot \vec{\mu}_{y=1} - \vec{m}' \cdot \vec{\mu}_{y=0})^2}{\vec{m}'^T \Sigma_{y=1} \vec{m}' + \vec{m}'^T \Sigma_{y=0} \vec{m}'} \\
&= \underset{\vec{m}'}{\operatorname{argmax}} \frac{(\vec{m}' \cdot (\vec{\mu}_{y=1} - \vec{\mu}_{y=0}))^2}{\vec{m}'^T (\Sigma_{y=0} + \Sigma_{y=1}) \vec{m}'}.
\end{aligned}$$

The maximum separation occurs when $\vec{m} = (\Sigma_{y=0} + \Sigma_{y=1})^{-1}(\vec{\mu}_{y=1} - \vec{\mu}_{y=0})$ and the corresponding c can be calculated as [70]

$$\begin{aligned}
c &= \vec{m} \cdot (\vec{\mu}_{y=0} + \vec{\mu}_{y=1})/2 \\
&= \frac{1}{2}\vec{\mu}^t_{y=1} \Sigma^{-1} \vec{\mu}_{y=1} - \frac{1}{2}\vec{\mu}^t_{y=0} \Sigma^{-1} \vec{\mu}_{y=0}.
\end{aligned}$$

To classify the arbitrary feature vector $\vec{x}$, LDA uses the discriminant plane in Equation (2.4). If the output of function f is greater than 0, then $\vec{x}$ is classified in class $y = 1$, otherwise $\vec{x}$ is classified in class $y = 0$.

One note is that when the dimensionality of the data is far larger than the number of data points, [43] suggests, in cases like this, to perform PCA first before using LDA for classification.

2.4. Support Vector Machine

Support Vector Machine (SVM) is a widely accepted tool for classification in many fields, including machine learning [42], communication and mobile computing [104], biology [27], economics [53], nuclear applications [16] and x-ray spectrometry [66].

Given training data $\mathcal{T}$

$$\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\ y_i \in \{-1, 1\}, i = 1...n\},$$

SVM finds the maximum-margin hyperplane that divides the points with $y_i = 1$ from those with $y_i = -1$. Like in the linear discriminant analysis, a hyperplane can be written as

$$\vec{m} \cdot \vec{x} - b = 0,$$

where $\vec{m}$ is the normal vector to the hyperplane and $\frac{b}{\|\vec{m}\|}$ is the translation of the hyperplane along the normal vector $\vec{m}$. SVM tries to find $\vec{m}$ by solving the following (primal) optimization problem:

$$\begin{aligned} &\underset{\vec{m}', b}{\text{minimize}} && \frac{1}{2}||\vec{m}'||^2 \\ &\text{subject to} && y_i(\vec{m}' \cdot \vec{x}_i - b) \geq 1 \text{ for } i = 1, ..., n. \end{aligned} \tag{2.5}$$

It can be shown that the $\vec{m}$ found by the above optimization constructs a discriminant plane furthest from both data points with $y_i = 1$ and data points with $y_i = -1$ [25].

The dual of the optimization problem (2.5) can be formulated first by adding Lagrangian multipliers α_j, $j = 1...n$ corresponding to the constraints in Problem (2.5) and writing the unconstrained problem as

$$\underset{\vec{m}', b}{\text{minimize}}\ \underset{\vec{\alpha} \geq 0}{\text{maximize}} \{\frac{1}{2}||\vec{m}'||^2 - \sum_{i=1}^{n} \alpha_i [y_i(\vec{m}' \cdot \vec{x}_i - b) - 1]\}.$$

Differentiating the unconstrained problem with respect to $\vec{m}'$ and b separately and set the derivatives to zero, we obtain relationships $\vec{m}' = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$. Using $\vec{m}' = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$, we can derive the dual of (2.5) by writing the unconstrained problem in terms of $\vec{\alpha}$ only. After adding the constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$, the dual can be written as:

$$\begin{aligned} &\underset{\vec{\alpha}}{\text{maximize}} && \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j) \\ &\text{subject to} && \alpha_i \geq 0, \text{ for } i = 1, \ldots, n \\ & && \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{2.6}$$

Note that in the dual, we use the notation $k(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$, called *kernel*. There are also other kernels, besides $k(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$, that can be used in (2.6). It is shown in [25] that we can obtain non-linear *decision boundaries* that separate data points with $y_i = 1$ from data points with $y_i = -1$ by using other forms of kernels. Two of the commonly used kernels are polynomial and radial basis function (RBF) kernels. The polynomial kernel's parameter is the degree of the polynomial, d, and RBF uses a derivative parameter, $\sigma > 0$:

1. $k(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j)^d$ (polynomial kernel)
2. $k(\vec{x}_i, \vec{x}_j) = \exp(-\sigma ||x_i - x_j||^2)$ (RBF kernel).

$\vec{m}$ can be found by solving for $\vec{\alpha}$ in Problem (2.6) [25] then substituting the value of $\vec{\alpha}$ in

$$\vec{m} = \sum_i \alpha_i y_i \vec{x}_i.$$

Only a few α_i will be greater than zero [25]. Each α_i corresponds to a constraint in Problem (2.5). By Karush Kuhn Tucker (KKT) conditions, the constraint in primal Problem (2.5) corresponding to a non-zero α_i in dual Problem (2.6) satisfies $y_i(\vec{m} \cdot \vec{x}_i - b) = 1$ and the $\vec{x}_i$ in the constraints are called a *support vectors*. Support vectors (SV) can be used to calculate the variable b in the decision boundary

$$\vec{m} \cdot \vec{x} - b = 0$$

by [25]

$$b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (\vec{m} \cdot \vec{x}_i - y_i)$$

where N_{VC} is the number of support vectors.

Problem (2.6) can be solved by standard quadratic programming techniques and sequential minimal optimization (SMO) techniques.

Corinna Cortes and Vladimir N. Vapnik [22] suggested a modified formulation of SVM that controls the sensitivity of SVM to possible outliers in the training data: It introduces slack variables ξ's. Problem (2.5) becomes

$$\begin{aligned} \underset{\vec{m}', b}{\text{minimize}} \quad & \frac{1}{2}||\vec{m}'||^2 \\ \text{subject to} \quad & y_i(\vec{m}' \cdot \vec{x}_i - b) \geq 1 - \xi_i \text{ for } i = 1, ..., n \\ & \xi_i \geq 0 \text{ for } i = 1, ..., n. \end{aligned} \tag{2.7}$$

This formulation allows some amount of slackness in constraints in comparison to Problem (2.5). In formulation of (2.5), misclassified data are not allowed, while Problem (2.7) allows misclassified data if $\xi > 0$.

The dual of Problem (2.7) introduces a soft margin penalty parameter, C that is added to Problem (2.6) [14, 25]:

$$\begin{aligned} \underset{\vec{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \ \text{ for } i = 1, \dots, n \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{2.8}$$

The selection of the SVM's parameters follows an exhaustive search. The work of [46] provides some guidance on how to search for optimal parameters of SVM. It has been shown there that the optimal value of C for a SVM formulation is always in a bounded interval. One way to tune the parameters of SVM is by setting them in a grid, $\{2^{-7}, 2^{-6}, 2^{-5}, \cdots, 2^{6}, 2^{7}\}$, the searching range used in [68].

One should note that by convention, the SVM parameters tuning times are *not* included in the computation times that are used to compare the different methods. The SVM classification is performed for all possible parameters' combinations. The parameters' set that produces the best classification results is the one used for the evaluation. This tuning procedure gives the best possible accuracy of a particular SVM classifier among the tested values [14, 25]. One of the implementation packages used in computational experiments in this thesis is LIBSVM [20]. It is important to note, that while these tuning times are not included in our runtime comparison, the tuning process is time consuming.

There are also several SVM based methods, which incorporate the feature selection process as an inherent part of the formulation. These SVM procedures not only produce the classifier (as the regular SVM procedure does), but also the subset of features that are used for constructing this classifier. These include: (1) a feature-reducing linear kernel 1-norm SVM (SVM-1) [115]; (2) a recursive feature elimination SVM (SVM-RFE), where RFE stands for recursive feature elimination [45]; and (3) a feature-reducing newton method for LP SVM (SVM-NLP) [41]. These methods are shown to improve SVM's prediction power by removing features that are of the least relevance.

In this thesis, SVM-1 is implemented by using MATLAB function `svmtrain` with 1-norm option. SVM-RFE is obtained by modifying MATLAB code from [89] (the original code has extra functionalities). Software for SVM-NLP is downloaded from [40].

2.5. Artificial Neural Network

Artificial Neural Network (ANN) is one of the most used machine learning tools in recent years. It has been demonstrated to work well in several disciplines including material science [7], imaging processing [71], analog computation [96] and more.

In machine learning, ANN can be used both as supervised and unsupervised learning methods. Here we are interested in the supervised setting in pattern recognition (classification). In this context, feedforward network - also known as multilayer perceptron is the method of choice [107]. The basic model maps sets of input data onto a set of outputs. The model consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Figure 2.2 displays an example of a feedforward network. Note that there is only one layer between input layer and output layer which is called a hidden layer, but in general, there can be more than one hidden layer. Except for the input nodes, each node is called *neuron* and contains a nonlinear function called an activation function and each arc has a weight associated with it. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold such as zero, the neuron fires and takes the activated value of 1; otherwise it takes the deactivated value, typically -1. The typical nonlinear activation function is logistic function in the form of

$$\phi(x) = (1 + e^{-x})^{-1}$$

where x is the sum of the products of the weights and the values from the previous layer.

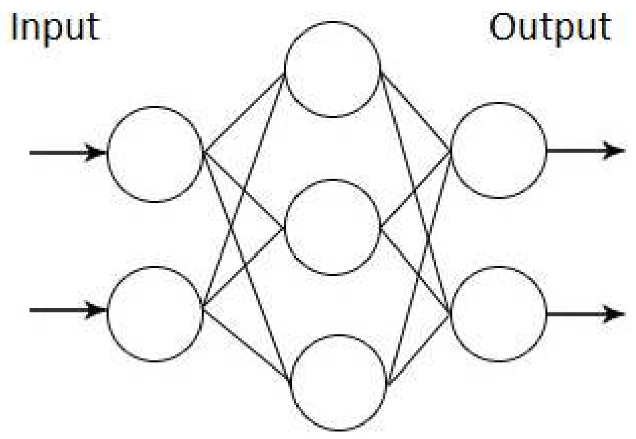


Figure 2.2: A graph representation of a feedforward network. There is only one hidden layer between the input and output nodes.

To construct a feedforward network, we need to determine the best values for parameters of interest of the network. Since the number of hidden layers can be more than one, the parameters of interest in this network are both the sizes of the hidden layers and the number of layers in between input and output layers. The optimal size of a hidden layer can be determined using grid search by specifying the search range for the size and utilizing the training to determine the best number. During the process of constructing the network, the technique for finding the weight on each arc is backproporgation algorithms (see [91] for details). One of the best backproporgation algorithms to find arc weights is scaled conjugate gradient method [76]. An implementation of this procedure can be found in MATLAB neural network toolbox by the function name `patternnet`.

2.6. Linear Regression and Logistic Regression

Given training data $\mathcal{T}$

$$\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\, y_i \in [-1, 1], i = 1...n\}\,.$$

A linear regression (LIN) model assumes that the relationship between the dependent *continuous* variable y_i and regressors $\vec{x}_i$ is linear. This relationship is modeled through an error variable ϵ, an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$\vec{y} = \mathbf{X}\vec{\beta} + \vec{\epsilon},$$

where $\mathbf{X}$ is the data matrix with rows as data points and columns as features. $\beta_j \in \mathbb{R}^p+$ are coefficients for the features.

There are many different algorithms for parameter estimation. One of the common techniques is to minimize the sum of squared residuals. This leads to a closed-form expression for the estimated value of the unknown parameters β_j [29],

$$\hat{\vec{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\vec{y}.$$

Other methods include variations of maximum-likelihood estimations and other forms of least squares methods (see [29] for a comprehensive list).

When y_i is binary in a classification problem, Linear Regression can be used for approximating the classification results. The model is built by first treating the predictive variables in the training set as continuous. When the final model is applied to a new data point in the testing set, we discretize the returned continuous predictive variable by thresholding, i.e. setting a threshold value and if the returned continuous predictive variable is greater than the threshold, then we discretize the ouput as 1, otherwise -1. An implementation in MATLAB `LinearModel.fit` can be used.

A related technique, logistic regression (LOG) can be used for classification problems. Given a data point $\vec{x}$ and two possible classes 0 and 1, we denote the probability of the data point belonging to class 1 as $\pi(\vec{x})$. In the training set $\mathcal{T}$, $\pi(\vec{x})$ can be estimated by calculating the proportion of training data points that belong to class 1. Logistic regression uses the ideas already developed in linear regression. For each data point $\vec{x}$, the estimated log-odds, or the natural log of ratio $\frac{\pi(\vec{x})}{1-\pi(\vec{x})}$, is used in place of predictive variable y in linear regression:

$$\log \frac{\pi(\vec{x})}{1-\pi(\vec{x})} = \vec{x}^T\vec{\beta}. \tag{2.9}$$

Notice that (2.9) can also be written as

$$\pi(\vec{x}) = \frac{1}{1+e^{-\vec{x}^T\vec{\beta}}}. \tag{2.10}$$

A common method to obtain $\vec{\beta}$ in (2.9) is to use Newton's method to maximize the *log likelihood* of $\vec{\beta}$ given the training set $\mathcal{T}$, $l(\vec{\beta} \mid \mathcal{T})$ [73]. To elaborate, the likelihood of $\vec{\beta}$ given the training set $\mathcal{T}$ can be written as

$$\Pi_{i=1}^{n}\pi(\vec{x}_i)^{y_i}(1-\pi(\vec{x}_i))^{(1-y_i)}$$

where $\vec{x}_i$ and y_i correspond to the training set $\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\ y_i \in [-1, 1], i = 1...n\}$.

The *log* likelihood is

$$\begin{aligned}
l(\vec{\beta} \mid \mathcal{T}) &= \log(\Pi_{i=1}^{n} \pi(\vec{x}_i)^{y_i} (1 - \pi(\vec{x}_i))^{1-y_i}) \\
&= \sum_{i=1}^{n} y_i \log(\pi(\vec{x}_i)) + (1 - y_i) \log(1 - \pi(\vec{x}_i)) \\
&= \sum_{i=1}^{n} y_i [\log(\pi(\vec{x}_i)) - \log(1 - \pi(\vec{x}_i))] + \log(1 - \pi(\vec{x}_i)) \\
&= \sum_{i=1}^{n} y_i \log \frac{\pi(\vec{x}_i)}{1 - \pi(\vec{x}_i)} + \log(1 - \pi(\vec{x}_i)) \\
&= \sum_{i=1}^{n} y_i \vec{x}_i^T \vec{\beta} - \log(1 + e^{\vec{x}_i^T \vec{\beta}})
\end{aligned}$$

where in the next-to-last step we use (2.9) and (2.10).

To maximize $l(\vec{\beta} \mid \mathcal{T})$, we differentiate $l(\vec{\beta} \mid \mathcal{T})$ and set the result after differentiation to zero. However, there is no closed form solution to $\vec{\beta}$ from this approach [73]. A common method of solving $\vec{\beta}$ is to use Newton's method: we first start with an initial guess of $\vec{\beta}$, $\vec{\beta}_0$. Then at each iteration, we update $\vec{\beta}$ by

$$\vec{\beta}_{t+1} = \vec{\beta}_t - [Hl(\vec{\beta}_t \mid \mathcal{T})]^{-1} \nabla l(\vec{\beta}_t \mid \mathcal{T})$$

where $t \geq 0$ is the iteration number, $Hl(\vec{\beta}_t \mid \mathcal{T})$ is the Hessian matrix of $l(\vec{\beta}_t)$ and $\nabla l(\vec{\beta}_t \mid \mathcal{T})$ is the gradient of $l(\vec{\beta}_t \mid \mathcal{T})$.

This process terminates when $\vec{\beta}$ converges or a maximum number of iterations is reached. There are also modifications to this basic Newton's method. In addition, other methods, different from Newton's methods, also exist to solve for $\vec{\beta}$ (see [73] for details).

When solving both linear regression and logistic regression, alternative regularized versions of least squares and maximum likelihood formulation use Lasso regularization, which adds a constraint that $\|\vec{\beta}\|^1$, the L^1-norm of the parameter vector, is no greater than a given value. Quadratic programming can be used to obtain $\vec{\beta}$. The added constraint involving $\|\vec{\beta}\|^1$ imposes a L^1-regularization that penalizes large values of $\vec{\beta}$ [100]. One of the implementations used for logistic regression is MATLAB's `lassoglm` function.

2.7. Ensemble Method

A family of methods, combining multiple classifiers strategically to solve problems such as classification and regression, is the Ensemble Method. The basic paradigm of ensemble methods is to combine different models (or classifiers). The aim of ensemble methods in classification context is to reduce prediction error and improve performance of a model.

There are several types of ensemble methods. The most commonly used are Bagging, Boosting and AdaBoost [88].

Bagging or bootstrap aggregating, uses bootstrapped subsets of the training set $\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\, y_i \in \mathbf{C} = \{C_1, C_2, ...C_m\}, i = 1...n\}$ where $\vec{x}_i$ is the feature-vector for training data point i, y_i is the class label of i and $\mathbf{C}$ is a set of possible class labels. The pseudo code is displayed in Algorithm 1. The inputs are 1) a set of training data $\mathcal{T}$, 2) a classification algorithm $\mathcal{A} : \hat{\mathcal{T}} \rightarrow f$, where $\hat{\mathcal{T}} \subseteq \mathcal{T}$ and f is a classifier of the form $f : \vec{x} \rightarrow y$ where $\vec{x}$ is a feature vector and y is a class label, 3) Integer N specifying number of iterations and 4) a percentage F to create bootstrapped training data. Throughout the algorithm, we keep track of a set of classifiers or the ensemble and denote it as $\mathcal{E}$.

In the Bagging algorithm, N classifiers are constructed as follows: First, a subset is obtained by randomly drawing a fixed percentage F of training data with replacement. Each subset is used as training for the algorithm $\mathcal{A}$ that outputs a classifier f_t that works best (for the tuning parameters) on the subset, where t is the index for the number t^{th} classifier trained. After all N f_t are trained, individual classifiers are then combined by taking a simple majority vote of their decisions. For any given instance, the class label from the possible set of $\mathbf{C}$ chosen by most number of classifiers is the ensemble decision. Ties in the voting can be broken by repeating the above mentioned process of bootstrapping and training additional classifiers until some class label has votes of the majority of classifiers.

Algorithm 1 Bagging

```
Inputs: Training set $\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p, y_i \in \mathbf{C} = \{C_1, C_2, ...C_m\}, i = 1...n\}$, a classification algorithm $\mathcal{A}$, the number of iterations $N$ and the percentage $F$ to bootstrap.
Initialize: $\mathcal{E} \leftarrow \emptyset$, where $\mathcal{E}$ is a set of classifiers or the ensemble.
Do t = 1,...,N
    1. Take a bootstrapped subsets $\mathcal{T}_t$ by randomly drawing $F$ percent of $\mathcal{T}$.
    2. Call $f_t \leftarrow \mathcal{A}(\mathcal{T}_t)$.
    3. Add $f_t$ to the ensemble, $\mathcal{E} \leftarrow \mathcal{E} \cup \{f_t\}$.
End
Test: Simple Majority Voting - When an unlabeled instance $\vec{x}$ is to be classified
    1. Evaluate the ensemble $\mathcal{E} = \{f_1, ..., f_N\}$ on $\vec{x}$, $f_1(\vec{x}), ..., f_N(\vec{x})$.
    2. Let $v_{t,j} = \mathbb{1}\{f_t(\vec{x}) = C_j\}$, where $\mathbb{1}$ is the indicator function.
    3. Obtain total vote received by each class, $V_j = \sum_{t=1}^{N} v_{t,j}$ , $j = 1, ..., m$.
    4. Set $y \leftarrow C_{\text{argmax}_j\{V_j\}}$, the class that receives the highest total vote.
Output: $y$
```

Boosting like Bagging also resamples training data and creates an ensemble of classifiers using majority voting. The difference between Boosting and Bagging is that Boosting trains the classifiers in consecutive order, while bagging trains classifiers in parallel. A bigger difference is that only three classifiers are produced in Boosting, while in Bagging an arbitrary number of classifiers are produced. The pseudo code is displayed in Algorithm 2. Similar to Bagging, the inputs are 1) a set of training data $\mathcal{T}$ and 2) a classification algorithm $\mathcal{A}$.

In Boosting in Algorithm 2, there are three classifiers trained. The first classifier f_1 is trained by algorithm $\mathcal{A}$ using a random subset $\mathcal{T}_1$ of the available training data $\mathcal{T}$. The second training subset $\mathcal{T}_2$ that is used to train f_2 is created by using f_1. The third training subset $\mathcal{T}_3$ that is used to train f_3 is created by using both f_1 and f_2. The three classifiers are combined through a majority vote.

It is shown [93] that the error upper bound of boosting can be written in terms of error ϵ of $\mathcal{A}$ in Algorithm 2: $f(\epsilon) = 3\epsilon^2 - 2\epsilon^3$. For $\epsilon < \frac{1}{2}$, $f(\epsilon) \leq \epsilon$ – this implies that as long as $\mathcal{A}$ is better than random guessing ($\epsilon < \frac{1}{2}$), then the model constructed by boosting is better than using $\mathcal{A}$ alone ($f(\epsilon) \leq \epsilon$). However, the limitation of boosting is that it can only deals with binary classification problem.

AdaBoost (ENSEM) is the best known of all ensemble methods [39]. Like the previous two ensemble methods mentioned, AdaBoost uses bootstrapped training data samples. Unlike the prvious two methods, each training data point in AdaBoost is assigned with a weight. The weights of all training data points are equal in the first iteration and these weights are summed to one, i.e. the weights are all one over the number of training data points. At each subsequent iteration, the weights are updated so that the previously misclassified training data points get higher weights than the previously correctly classified points. By increasing the weights of misclassified points, AdaBoost forces the subsequent classifiers to focus more on these points. After a pre-specified number of classifiers are constructed, all of them are combined through weighted majority voting.

