CRRAO Advanced Institute of Mathematics, Statistics and Computer Science (AIMSCS)

Research Report

Author (s): **Ariful Azad, Arif Khan, Bartek Rajwa, Saumyadipta Pyne and Alex Pothen**

Title of the Report: **Classifying Immunophenotypes With Templates From Flow Cytometry**

Research Report No.: **RR2013-08**

Date: **June 25, 2013**

Prof. C R Rao Road, University of Hyderabad Campus,
Gachibowli, Hyderabad-500046, INDIA.
www.crraoaimscs.org

Classifying Immunophenotypes With Templates From Flow Cytometry

Ariful Azad
Computer Science
Purdue University
aazad@purdue.edu

Arif Khan
Computer Science
Purdue University
khan58@purdue.edu

Bartek Rajwa
Bindley Bioscience Center
Purdue University
brajwa@purdue.edu

Saumyadipta Pyne
C.R. Rao Advanced Institute
of Mathematics, Statistics and
Computer Science
Hyderabad, India
spyne@crraoaimscs.res.in

Alex Pothen
Computer Science
Purdue University
apothen@purdue.edu

ABSTRACT

Given a collection of flow cytometry data samples, consisting of fluorescence measurements of proteins that characterize different cell types, we describe an algorithm to dynamically classify the samples into several classes based on their immunophenotypes. Each sample is initially clustered to identify the cell populations present in it. Using a combinatorial dissimilarity measure between cell populations in samples, we compute meta-clusters that correspond to the same cell population across samples. The collection of meta-clusters in a class of samples then describes a template for that class. We organize the samples into a template tree, and use it to classify new samples into existing classes or create a new class if needed. As new samples are classified, we update the templates and their statistical parameters so that the new information is used to update the templates for the classes. We have used our dynamic classification algorithm to classify T Cell Phosphorylation data that shows increased presence of the proteins SLP-76 and ZAP-70, which are involved in a platform that assembles the signaling proteins in the immune response. We have also used the dynamic classification algorithm to show that variations in the immune system in individuals is a larger effect than variations in multiple samples from each individual.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Flow Cytometry, Classification, Clustering

Consider a collection of flow cytometry data samples, each sample consisting of fluorescence measurements of protein markers made at the single-cell level from hundreds of thousands of cells, indicating different cell types present in each sample. We describe an algorithm to dynamically classify such samples into several classes based on the cell populations present in the samples. Each class is summarized by means of a characteristic template that describes the cell populations and their statistical parameters. We build the class templates, organize the samples into a template tree data structure, update the templates (and the statistical parameters involved), and the template tree as new samples are classified. Our algorithm makes use of a combinatorial dissimilarity measure to compare two samples based on a mixed edge cover computed from a graph model of the problem. It then uses a hierarchical merging and matching algorithm similar in spirit to the UPGMA algorithm in phylogenetics to build templates descriptive of classes from multiple samples. The template tree is used to classify samples into different classes efficiently.

Flow cytometry is a platform for profiling the immune responses of biological samples, as the immune system responds to various stimuli, e.g., the development of the immune system, or the immune respone to a pathogen, a drug, or vaccine. The immune response is measured in flow cytometry (FC) by quantifying the abundances of multiple protein complexes (called Clusters of Differentiation, CD). Fluorophore-conjugated antibodies are used to target marker proteins on the surface or within the cells. The fluorescence intensities due to binding of specific antibodies convey information about the type and levels of expression of each of the protein markers per cell [22], and thus identify different cell types in the sample. Recent advances in FC technology allow us to measure the abundance of fifteen to twenty proteins simultaneously in each cell from a sample containing millions of cells [16]. This technology is now routinely used to understand how different kinds of immune cells develop [18], how the immune system responds at multiple levels to the presence of a pathogen, to clinically diagnose diseases of the immune system [19], and to develop novel vaccines (e.g., HIV) [21].

In conventional FC practice, cell populations are identified by visualizing cells by means of a collection of two-dimensional scatter plots (see Fig. 8 in this paper for an example). However, with the ability to monitor large numbers of protein markers simultaneously, this approach is not feasible for high-dimensional or high-throughput FC data. To address this problem, many automated data clustering algorithms to identify cell populations in FC samples have been described by a number of researchers, and [1] provides a state-of-the-art summary of the field. We extend this work to registration and classification problems associated with a large collection of samples. Here the cell populations with their statistical descriptions in each sample have been identified with a clustering algorithm, and we proceed to organize samples into a few classes using these descriptions. As new samples are acquired, we need to classify the new samples into existing classes, or create new classes if needed. We also need to update the class descriptions with the data from the new samples.