The weights of training data are the same at the first iteration, so that all instances have equal probability to be drawn. At each iteration t, a new training set is drawn, and a classification algorithm $\mathcal{A}$ is trained to produce a classifier f_t. The error of this classifier is calculated as the sum of distribution weights of the instances misclassified by f_t. If the error is greater than $\frac{1}{2}$, the algorithm aborts. Otherwise we update the weights of training data according to whether it is misclassified. If it is, then the weight is increased, otherwise it is decreased. Once the training is complete, test data are classified by this ensemble of N classifiers using weighted majority voting, where each classifier receives a voting weight that is inversely proportional to its error. The weighted majority voting then chooses the class receiving the highest total vote from all classifiers. A more detailed description is displayed in Algorithm 3 [88].

It is shown that [39], the training error of AdaBoost is bound above by

$$2^N \Pi_{t=1}^{N} \sqrt{\epsilon_t(1-\epsilon_t)}$$

where ϵ_t is the error of classifier f_t.

Algorithm 2 Boosting

```
Inputs: Training data $\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\ y_i \in \mathbf{C} = \{C_1, C_2\}, i = 1...n\}$ and a classification algorithm $\mathcal{A}$.
Training:
  1. Select $\mathcal{T}_1 \subset \mathcal{T}$ by sampling without replacement.
  2. Call $f_1 \leftarrow \mathcal{A}(\mathcal{T}_1)$.
  3. Create $\mathcal{T}_2 \subseteq \mathcal{T}$ by the following procedure:
     Initalize $\mathcal{T}_2 \leftarrow \emptyset$
     Set temporary set variable $\mathcal{T}_{temp} = \mathcal{T}$
     While ($\mathcal{T}_{temp} \neq \emptyset$)
       Call $r \leftarrow Rand(0, 1)$, generating a random number between 0 and 1.
       if ($r < 0.5$) then
         Randomly select $(\vec{x}_{temp}, y_{temp}) \in \mathcal{T}_{temp}$
         Update $\mathcal{T}_{temp} \leftarrow \mathcal{T}_{temp} \backslash (\vec{x}_{temp}, y_{temp})$
         if ($f_1(\vec{x}_{temp}) \neq y_{temp}$) then
           $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup (\vec{x}_{temp}, y_{temp})$
       else if ($r \geq 0.5$) then
         Randomly select $(\vec{x}_{temp}, y_{temp}) \in \mathcal{T}_{temp}$
         Update $\mathcal{T}_{temp} \leftarrow \mathcal{T}_{temp} \backslash (\vec{x}_{temp}, y_{temp})$
         if ($f_1(\vec{x}_{temp}) = y_{temp}$) then
           $\mathcal{T}_2 \leftarrow \mathcal{T}_2 \cup (\vec{x}_{temp}, y_{temp})$
  4. Call $f_2 \leftarrow \mathcal{A}(\mathcal{T}_2)$.
  5. Create $\mathcal{T}_3$ by selecting those instances for which $f_1$ and $f_2$ disagree:
     Initalize $\mathcal{T}_3 \leftarrow \emptyset$
     Set temporary set variable $\mathcal{T}_{temp} = \mathcal{T}$
     While ($\mathcal{T}_{temp} \neq \emptyset$)
       Randomly select $(\vec{x}_{temp}, y_{temp}) \in \mathcal{T}_{temp}$
       Update $\mathcal{T}_{temp} \leftarrow \mathcal{T}_{temp} \backslash (\vec{x}_{temp}, y_{temp})$
       if ($f_1(\vec{x}_{temp}) \neq f_2(\vec{x}_{temp})$)
         $\mathcal{T}_3 \leftarrow \mathcal{T}_3 \cup \vec{x}_{temp}$
  6. Call $f_3 \leftarrow \mathcal{A}(\mathcal{T}_3)$.
Test - When an unlabeled instance $\vec{x}$ is to be classified
  Classify $\vec{x}$ with $f_1$, $f_2$ and $f_3$ and assign the label to $\vec{x}$ with the label returned by the majority:
     Let $v_{t,j} = \mathbb{1}\{f_t(\vec{x}) = C_j\}$, where $\mathbb{1}$ is the indicator function.
     Obtain total vote received by each class, $V_j = \sum_{t=1}^{3} v_{t,j}$ , $j = 1, 2$.
     Set $y \leftarrow C_{\text{argmax}_j\{V_j\}}$, the class that receives the highest total vote.
Output: $y$
```

Algorithm 3 AdaBoost

```
**Inputs:** Training data $\mathcal{T} = \{(\vec{x}_i, y_i) \mid \vec{x}_i \in \mathbb{R}^p,\ y_i \in \mathbf{C} = \{C_1, ..., C_m\}, i = 1...n\}$, a classification algorithm $\mathcal{A}$ and an integer $N$ or the number of classifiers.
**Initialize** $D_1(i) = \frac{1}{n}; i = 1, ..., n$, the weight of each training data point.
**Do for** $t = 1, 2, ..., N$:
    1. Draw bootstrap training data subset $\mathcal{T}_t$ according to current weight vector $D_t$.
    2. Call $f_t \leftarrow \mathcal{A}(\mathcal{T}_t)$.
    3. Calculate the error of $f_t : \epsilon_t = \sum_{i=1}^{n} \mathbb{1}\{f_t(\vec{x}_i) \neq y_i\} D_t(i)$.
       if $\epsilon_t > \frac{1}{2}$ then abort.
    4. Calculate normalized error:
       $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
    5. Update weights $D_t$:
       $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } f_t(\vec{x}_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
       where $Z_t$ is a normalization constant chosen such that the components of the vector $D_{t+1}$ are a set of non-negative numbers that sum to 1.
**Test - Weighted Majority Voting:** When an unlabeled instance $\vec{x}$ is to be classified
    1. Obtain total weighted vote received by each class: $V_j = \sum_{t: f_t(\vec{x}) = C_j} log(\frac{1}{\beta_t})$,
       $j = 1, ..., m$
    2. Choose the class that receives the highest total weighted vote as the final classification:
       $y \leftarrow C_{\text{argmax}_j \{V_j\}}$.
**Output:** $y$
```

Chapter 3

Nuclear Material Identification with Supervised Normalized Cut (SNC)

3.1. Background and Chapter Outline

The detection of illicit nuclear materials is of great interest in the efforts to deter and prevent nuclear terrorism. Today's typical approaches to Special Nuclear Material (SNM) detection primarily employ fixed inspection portals, installed at national borders, sea-ports, and traffic and railway checkpoints within the national interior. Although one can detect the presence of radioactive material using simple gamma-ray counting equipment, such as a Geiger counter, this creates a great deal of false-positive errors as some legitimate cargoes such as bananas, fertilizers, kitty litter, tiles and ceramics (containing potassium, ^{40}K), smoke detectors (with americium, ^{241}Am) and colored glass (containing natural uranium) may also generate high radioactivity levels. It is therefore important to identify, not only the presence of a radioactive material, but also its identity. One way of identifying the source is by examining the radiation's spectrum, the number of gamma-rays detected at each energy interval, and finding the best match for that spectrum in a set of spectra obtained from several known SNMs. With low-resolution detectors this task is challenging, even for human experts. It is therefore important to enhance the capabilities for identifying the nuclear material based on the radiation spectrum.

In comparing a given spectrum with that of a set of known SNMs the latter is used as a so-called training set. As such, the illicit SNM detection problem can be cast as a machine learning classification problem. The goal is to classify the target material examined by its acquired spectrum, or a set of spectra gathered by various sensors or in different time intervals, so as to generate information about the material and discern whether or not it poses a threat in a relatively quick manner.

In real-life scenarios of shipping cargo screening, training sets are usually used in several screening methods based on passive radiation counting. In these cases, the training set consists of spectra acquired by detectors when the content of the examined cargo is known.

Thus, different containers, which are known to contain a specific SNM, as well as containers with benign substances (such as bananas, colored glass or without any radioactive material) are placed in front of the detector. For each of these materials a set of spectra is acquired and labeled accordingly. Upon an arrival of a new container with unknown content a new set of spectra (with unknown labels) is acquired. The purpose of the classification is to group these unknown spectra with the best matched samples from the training set.

There are two methods of detection - passive and active interrogation. Passive interrogation measures a material's emitted radiation. As such, passive interrogation is limited by the rate and the energy of natural radioactivities and their attenuation through shielding. Due to these shortcomings, the active interrogation alternative was proposed for the nuclear material detection task [6, 81], especially for use on cargo at ports of entry. In active interrogation, the target is irradiated by bremsstrahlung x-rays [6] or highly penetrating neutrons [81] in order to produce spectra that are characteristic to each SNM. Still, even with active interrogation the identification of nuclear materials by its acquired spectra is difficult due to physical limitations of nuclear radiation detectors, the presence of background noise, and intervening shielding materials.

Different types of detectors deliver spectra with different merits. High Purity Germanium (HPGe) gamma detectors have excellent energy resolution. However they are expensive and require cryogenic cooling, making field use cumbersome. Sodium Iodide (NaI) detectors are less expensive and do not require cooling, and the quality of the delivered spectra is lesser. Plastic scintillators are detectors which do not require cooling nor high maintenance and as such are more practical for nuclear field detection applications. The trade-off is that these detectors produce low-resolution spectra which are very challenging to analyze. Even for human experts the differentiation between the spectra produced by plutonium and those produced by uranium is subtle. Data mining and pattern recognition techniques tailored for nuclear data have the potential of enhancing the ability to differentiate between different SNMs and make up for the hardware shortcomings. Several such methods reported in the literature include: artificial neural networks [58]; naive Bayesian framework classification [16]; support vector machine [44]; and graph theory based techniques [74]. However, all these tools were used on measurements recorded by high-resolution HPGe detectors. Other than the references above, we find no systematic efforts in the literature to construct a robust automated technique to identify nuclear threats. In addition to the problems such as the effect of intervening cargo on signal distortion, one reason for this is the lack of a comprehensive data set of SNM spectral signatures.

Swanberg et al. [97], have recently acquired spectral data from a plastic scintillator detector by active interrogation. They produced the only data set currently available that presents spectra of SNMs acquired by a low-resolution plastic detector. This data set consists of spectra of plutonium, uranium, latite (rock material) and blank. The challenging task, within the scope of this work, is to distinguish between plutonium and uranium. Recent studies [69] demonstrated that a classification of plutonium's versus uranium's spectra can be accomplished when high-resolution HPGe detectors are employed. Here we show that this classification task can be accomplished by employing appropriate data-mining techniques on low-resolution spectra, acquired by plastic detectors.

The data sets obtained by Swanberg et al. are the basis for our computational study, which presents, for the first time, a graph theory based method for classifying low-resolution spectra. This provides preliminary evidence that the use of the inexpensive and low-maintenance plastic detectors with data mining techniques for the purpose of detecting illicit nuclear material is practical. Furthermore, our results appear to be promising enough to encourage the testing of the technique we propose for this task, the *supervised normalized cut*, in other areas of data mining and classification contexts as well.

The SNC method is compared here with two traditional data mining techniques – three variants of support vector machine (SVM) and linear discriminant analysis (LDA). The results of this study suggest that SNC is preferred to SVM (with or without feature reduction) for the task of nuclear material detection and might be better suited than these techniques for other classification problems.

In this study, the feature-vectors represent the spectra from the dynamic decay of fission products of ^{239}Pu and ^{235}U, the radioactivities induced in latite, and the radioactivities induced in background materials; the y's are plutonium, uranium, latite and blank.

This chapter is organized as follows: Section 3.2 describes the SNC method. Section 3.3 describes how the data were generated and explores different ways to present the acquired data. Section 3.4 presents the classification results, both in terms of accuracy and running times. Section 3.5 concludes the chapter.

3.2. Supervised Normalized Cut

Binary Classification

The binary classification problem is formalized as a graph bi-partitioning problem. As introduced in 1.3, An undirected (complete) graph $G = (V, E)$ is constructed, where each node $v \in V$ corresponds to a data point – in our case, a feature-vector (see Section 3.3) associated with a set of spectra acquired from a material sample.

Recall that from Chapter 2.1, we know that normalized cut prime (NC′) can be written as

$$NC'(S^*) = \min_{S \subset V} \frac{C(S, \bar{S})}{C(S, S)}. \tag{3.1}$$

The NC′ solution procedure requires to assign, in advance, a single node which will be included in the source S (or sink $\bar{S}$) set (see [49] for details). This node is referred to as a *seed node*. Here, we exploit the seed node mechanism in order to force a-priori the training data to be in either in the source S or in the sink $\bar{S}$, based on the material from which they were acquired. Specifically, the input consists of three sets: two sets of nodes, A and B, which are associated with feature-vectors acquired from two different known materials, M^1 and M^2, and a third set, I, corresponding to feature-vectors acquired from an unknown material or materials. The goal of the binary classification problem is to associate each feature-vector in I with either M^1 or M^2.

The input to the classification problem is the complete graph, $G = (V, E)$, defined on the set of objects $V = A \cup B \cup I$ and the similarity weights associated with each pair of nodes (edge) $[i, j] \in E$. Two nodes s and t are added to the graph with an arc of infinite weight from s to each node in A and from each node in B to t. On this graph we seek a partition that minimizes the NC$'$ criterion so that $s \in S, t \in \bar{S}$. The nodes in I which end up in S are classified as A, i.e., acquired from material M^1 and nodes in I which end in $\bar{S}$ are classified as B, thus acquired from M^2. This process is illustrated in Figure 3.1.

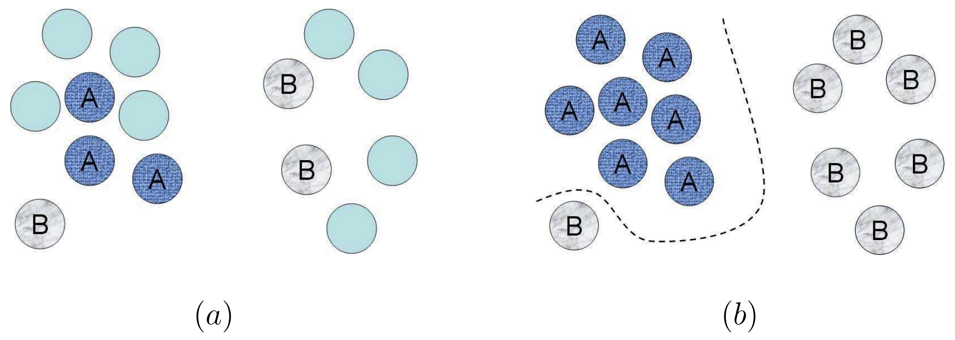


Figure 3.1: (a) The input with the training sets A (dark-blue) and B (light-gray) and the unclassified nodes C (light-blue); (b) The solution: the two sets are separated by a minimum cut. The set on the left consists of the nodes, classified as A nodes, forms the set S; and the set on the right of the B nodes is $\bar{S}$, where the similarity within S and the dis-similarity between the two sets are high.

The adjustment of NC$'$ to a supervised context, as described above, is a new *supervised* classification methodology, which takes advantage of the solvability of NC$'$, and broadens the application of NC$'$ to a wider class of problems.

As mentioned in 2.1, the efficiency of NC$'$ algorithm was established in [49], where it was shown that NC$'$ is solvable in the running time of a minimum s,t-cut problem, which is strongly polynomial and combinatorial. In the context of Supervised Normalized Cut, the only additional step before performing NC$'$ solution procedure is to separate and assigning training data to belong to the source or the sink set, which does not affect the time complexity. Thus SNC is efficient, running in polynomial time.

Multi-classification

The binary classification with NC$'$ is used here as a subroutine for solving multi-classification problems, involving three or more different classes. Multi-classification is more realistic in the context of nuclear threat detection, as it is necessary to identify, e.g. the contents of cargo, as one of an array of possible materials.

Since the number of illicit radioactive substances is finite and well defined [55], the respective multi-classification problem is to classify into K classes, where K is known in advance. Here we solve the multi-classification problem by repeated calls to a binary classification subroutine. This is a common practice in many multi-classification techniques [24, 59].

For multi-classification we utilize a scheme generally referred to as one-vs-all decomposition (e.g., [30, 90]). For a problem with K different classes, we create K different binary problems. The k^{th} binary problem is to classify the unknown nodes, I, into two classes - material M^k, or not-M^k, E (stands for *Else*). Each node is classified by K binary classifiers. The label of the node is determined to be class k, if it was classified as M^k. If the node was classified as material more than once, all possible materials are reported. If the node is classified as E for all k, then the label is undecided.

In the case where all the nodes in I are acquired from the same container, a voting can be used to determine the final grouping of all nodes in I. In this case, the label of a node is determined to be the class k which has the highest score. If there is more than one class with the highest score, the tie is broken arbitrarily or both materials are reported as possible classification. If the highest score is zero, then the label of the node is undecided.

Figure 3.2 demonstrates multi-classification to four possible materials or classes A, B, C and D. Three unclassified data points need to be classified. Training data are provided for each class as shown in Figure 3.2 (i). Figure 3.2 (a) – (d) are the binary classifications for each class A through D. Figure 3.2 (r) shows the combined results of all subproblems, two of the unknown points are successfully classified to A and B. The middle unclassified node is undecided, because it has E for all its labels.

3.3. Data and Experimental Setup

The measurement data for the nuclear classification problem were acquired in a controlled environment with plastic detectors. We use here a data set of active interrogation of plutonium and uranium made available by Swanberg et al. [97]. In this experiment, a sample of, either blank; 0.19 grams of ^{235}U; 0.568 grams of ^{239}Pu; or 3 grams of latite, an igneous rock material; was placed in a cave and irradiated for 30 seconds with neutrons generated by the 88 inch cyclotron at Lawrence Berkeley Laboratory [81]. When irradiated with neutrons, materials may become radioactive or undergo nuclear fission. Activation products and fission products from different materials have different characteristic gamma rays and decay times. These are the characteristics that we use as the pattern distinguishing a specific nuclear material from others.

The target was exposed to the detectors for a total of 25 seconds and each 2.5-second interval yields a cumulative energy spectrum measurement for that interval. The detector system measured energies in the range from approximately 100 keV to 14 MeV using 1024 channels.

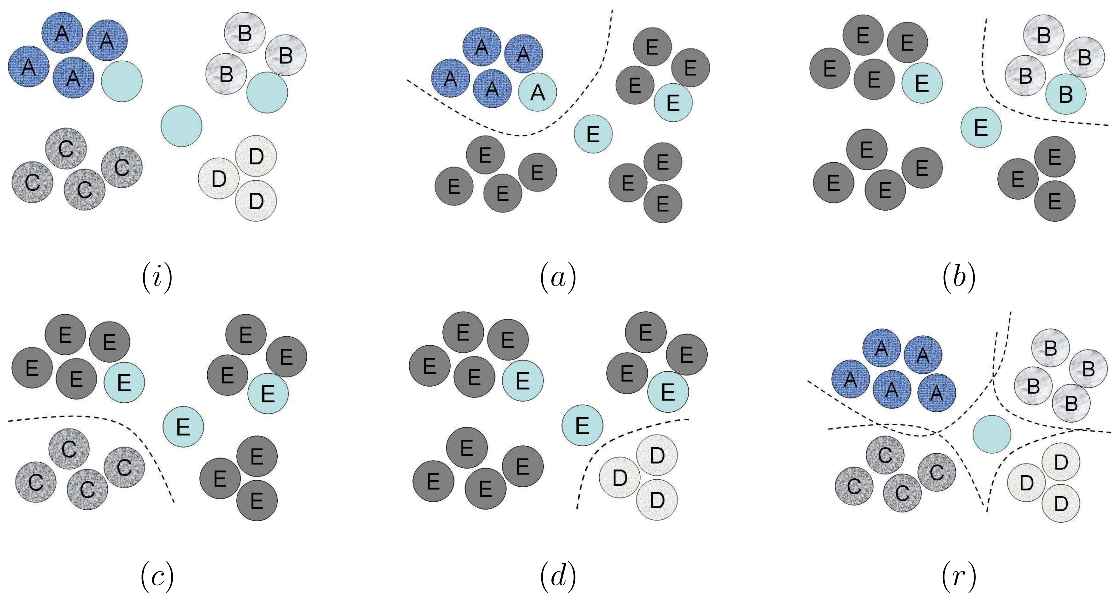


Figure 3.2: Multi-classification: (i) A, B, C, and D are four distinct materials or classes. (a)–(d) are $A - D$ decomposed binary classifications. Each classification gives a label to the unknown points. (r) The final multi-classification combines all the labels for a given unknown point.

The data set used in our experiments consists of the measurements reported in [97] and additional measurements that were made available by the same authors. The additional data include 10 measurements for each run. In total, 275 runs were conducted: 20 with blank, 22 with latite, 92 with uranium, ^{235}U and 140 with plutonium, ^{239}Pu, resulting in a total of 2750 acquired spectra.

The Detector Live-time

During the data acquisition, the operator set the detector to a nominal run-time of 2.5 seconds. The actual acquisition time of the detector, however, depends on the particularities of each run. Specifically, when the gamma-rays arrive at high frequency, the detector does not process all of them due to hardware limitations. Therefore the length of the actual run-time, or the so-called *live-time*, is shorter than the nominal run-time or real time. To correct this inherent hardware bias, we adjust for the live times of the detector in each run by rescaling each gamma-ray count by the instrument's live time produced with the data. This means that the gamma-ray counts in each spectrum are scaled (divided) by the live-time associated with that spectrum measurement. The results presented here use such scaled data.

Feature-vectors and Data Analysis

Each target was placed in front of the detector for a run of 25 seconds. The spectrum, which is the energy histogram of the gamma-ray received by the detector, was recorded every 2.5 seconds. Each entry in the histogram corresponds to a different energy band (channel). Hence 10 spectral measurements, taken at consecutive periods of time, were produced in every run.

The obtained data can be regarded as a two-dimensional array composed of the gamma-ray counts for each energy channel in each of the 10 given measurements. For $1,024$ energy channels and 10 consecutive measurements, each such sample is an array with $1,024$ columns and 10 rows. Each row vector is denoted by $\vec{s}_i$, for $i = [1, 2, \cdots, 10]$, where $\vec{s}_1$ corresponds to the first 2.5-second interval and $\vec{s}_i$ corresponds to the i^{th} 2.5-second interval. To convert this $2D$ array into a feature-vector, four different methods are considered:

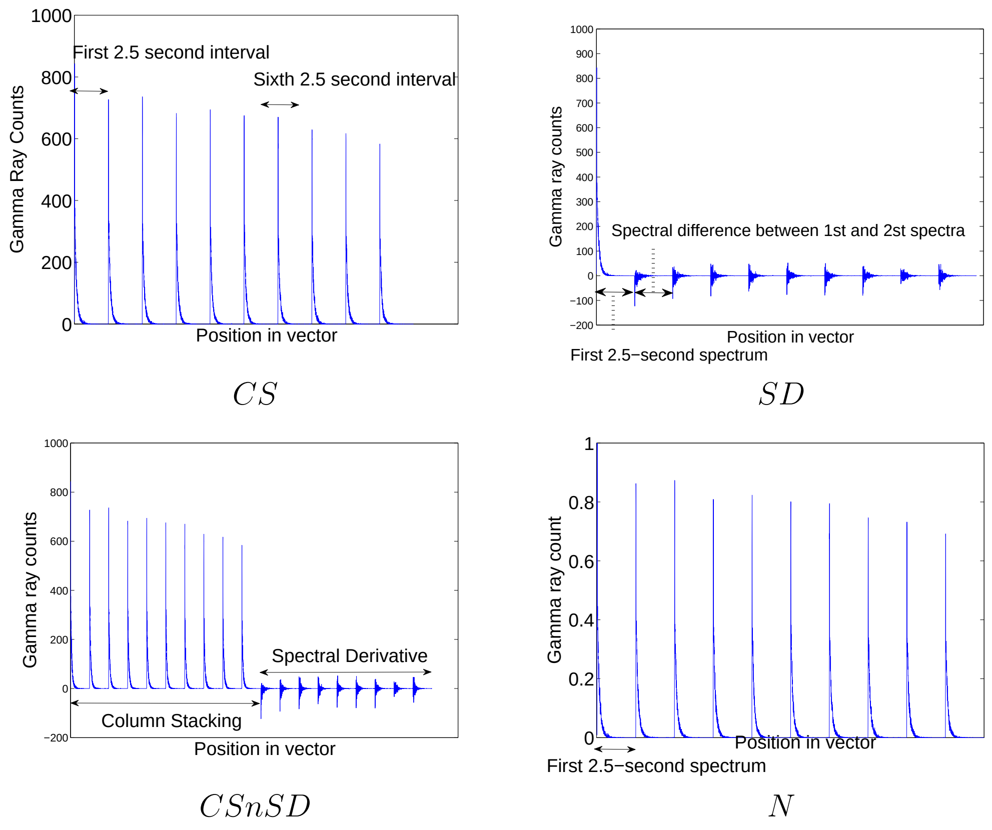


Figure 3.3: Feature-vectors produced by column stacking (CS); spectral difference (SD); column stacking and spectral difference (CSnSD) and Normalization of the data (N). For CS, SD and N, there are 10240 indices in the vectors; for CSnSD, there are 19456 indices.

- Column Stacking (CS): The vectors $\vec{s}_1, ..., \vec{s}_{10}$ are concatenated to a vector of length 10240. The feature-vector produced is $(\vec{s}_1, \vec{s}_2 ..., \vec{s}_{10})$ as in Figure 3.3 (top left).

- Spectral Difference (SD): The purpose of this method is to emphasize the radioactive decay captured in the temporal domain. To do that, we concatenate the first spectrum vector, followed by the difference between the second spectrum vector and the first spectrum vector, and in general the ith entry is the difference between the i^{th} vector and the $(i-1)^{th}$ vector. The feature-vector produced is then $(\vec{s}_1, \vec{s}_2 - \vec{s}_1, \vec{s}_3 - \vec{s}_2, \ldots, \vec{s}_{10} - \vec{s}_9)$ as in Figure 3.3 (top right).

- Column Stacking and Spectral Difference (CSnSD): Here we join (concatenate) the feature-vectors resulting from CS and SD into a single feature-vector. The feature-vector produced is $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_{10}, \vec{s}_2 - \vec{s}_1, \vec{s}_3 - \vec{s}_2, \ldots, \vec{s}_{10} - \vec{s}_9)$ as in Figure 3.3 (bottom left).

- Normalization (N): In order to remove the dependence on absolute counts of the spectra, which grow with the sample quantity and weight, while preserving the general patterns, we *normalize* the CS feature-vector. This is done by dividing the entries of the feature vector by the largest entry of that vector. Let $s_{max} = \max_{i=1,\ldots,10; j=1,\ldots,1024} (\vec{s}_i)_j$, then the N feature vector is $(\vec{s}_1/s_{max}, \vec{s}_2/s_{max}, \ldots, \vec{s}_{10}/s_{max})$ as in Figure 3.3 (bottom right).

The intuition behind using the above four methods is that they are simple and the first step attempt to incorporate both spatial and temporal dimensions of our dataset. Using these feature-vectors serves two goals: (i) converting the $2D$ data to $1D$; and (ii) capturing the local temporal changes of spectra. To achieve these two goals, one could apply more sophisticated signal processing methods, such as Pyramid Transform, Discrete Wavelets Transform (DWT) or Discrete Cosine Transform (DCT). A possible advantage of using these signal processing transformations is that they may result in better posed covariance matrices of the data, which can potentially improve the classification accuracy of PCA and LDA. However, these signal transformations require more computation time than the simple methods utilized here, and the trade-off between the added computational complexity and better accuracy should be investigated. Within the scope of this chapter we have utilized only the aforementioned feature-vectors construction methods, which are simple and computationally efficient.

3.4. Results

In this section, results concerning different aspects of our method are presented: Section 3.4 establishes standards for measuring the quality of a classification technique in order to compare across different methods. Section 3.4 describes the classification methods used here to compare to SNC and addresses practical aspects of classification methods such as feature selection. In Section 3.4, classification methods, including various versions of support vector machine (SVM), Linear Discriminant Analysis (LDA) and SNC are presented for the Swanberg et al. nuclear data [97]. Section 3.4 gives a more detailed account on the results of SNC. Section 3.4 compares the methods in terms of running times. Multi-classification results are shown in Section 3.4. Finally, Section 3.4 presents the influence of different constructions of feature-vectors on the different algorithms running time.

As described in Chapter 1, a classification procedure consists of two stages: "training" stage, where one tries to infer a functional relation (i.e. $y = f(x)$) between a training set $(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)$ and its known labels, y's; and "testing" phase in which the labels $y_{m+1}, y_{m+2}, ...$ of unlabeled input vectors $\vec{x}_{m+1}, \vec{x}_{m+2}, \cdots$ are estimated. The outcome of the training phase is a classifier that is used for labeling new data points in the testing phase. For SVM and PCA the testing is very quick. We produce an analogous testing phase for SNC. The output of the training phase of SNC is a bipartition of the training and other data points. When a new data point becomes available the testing phase assigns that point to the side of the bipartition that increases the least the objective value of NC$'$ criterion ($\frac{C(S,\bar{S})}{C(S,S)}$, the objective in $Eq.$2.1). This process involves a comparison of few values and it is at least as fast as the testing phase for PCA and SVM.

Our study establishes that SNC is faster than SVM and PCA in the training phase, and all three methods are almost instantaneous, and thus on par with each other, in the testing phase. Since SNC is also at least as accurate as the other methods, it should be the preferred technique. The speed of the training phase with SNC makes the re-calibration of the algorithm with changing conditions easy and fast, and therefore can be done more frequently than if one is to use the other techniques. Therefore with SNC it is possible to retain a more accurate and updated classification model.

Quality of Classification

In the machine learning community, the quality of a classification method is generally measured by simulation–applying the method to a known data set. Here we use an extended version of the data reported in [97] consisting of 275 data points in the form of feature-vectors, each labeled with its underlying material: blank (20 samples), latite (22 samples), ^{235}U (92 samples), or ^{239}Pu (140 samples).

To study the performance of the method we apply *random sub-sampling* which divides the data set into two subsets: training and testing. The training data are used to construct a classifier, and the labels of the testing data are hidden–i.e., we pretend that the labels are unknown. The classifier provides *predicted* labels, which are then compared to the true labels. The accuracy of a classifier is the fraction of the correct predictions across all testing data. For statistical significance purposes, this sub-sampling and classification are repeated 100 times. Each time the training and testing sets are re-sampled. The *accuracy of a classification method on a certain data set* can then be defined as the mean accuracy of these 100 runs. In addition, standard deviation and 95% confidence interval of these runs can be calculated. These measure the *consistency of a classification method on a certain data* *set*. The lower the standard deviation and the confidence interval are, the more consistent the method.

Random sub-sampling can involve different training-testing ratios, e.g. $40\% - 60\%$. A $40\% - 60\%$ ratio means that 40% of the total data are used for training and the other 60% are used for testing. In our experiments we used the following ratios: $50\% - 50\%$, $40\% - 60\%$, $30\% - 70\%$, $20\% - 80\%$, and $10\% - 90\%$. As the size of the training data decreases, less information is provided to construct the corresponding classifier. Thus the accuracy of the classifier decreases. A more *robust* classification method is one that is less affected by the decreasing size of the training data.

Classification Methods

Chapter 2 has already described several machine learning methods. Out of those methods, the feature reduction technique we test here is principal component analysis (PCA). Recall that in order to evaluate the principal components, one has to compute the covariance matrix for all acquired spectra. Since the feature-vectors used here contain more than $11,200$ coefficients each (see Section 3.3), finding this covariance matrix and its eigen-vectors is computationally intractable. When evaluating smaller feature-vectors (with $5,000$ coefficients) the SNC method is 150 times faster than PCA. In addition, the gain in accuracy results from applying PCA before applying SVM is less than 1%. Therefore, for the task in hand, the use of PCA in this context is unlikely to improve the overall quality of the detection while significantly slowing down the detection speed.

The machine learning methods we choose to compare with our SNC are classical SVM, feature-reducing linear kernel 1-norm SVM (SVM-1), recursive feature elimination SVM (SVM-RFE), feature-reducing newton method for LP SVM (SVM-NLP), and linear discrminant anaylsis (LDA). As described in Chapter 2 We tune the parameters of SVM by setting them in a grid, $\{2^{-7}, 2^{-6}, 2^{-5}, \cdots, 2^{6}, 2^{7}\}$, the searching range used in [68]. The SVM classification is then performed for all possible parameters' combinations. The parameters' set that produces the best classification results is the one used for the evaluation. This tuning procedure gives the best possible accuracy of a particular SVM classifier [14, 25]. It is important to note, that while these tuning times are not included in our runtime comparison, the tuning process is time consuming. The variants of SVM are tuned similarly. The SVM-RFE method requires an additional parameter – the number of remaining features. We tune this parameter by exhaustive search in the space $\{2^1, 2^2, 2^3, ... 2^{13}\}$.

As for linear discriminant analysis (LDA), since the dimensionality of our data (the combined number of energy channels) is far larger than the number of data points, the suggested method is used in Chapter 2 is used to perform PCA first before using LDA for classification.

To compare PCA-LDA, SVMs and SNC, we use the four types of feature-vectors, {CS, SD, CSnSD, N}, described in Section 3.3. For PCA-LDA, the reduced version of PCA – mentioned above, for the reason of computational cost – is used: the PCA is performed first by centering data points and adding first few important principal components until the total variance accounted for is just above 80%. Then the standard LDA is used for training classifier and classifying the test data. One note here is that the data is not scaled before PCA is applied: this is because, by scaling data, in many instances, the resulting covariance matrix becomes singular for the training data from our dataset – thus LDA could not proceed. Therefore, the decision is made to use only centered data while performing PCA before LDA.

Binary Classification

The performance of the binary classification method, SNC, is tested here. It is compared to two common classification methods: Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA), as well as the three specialized feature-reducing SVMs: 1-NORM SVM (SVM-1), recursive feature elimination SVM (SVM-RFE) and Newton method LP SVM (SVM-NLP).

To solve SNC, the graph construction is written in MATLAB. The resulting minimum cut problem is solved with Hochbaum's PseudoFlow algorithm, HPF, the implementation of which is downloaded from [48]. The similarity between two feature-vectors v_i and v_j is quantified by:

$$w_{ij} = \frac{1}{||v_i - v_j||_2 + \epsilon},$$

for $0 < \epsilon \ll 1$.

Table 3.1 displays the accuracy and the precision of the supervised normalized cut for varying training-testing ratios with different types of feature-vectors. For $50\% - 50\%$ ratio, all four types of feature vectors produce similar results. These results for the different feature vectors are statistically identical as confirmed by ANOVA test failing to reject null hypothesis at 95% significant level for every pair of vectors. However, as the training proportion decreases, the CS feature-vector gives the highest accuracy. This behavior characterized also the standard deviation and the 95% confidence interval, which are also the best for CS.

Table 3.2 details the corresponding results for SVM and specialized SVMs when using either Radial Basis Functions (RBF) or polynomial kernels. Each of these kernels takes user defined parameters including a parameter for soft margin penalty, C. In addition, RBF uses a derivative parameter, σ. For SVM-RFE, the optimal number of features is also displayed (NumFeat) and for SVM-NLP, another parameter ν is included. The table lists the best accuracy results for all the methods and training-testing ratios involved and the corresponding parameters.

Similarly to the results of SNC: CS gives high accuracy in most cases. Unlike the results for SNC, for 50%-50% ratio, CSnSD gives better accuracy when run with SVM-RFE. Still, using CS feature-vectors gives highly accurate results for both the SNC and the SVM methods. We conclude that column stacking (CS) is the best suitable of the feature-vectors to use. Furthermore, for all methods involved that take the RBF kernel (SVM and SVM-1), RBF consistently presents better results than polynomial kernels.

	CS	***SD***	***CSnSD***	***N***
50% – 50%				
mean	99.62%	99.61%	99.52%	99.17%
std	0.43%	0.43%	0.43%	2.83%
95% CI	0.08%	0.08%	0.08%	0.55%
40% – 60%				
mean	99.25%	92.02%	99.62%	95.73%
std	0.34%	16.80%	0.36%	6.76%
95% CI	0.07%	3.29%	0.07%	1.33%
30% – 70%				
mean	99.62%	46.48%	99.56%	92.38%
std	0.30%	6.89%	0.28%	8.27%
95% CI	0.06%	0.08%	0.08%	0.55%
20% – 80%				
mean	99.59%	42.02%	98.98%	80.08%
std	0.23%	1.22%	3.31%	6.43%
95% CI	0.05%	0.24%	0.65%	1.26%
10% – 90%				
mean	98.61%	41.14%	85.91%	50.37%
std	4.24%	1.37%	11.55%	16.77%
95% CI	0.83%	0.27%	2.26%	3.29%

Table 3.1: SNC runs for the feature-vectors {CS, SD, CSnSD, N} with five training-testing ratios. mean is the average accuracy of a prediction based on 100 runs; std and 95% CI are the standard deviation and the 95% confidence interval of the prediction. A higher average indicates a more accurate prediction, while a lower standard deviation and a lower confidence interval indicate higher consistency in the prediction.

Among SVM and specialized SVMs, SVM-1 appears to improve the results of SVM, while SVM-RFE improves the results only in some cases and SVM-NLP does not appear to improve SVM on this set of data.

Table 3.3 displays the detailed results for PCA-LDA. We observe that the results indicate that SD is a more appropriate feature construction for PCA-LDA. This is in contrast to CS as the best and most accurate feature construction for both SNC and SVM. The only case for which SD is not best for PCA-LDA is when training-testing percentage is 10%-90%. In this instance, N gives better accuracy, but SD still gives the most consistent results.

	SVM	***SVM-1***	***SVM-RFE***	***SVM-NLP***
50%	99.59% (CS,C=32,RBF,σ=128)	99.54% (CS,C=32,RBF,σ=128)	99.38% (CSnSD,C=1,F=256)	98.27 (N,C=100,ν=256)
40%	99.60% (CS,C=16,RBF,σ=128)	99.68% (CS,C=16,RBF,σ=128)	99.10% (CS,C=2,F=128)	94.75% (CS,C=100,$\nu=2^{-11}$)
30%	99.46% (CS,C=128,RBF,σ=128)	99.58% (CS,C=64,RBF,σ=128)	98.92% (CS,C=32,F=128)	94.56% (CSnSD,C=100,ν=1)
20%	99.43% (CS,C=64,RBF,σ=128)	99.56% (CS,C=64,RBF,σ=128)	98.88% (CS,C=16,F=512)	94.92% (CS,C=100,ν=0.031)
10%	98.13% (CS,C=128,RBF,σ=128)	98.80% (CS,C=64,RBF,σ=128)	96.74% (CS,C=0.008,F=1024)	93.18% (CS,C=100,ν=0.004)

Table 3.2: SVM, SVM-1, SVM-RFE and SVM-NLP runs with five training percentages – the testing percentage is one minus the training percentage. The best accuracy result for each method and each ratio is listed along with the optimal parameters. F is the optimal number of features for SVM-RFE.

Figure 3.4 summarizes the results of Tables 3.1 to 3.3. In the figure, the highest accuracy is presented for the different training-testing ratios. Examining the graph in Figure 3.4 shows that SNC, in terms of accuracy, is on par or superior both in accuracy and robustness to the methods compared, except SVM-1. SVM-1 for 10%-90% ratio has (slightly) higher accuracy which comes at a price of substantially increase in running time (see Section 3.4). In terms of robustness, which is measured by the decrease in accuracy as the sample size decreases, SNC presents better results than most methods. For example, in the case of 10%-90% training-testing ratio, SNC has a more than 1% accuracy lead over SVM, 2% lead over SVM-RFE and 4% lead over PCA-LDA.

SNC Misclassifications Analysis

A *confusion matrix* is a presentation of the data that helps to identify the source of prediction errors. In a confusion matrix, the rows are the true labels of the data points and the columns are the predictions. For example, an entry at position (Pu, U) corresponds to the average number of data points, over 100 runs, that are incorrectly labeled as uranium (U) when their true identity is plutonium (Pu). According to this definition, the sum of diagonal entries is the number of correctly predicted points. Table 3.4 is the confusion matrices of the results of SNC with CS feature-vectors for the different training-testing ratios. The table shows that all samples acquired in the presence of uranium were labeled with 100% accuracy. It is interesting to observe that the only source of error for all matrices is the misclassification of plutonium samples as uranium samples.

	CS	***SD***	***CSnSD***	***N***
50% – 50%				
mean	76.91%	97.04%	77.56%	93.00%
std	3.39%	1.26%	4.46%	3.72%
95% CI	0.66%	0.25%	0.87%	0.73%
40% – 60%				
mean	76.21%	96.96%	76.03%	93.42%
std	4.02%	1.46%	4.10%	3.86%
95% CI	0.79%	0.29%	0.80%	0.76%
30% – 70%				
mean	76.83%	96.61%	75.55%	93.28%
std	4.33%	1.77%	5.10%	4.25%
95% CI	0.85%	0.35%	1.00%	0.83%
20% – 80%				
mean	74.68%	95.46%	74.55%	93.43%
std	5.45%	2.25%	5.15%	5.66%
95% CI	1.07%	0.44%	1.01%	1.11%
10% – 90%				
mean	74.13%	91.34%	74.76%	94.00%
std	5.90%	4.86%	5.39%	5.05%
95% CI	1.16%	0.95%	1.06%	0.99%

Table 3.3: PCA plus Linear Discriminant Analysis runs for the feature-vectors {CS, SD, CSnSD, N} with five training-testing ratios. "mean" is the average accuracy of a prediction based on 100 runs; "std" and "95% CI" are the standard deviation and the 95% confidence interval of the prediction. A higher average indicates a more accurate prediction, while a lower standard deviation and a lower confidence interval indicate higher consistency in the prediction.

	50% training		*40% training*		*30% training*		*20% training*		*10% training*	
	Pu	**U**	**Pu**	**U**	**Pu**	**U**	**Pu**	**U**	**Pu**	**U**
Pu	69.56	0.44	83.88	0.67	97.38	0.62	111.24	0.76	123.1	2.9
U	0	46	0	55	0	65	0	74	0	83

Table 3.4: Confusion matrices for different training sizes.

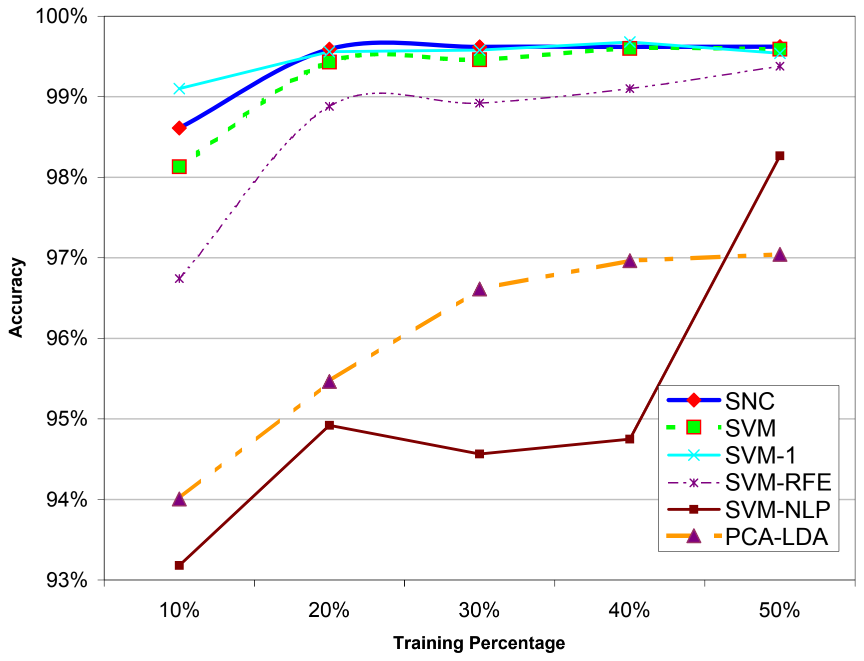


Figure 3.4: The best classification accuracy for SNC, SVM, specialized SVMs and PCA-LDA with different training sizes.