Samples belonging to the same class are summarized by a template, which represents a generalization of the sample's expression pattern [3, 11, 20]. The concept of cell clusters or populations in a sample can be extended to *meta-clusters* in a collection of similar samples, representing generic cell populations that appear in each sample with some sample-specific variation. A *template* is a collection of relatively homogeneous metaclusters commonly shared across samples of a given class, thus describing the key immune-phenotypes of an overall class of samples in a formal, yet robust, manner. Given the inter-sample variations due to innate biological variations among individuals or Poisson and Gaussian noise from the FC measurements, a few templates can concisely represent a large cohort of samples by emphasizing their major characteristics while hiding statistical noise and unnecessary details. Thereby, overall changes across multiple conditions can be determined rigorously by comparing the cleaner and fewer class templates rather than the large number of noisy samples themselves [3, 20]. We show that the use of templates leads to efficient classification algorithms as well.

Beside high-level visualization and cross-class comparisons, templates can be used to classify new samples with unknown status. In this paper we use this approach to classify samples in terms of their expression of markers of the immune system. In the static classification approach, we build a fixed number of templates each representing samples from a particular class, and orgnize them into a template tree data structure. A new sample is then predicted to come from a class whose template is it most similar to. In the dynamic classification approach, we update the templates and also the template tree, as new samples are classified. Hence the dynamic approach can be used to summarize and classify samples when they are being continuously collected, as in a flu epidemic. Another context where the dynamic classfication approach is useful is when the samples are collected at a large number of hospitals or labs; the data at each hospital can be analyzed *in situ*, and only the summaries (templates and template trees) need to be shared among the hospitals, thus avoiding issues with confidentiality.

We demonstrate the use of template-based classification on two different datasets in this paper. The first dataset measures the differences in phosphorylation events before and after stimulating T cells in human whole blood with an anti-CD3 antibody. By creating pre-stimulation and post-stimulation templates we are able to classify samples according to their stimulation status. The second dataset studies the natural variations among different subsets of cells in five healthy individuals. Five technical replicates were created from each subject, and we show that technical replicates from each subject are correctly classified to the corresponding template for the individual, thus showing that these technical variations are relatively small when compaed with the natural biological variations across individuals.

Template based classification has been used in several areas such as face recognition, speech recognition, character recognition, etc. In face recognition [6], a template library is created with one or more digital images from each person. When attempting recognition, an unclassified image is compared in turn to each database image by computing correlations of different features (eyes, nose, mouth etc.) and is classified as the one giving the highest cumulative score. In speech recognition [8, 9], a template is created for each speaker by a sequence of consecutive acoustic feature vectors and an incoming signal is classified by comparing it with the templates using the Dynamic Time Warping algorithm. In character recognition [7], representative prototypes for each character are created from different writing styles and an incoming character is classified by comparing it to existing prototypes using feature matching algorithm. Our approach has similarities to these approaches in principle but differs from them significantly in how the templates are created, represented and compared to incoming samples. In contrast to these approaches, we maintain a hierarchy of the training samples in order to use their relationships in future classifications. We represent a template with the shared features of all samples in a class whereas the methods discussed above use representatives from the training set. Furthermore, with the dynamic template algorithm we continuously update templates as new samples are classified, which improves the accuracy of future classifications.

Flow cytometry data is continuous, impacted by Poisson and Gaussian noise, high-dimensional (each column corresponds to a fluoresence from a protein being measured), and provides data at the single cell level for millions of intact cells in a sample. Microarray data is similar in that it is continuous, but corresponds to gene expression rather than protein. Microarray data measures the expression of a large number of genes under different conditions, whereas FC data measures a smaller number of proteins characteristic of a few immunophenotypes, across a large number of samples due to its lower cost and widespread clinical use. The nature of the data, its pre-processing, and algorithms for downstream analysis are all substantially different for the two technologies.

1. BACKGROUND

1.1 Identifying cell populations

A flow cytometry sample measuring the levels of expression of p (protein) markers for n cells is represented by an $n \times p$ matrix A. The matrix element $A(i, j)$ represents the abundance of the j^{th} marker in the i^{th} cell. Each sample consists

of several clusters of cells, with each cluster of cells expressing one or more phenotypes in the measured marker space. Such a cluster of cells represents a particular cell type and is called a *cell population* in cytometry. We model this collection of clusters with a finite mixture model of multivariate normal distributions. In the mixture model, each cell population is characterized by a multi-dimensional normal distribution, represented by two parameters $\boldsymbol{\mu}$, the p-dimensional mean vector, and Σ, the $p \times p$ covariance matrix [14].