Run Times

We report on the run times of SVM and specialized SVMs, that *exclude* the time required to find the best tuned parameters, including only run times for training and classification. Figure 3.5 graphs the running times of SNC, SVM and PCA-LDA. Since the complexity of SVMs depends on the number of training data points, the smaller the training data, the shorter SVMs' running times. SVM-RFE and SVM-NLP have the longest running times – around 80 times more than that of SNC. SVM-1, whose accuracy result is a bit better at 10%-90% ratio than SNC, is 4 times slower than SNC at that ratio. The running time of SVM-RFE is influenced not only by the ratio, but also by the number of features used. For SNC, the run time of the algorithm is dominated by the graph construction, and therefore appears constant regardless of the size of the training set. PCA-LDA has the running time more than 40 times than that of SNC – this is primarily due to the using of PCA, which is the reduced version; the full version of PCA takes even longer. Figure 3.5 clearly shows that SNC is significantly more efficient (factor of $2-80$) than SVMs and factor of 40 than PCA-LDA under the same hardware setup (1.3GHz Intel SU7300 Core 2 Duo ULV Processor with 1GB $1,066$MHz RAM).

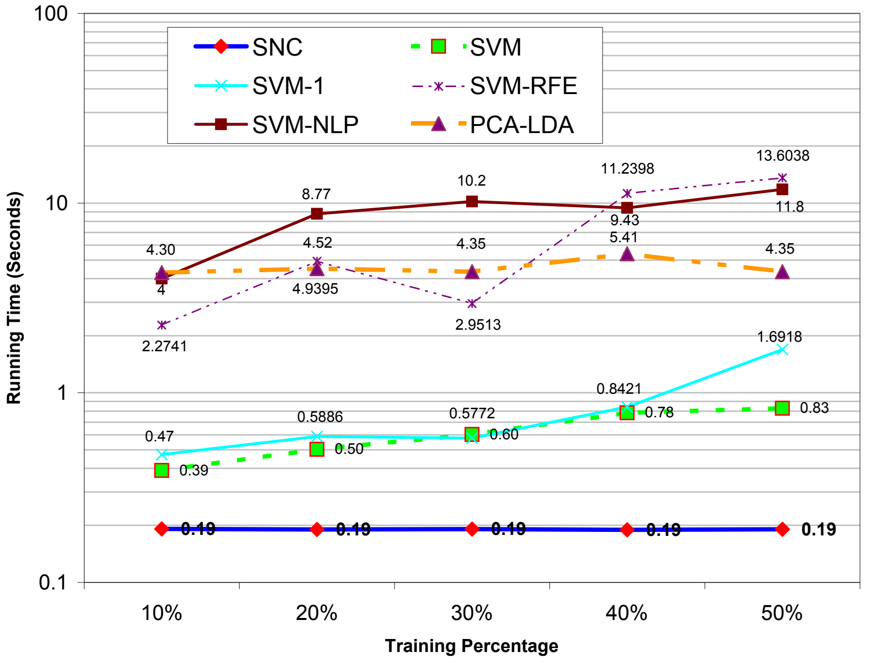


Figure 3.5: Computation times of SNC, SVM, specialized SVMs and PCA-LDA for different training sizes. The graph is drawn in log scale and the running times are labeled next to the curves.

Multi-classification

In this section we evaluate the performance of SNC with respect to SVM and PCA-LDA for solving multi-classification problems. As both SNC and SVM use binary classification as a subroutine for solving the multi-classification problem, we apply the voting mechanism described in Section 3.2 for both methods. Specialized SVMs have less established extension from binary to multi-classifications and are left for future investigations. For the evaluation process we use three subsets of the [97] data set: 1) blank, plutonium, and uranium; 2) latite, plutonium, and uranium; 3) blank, latite, plutonium, and uranium. Note that the latter consists of the entire data set.

The results for SNC of 50%-50% training-testing case, given in Table 3.5, show that SNC-based multi-classification gives highly accurate and consistent prediction for several sets of data. When all four materials are used, the best prediction accuracy is 98.65% under $CSnSD$ feature-vectors. In fact across all permutations, $CSnSD$ is the best feature-vector in terms of both accuracy and consistency. We note that the presence of latite affects the quality of the classification more adversely than blank or background noise, as observed in the difference between the first row and the second row of the table. When all materials are present, the prediction improves from that with latite, plutonium and uranium.

Figure 3.6 displays the comparison among SNC, SVM and PCA-LDA, for 3 different

	CS		*SD*		*CSnSD*		*N*	
	Mean	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**	**Mean**	**Std.**
L, Pu, U	94.85%	3.40%	98.34%	0.98%	98.91%	0.78%	84.61%	0.27%
B, Pu, U	100.00%	0.00%	99.88%	1.21%	100.00%	0.00%	87.37%	0.54%
All four	98.65%	1.25%	98.32%	1.75%	98.65%	1.12%	86.22%	0.62%

Table 3.5: Multi-classification for different permutations of the data set (B: blank; L: latite; Pu: plutonium; U: uranium; all four: B, L, Pu and U) and the four different kinds of feature-vectors {CS, SD, CSnSD, N}. The sub-sampling ratio is 50% − 50%.

classification problems. The results presented in this Figure, for each training portion, are the best results achieved across all processing methods (CS, SD, CSnSD and N). As can be seen in Figure 3.6 for all classification setups and at each training portion SNC produces better accuracy than SVM and PCA-LDA. Furthermore, in terms of robustness, SNC is superior to SVM and PCA-LDA. These results are in agreement with the results of binary classification, which are reported in Section 2.2.

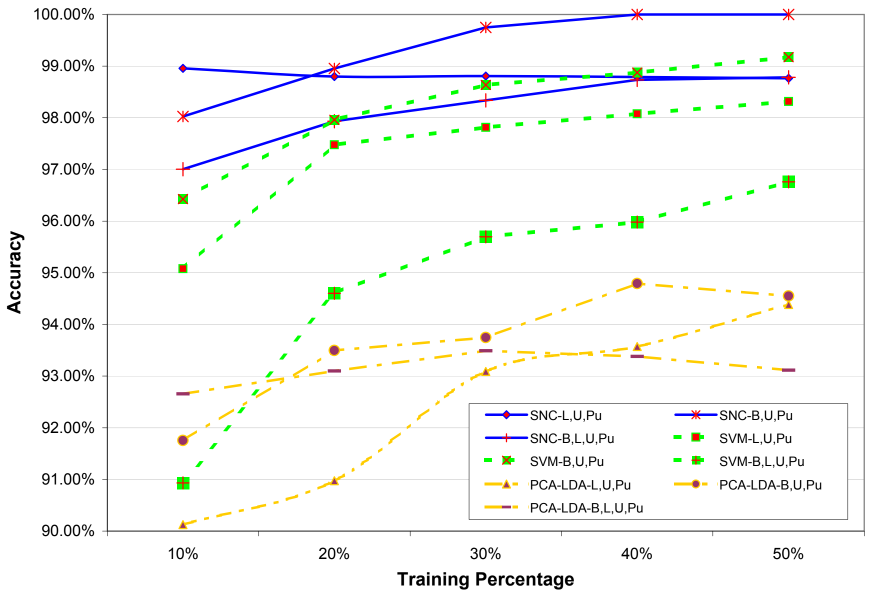


Figure 3.6: The best multi-classification accuracy for SNC, SVM and PCA-LDA with different training sizes.

Figure 3.7 displays the run times of SNC, SVM and PCA-LDA. It is noted that the SVM method requires extensive tuning of parameters for each data set and each feature-vector representation method (Section 3.3). The running times of SVM, reported in Figure 3.7 do not include the time it takes for tuning of the various parameters. Were these times to be included, the superiority of SNC's efficiency would have been even more pronounced.

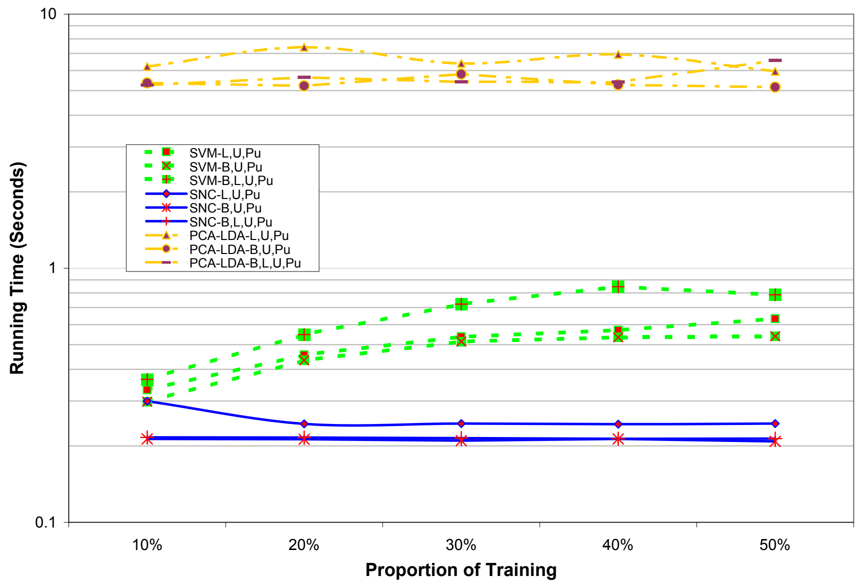


Figure 3.7: Computation times of SNC, SVM and PCA-LDA for different training sizes. The running times of B,U,Pu and L,U,Pu overlap for SNC.

Constructions of Feature-vectors and Running Time

We conclude this section with the presentation of running time results for the different feature-vectors presented in Section 3.3. The major reason of choosing these methods is their simplicity: despite the fact that more complex derived features such as those from signal processing may produce better results, the four methods, proposed here, are simple and incorporate both spatial and temporal dimensions of our data set.

Figure 3.8 displays the running times of various feature-vector constructions with different algorithms – SVM, SNC and PCA-LDA for the 50%-50% training-testing ratio of binary classification problem. Overall, the graph shows, as concluded in the previous sections: SNC is the fastest algorithm involved; SVM is the next fastest, while PCA-LDA is the slowest. We can also observe how sensitive the running times depend on different feature-vector constructions: CSnSD appears to require the most computation than other methods – this is confirmed by the fact that CSnSD gives the longest feature-vectors among all methods. The next method that constructs feature-vectors and requires long running time for all methods is N, followed by CS and SD, which are the fastest for PCA-LDA and SVM respectively and tied for SNC. One should note that the SNC algorithm takes the edge weights as input and thus is not affected by the length of the feature-vectors. The computation times of the weights however, are proportional to the feature-vector lengths. SVM and PCA-LDA have different running times when applied on the different feature-vectors. This observation strongly suggests that if one considers to replace the method for generating the feature-vectors, he should consider not only the method's running times for creating the vectors, but also the total run times with the new vectors.

These observations, combined with the accuracy results from Section 2.2, conclude that CS is the most appropriate feature-vector construction method – using feature-vectors constructed by CS allows algorithms to have fast and accurate classification results.

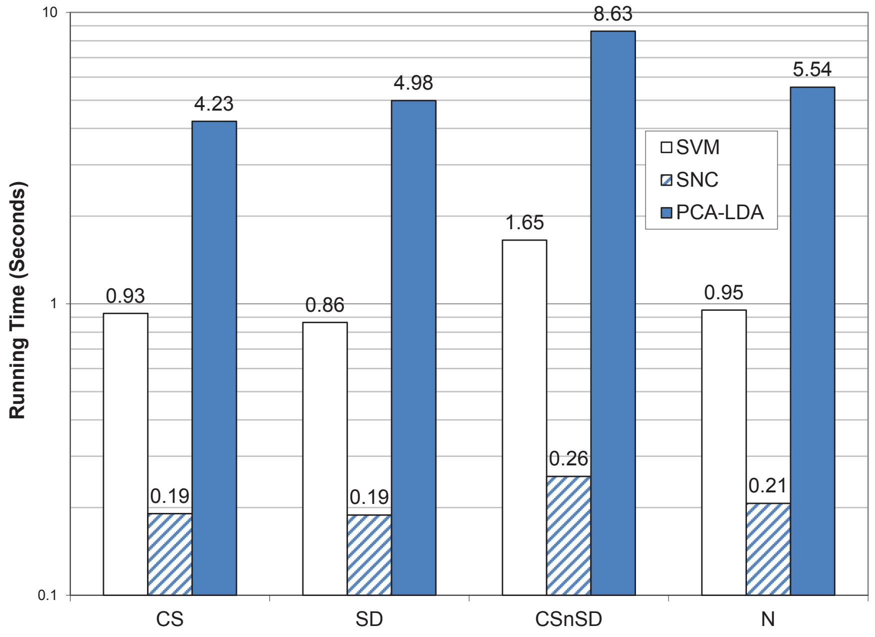


Figure 3.8: Computation times of SNC, SVM and PCA-LDA for different feature-vector constructions {CS,SD,CSnSD,N} for the 50%-50% training-testing ratio of binary classification problem. The plot is drawn in log scale and individual running times are marked above their respective bars.

3.5. Conclusions

We present here a new technique for classification and clustering, devised for the purpose of enhancing the identification of nuclear threats with low-resolution detectors. The technique builds on a bi-partitioning procedure called normalized cut prime, NC′. The solution method proposed here incorporates training data and as such it is a supervised classification method, *supervised normalized cut*, SNC. We test SNC and compare it with support vector machine, SVM, specialized SVMs and linear discriminant analysis (LDA) on data of low-resolution nuclear spectra. The results demonstrate that SNC is comparable or superior to SVM methods in terms of accuracy and much superior in terms of efficiency and robustness for either the binary or the multi-classification problem of the nuclear data set.

Chapter 4

Drug Ranking with Fractional Adjusted Bi-partitional Score (FABS)

4.1. Background and Chapter Outline

Automated microscopy is increasingly used in drug discovery, especially predicting the toxicity of new drugs [86]. The so-called high-content screening (HCS) has greatly enhanced investigators' capability of discerning the response of cells treated by various drugs [21, 26, 61, 75, 78, 99, 32]. HCS accomplishes this by analyzing phenotypic features of cells from tens of thousands cell images produced by HCS. In addition, the decreasing cost of such a method means a wide-spread application [63]. HCS employs cell imaging assays, tagged with fluorescent dyes - each field of cells contains these tags for its different macromolecules. Automated microscopy is performed to produce a large amount of visual information.

There are three steps during this process [110, 75]: fluorescence-tagging, automated microscopy and identification and measurement of target phenotypic feature(s) for further analysis. The analysis step usually poses the most challenge. To extract meaning out of a gigantic image database, traditional tools usually need to be tailored to specific known phenotype's features, instead of unknown yet more informative differences. For example, it has been reported that applying an analysis method that only distinguishes phenotypic changes in cellular level misses on the detecting meaningful morphological modification on subcellular structures [98, 114].

In high-throughput drug screening assays, typically a quantity, such as normalized intensity of a reporter fluorescent protein [77], is assumed to be measurable. Differences between samples of two distinct cell populations (such as treated versus untreated) are estimated and tested for significance. Methods using statistics like Z'-factor [113] to evaluate reliability of the measurements have been developed. Comparison of the difference is usually done by performing a multivariate F-test to test whether two populations are distributed differently. But F-test may introduce high errors when the distributions are not normal, which is expected to be the case in many types of cell responses. Moreover, in image-based assays, the use of a measurable quantity is no longer applicable when this quantity is not straightforward to obtain directly and the measurement itself can never be perfect. For example, to measure the composition of morphological subtypes of mitochondria requires pattern recognition algorithms to accurately detect and quantify target events [85]. Though many advanced algorithms have been developed for years, these pattern recognition algorithms usually require nontrivial tuning and optimization for each study because they may generalize poorly, sometimes not even generalize within a well, due to noise and systematic bias introduced during the sample preparation and imaging process steps, inducing additional overhead when attempts are made to scale up the assay to high-throughput.

Another challenge is when a multiplex approach is required, where multiple independent quantities are measured for each single-cell. In these cases, response of each single-cell will be a multi-dimensional vector. How to measure difference between these vectors become an issue because simple Euclidean distance in the multi-dimensional space may not serve the need. One solution is to come up with an appropriate "metric" to convert multi-dimensional vectors into a scalar that reflects the difference. There is, however, no generally-applicable solution about how to come up with this metric. Usually, one or more dimensions in the vector come from an imperfectly measured quantity, such as one that requires advanced pattern recognition in order to automatically extract, as discussed in the previous paragraph. Another issue is that our observation is the result of sampling, which inevitably introduces sampling errors and is further complicated by possible heterogeneous responses by cells [3].

The focus of this research is to address the issues mentioned above for the application of HCS in drug ranking. Drug ranking refers to the ordering of a group of different drugs according to their effectiveness by certain criteria. One of the most used criteria is the relative toxicity among drugs [84]. Ideally, this provides the important scale to assess relative merit of each candidate drug. However, each cell responds to a certain drug differently, thus making the outcome of any ranking highly dependent on sampling and noise. A conspicuous example is the fragmentation of cells or organelles: the intact and the completely fragmented states are easy to recognize while the degree of partial fragmentation is difficult to gauge, thus often involving human experts and time-consuming manual processes. This is infeasible for high-throughput screening such as HCS [63, 85].

Our objective is to develop an efficient and accurate ranking measure (metric learning) that can be used to order candidate drugs according to their effectiveness. To this end, we developed a framework called *Fractional Adjusted Bi-partitional Score* (FABS). This general strategy, introduced here for the first time, takes advantages of graph-based formulations and solutions and avoids many shortfalls of traditionally used methods in practice. We use such a scheme because graph-based construction works well in several areas of data mining [105], machine learning [57] and image processing [47], while a recent publication [63] also confirms its usefulness in the HCS context.

In order to apply our FABS to the images, we use a feature extraction tool first presented in [85]. This tool takes cell images and output several vectors that represents important geometric and other features of the target images - these vectors are then used as inputs for getting FABS.

One feature of FABS is that it has, as part of the input and as training data, extreme cases labeled as positive and negative controls, which in our case are the intact and the completely fragmented states mentioned previously. The algorithm does not involve any training from in-between cases, which are hard to come by. This completely sidesteps the common problem of a laborious and time-consuming annotation step, performed by experts to assess the relative merit of drugs for a small sample of images used as a training group. Furthermore, our measure takes the advantage of high-volume nature of the dataset, using all available images for computation of FABS for each drug. This reduces the effect of noise and sampling bias. This framework can potentially be used for any task that requires to quantify subtle and implicit differences between populations of high-dimensional feature vectors. By formulating the problem as a biparition problem as in FABS, there is no need to solve an image-based drug ranking problem as a regression problem. Our preliminary formal analysis of FABS shows that the expected error and variance of the estimated scores by FABS will be within a manageable range given the classification error by the bipartition.

To empirically evaluate our framework, we use a model of (NC') and the respective algorithm. That algorithm runs efficiently and is furthermore combinatorial. This latter feature differentiates it from e.g. [63] in which a spectral techniques is used to achieve a bipartitioning. Combinatorial solutions are superior than spectral ones in several regards such as being more efficient and accurate [51, 49], as shown in our experimental results.

4.2. Methodology

This section presents a general framework for quantifying the difference in morphological composition between populations of cells. The proposed framework utilizes a procedure named FABS-$\mathcal{A}$ where $\mathcal{A}$ stands for a bipartition algorithm, and FABS stands for *Fractional Adjusted Bi-partitional Score* (*FABS*). We show that using certain graph theoretical formulations for the bipartition algorithm avoids many shortfalls of the methods used in practice. Its importance lies in teasing apart cell groups based on morphological composition and in detecting whether or not such differences exist.

As previously mentioned, we use a feature extraction tool, capable of processing cell images with different dimensionalities (from static 2D to animated 3D with multiple channels) to generate high-dimensional (in our experiments, 134-D) output vectors, called *feature vectors*. Each feature vector, corresponds to an image of a single cell and contains measurements for the image characteristics, such as the intensity of the image, the shape of a particular object in the image, etc. Each group of cell images (and their corresponding feature vectors) can be associated with a certain population (e.g. populations representing cells to whom a certain drug has been applied).

The method proposed in this section, FABS-$\mathcal{A}$, is capable of receiving - as input - the feature vectors from cells representing different populations and detecting and quantifying the differences between these populations. For example, given the features extracted from the mitochondrial images of two populations of cells, one derived from diseased tissues and the other from healthy tissues, FABS-$\mathcal{A}$ will tell us to what extent the fragmentation levels of their mitochondria are different and estimate the significance of the difference.

We then perform FABS-$\mathcal{A}$ on the processed feature-vectors. The input to FABS-$\mathcal{A}$ is the processed feature-vectors by Principal Component Analysis (PCA) to reduce dimensionalities of the original data, each of which belongs to a certain population set, namely, P_i, and training data. The training data consists of feature vectors belonging to two populations on the opposite ends of the spectrum, R_1 and R_2. These two population sets represent positive and negative controls, which in this experiment are the completely fragmented and the completely intact mitochondria cell populations.

Computation of FABS-$\mathcal{A}$, the details of which will be discussed shortly, consists of three steps: The first step is to construct a graph from the input data. The second step is to apply a blackbox algorithm ($\mathcal{A}$) to find a *bipartition* on the resulting graph. The third step is to recover a scalar score for each population, based on the fraction of the cases that fall in the side of the partition boundary (cut) that contains positive controls. The blackbox can be any appropriate bipartitioning algorithm available. The algorithm we propose to use for the blackbox solves the *normalized cut prime problem* (NC$'$). We shall see in the Results section that this bipartitioning algorithm, in the context of FABS-$\mathcal{A}$ (FABS-NC$'$), outperforms Support Vector Machine (SVM) algorithm (FABS-SVM). This overall framework provides a flexible general strategy for quantifying the differences among population groups.

The major advantages of FABS-$\mathcal{A}$ include

1. It is capable of efficiently processing the high-dimensional input data acquired from the images using feature extraction tool from [85];

2. The generated output is one-dimensional, in that a single scalar score is generated for each population of multidimensional vectors. As such, the difference between the scores can be used to quantify population differences in an unambiguous way;

3. The calculation of the output scores is done in a way that reduces the effects of outliers in distinguishing cell populations;

4. Unlike many statistical tests, it does not assume any underlying distribution for the populations;

5. The labeled training data set required is minimal and easily obtainable, requiring minimum intervention from the experts;

6. It scales well in high-throughput applications.

In what follows we describe the three steps of FABS-$\mathcal{A}$ in more details.

The FABS-$\mathcal{A}$ Algorithm

Step 1: Graph Construction

As mentioned previously, the input to FABS-$\mathcal{A}$ consists of n (pre-processed) feature vectors, V=$\{v_1, ..., v_n\}$, each associated with an HCS image, obtained after feature extraction and PCA pre-processing. This input includes k population sets, $\{P_1, ..., P_k\}$. Each population set in this case represents a set of feature vectors corresponding to cells treated with a certain drug. Each feature vector v_i belongs to one of the population sets, indicating in this case what drug has been applied to the particular cell the vector is representing. The input also contains two training sets $\{R_1, R_2\}$, representing the extreme cases such as the completely fragmented and the completely intact mitochondria cell populations. In the graph construction step of FABS-$\mathcal{A}$, an undirected graph $G = (V, E, \mathbf{l}, \mathbf{w})$ is created, where each node $v_i \in V$ corresponds to a feature vector. The set of all possible pairs correspond to the set of edges of the graph $E = V \times V$ that form a complete graph. Each feature vector v_i is labeled with l_{v_i}, which is the index of the population set it belongs to. The labeling function, l_{v_i}, assigns a mapping from each feature vector, v_i, to its corresponding population set, determinewhich population it belongs to. A weight function $w : V \times V \rightarrow \Re^+$ associates with each pair of nodes $\{i, j\}$ (an edge) its encoding connection strength, or the similarity strength between the two nodes. For each edge $[i, j]$, the weight w_{ij} and the distance between the two points v_i and v_j have the relationship: one goes up as the other goes down (or vice versa) - this also means that w_{ij} and the similarity between v_i and v_j both go up or down together. Several distance measures can be used for this purpose, among them, Euclidean, city block, and Minkowski distances. Notice that constructing these similarity measures makes the dimensionality of the vectors irrelevant to our algorithm.

Step 2: Bipartitioning the graph using NC′

As previously mentioned, in the second step of FABS-$\mathcal{A}$, we use a blackbox algorithm to find a bipartition on the graph. A bipartition algorithm aims at finding the cut that separates the graph into S and $\bar{S}$, according to some underlying objectives. There are many different objectives that can be selected. For instance, the bipartition algorithm for the well known minimum cut problem is defined with the goal of separating the graph into S and $\bar{S}$ such that $C(S, \bar{S})$ is the minimum among all possible non-empty subsets S and $\bar{S}$. Since the goal is to obtain a bipartition for the FABS-$\mathcal{A}$ calculation process, any bipartition algorithm can be used as a blackbox. However, an extra requirement has to be imposed (either by the internal working of the algorithm or by an external constraint) listed as follows.

Requirement 1 *All positive controls R_1 must be in S (or $\bar{S}$) and all negative controls R_2 must be in $\bar{S}$ (or S).*

For a particular blackbox implementation of FABS-$\mathcal{A}$ in Step 2 of Algorithm 4, we choose the previously mentioned bipartitioning algorithm, called *normalized cut prime* (NC′), and adjust it to guarantee that the constraint listed in Requirement 1 is satisfied. The resulting FABS-NC$'$ is semi-supervised in nature and incorporates all information of the corresponding graph.

Recall from Chapter 2, the objective of normalized cut prime is $\min_{S \subset V} \frac{C(S,\bar{S})}{C(S,S)}$ on a given graph. In addition, seed nodes are used in the algorithm to solve. For a graph $G = (V, E)$, we denote $\text{NC}'(G) = \min_{S \subset V} \frac{C(S,\bar{S})}{C(S,S)}$. In the adaptation of the parametric s, t cut algorithm for the FABS-$\mathcal{A}$ framework, the positive and negative control data are used as *seed nodes* that are forced to join s and t in the graph. This is achieved through setting the nodes in R_1 to be "infinitely similar" to the source node s, and the nodes of R_2 to be "infinitely similar" to the sink node t. In terms of the graph that means that we add edges of infinite weight between the source node s and all nodes in R_1, and edges of infinite weight between the nodes of R_2 and t.

Since NC$'$ can be solved in the running time of a minimum s,t-cut problem [49], our FABS-NC$'$ implementation is efficient, solving in polynomial time. We later compare the performance of FABS-NC$'$, with FABS-SVM, where the bipartitioning algorithm used is Support Vector Machine (SVM), whose objective is to find a high dimensional hyperplane that is as wide as possible to separate data of different labels [25].

Step 3: Computing FABS scores

After a bipartition algorithm has been applied on G, all feature vectors in the graph are partitioned into S and $\bar{S}$. In the third step of FABS-$\mathcal{A}$, a scalar score, $\texttt{FABS}_{P_i}$, is calculated for each population set P_i. $\texttt{FABS}_{P_i}$ is the fraction of the number of feature vectors in P_i that fall in the set S, to the total number of feature vectors in P_i. Formally,

$$\texttt{FABS}_{P_i} = \frac{|S \cap P_i|}{|P_i|}.$$

This is shown pictorially in Figure 4.1. The FABS scores of the populations are then used to rank them: The higher the FABS score the closer is the population to R_1. The FABS scores are therefore ordered so that $\texttt{FABS}_{P_{\pi(1)}} \geq \texttt{FABS}_{P_{\pi(2)}} \geq \ldots \geq \texttt{FABS}_{P_{\pi(k)}}$, where $(\pi(1), \ldots \pi(k))$ is a permutation of $(1, 2, \ldots, k)$. The ranking of the populations is then given by $(\pi(1), \ldots \pi(k))$.

The entire procedure is summarized in Algorithm 4.

Significance Test

One can further use the FABS scores to test statistical significance of the difference between the effects of two drugs. The idea is to apply bootstrapping to obtain FABS scores from a large number of resampling trials and then perform hypothesis test on the difference of the distributions of FABS. Algorithm 5 gives the test procedure, which takes resulting FABS's from repeated experiment and calculate p-values from a t-test for each drug. The obtained p-value is then transformed into a log score $-\log p$.

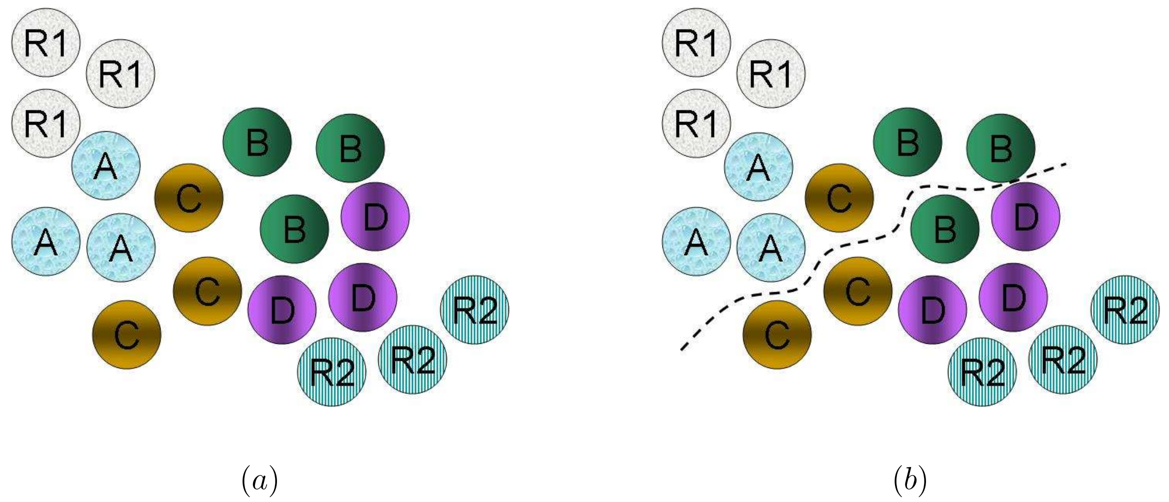


Figure 4.1: (a) The input with the feature vectors of images associated with positive and negative controls $\mathtt{R}_1$ and $\mathtt{R}_2$ and four different drugs `drug A`, `drug B`, `drug C` and `drug D`; (b) The bipartition boundary after the cut is found: if $\mathtt{R}_2$ contains negative controls, such as the completely fragmented state of mitochondria for toxicity criterion, while $\mathtt{R}_1$ contains positive controls, representing cells in a desired normal healthy state with mitochondria rescued from the completely fragmented, then $\mathtt{FABS}_{\mathtt{drug\ A}} = 1$, $\mathtt{FABS}_{\mathtt{drug\ B}} = \frac{2}{3}$, $\mathtt{FABS}_{\mathtt{drug\ C}} = \frac{1}{3}$, and $\mathtt{FABS}_{\mathtt{drug\ D}} = 0$. Our ranking of the drugs will be: `drug A` $>>$ `drug B` $>>$ `drug C` $>>$ `drug D`, where $x >> y$ indicates that x is more effective than y.

Algorithm 4 Fractional Adjusted Bi-partitional Score (FABS-$\mathcal{A}$) Algorithm

Inputs: The feature vectors $\{v_1, ..., v_n\}$ extracted from images (possibly after PCA pre-processing), and their corresponding population sets $\{P_1, ..., P_k\}$; The training data (or extreme sets) $\{R_1, R_2\}$

Step 1: Construct $G = \{V, E, \mathbf{l}, \mathbf{w}\}$, a complete graph from feature vectors;

Step 2: Use a bipartitioning algorithm $\mathcal{A}$ to find a bipartition $(S, \bar{S})$ on G such $R_1 \subseteq S$ and $R_2 \subseteq \bar{S}$;

Step 3: $\forall P_i$, calculate $\mathtt{FABS}_{P_i} = \frac{|S \cap P_i|}{|P_i|}$

Step 4: The FABS scores are ordered so that $\mathtt{FABS}_{P_{\pi(1)}} \geq \mathtt{FABS}_{P_{\pi(2)}} \geq \ldots \geq \mathtt{FABS}_{P_{\pi(k)}}$, where $(\pi(1), \ldots \pi(k))$ is a permutation of $(1, 2, \ldots, k)$. The ranking of the populations is then given by $(\pi(1), \ldots \pi(k))$;

Output: An ordered array of population sets based on their FABS score, $\{R_1, P_{\pi_1}, ..., P_{\pi_k}, R_2\}$.

To see if t-test is appropriate, there are several important assumptions to check. First the sets of FABS of two drugs must each be normally distributed. We plotted a histogram of FABS scores obtained by our FABS-SVM implementation and observed that the distributions for each drug in our test data are roughly bell-shaped. In addition, for Z-IETD and Z-LEHD, the p-values obtained through Jarque-Bera test [56] are 0.5 and 0.0718 respectively, indicating approximate normality for both. Another assumption is that variance for each group must be equal. Though this is usually not the case in drug profiling applications, t-test is robust against unequal variances if the sample sizes are approximately equal for each group, which can be enforced in drug profiling applications. Other assumptions, such as that sample means and sample variances must be statistically independent, can be compensated when the sample is moderately large or larger, which is always the case for HCS. Consequently, the t-test is appropriate for our purposes. When the number of population is high, we can apply Bonferroni correction to avoid errors due to multiple comparisons.

Algorithm 5 Significance Test

```
Step 1: Collect FABS from all subsampling trials for each drug, i.e. randomly sample certain percentages of controls and drugs with replacement from the original database repeatedly and calculate FABS score per drug each time;
Step 2: Perform t-test on FABS obtained with any two different drugs. T-test of drug A and drug B returns a p-value, $p_{(\texttt{drug A},\texttt{drug B})}$;
Step 3: Return $-\log p_{(\texttt{drug A},\texttt{drug B})}$
```

Data Preparation

We use a subset of a large image database of Chinese Hamster Ovary (CHO) cells published in [85]. The cells are divided into four groups according to the drug treatments they have received – control, squamocin, squamocin and z-IETD (shortened as z-IETD), and squamocin and z-LEHD (shortened as z-LEHD). Squamocin is known to induce mitochondrial fragmentation and cell apoptosis (i.e., programmed cell death). z-IETD and z-LEHD are inhibitors of caspases that play important roles in mitochondrial fragmentation. The goal of the study was to investigate whether z-IETD and z-LEHD can recover mitochondria from squamosin-induced fragmentation. Figure 4.2 shows some example cell images of mitochondria at different fragmentation stages. Intact mitochondria usually appear like threads, as shown in the images at the top row, while fragmented mitochondria appear like small globules as shown at the bottom row. Even though the totally intact and totally fragmented mitochondria (extreme set cases) can be easily distinguished by visual inspection, it is very hard (if not impossible) to look at a set of mitochondria images that are neither totally intact nor totally fragmented (e.g. a set of mitochondria images representing a population set of say cells treated by a certain drug) and distinguish between these different population sets and determining which extreme sets they are closest to and how they compare against each

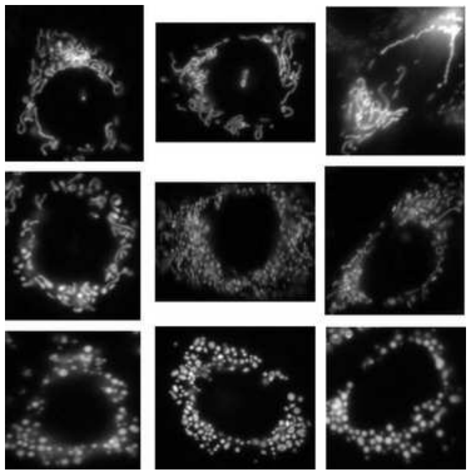


Figure 4.2: Example cell images show different fragmentation stages of mitochondria, tagged with a fluorescent dye. Images at the bottom row are cells with the completely fragmented mitochondria, at the top row are those without fragmentation, those in the middle are partially fragmented. From [63].

other (in terms of level of fragmentation). Another challenge is to automate this process. The automation process is critical, because the biological data sets available are very large and screening them manually could be a very time consuming and laborious task.

The challenge is to quantify and rank partial fragmentation as shown in the middle row. [85] concluded that z-LEHD was more effective than z-IETD in rescuing mitochondria from squamocin-induced fragmentation. This conclusion was used as the ground truth to assess the prediction accuracy of different methods later and images treated by squamocin and control were used as extreme cases.

Our database contains 257 images of cells treated with squamocin, 239 with z-IETD, 262 with z-LEHD and 238 control. We applied a feature extraction method to extract 135 features from each cell image to form the feature vector to represent each cell. This feature extraction method is the same as the one that was used to extract *strong detectors* from cell images to determine protein subcellular localization as described by [62]. Strong detectors include general-purpose features derived from image transformations such as Haralick texture features and geometric features of the objects extracted from the input image. These features have been shown to be useful in problems like recognizing fluorescent patterns of subcelluar organelles in protein subcellular localization [54].

4.3. Results

Formal Analysis of FABS-$\mathcal{A}$

Here we formally define the drug ranking problem and report a bias-variance analysis of FABS-$\mathcal{A}$ as a solution to this problem. The drug ranking problem can be considered as a regression problem, where given a multi-dimensional observation $v_i = X \in \Re^d$, we assume that a quantity $Y \in [-1, +1]$ is associated with X as our target metric of X. A solution of this regression problem is to learn a regression model from examples that compute Y given X. With the metric quantity Y, given two treatments a and b with population distributions $\mathcal{P}_a$ and $\mathcal{P}_b$, respectively, if

$$\mathbb{E}_{\mathcal{P}_a}(Y|X) - \mathbb{E}_{\mathcal{P}_b}(Y|X) \geqslant 0, \tag{4.1}$$

then treatment a will be considered to be more effective than treatment b, assuming that $Y = +1$ is the desired phenotypic outcome.

However, it is usually infeasible to manually assign score Y for a sufficient number of training examples consistently. Instead, FABS-$\mathcal{A}$ simplifies the problem as a bipartition problem. In our bipartition scheme, our model will assign $Y_c = 1$ to a given X if $Y \geqslant 0$ and $Y_c = -1$ otherwise, and then use empirical population mean as the estimated population mean of Y. In a drug screening application, this quantity will be used to rank the effectiveness of a treatment.

More formally,

$$Y_c = Y + \mathit{compl}(Y)$$

where

$$\mathit{compl}(Y) = \begin{cases} 1 - Y & \text{if } Y \geqslant 0 \\ -1 - Y & \text{if } Y < 0 \end{cases}$$

Instead of directly comparing the expectation of Y, FABS-$\mathcal{A}$ compares the expectation of Y_c to determine which treatment is more effective.

$$\mathbb{E}_{\mathcal{P}_a}(Y_c|X) - \mathbb{E}_{\mathcal{P}_b}(Y_c|X) \geqslant 0, \tag{4.2}$$

Like Y, Y_c is unknown and must be estimated with a model learned from data. Let $\widehat{Y_c}$ be the estimation of Y_c. Then

$$\widehat{Y_c} = \begin{cases} Y + \mathit{compl}(Y) & \text{if correctly classified} \\ Y + 1 & \text{if incorrect and } Y < 0 \\ Y - 1 & \text{if incorrect and } Y \geqslant 0 \end{cases}$$

An analysis of bias and variance of the bipartition scheme is as follows. The absolute error made by bipartition instead of regression is

$$|Y - Y_c| = |\Delta\widehat{Y_c}| = \begin{cases} |\mathit{compl}(Y)| = 1 - |Y| & \text{if correct} \\ 1 + |Y| & \text{otherwise} \end{cases}$$

Let ε be the classification error rate of the bipartition model.

$$\begin{aligned}\mathbb{E}(|\Delta\widehat{Y_c}|) &= (1-\varepsilon)(1-\mathbb{E}(|Y|) + \varepsilon(1+\mathbb{E}(|Y|)) \\ &= 1 + (2\varepsilon - 1)\mathbb{E}(|Y|) \leqslant 1 \left({}^{\text{when } \varepsilon = 0.5}_{1+(1-1)|Y|=1} \right)\end{aligned}$$

The expectation of the absolute error is bounded below one when we use a weak classifier for the bipartition that simply guesses a label randomly.

The variance of the absolute error is

$$\begin{aligned}Var(|\Delta\widehat{Y_c}|) &= \mathbb{E}(|\Delta\widehat{Y_c}|^2) - (\mathbb{E}(|\Delta\widehat{Y_c}|))^2 \\ &= 4\mathbb{E}(|Y|)^2\varepsilon(1-\varepsilon),\end{aligned}$$

which turns out to be the variance of Bernoulli trial scaled with the square of the expected scale of Y. Again, this is bounded by 1 when $\varepsilon = 0.5$ and $\mathbb{E}(|Y|) = 1$.

Next, we consider the expectation of $\widehat{Y_c}$, which is interesting because we can infer the expected difference between regression (Eq. 4.1) and bipartition (Eq. 4.2).

$$\Delta\widehat{Y_c} = Y - \widehat{Y_c} = \begin{cases} 1 - Y & \text{if } Y \geqslant 0 \text{ and correctly classified} \\ -1 - Y & \text{if } Y < 0 \text{ and correctly classified} \\ -1 - Y & \text{if } Y \geqslant 0 \text{ and incorrect} \\ 1 - Y & \text{if } Y < 0 \text{ and incorrect} \end{cases}$$

Let $P_+ = \Pr(Y \geqslant 0|X)$, the probability that $Y \geqslant 0$, and $\overline{Y} = \mathbb{E}(Y|X)$. We have

$$\begin{aligned}\mathbb{E}(\Delta\widehat{Y_c}) =&(1-\varepsilon)P_+(1-\overline{Y}) + (1-\varepsilon)(1-P_+)(-1-\overline{Y}) + \\ &\varepsilon P_+(-1-\overline{Y}) + \varepsilon(1-P_+)(1-\overline{Y}) \\ =&(2-4\varepsilon)P_+ - 1 + 2\varepsilon - \overline{Y}.\end{aligned}$$

The result above implies that when we have a weak classifier $\varepsilon \to 0.5$, $\mathbb{E}(\Delta\widehat{Y_c}) \to -\overline{Y}$ and $\mathbb{E}(\widehat{Y_c}) = \overline{Y} + \mathbb{E}(\Delta\widehat{Y_c}) = 0$. That is, regardless of the population, random guessing will not give any distinction between any populations and provide no discerning power. In contrast, when we have a perfect classifier with $\varepsilon \to 0$, $\mathbb{E}(\widehat{Y_c}) \to 2P_+ - 1$, which is to scale the true probability of $Y \geqslant 0$ for the population to $[-1, 1]$, perfectly matching our desire. Consequently, given an accurate bipartition algorithm, FABS-$\mathcal{A}$ can reasonably approximate effectiveness of drugs without exact scores the effectiveness.

Performance of Ranking

We compared the performance of FABS-NC$'$ with four other baselines that has been used in HCS – center ranking, PCA ranking, and graph transition energy method (GTEM) [63]. Center ranking first finds the center, which can be the mean, the median or any other measure of the center, of all feature vectors associated with a particular drug or an extreme case, then calculate the distance, such as euclidean distance, between all pairs of centers. The ranking of the drugs are performed by ordering the drugs according to the centers of the closest to the farthest from the center of the desired extreme case (such as the completely fragmented state for toxicity criterion). PCA ranking is similar to center ranking, except it first projects the feature vectors onto the first few important principal components, then performs center ranking. The Graph transition energy method (GTEM) [63] is also a graph-based approach. GTEM defines graph transition energy as the distance metric and utilizes a spectral graph theoretic regularization to transform the feature space so that extreme cases will be separated widely before ranks populations of cells under different treatments.

In addition to use NC′ (solved with Hochbaum's PseudoFlow algorithm, HPF, the implementation of which is obtained from [48, 51, 19]) as our bipartition algorithm in the FABS framework, we also tested other bipartition procedures. One classical technique is the support vector machines (SVM) [14, 25]. When using SVM for FABS, we satisfy Requirement 1 by setting training data as the positive and negative controls: all $\mathtt{R}_1$ points are in S and all $\mathtt{R}_2$ points are in $\bar{S}$. To see the performance of this particular implementation of (FABS-SVM), the kernel used is radial basis function (RBF) and the parameters are the following: C value is 10^4 and the kernel parameter is 1. The implementation package used is LIBSVM [20].

Another approach, often used in image segmentation is based on finding the Fielder eigenvector of the graph (referred to as the *spectral technique*) as a heuristic solution for the normalized cut problem [95]. The spectral technique however is unsupervised, and thus does not satisfy Requirement 1. To resolve this issue, we modified the weights of the graph to ensure that Requirement 1 is satisfied. The implementation package used is Normalized Cuts Segmentation Code [48]. However, its performance was much worse than all other methods and was removed from the results.