We identify the cell populations in a sample and estimate their distribution parameters by applying a two step clustering algorithm (known as automated gating in flow cytometry). In the first step, we identify the optimal number of cell clusters, k^*, by applying the k-means clustering algorithm for a wide range of values for k, and select the optimal number of clusters by optimizing both the Calinski-Harabasz and S_Dbw cluster validation criteria [13]. The distribution parameters for each cluster are then estimated using the Expectation-Maximization (EM) algorithm [15] (with software obtained from Dr. Tsung-I Lin from National Chung Hsing University, Taiwan, personal communication). We have used other clustering algorithms such as the Dirichlet Process Mixture (DPM) model, and have obtained similar results. We have also stabilized the variances of the cell populations, and this is key step for downstream statistical processing, but this will be discussed in another publication [4].

1.2 Dissimilarity between cell populations

We can calculate the dissimilarity between a pair of cell populations by any measure that computes the dissimilarity between a pair of multivariate probability distributions. In this study, we used the symmetrized Kullback-Leibler (KL) divergence, also known as the relative entropy in information theory. Let $c_1(\boldsymbol{\mu_1}, \Sigma_1)$ and $c_2(\boldsymbol{\mu_2}, \Sigma_2)$ be the mean vector and covariance matrix of two cell clusters modeled by normal distributions. The dissimilarity $d(c_1, c_2)$ between the pair of clusters c_1 and c_2 is the symmetrized KL-divergence for normal distributions:

$$\begin{aligned} d(c_1, c_2) &= \frac{1}{2}(\boldsymbol{\mu_1} - \boldsymbol{\mu_1})^\top (\Sigma_1^{-1} + \Sigma_2^{-1})(\boldsymbol{\mu_1} - \boldsymbol{\mu_1}) \\ &\quad + \frac{1}{2}\mathrm{tr}(\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1 - 2\mathrm{I}). \end{aligned} \quad (1)$$

The symmetrized KL divergence is not a metric because it does not satisfy the triangleinequality. Nevertheless, it enables our goal of computing the dissimilarity between two probability distributions.

1.3 Dissimilarity between a pair of samples

In the model described in Section 1.1, a sample is characterized by a mixture of cell populations. We compute the dissimilarity between a pair of samples by optimally matching (in a graph-theoretic model) similar cell clusters and summing up the dissimilarities of the matched clusters. However, it is possible for a cell population from one sample either to be absent from another sample or to split into two or more cell populations in the second sample. These can happen due to biological reasons or due to errors in clustering. Thus it should be possible to match a cluster in one sample to zero, one, or more clusters in a second sample to compute the dissimilarity between the samples. We have developed a robust variant of a graph matching algorithm called the Mixed Edge Cover (MEC) algorithm that allows a cluster to be matched with zero or more clusters in the paired sample [2].

More formally, let S_1 and S_2 be two flow cytometry samples characterized by mixtures of n_1 and n_2 cell populations such that $S_1 = \{c_1^1, c_2^1, ..., c_{n_1}^1\}$, and $S_2 = \{c_1^2, c_2^2, ..., c_{n_2}^2\}$, where $c_i^j(\mu_i^j, \Sigma_i^j)$ is the i^{th} cluster from the j^{th} sample (here j=1 or 2). The mixed edge cover computes a correspondence $\texttt{mec}$, of clusters across S_1 and S_2 such that $\texttt{mec}(c_i^1) \in \mathcal{P}(S_2)$ and $\texttt{mec}(c_j^2) \in \mathcal{P}(S_1)$, where $\mathcal{P}(S_1)$ ($\mathcal{P}(S_2)$) is the power set of clusters in S_1 (S_2). When a cluster c_i^j remains unmatched, i.e., $\texttt{mec}(c_i^j) = \emptyset$, we set $d(c_i^j, -) = \lambda$ where the cost λ is a penalty for leaving a vertex unmatched in the mixed edge cover, and is set to a value such that the number of such clusters remains small. The cost of $\texttt{mec}$ is the sum of the dissimilarities of all pairs of matched clusters and the penalties due to the unmatched clusters. A minimum weight mixed edge cover is a mixed edge cover with the minimum cost. We use the cost of a minimum weight mixed edge cover as the dissimilarity $D(S_1, S_2)$ between a pair of samples S_1 and S_2:

$$D(S_1, S_2) = \quad (2)$$

$$\min_{\substack{\text{mixed edge} \\ \text{covers, mec}}} \frac{1}{2}\left(\sum_{\substack{c_k^2 \in \texttt{mec}(c_j^1) \\ 1 \le j \le n_1}} d(c_j^1, c_k^2) \; + \sum_{\substack{c_j^1 \in \texttt{mec}(c_k^2) \\ 1 \le k \le n_2}} d(c_j^1, c_k^2) \right),$$

where $d(c_j^1, c_k^2)$ is computed from Equation (1).

Our dissimilarity measure between a pair of samples can be compared with the partition distance (also called R-metric or transfer distance) that computes the minimum number of augmentations and removals of elements needed to transform one partition of a sample into another [12]. However, the partition distance compares two partitions (clusterings) of the same sample whereas our measure can work with partitions from the same sample or two different samples. In contrast to partition distance that matches a cluster to at most one cluster, MEC is able to match a cluster to zero or more clusters. The partitions distance does not accommodate the dissimilarities between elements in a natural way.