The comparative study that we performed used the median for all center measures and Euclidean distance for all distance measures. The edge weights between two feature vectors v_i and v_j increase or decrease in the opposite direction with respect to the distance between them and is quantified by $w_{ij} = e^{-||v_i - v_j||_2 + \epsilon}$, for $0 < \epsilon \ll 1$.

Prior to feeding the input feature vectors extracted from the images into FABS-$\mathcal{A}$, we first pre-process these vectors to transform them from a high-dimension space to a space of fewer dimensions. In this process, the data is reduced to fewer dimensions, and we only preserve the dimensions that are of the most significance to our experiment. The dimension reduction is performed by using Principal Component Analysis and the number of principal components used is 80% of the total variation in the data set considered. We also tested whether applying GTEM's feature transformation step as a preprocessing step before applying FABS-NC′ may improve the performance.

To guarantee statistical validity of our comparison, we subsampled the available cell images from the entire database, i.e. we drew samples with replacement for certain percentage from the database to test methods. 30% , 60% 70% and 80% were the subsampling percentages tried for drug images (501 images). For each fixed drug percentage, we changed percentages of labeled controls by increasing from 10% to 100% to see the effects of the

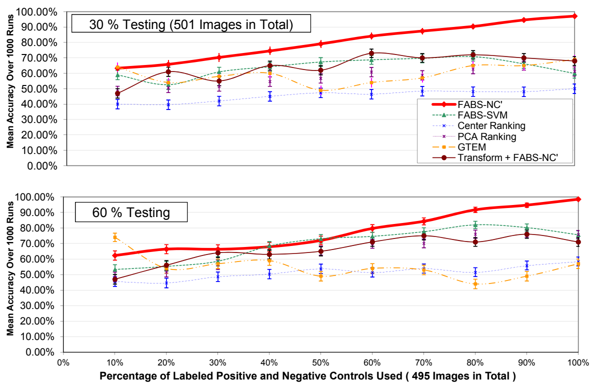


Figure 4.3: The accuracy comparison among different ranking methods. The vertical bars in the graph are 95% confidence intervals. The testing percentages used are: 30 % and 60 %.

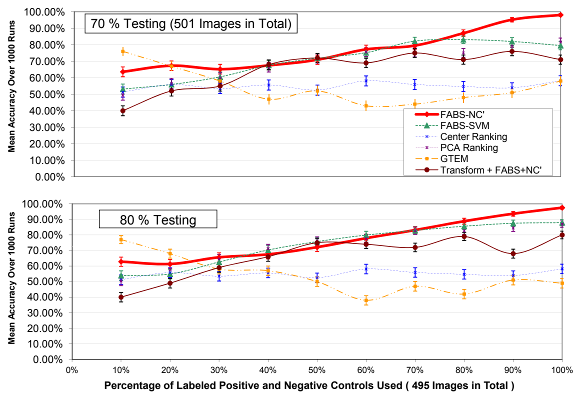


Figure 4.4: (Continued) the testing percentages used are: 70% and 80 %.

number of labeled controls on the final prediction accuracy of the ranking (495 images in total). The subsampling trials are performed 1000 times for each combination. The *prediction accuracy* of any ranking method is the fraction of correctly ranked trials — this can be determined, since we have the ground truth — out of the grand total of 1000 trials.

Figures 4.3 and 4.4 graphically summarize the results in the experiment. Each graph shown is for 30%, 60%, 70% or 80% fixed drug percentage (testing percentage). The x-axis is the percentage of labeled controls used, while y-axis displays the average prediction accuracy over 1000 trials described in Section 4.3.

Each curve in the graphs indicates a particular ranking method - they include FABS-NC′, FABS-SVM, Center ranking, PCA ranking, graph transition energy method (GTEM). The results of FABS-Spectral is poor with our particular implementation and from the figures. The vertical lines in Figures 4.3 - 4.4 are 95 % confidence intervals for the accuracy of each ranking method.

For all testing percentages, the prediction accuracy of FABS-NC′ steadily increases as more labeled controls become available, especially when more images are tested (70% and 80%) - the slope increases then levels off from the left to the right. The overall accuracy is nearly 98% for all graphs at the end of the x-axis, indicating that the method is highly accurate with as little as 500 labeled controls. It is also robust considering that the trend of prediction curve remains the same for different testing percentages.

Moreover, we can see that FABS-NC′ has an advantage over other ranking methods for this particular mitochondria dataset. Its curve is often above all other methods, except for 10% labeled controls; testing percentage 70%: 70% labeled controls; and testing percentage 80%: 10% and interval 40% to 50%. Notice that for the low number of testing (30%), FABS-NC′ outperforms all other methods - when using all labeled controls for ranking, it is over half more accurate than the next best algorithm.

Overall, FABS-SVM also performs well, although sometimes trailing behind FABS-NC′ by a large margin. PCA ranking performs poorly when testing images are few (30%). Center-ranking is generally of low quality, giving small accuracy for all testing percentages. Notice, however, GTEM gives the best results when the number of labeled controls is very low (10%), indicating its usefulness when training data are few - neverthless, its advantage dimishes as more labeled training cases becomes available, producing inaccurate rankings comparing to FABS. The results show that applying the feature transformation step of GTEM as a preprocessing step of FABS-NC′ performs better than GTEM but not as well and as stable as FABS-NC′.

The experimental results suggest that, overall, FABS with NC′ implementation is the best ranking method among all for this particular mitochondria database. Remarkably, FABS-NC′ generalizes better than any other methods as more training and test examples become available.

Significance

Table 4.1 displays the significance score $-\log p$ between different pairs of drugs for FABS-NC′ and FABS-SVM implementations when we sub-sampled 30% of labeled controls and 30% of drug treatment results. An infinity score (∞) is obtained when p is very close to zero, indicating that the distance between the two corresponding drugs is very large. The results show that FABS-NC′ is more discriminant then FAB-SVM because the significance scores for FABS-NC′ are larger than for FABS-SVM.

We also performed a Monte Carlo simulation to test whether the observed difference of the FABS-NC′ scores of 30% of Z-IETD and Z-LEHD data using 80% of control data for training is significant against pairs of null data sets sampled from the same drug treatment populations. In 1000 random resamplings, no difference of the scores of the null data set pairs is higher than the observed score, yielding a close to zero p-value.

FABS-SVM	squamocin	Z-IETD	Z-LEHD
squamocin	0	∞	∞
Z-IETD	∞	0	3.43
Z-LEHD	∞	3.43	0

FABS-NC$'$	squamocin	Z-IETD	Z-LEHD
squamocin	0	∞	∞
Z-IETD	∞	0	4.36
Z-LEHD	∞	4.36	0

Table 4.1: Matrices of GDM between different pairs of drugs for different implementations of FABS - SVM and FABS-NC$'$.

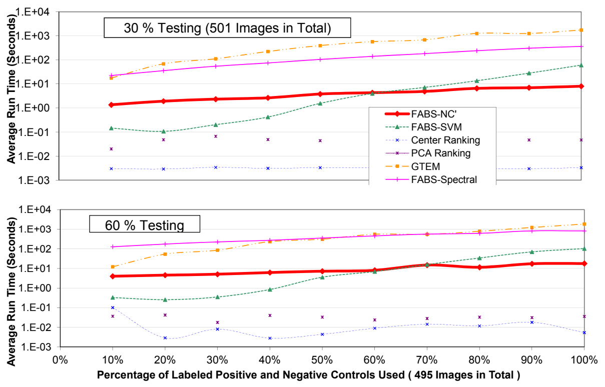


Figure 4.5: The running time comparison among different methods. The testing percentages used are: 30 % and 60 %.

Comparison of Running Time

In this section, we compare the running times of three FABS-$\mathcal{A}$ procedures, where $\mathcal{A}$ here, as mentioned in previous sections, is one of bipartition algorithms including NC$'$ [49], SVM [25] and Spectral [95], among themselves and against PCA Ranking, Center Ranking, and GTEM. The specification of the computer environment for this comparison is a Windows computer with 2.4GHz Intel(R) Core(TM)2 Duo CPU 2.40GHz and 2GB memory.

Figures 4.5 and 4.6 display running times of various methods, excluding the times for subsampling - which have a median of 0.01 second, maximum of 0.02 second and minimum of 0.006 second - for different testing percentages: x-axis increases with the number of positive controls and negative controls used, representing more and more training data becoming available, while y-axis is the running time. The six curves in the figures are the different methods including various implementations of FABS-$\mathcal{A}$ – notice that FABS-NC$'$ is represented by the thickest curve. There are 501 testing data: 265 Z-IETD and 291 Z-LEHD.

From the figures, among FABS-$\mathcal{A}$, we can observe that for all testing percentages con-

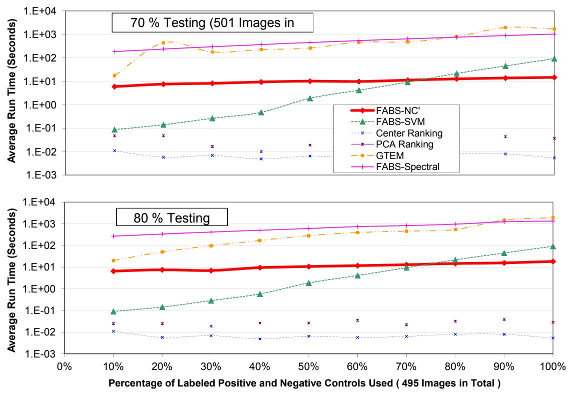


Figure 4.6: (Continued) the testing percentages used are: 70% and 80 %.

sidered, FABS-Spectral takes the most running time, lagging behind both FABS-NC′ and FABS-SVM by large margins. For FABS-NC′, the running time steadily lengthens as the number of positive and negative controls increases, however, not as dramatic as FABS-SVM, whose running time, shorter than these of other procedures initially, grows exponentially – in one case (testing percentage 70%), running 100% of positive and negative controls requires around 1000 times more seconds than running 10% of positive and negative controls. This is to compare with FABS-NC′: for the same testing percentage, using all positve and negative controls only requires twice as much running time than that of using only 10% – 10% corresponds to around 50 controls in total, a relative small number of images that can be obtained through HCS. This observation, combined with the results from Section 4.3, indicates that even though FABS-SVM has the initial advantage for running time, this is off-set by the initial more accurate results produced by FABS-NC′. Moreover, it appears that FABS-NC′ scales much better with increasing input data than FABS-SVM. Looking at the other methods besides FABS-$\mathcal{A}$, we can observe that GTEM takes relatively long time on the par with FABS-Spectral - this is in contrast with PCA Ranking and Center Ranking whose running times are the lowest among all methods: this result is expected, since FABS-$\mathcal{A}$ use PCA for pre-processing (i.e. doing PCA is already added as a part of computational costs), therefore FABS-$\mathcal{A}$ can only take longer time than PCA ranking. However, from Section 4.3, it is clear that this extra computational costs bring significant improvements in accuracy, which combined with scalability of FABS-NC′, makes FABS-NC′, overall, an attractive candidate for ranking this database.

4.4. Discussions

In this chapter, we describe a new drug ranking framework called fractional adjusted bi-partitional score (FABS). It is graph-based, producing a single scalar score for each drug for ranking. The formulation and solution sidesteps many pitfalls of other traditional methods.

The chapter also reports FABS-NC$'$ semi-supervised implementation and its comparative study. Not only is this implementation better than four other considered methods, it also outperforms FABS-SVM and FABS-Spectral implementations on a mitochondria databases. This preliminary result suggests that FABS-NC$'$ is good for ranking toxicity of drugs targeting mitochondria for a specific database.

There are some advantages of our measure. First, FABS is one dimensional, that is, a single scalar, giving an unambiguous way to rank drugs. Its computation considers all samples of each drug and uses a fraction as the final score. This diminishes the effect of outliers and noise, because, if the number of images is large for each drug, as in the case of HCS, outliers, which are few in number, can not influence the result - a fraction, in a significant way. This similarly is the reason for noise reduction. More importantly, our measure FABS-NC$'$ is acquired through a combinatorial algorithm, which is efficient. This is essential since the number of cells observed in a HCS is large and the applicability of any metric learning algorithm is greatly reduced if it cannot process them sufficiently fast. The last noteworthy advantage of our framework is that the training data for the semi-supervised formulation are the positive and negative controls, which are easily recognizable and obtained without time-consuming annotation, sidestepping the limitation of training sample size of many metric learning algorithms.

Chapter 5

Target Tracking in Video Sequences

5.1. Background and Chapter Outline

Target tracking is a fundamental problem in computer vision. The goal is to isolate a target object from its background across a sequence of frames. The tracking problem is three dimensional in that it incorporates the time dimension. As such, the computational efficiency of any suggested solution is a major challenge. The method presented here is efficient enough to process videos under near real-time constraints.

Tracking algorithms in the literature are categorized into three main classes: (i) active-contour approach, (ii) statistical and stochastic methods and (iii) graph-theory based tracking. The first class includes variational motion segmentation with level sets, [11], and fast geodesic active contour method, [82]. At heart of this variational computation approach is the use of continuous models coupled with consistency constraints to delineate boundary of a target object. However, digital videos are innately discrete. The conversion of these real-numbers solutions to discrete ones is not straight forward and often requires heuristics and further processing.

The second class of tracking techniques incorporates statistical and stochastic elements. To this end, a stochastic gradient decent method on a cost function that consists of both intensity and motion information was suggested for tracking [8]. Nillius et. al [80] presented target tracking by Bayesian network formulation. Tracking through Markov-Chain Monte-Carlo Data Association (MC-MCDA) was devised by Benfold and Reid [4]. Their scheme operates on a set of Histogram of Oriented Gradients (HOG) descriptors rather than color or motion information. Breitenstein et. al utilized particle filtering framework for addressing the unreliability of the input data [9, 10]. A two-phase tracking algorithm for multi-target tracking in crowded scenes was reported by Yu et. al., [111]. In this method, the first phase extracts a set of continuous trajectories (tracklets). The second phase employs a Bayesian formulation to find the most probable set of tracklets for a single object. While an extensive work has been put into statistical and stochastic methods, all of the aforementioned schemes rely heavily on iterative steps that are computationally intense and do not guarantee optimal solution nor consistency over sequential runs on the same input data.

The third approach, on which we focus here, formulates the problem as a graph problem. In our case, we use a variant of the normalized cut that is solved with a minimum cut on an associated graph [49]. This formulation is solved optimally (i.e., no local minima are picked as a solution) and with low computational cost [49, 19, 35]. The use of graph cuts for graph-cuts for object tracking was first introduced by Xu et al. [108], where the tracked object's contour in frame t was sought in a narrow region in frame $t+1$. This method did not utilize motion information and therefore faced difficulties when dealing with large displacements and occlusions. A few graph-cuts based tracking algorithms that utilize motion data were reported [38, 67, 13, 83]. Freedman and Turek, [38] suggested a two-phase tracking mechanism. At the first stage the motion in the sequence is extracted by graph-cuts based optical flow. Then all the motion vectors are grouped into spatio-temporal blobs, each representing a moving object (i.e., tracking). The grouping process is done through propagation, hence if pixel $I_{x,y,t}$ was found to move to location $(x\prime, y\prime)$ at frame $t+1$, than both pixels $(I_{x,t,y}, I_{x\prime,y\prime,t+1})$ are grouped into one object. This scheme, however, does not guarantee that the tracked objects do not break into several small components within a few frames. Malcolm et. al [67] used an autoregressive model to provide a prediction of the target's location in the succeeding frames. This prediction is then used to construct the spatial constraints on the object's expected location in these frames. This algorithm, however, does not handle occlusions well, since it does not introduce a specific process for dealing with interacting objects (see [83] for such example). In [13] Bugeau and Pérez utilized Lucas and Kanade's optical flow algorithm [65] in a two-phase tracking mechanism. At the first stage all objects are tracked individually. Then at the second stage objects that might have been merged in the first phase are segmented. An extension of the latter work that addresses occluded objects was introduced by Papadakis and Bugeau [83].

All the video tracking schemes mentioned above, that utilize motion data, use *optical-flow* methods for motion estimation. While considered to be most accurate, these optical-flow methods require the minimization of an energy functional. In order to solve the resulting large sparse systems numerically, classical iterative methods are commonly used (e.g. [12]). While these are simple to implement, their convergence is slow, and often thousands of iterations are necessary to get sufficiently close to the global minimum of the energy functional. This is the reason why optical flow methods are slow and unsuitable for time-critical applications.

In this chapter we suggest a generic robust graph-theory-based tracking scheme in videos. The suggested method casts the tracking problem as a variant of the normalized cut (NC$'$) problem [49]. This approach is unique in that it solves optimally an optimization problem (i.e., neither heuristics nor approximations are applied). Previous tracking methods [67, 13, 83], even if they cast the tracking task as an optimization problem, deliver a heuristic solution to the problem posed.

The suggested scheme is so robust that it allows for incorporating the computationally cheaper Moving Picture Experts Group (Rev. 4), MPEG-4, block-matching, motion estimation schemes. Although block matching techniques generate noisy and coarse motion fields, their use here has two advantages: (i) Faster computation times as broad variety of off-the-shelf software and hardware components that specialize in performing this task and can easily be incorporated into the segmentation scheme are available; and (ii) If the videos are already compressed, then the motion information is inherent in their compressed form, and it is available from the video encoder. In that case there is no need to apply *any* motion estimation algorithm. This approach of using the motion field coded within the compressed sequence was suggested for video enhancement (see for example [60, 36]). Graph-based object detection and tracking in H.264/AVC bitstreams was recently suggested by Sabrin et. al [92]. However the graph there is used only to build the association of the data. No graph-based algorithms, which could have enhanced the performance of the algorithm, were exploited for the task.

Another direct advantage of our NC$'$ approach is that motion is treated as one of the similarity measures. This is in contrast to the aforementioned methods, [38, 67, 13, 83] which presented motion as consistency constraints. Because of that, there is no need for the heuristics commonly used in dealing with difficulties associated with this type of constraints. Similar notions can be found in human action recognition algorithms, where the similarity between nodes is measured either by using descriptors [2] or by the motion field computed by optical flow [64].

Consequently, the contribution is two-fold: Firstly, it formulates the tracking problem as a graph problem; secondly, it demonstrates that the graph-theory-based tracking scheme developed here is robust enough, allowing using coarse and noisy block matching motion fields as a feature rather than as a constraint. Incorporating this information into tracking has the advantages mentioned above. The results here demonstrate that our scheme can support a fast and accurate video tracking, thus making it amenable to real-time applications.

The chapter is organized as follows: Section 5.2 formulates the tracking problem as NC$'$ problem and describes how it is solved through graph-cuts. Section 5.3 addressed practical aspects of the system; Section 5.4 presents a performance evaluation of the algorithm on real-life benchmark videos and compares it to the state of the art. Section 5.5 provides concluding remarks.

5.2. Problem Formulation

A graph representation of video tracking

Target tracking is presented as a bi-partitioning problem in a graph representing the video, where one set of the bi-partition represents the tracked object and through this tracking is achieved. Specifically, the problem is presented on an undirected graph $G = (V, E)$ with a set of nodes V, representing pixels in their spatiotemporal position, and a set of edges E, connecting each node to its adjacent pixels. For videos, one typically considers three dimensional graphs with pixels arranged along a grid. The 6-neighbors set up is a commonly used adjacency rule with each pixel having 6 neighbors – two along the vertical axis, two along the horizontal axis and two along the temporal axis. The 26-neighbors arrangement, which includes the adjacent pixels along the diagonal axes is also a common setup. We use here a 10-neighborhood model: Each pixel has a total of 10 neighbors: 4 in the current frame (up, down, left and right) and three additional neighbors in the pixel's corresponding locations in the 3 preceding and 3 subsequent frames. Notice the graph G is *not* a complete graph – different from that in Notation section of Chapter 1. The construction is illustrated in Figure 5.1. The similarity is computed for each pair of neighboring pixels. All edges between non-neighboring pixels are assigned zero weights.

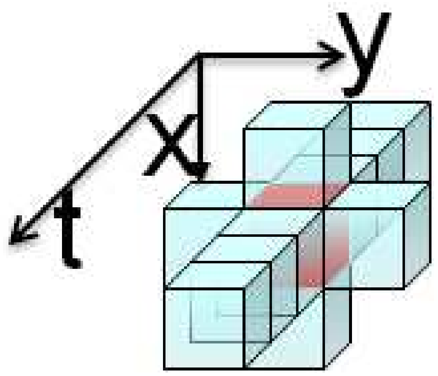


Figure 5.1: A pixel and its spatio-temporal neighborhood.

The edges in the graph carry similarity weights. This similarity may take into account multiple pixels' features such as the pixel's neighborhood texture, its intensity, corresponding motion and its color or brightness. In terms of the graph, each edge $[i, j]$ is assigned a similarity weight w_{ij} that increases as the two pixels i and j are perceived to be more similar. Low values of w_{ij} are interpreted as *dissimilarity.*

The tracking problem is cast as finding a bi-partition, $\big(S, \bar{S}\big)$ that optimize NC$'$ defined in Chapter 2. Recall that, NC$'$ is defined as

$$NC'(S^*) = \min_{S \subset V} \frac{C(S, \bar{S})}{C(S, S)}. \tag{5.1}$$

where S set corresponds to the pixels in foreground (or background) and $\bar{S}$ set corresponds to the pixels in background (or foreground).

The target of interest to be tracked is not always the salient nor the only feature in the frame. In order to specify the object of interest, one or more pixels are a priori assigned as foreground or background. These pixels correspond to the *seed nodes* in the graph. Seeds may be selected with either a manual or automatic procedure. It is possible to run the NC' segmentation with a single foreground and a single background nodes. In cases where the segmentation criterion, (5.1), results in an unsatisfactory results, such as a very large $|S|$, one can add few more nodes (by clicking on relevant pixels in the sequence), which often results in a significant improvement. This simulates the course of action of a human operator, where the target of interest is indicated in the first few frames by the operator's mouse clicks and then the algorithm delineates and tracks the object in all subsequent frames.

The corresponding graph is constructed according to Chapter 2. In order to solve the minimum parametric s, t-cut problem associated with the graph, we use the Hochbaum's PseudoFlow (*HPF*) algorithm [51]. The HPF algorithm has a strongly polynomial complexity and it was found to outperform any other solution approaches in general, [19], and for vision problems in particular, [35]. The output of the minimum s, t-cut is a bi-partition that divides the spatiotemporal pixels into two groups: one group is the delineated target object, and the other corresponds to background. HPF can be downloaded from [48].

5.3. Implementation Considerations

Similarity Measures

The system's input consists of two *vectors*: $\vec{I}_{klt}$ and $\vec{m}_{klt}$. $\vec{I}_{klt}$ is the color representation vector of the pixel coordinates (k, l) of frame t. The color representation can be in any form (e.g, R-G-B, Y-Cb-Cr, H-S-V or L-a-b). The vector $\vec{m}_{klt}$ is the motion component which typically contains two components: the horizontal and vertical motions. For the subsequent processing stages, the translation vector is presented in polar coordinates, hence magnitude, A_{klt}, and angle, φ_{klt}, of the motion vector.

We incorporate these two vectors for each pixel in a $5-D$ *feature vector* consisting of 5 parameters - 3 for color representation and two for motion. These features quantify the resemblance between pairs of pixels. To this end, several quantifiers can be used to measure the similarity: correlation, t statistical test and L^1 or L^2 norms. Here we use the L^2 norm as a measure of dissimilarity: the larger the norm the greater the difference between the two pixels. Consequently, the reciprocal of this quantity is a measure of similarity between two pixels i and j,

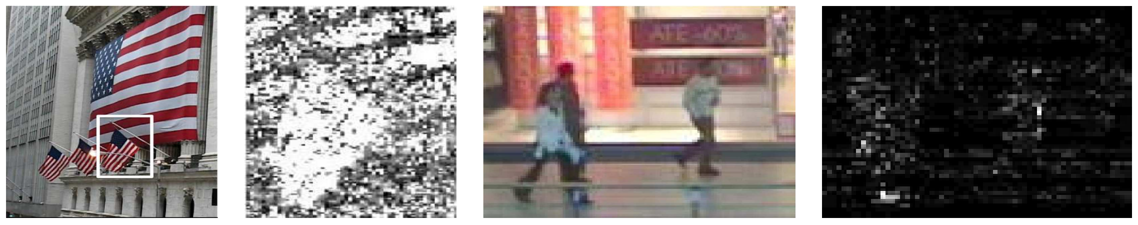
$$w_{ij} = 1/(\|\vec{F}_i - \vec{F}_j\|_2 + \epsilon). \tag{5.2}$$

Block-Matching-Based Motion Estimation Techniques

The concept behind block matching motion estimation is to divide the current frame into a matrix of macro-blocks. The translation vector of each of these blocks is estimated by searching the most similar block in the preceding frame. The matching is based on the output of a cost function. The location in the previous frame that results in the least cost is the one that matches best the current block. There are various cost functions, of which the most popular and least computationally expensive is the sum of absolute difference (SAD). Another common cost function is the mean squared error (MSE).

Several block-matching high efficiency algorithms were presented (e.g., [79, 101, 116]). By applying certain assumptions on the error function, such as smoothness and global minima, these methods reduce the computational complexity: The number of possible matching candidate blocks, examined within the entire previous frame or within a bounded search area, is reduced by using efficient location patterns for canadidate blocks, such as diamond or spiral; and by introducing maximum desirable error value, an early-stopping criterion is applied. These improvements are traded off with possible degradation in motion estimation accuracy and the presence of noise in the computed motion field. The degradation is substantiated by the tremendous reduction in running times. Specifically, we use here the x.264 [103] implementation of *diamond search* motion estimation algorithm [116], which is commonly used in MPEG-4 video compression standard.

Figure 5.2 illustrates the block matching motion field computed by diamond search for two sequences, one sequence taken from the CAVIAR [18] data set and a sequence of the New York Stock Exchange's facade [33]. Figure (a) shows a representing frame from the CAVIAR sequence. Figure (b) presents the corresponding motion field. Figure (d) presents a blowup of the motion field of the small segment marked on Figure (c). Both examples clearly show that the motion fields, generated by the block matching diamond search motion estimation technique, are coarse and noisy. In spite of these characteristics of the motion field, the tracking scheme presented is robust and manages to utilize the motion field for the tracking task. This results in a computationally efficient mechanism as both the computation of the motion field and the tracking realization are extremely efficient.



(a) (b) (c) (d)

Figure 5.2: New-York Stock Exchange facade sequence (a) and the motion amplitudes (b) of the flag fragment in (a), brighter pixels correspond to larger amplitudes. Figure (c) is a frame from a surveillance sequence extracted from the CAVIAR data set [18] and (d) presents its corresponding motion field

Segmentation over Long Image Sequences

The method described here takes in a fixed number of image frames, and produces a segmentation of this batch. In this way the algorithm can incorporate information across several frames to produce the best partition. When considering long image sequences the algorithm processes the sequence in a temporal moving window fashion, where N frames are processed at each window's location. The process is described in Figure 5.3. As described in Section 5.2, few nodes are a priori tagged as foreground or background (seed nodes). After the required seed nodes are indicated, a window of N frames is processed. Then the tracking results of the last frame in the first batch are used as seed nodes for the segmentation of the next $N-1$ frames. This process is repeated till all frames are processed. This mode of operation is prone to error propagation over time. This can be compensated by additional user inputs in any window's position. It is important to note that while additional user input throughout the process may improve the tracking results, the user's input is required only at the beginning of the process for identifying the target of interest. Our experiments show that a window size of $N = 10$ was a good tradeoff between computation time and accuracy. Following the discussion in Section 5.2, pixel's neighborhood is defined over 7 frames. If $N < 7$, then the pixel's neighborhood is truncated symmetrically around the central pixel.

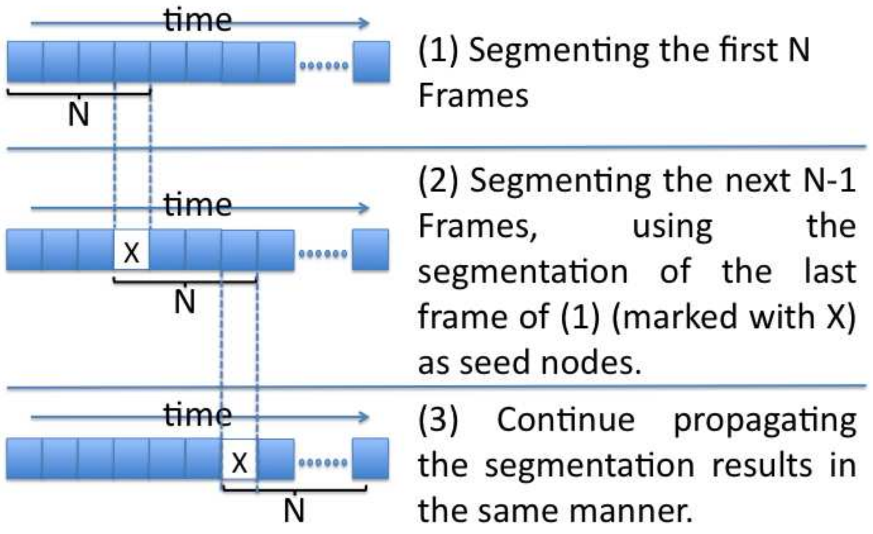

time
(1) Segmenting the first N Frames
N
time
(2) Segmenting the next N-1 Frames, using the segmentation of the last frame of (1) (marked with X) as seed nodes.
X
N
time
(3) Continue propagating the segmentation results in the same manner.
X
N

Figure 5.3: Segmenting Long Video Sequences by propagating the segmentation results over consecutive moving window positions

5.4. Experimental Results

Data Sets and Performance Measures

The suggested method was tested on a broad variety of standard and non-standard test scenarios. Standard test scenarios sequences were taken from: the Context Aware Vision using Image-based Active Recognition (CAVIAR) database [18]; the HDTV TRICTRAC test sequences [102]; and the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) benchmark scenarios [87].

For quantitative comparison of our method's performance to the tracking algorithms presented by Breitenstein et. al [9, 10], and Yu et. al [111] we employed the Multi Object

Tracking (MOT) measures [5] for Precision (MOTP) and accuracy (MOTA). As our algorithm does not tag (identify) the different objects, this quantity is omitted in the MOTP calculations. As a base for the quantitive comparison, the PETS2009 [87] S2.L1 and S2.L2 benchmark sets were arbitrarily selected.

Results

The tracking of the flag (marked with a square in Figure 5.2(a)) in the NYSE sequence, which is characterized by heavy global motion, is done with a reduced feature space: As color information, we use the pixels' intensity levels; while for motion we use only motion amplitudes (shown in Figure 5.2(b)). The segmentation results for three consecutive frames of the NYSE sequence are presented in Figure 5.4. Column (a) shows a sequence of the original frames; the second column, (b), shows the video segmentation produced by using only pixels' intensities; The segmentation errors in these frames, mainly noticeable in the lower-left part of the images, are associated with the similarity of the color schemes of the foreground and background flags. The segmentation results using solely motion data are given in column (c). Here the error is attributed to similarity in the motion behavior: The top part of the foreground (small) flag, that is anchored, exhibits slower motion in comparison to the rest of the flag. In that it has similar motion behavior to that of the background (big) flag, explaining the segmentation error. The segmentation resulted by using both intensities and motion is presented in the fourth column (d) of Figure 5.4. This final output presents better and more accurate segmentation than the previous two. Thus it is evident that using both color and motion results in the best segmentation. This notion is substantiated by the tracking errors that appear in the same image regions both in column (b) and (c), just left to the small flag. Hence, solely color and motion can not make the separation between the flag and its background. However, when both are combined the delineation becomes more accurate.

Figures 5.5 presents the tracking results for two surveillance sequences taken from the CAVIAR data set [18]. These sequences present two scenarios, where the target of interest is moving (first row) and where it is standing (second row). In both cases the target of interest is occluded part of the time. The tracker position is marked with a rectangle, as can be seen in the tracker sticks to the target of interest even under occlusion.

A synthetic sequence, taken from the standard HDTV TRICTRAC data set [102] is presented in Figure 5.6. Since this sequence is a synthetic one, the players' shirts' color schemes are identical. The target of interest is the red player sprinting to the right. When color data is used, both players in red are delineated, while when incorporating motion, as can be seen in the Figure 5.6, only the player who is the target of interest is tracked.

For quantitative evaluation we use the PETS 2009 data set [87]. This set consists of sequences acquired from several synchronized cameras, from which we use three different views: $View$ 1, $View$ 2 and $View$ 8. The PETS sequences present several challenges: (i) there are significant complete and partial occlusions; (ii) the motion of some targets is highly dynamic, as they are suddenly stopping, moving backwards, or in circles; (iii) target

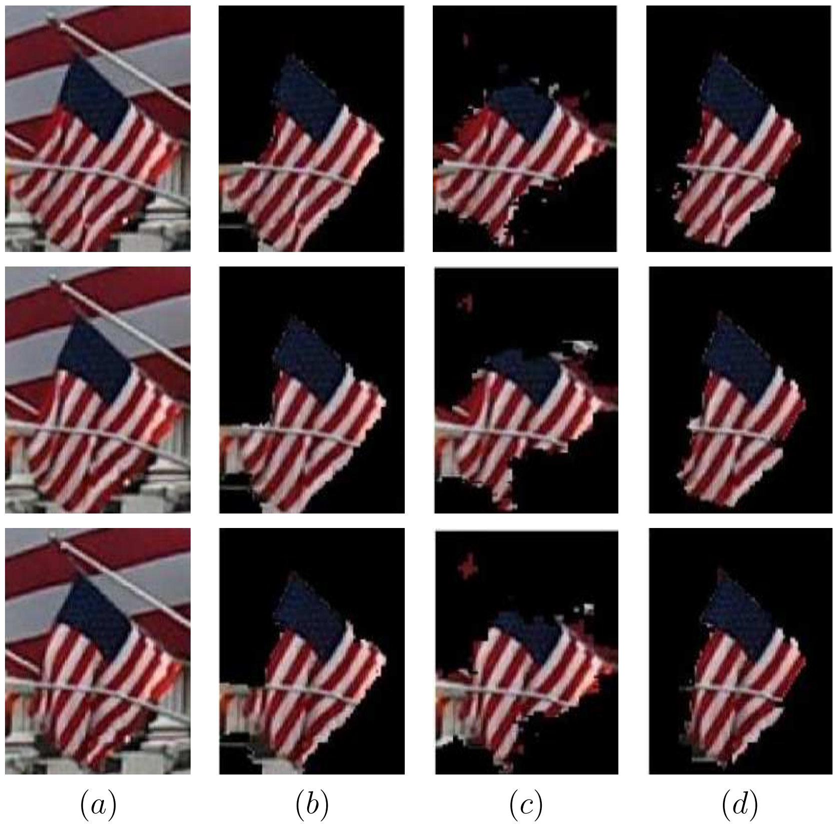


Figure 5.4: Flag Tracking Results - Column (a) - original frames; (b) results produced by using only pixels' intensities; (c) results using solely motion data; and (d) results by using both intensities and motion

appearance changes heavily, caused by different lighting conditions in different image areas, or when a target turns with respect to the camera position; and (iv) the people in the crowd walk very close to each other, regularly occluding each other. Our algorithm can detect any number of targets automatically as a function of the number and locations of the seed nodes placed by the user. Figure 5.7 presents the results and all trajectories computed for the different views. The ground-truth for each object is marked with a colored line, where the calculated paths are presented in black. Our algorithm does not assume any motion patterns, therefore it handles robustly the abrupt changes in motion and direction. The occlusions, and changes in appearance are addressed by processing a batch of succeeding frames within a temporal window together.

We compare the performance of our algorithm (NC-Track) with the following state-of-the-art tracking algorithms: Breitenstein et. al -[9] (*BR-1*) and [10] (*BR-2*), and the tracking algorithm presented by Yu et. al [111] (*Yu*). The results are summarized in Table 5.1. The

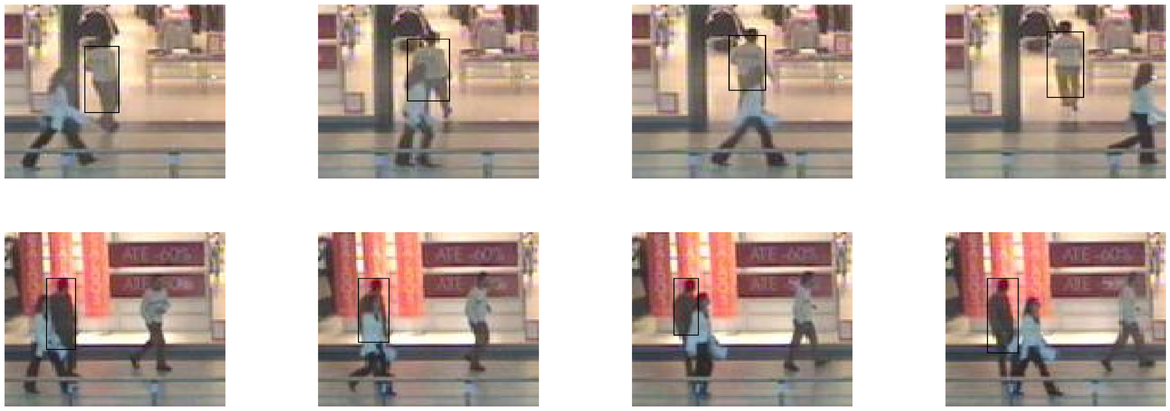


Figure 5.5: Surveillance Sequences Tracking Results. 4 representing frames from surveillance sequences taken from the CAVIAR data sets [18]

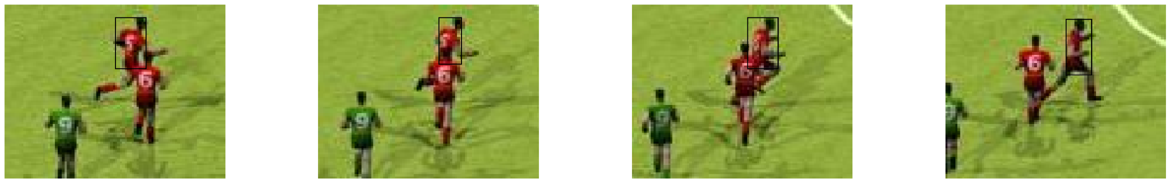


Figure 5.6: Tracking in Synthetic Video, of object with highly similar color scheme to other objects in the sequence

results demonstrate that even though we utilize a coarse and noisy motion field we get comparable results to the other methods. In one case (view 8), NC-Track provides superior results. The observed results provide evidence that our method gives reasonably accurate tracking compared to other methods. Given the utilization of the MPEG-4 motion scheme, the method is a great candidate for real-time applications or when processing compressed videos. One caveat of NC-Track, mentioned previously, is that the algorithm does not label identifications for target, a feature that may be of importance in some applications.

5.5. Conclusions

We show here a scheme for target tracking in videos that incorporates both color and motion data. The scheme presented is based on the *normalized cuts'* segmentation criterion [49],

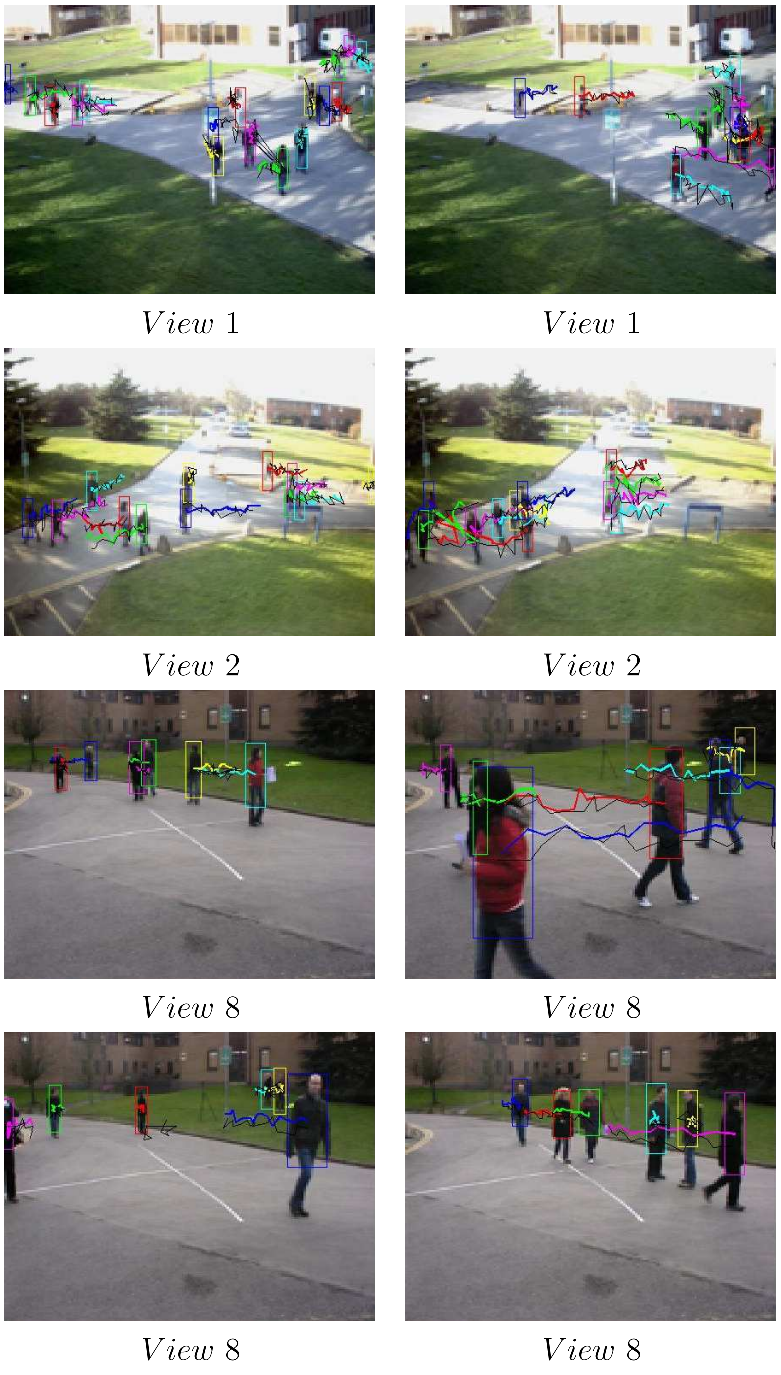


Figure 5.7: The resulting trajectories for the PETS09 [87], tasks S2.L1 and S2.L2.

Table 5.1: Tracking Results - Precision (MOTP) and Accuracy (MOTA) compared with [9] (BR-1); [10] (BR-2) and [111] (Yu)

View	NC-Track		BR-1		BR-2		Yu	
	MOTP	MOTA	MOTP	MOTA	MOTP	MOTA	MOTP	MOTA
1	65.54%	49.35%	51.30%	50.00%	–	–	79.10%	55.10%
2	71.07%	56.76%	51.30%	50.00%	–	–	79.10%	55.10%
8	81.54%	83.02%	56.30%	79.70%	56.70%	74.90%	–	–

which is solved by solving the s, t-cut problem using the HPF polynomial time algorithm.

The segmentation scheme presented in this chapter is highly robust, thus permitting the utilization of block-matching motion estimation techniques, which are computationally efficient. The evaluation of the method on standard and non-standard benchmark videos clearly shows that the method presents comparable results to other state-of-the-art methods, while incorporating coarse and inaccurate motion field. These, along with the time efficiency of the algorithm, make our scheme a perfect choice for many online video tracking applications.

Chapter 6

Conclusion

This thesis explores machine learning techniques in several important application areas: nuclear material detection, drug ranking and target tracking. Supervised Normalized Cut (SNC), Fractional Adjusted Bipartitional Score (FABS) and NC-Track are used to solve these problem respectively. They are all based on a new algorithm, in [49], Normalized Cut Prime (NC′). We divide the techniques into three different categories: clustering [34], classification [109] and ranking [52].

We explore clustering in the context of target tracking. The target tracking problem is to cluster pixels into two groups, the background and the foreground in a sequence of video frames. We find that a graph-cut formulation incorporating intensity and motion data has the highest performance. Tests on real-life benchmark videos show that this graph-cut technique is more efficient than many existing techniques, and that it delivers good quality results.

We explore classification in the context of detecting concealed illicit nuclear material. Supervised Normalized Cut (SNC) is used to classify measurements obtained from very low resolution plastic scintillation detectors, among others. The classification problem is to use training data to accurately determine the identity of a given material. SNC method is proved to be appropriate for this task. In terms of accuracy, the SNC method is on par with alternative approaches, yet SNC is computationally more efficient.

We explore drug ranking of several drugs treating the same disease according to their effectiveness, using data directly from experimental images. The framework used in drug ranking producing graph theoretic descriptors, automatically ordering the performance of drugs, is called fractional adjusted bi-partitional score (FABS). Computational experiments show that FABS framework implemented with normalized cut prime (FABS-NC′) outperforms other implementations of FABS and alternative methods currently used for ranking that are unrelated to FABS.

Overall, this thesis contributes to the field of machine learning and data mining by 1) extensive discussion of three distinct application areas which can benefit from machine learning techniques, 2) detailed descriptions and analysis of techniques developed for these three areas, and 3) computational studies in the three application areas of nuclear material detection, drug ranking and video tracking, comparing the techniques used in these three areas with different techniques well known to the machine learning community.

Several extensions that can be explored in the future are:

1. In nuclear material detection problem, the analysis presented in the thesis provides a proof of concept that the SNC approach is worth investigating further in this context, and with more detailed and advanced data sets. It should also be pursued as an approach for identifying isotopes in spectra obtained even with higher resolution detectors. Future research will test supervised normalized cut on additional nuclear data sets on a vaster variety of SNMs, as they become available.

2. In drug ranking, our current graph construction only considers weights on arcs. No weights are assigned to nodes. The future work is to investigate whether the introduction of node weights can benefit the prediction results. Moreover, we can expand our FABS application beyond drug ranking into other ranking problems in different areas.

Bibliography

[1] D. W. Aha. “Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms”. In: *International Journal of Man-Machine Studies* 36.2 (1992), pp. 267–287.

[2] S. Ali and M. Shah. “Human Action Recognition in Videos Using Kinematic Features and Multiple Instance Learning”. In: *IEEE Trans, Pattern Analysis and Machine Intelligence* 32.2 (2010), pp. 288–303.

[3] S. J. Altschuler and L. F. Wu. “Cellular Heterogeneity: Do Differences Make a Difference?” In: *Cell* 141, issue 4 (2010), pp. 559 –563.

[4] B. Benfold and I. Reid. “Stable multi-target tracking in real-time surveillance video”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* 2011, pp. 3457 –3464.

[5] K. Bernardin and R. Stiefelhagen. “Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics”. In: *EURASIP Journal on Image and Video Processing* 2008 (2008).

[6] W. Bertozzi and R. J. Ledoux. “Nuclear resonance fluorescence imaging in non-intrusive cargo inspection”. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 241.1-4 (2005), pp. 820 –825.

[7] H. Bhadeshia. “Neural Networks in Materials Science”. In: *ISIJ International* 39 (10) (1999), pp. 966–979.

[8] M. Black. “Combining intensity and motion for incremental segmentation and tracking over Combining intensity and motion for incremental segmentation and tracking over long image sequences”. In: *ECCV 1992.* 1992, pp. 485 –493.

[9] M. D. Breitenstein et al. “Markovian tracking-by-detection from a single, uncalibrated camera”. In: *IEEE Int. Workshop Performance Evaluation of Tracking and Surveillance.* 2009.

[10] M. D. Breitenstein et al. “Online Multi-Person Tracking-by-Detection from a Single, Uncalibrated Camera”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010). in press.

[11] T. Brox et al. “Colour, texture, and motion in level set based segmentation and tracking”. In: *Image and Vision Computing* 28.3 (2010), pp. 376 –390.

[12] T. Brox et al. “High Accuracy Optical Flow Estimation based on Theory for Warping”. In: *ECCV 2004*. Vol. 4. 2004, pp. 25–36.

[13] A. Bugeau and P. Pérezz. “Track and cut: simultaneous tracking and segmentation of multiple objects with graph cuts”. In: *J. Image Video Process.* 2008 (2008), 3:1–3:14.

[14] C. J. C. Burges. “A Tutorial on Support Vector Machines for Pattern Recognition”. In: *Data Mining and Knowledge Discovery* 2 (2 1998). 10.1023/A:1009715923555, pp. 121–167.

[15] G. Carneiro et al. “Supervised Learning of Semantic Classes for Image Annotation and Retrieval”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.3 (2007), pp. 394–410.

[16] T. Carpenter et al. “Safety and Risk Modeling and Its Applications”. In: ed. by Hoang Pham. Springer Series in Reliability Engineering. Springer London, 2009. Chap. Sensor Management Problems of Nuclear Detection, pp. 299–323.

[17] R. Caruana and A. Niculescu-Mizil. “An empirical comparison of supervised learning algorithms”. In: *Proceedings of the 23rd international conference on Machine learning.* ICML ’06. Pittsburgh, Pennsylvania: ACM, 2006, pp. 161–168.

[18] CAVIAR Data Set. *http://homepages.inf.ed.ac.uk/rbf/CAVIAR*. 2001.

[19] B. G. Chandran and D. S. Hochbaum. “A Computational Study of the Pseudoflow and Push-Relabel Algorithms for the Maximum Flow Problem”. In: *Oper. Res.* 57.2 (2009), pp. 358–376.

[20] C.-C. Chang and C.-J. Lin. “LIBSVM: A library for support vector machines”. In: *ACM Transactions on Intelligent Systems and Technologies* 2 (3 2011), 27:1–27:27.

[21] C. Conrad and D. W. Gerlich. “Automated microscopy for high-content RNAi screening”. In: *J. Cell Biol.* 188 (2010), pp. 453–461.

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

[23] I. J. Cox, S. B. Rao, and Y. Zhong. “Ratio Regions: A Technique for Image Segmentation”. In: *Proc. Int. Conf. on Pattern Recognition.* Vol. B. 1996, pp. 557–564.

[24] K. Crammer and Y. Singer. “On the Learnability and Design of Output Codes for Multiclass Problems”. In: *Machine Learning* 47.2-3 (2 2002), pp. 201–233.

[25] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.* Cambridge University Press, 2000.

[26] P. Denner, J. Schmalowsky, and S. Prechtl. “High-content analysis in preclinical drug discovery”. In: *Comb. Chem. High Throughput Screen* 11 (2008), pp. 216–230.

[27] Y. S. Ding, T. L. Zhang, and K. C. Chou. “Prediction of Protein Structure Classes with Pseudo Amino Acid Composition and Fuzzy Support Vector Machine Network”. In: *Protein and Peptide Letters* 14.8 (2007), pp. 811–815.

[28] P. A. Dowd and E. Pardo-Iguzquiza. "Estimating the boundary surface between geologic formations from 3D seismic data using neural networks and geostatistics". In: *Geophysics* 70.1 (2005), pp. 1–11.

[29] N. R. Draper and H. Smith. *Applied Regression Analysis (3rd ed.)* John Wiley, 1998.

[30] K.-B. Duan and S. S. Keerthi. "Which Is the Best Multiclass SVM Method? An Empirical Study". In: *Multiple Classifier Systems*. Ed. by Nikunj C. Oza et al. Vol. 3541. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, pp. 732–760.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. "Unsupervised Learning and Clustering". In: *Pattern classification (2nd edition)*. Wiley, New York, 2001. Chap. 10, pp. 517 –600.

[32] Y. Feng et al. "Multi-parameter phenotypic profiling: using cellular effects to characterize small-molecule compounds". In: *Nat. Rev. Drug Discov.* 8 (2009), pp. 567–578.

[33] B. Fishbain. *New York Stock Exchange Facade, video and motion.* 2012. URL: `http://fishbain.net.technion.ac.il/datasets/`.

[34] B. Fishbain, D. Hochbaum, and Y. Yang. "A new approach for real-time target tracking in videos." In: *SPIE Newsroom* (Jan 2013).

[35] B. Fishbain, D. S. Hochbaum, and S. Mueller. *The Pseudoflow algorithm for minimum cut in vision problems.* 2009.

[36] B. Fishbain, L. P. Yaroslavsky, and I. A. Ideses. "Real-time stabilization of long range observation system turbulent video". In: *Journal of Real-Time Image Processing* 2.1 (2007), pp. 11–22.

[37] L. Ford and D. Fulkerson. "Maximal flow through a network". In: *Canadian Journal of Mathematics* (1956), 8:339–404.

[38] D. Freedman and M. W. Turek. "Illumination-invariant tracking via graph cuts". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* Vol. 2. 2005, 10 –17 vol. 2.

[39] Y. Freund and R. E. Schapire. "Decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of Computer and System Sciences* 55:1 (1997), pp. 119–139.

[40] G. Fung and O. L. Mangasarian. *A Feature Selection Newton Method for Support Vector Machine Classification.* Tech. rep. 02-03. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-01.ps. Madison, Wisconsin: Data Mining Institute, Computer Sciences Department, University of Wisconsin, 2002.

[41] G. L. Fung and O. L. Mangasarian. "A Feature Selection Newton Method for Support Vector Machine Classification". In: *Computer Optimization and Appliations* 28.2 (2004), pp. 185–202.

[42] G. M. Fung and O. L. Mangasarian. “Multicategory Proximal Support Vector Machine Classifiers”. In: *Machine Learning* 59.1-2 (1 2005), pp. 77–97.

[43] N. P. Geisinger. “Classification of Digital Modulation Schemes Using Linear and Nonlinear Classifiers”. PhD thesis. Naval Postgraduate School, Monterey, CA, USA, 2010.

[44] C.A. Gentile. “*US Patent* No. 7,711,661 B2, System and method for resolving gamma-ray spectra”. Pat. US 7,711,661 B2. 2010.

[45] I. Guyon et al. “Gene Selection for Cancer Classification using Support Vector Machines”. In: *Machine Learning* 46.1-3 (1 2002), pp. 389–422.

[46] T. Hastie et al. “The Entire Regularization Path for the Support Vector Machine”. In: *J. Mach. Learn. Res.* 5.1 (2004), pp. 1391–1415.

[47] D. S. Hochbaum. “An efficient algorithm for image segmentation, Markov Random Fields and related problems”. In: *Journal of the ACM* 48 (2001), pp. 686–701.

[48] D. S. Hochbaum. *HPF: Hochbaum’s Pseudo-Flow Algorithm Implementation: http://riot.ieor.berkeley.edu/riot/Applications/Pseudoflow/maxflow.html.* 2010, Last updated on July 26, 2010.

[49] D. S. Hochbaum. “Polynomial Time Algorithms for Ratio Regions and a Variant of Normalized Cut”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5 (2010), pp. 889–898.

[50] D. S. Hochbaum. “Solving integer programs over monotone inequalities in three variables: A framework for half integrality and good approximations”. In: *European Journal of Operational Research* 140.2 (2002), pp. 291 –321.

[51] D. S. Hochbaum. “The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem”. In: *Oper Res.* 56.4 (2008), pp. 992–1009.

[52] D. S. Hochbaum, C.-N. Hsu, and Y. T. Yang. “Ranking of multidimensional drug profiling data by fractional-adjusted bi-partitional scores”. In: *Bioinformatics* 28.12 (2012), pp. i106–i114.

[53] Z. Hua et al. “Predicting corporate financial distress based on integration of support vector machine and logistic regression”. In: *Expert Systems with Applications* 33.2 (2007), pp. 434 –440.

[54] K. Huang and R. F. Murphy. “Boosting accuracy of automated classification of fluorescence microscope images for location proteomics”. In: *BMC Bioinformatics* 5 (2004), p. 78.

[55] International Atomic Energy Agency. “Combating Illicit Trafficking in Nuclear and other Radioactive Material”. In: vol. 6. IAEA Nuclear Security Series. Vienna: International Atomic Energy Agency, Vienna, Austria, 2007. Chap. 7.3. SPECIFIC RADIONUCLIDES, pp. 66–68.

[56] C. M. Jarque and A. K. Bera. “A test for normality of observations and regression residuals”. In: *International Statistical Review* 55 (1987), 163172.

[57] M. Jordan, ed. *Learning in graphical models.* North Atlantic Treaty Organization. Scientific Affairs Division., 1996.

[58] L. J. Kangas et al. “The use of artificial neural networks in PVT-based radiation portal monitors”. In: *Nuclear instruments & methods in physics research. Section A, Accelerators, spectrometers, detectors and associated equipment* 587.2-3 (2008), pp. 398–412.

[59] S. Kotsiantis. “Supervised learning: A review of classification techniques”. In: *Informatica* 31.1 (2007), pp. 249–268.

[60] P. Kramer et al. “Super-resolution mosaicing from MPEG compressed video”. In: *Signal Processing: Image Communication* 22.10 (2007), pp. 845 –865.

[61] P. Lang et al. “Cellular imaging in drug discovery”. In: *Nat. Rev. Drug Discov* 5 (2006), pp. 343–356.

[62] C.-C. Lin et al. “Boosting multiclass learning with repeating codes and weak detectors for protein subcellular localization”. In: *Bioinformatics* 23.24 (2007), pp. 3374–3381.

[63] Y.-S. Lin et al. “A spectral graph theoretic approach to quantification and calibration of collective morphological differences in cell images”. In: *Bioinformatics* 26 (12) (2010), pp. i29–i37.

[64] J. Liu, S. Ali, and M. Shah. “Recognizing Human Actions Using Multiple Features”. In: *CVPR.* 2008.

[65] B. D. Lucas and K. Takeo. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *Intl. Joint Conf. on Artificial Intelligence (IJCAI).* 1981, pp. 674–679.

[66] L. Luo. “Chemometrics and its applications to x-ray spectrometry”. In: *X-Ray Spectrometry* 35.4 (2006), p. 215225.

[67] J. Malcolm, Y. Rathi, and A. Tannenbaum. “Multi-Object Tracking Through Clutter Using Graph Cuts”. In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* 2007, pp. 1 –5.

[68] O. L. Mangasarian and E. W. Wild. “Nonlinear Knowledge in Kernel Approximation”. In: *Neural Networks, IEEE Transactions on* 18.1 (2007), pp. 300 –306.

[69] R. E. Marrs et al. “Fission-product gamma-ray line pairs sensitive to fissile material and neutron energy”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 592.3 (2008), pp. 463 –471.

[70] A. M. Martinez and A. C. Kak. “PCA versus LDA”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.2 (2001), pp. 228–233.

[71] T. Masters. *Signal and Image Processing with Neural Networks.* ISBN 0-471-04963-8. John Wiley & Sons, Inc., 1994.

[72] G. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* Wiley Interscience, New-York, NY, USA, 2004.

[73] S. W. Menard. *Applied Logistic Regression (2nd ed.)* SAGE, 2002.

[74] L. Mihailescu et al. “Dynamic stand-off 3D gamma-ray imaging”. In: *12th Symposium on Radiation Measurements and Applications (SORMA).* National Nuclear Security Administration, Washington DC, USA, 2010, p. 106.

[75] T. J. Mitchison. “Small-molecule screening and profiling by using automated microscopy”. In: *ChemBioChem* 6 (2005), pp. 33–39.

[76] M. F. Mller. “A scaled conjugate gradient algorithm for fast supervised learning”. In: *Neural networks* 6.4 (1993), pp. 525–533.

[77] M. M. Morelock et al. “Statistics of assay validation in high throughput cell imaging of nuclear factor κB nuclear translocation”. In: *Assay and Drug Development Technologies* 3.5 (2005), pp. 483–499.

[78] A. Nichols. “High content screening as a screening tool in drug discovery”. In: *Methods Mol. Biol.* 356 (2007), pp. 379–387.

[79] Y. Nie and K.-K. Ma. “Adaptive rood pattern search for fast block-matching motion estimation”. In: *IEEE Trans. Image Processing* 11.12 (2002), pp. 1442–1449.

[80] P. Nillius, J. Sullivan, and S. Carlsson. “Multi-Target Tracking - Linking Identities using Bayesian Network Inference”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on.* Vol. 2. 2006, pp. 2187 –2194.

[81] E. B. Norman et al. “Signatures of fissile materials: high-energy [gamma] rays following fission”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 521.2-3 (2004), pp. 608 –610.

[82] J. I. Olszewska, C. De Vleeschouwer, and B. Macq. “Multi-feature vector flow for active contour tracking”. In: *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on.* 2008, pp. 721 –724.

[83] N. Papadakis and A. Bugeau. “Tracking with Occlusions via Graph Cuts”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.1 (2011), pp. 144 –157.

[84] K. D. Paull et al. “Identification of Novel Antimitotic Agents Acting at the Tubulin Level by Computer-assisted Evaluation of Differential Cytotoxicity Data”. In: *Cancer Res* 52 (1992), p. 3892.

[85] J.-Y. Peng et al. “Automatic Morphological Subtyping Reveals New Roles of Caspases in Mitochondrial Dynamics”. In: *PLoS Comput Biol* 7.10 (Oct. 2011), e1002212.

[86] Z. E. Perlman et al. “Multidimensional Drug Profiling By Automated Microscopy”. In: *SCIENCE* 306 (2004), pp. 1194–1198.

[87] PETS 2009 Benchmark Dataset. *http://www.cvg.rdg.ac.uk/PETS2009/index.html.* 2009.

[88] R. Polikar. “Ensemble learning”. In: *Scholarpedia* 4.1 (2009), p. 2776.

[89] H. W. Resson, R. S. Varghese, and R. Goldman. “Computational Methods for Analysis of MALDI-TOF Spectra to Discover Peptide Serum Biomarkers”. In: *The Protein Protocols Handbook* IV (2009), pp. 1175–1183.

[90] R. Rifkin and A. Klautau. “In Defense of One-Vs-All Classification”. In: *Journal of Machine Learning Research* 5.1 (2004), pp. 101–141.

[91] F. Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms.* Tech. rep. DTIC Document, 1961.

[92] H. Sabirin, J. Kim, and M. Kim. “Graph-based object detection and tracking in H.264/AVC bitstreams for surveillance video”. In: *Multimedia and Expo (ICME), 2011 IEEE International Conference on.* 2011, pp. 1 –6.

[93] R. E. Schapire. “The strength of weak learnability”. In: *Machine learning* 5.2 (1990), pp. 197–227.

[94] E. Sharon et al. “Hierarchy and adaptivity in segmenting visual scenes”. In: *Nature* 442.7104 (2006), pp. 810–813.

[95] J. B. Shi and J. Malik. “Normalized cuts and image segmentation”. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905.

[96] H. T. Siegelmann and E. D. Sontag. “Analog computation via neural networks”. In: *Theoretical Computer Science* 131(2) (1994), pp. 331–360.

[97] E. Swanberg et al. “Using low resolution gamma detectors to detect and differentiate ^{239}Pu and ^{235}U fissions”. In: *Journal of Radioanalytical and Nuclear Chemistry* 282.3 (3 2009). 10.1007/s10967-009-0283-4, pp. 901–904.

[98] N. Taguchi et al. “Mitotic phosphorylation of dynamin-related gtpase drp1 participates in mitochondrial fission”. In: *Biol. Chem.* 282 (2007), pp. 11521–11529.

[99] D. L. Taylor and J. R. Hsaskins. *High Contnet Screening: A Powerful Approach to Systems Cell Biology and Drug Discovery.* Ed. by K. A. Giuliano. Totowa, NY: Humana, 2007.

[100] R. Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[101] A. M. Tourapis, O. C. Au, and M. L. Liou. “Highly efficient predictive zonal algorithms for fast block-matching motion estimation”. In: *IEEE Trans Circuits and Systems for Video Technology* 12.10 (2002), pp. 934–947.

[102] TRICTRAC Test Data Sets. *http://www.multitel.be/trictrac/.* 2004.

[103] VideoLAN Organization. *x264 video encoding into the H.264/MPEG-4 AVC format.* 2010. URL: `http://www.videolan.org/developers/x264.html`.

[104] F. Wang et al. "A Model Based on Hybrid Support Vector Machine and Self-Organizing Map for Anomaly Detection". In: *Communications and Mobile Computing, International Conference on* 1 (2010), pp. 97–101.

[105] T. Washio and H. Motoda. "State of the art of graph-based data mining". In: *ACM SIGKDD Explorations Newsletter* 5 (2003), pp. 59–68.

[106] D. West. "Neural network credit scoring models". In: *Computers & Operations Research* 27.11-12 (2000), pp. 1131 –1152.

[107] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd. San Francisco: Morgan Kaufmann, 2005.

[108] N. Xu, N. Ahuja, and R. Bansal. "Object segmentation using graph cuts based active contours". In: *Computer Vision and Image Understanding* 107.3 (2007), pp. 210 –224.

[109] Y. T. Yang et al. "The Supervised Normalized Cut Method for Detecting, Classifying and Identifying Special Nuclear Materials". In: *INFORMS J Computing* (to appear, online version April, 2013).

[110] J. C. Yarrow et al. "Phenotypic screening of small molecule libraries by high throughput cell imaging". In: *Comb Chem High Throughput Screen* 6 (2003), pp. 279–286.

[111] J. Yu, D. Farin, and B. Schiele. "Multi-target Tracking in Crowded Scenes". In: *Pattern Recognition*. Ed. by Rudolf Mester and Michael Felsberg. Vol. 6835. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, pp. 406–415.

[112] J. Zhang, X.-D. Zhang, and S.-W. Ha. "A Novel Approach Using PCA and SVM for Face Detection". In: *Fourth International Conference on Natural Computation, (ICNC '08)*. Vol. 03. IEEE Computer Society, 2008, pp. 29–33.

[113] J. H. Zhang, T. D. Y. Chung, and K. R. Oldenburg. "A simple statistical parameter for use in evaluation and validation of high throughput screening assays". In: *Journal of Biomolecular Screening* 4 (1999), pp. 67–73.

[114] X. Zhou and S. T. Wong. "Informatics challenges of high-throughput microscopy". In: *IEEE Signal Proc. Mag.* 23 (2006), pp. 63–72.

[115] J. Zhu et al. "1-norm Support Vector Machines". In: *Neural Information Processing Systems*. MIT Press, 2003, pp. 16–23.

[116] S. Zhu and K.-K. Ma. "A new diamond search algorithm for fast block-matching motion estimation". In: *IEEE Trans. Image Processing* 9.2 (2000), pp. 287–290.