A mixed edge cover can be computed by a modified minimum weight perfect matching algorithm in $O(k^3 \log k)$ time where k is the maximum number of clusters in a sample [2]. The number of cell clusters (k) is usually small (fewer than fifty in typical experiments), and the dissimilarity between a pair of samples can be computed in seconds on a desktop computer.

2. CLASSIFYING SAMPLES WITH STATIC TEMPLATES

Given a collection of samples that can be grouped into a few representative classes, we build a template to describe each class. Different classes could represent different experimental conditions, disease status, time points, etc. Just as a sample is represented by a mixture of normal distributions corresponding to different cell populations, a template is characterized by a finite mixture of normal distributions

corresponding to distinct cell populations. Each component of a template is a meta-population or a meta-cluster, formed by combining cell populations expressing similar phenotypes in different samples. Hence a meta-cluster is characterized by a normal distribution, with parameters computed from the distributions of the clusters included in it. Note that clusters in a meta-cluster represent the same type of cells and thus have overlapping distributions in the measured marker space.

We build templates from a collection of samples by a hierarchical algorithm that repeatedly merges the most similar pair of samples or partial templates obtained by the algorithm thus far. The algorithm builds a binary tree called the *template-tree* denoting the hierarchical relationships among the samples. A leaf node of the template-tree represents a sample and an internal (non-leaf) node represents a template created from the samples. Figure 1 shows an example of a template-tree created from four hypothetical samples, S_1, S_2, S_3, and S_4. An internal node in the template-tree is created by matching similar cell clusters with the MEC algorithm across the two children and merging the matched clusters into meta-clusters. For example, the internal node $T(S_1, S_2)$ in Figure 1 denotes the template from samples S_1 and S_2. The mean vector and covariance matrix of a meta-cluster are computed from the means and covariance matrices of the clusters participating in the meta-cluster.

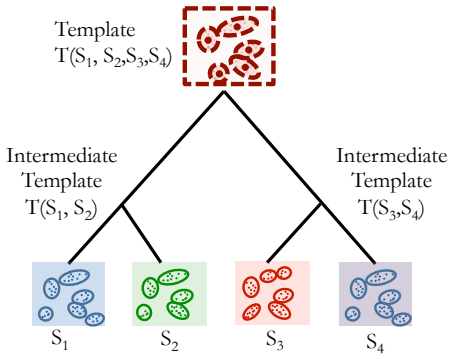

Figure 1: An example of a hierarchical template tree created from four hypothetical samples S_1, S_2, S_3 and S_4. A leaf node of the template-tree represents a sample and an internal (non-leaf) node represents a template created from its children in the tree. The children could be templates if they are interior nodes, or samples if they are leaves.

2.1 Algorithm to construct a template-tree

We designed a hierarchical matching-and-merging (HM&M) algorithm that builds a binary *template-tree* [3] from a given, fixed, set of n samples. A node in the tree represents either a sample (leaf node) or a template (internal node). In both cases a node is characterized by a finite mixture of multivariate normal distributions each component of which is a cluster or meta-cluster. Therefore, the dissimilarity of a pair of nodes can be computed by the mixed edge cover discussed in Section 1.3.

During the construction of a template-tree, a node is called an "orphan" if it does not have a parent. Such an orphan node could be one of the samples or one of the current templates. At each stage of the algorithm, the dissimilarity between each pair of orphan nodes is either computed with the MEC algorithm, or it is unchanged from the previous stage, and a pair of nodes (v_i, v_j) with the minimum dissimilarity is *merged* to form a new node v_k. The newly created node, v_k, represents a template and is assigned as the parent of v_i and v_j.

Let a node v_i consist of n_i clusters or meta-clusters c_1^i, c_2^i, ..., $c_{n_i}^i$. The HM&M algorithm can then be described in the following three steps:

1. *Initialization*: Create a node v_i for each of the n samples S_i. Define the set of orphan nodes, $\texttt{Orphan} = \{v_1, v_2, ..., v_n\}$. Then repeat the matching and merging steps until a single orphan node remains.

2. *Matching*: Compute the dissimilarity $D(v_i, v_j)$ between every pair of nodes v_i and v_j in the current $\texttt{Orphan}$ set with the mixed edge cover procedure described in Section 1.3.

3. *Merging:* Find a pair of orphan nodes, (v_i, v_j), with minimum dissimilarity $D(v_i, v_j)$. Create a new node $v_k = \{c_1^k, c_2^k, ..., c_{n_k}^k\}$ where each meta-cluster $c_z^k, 1 \le z \le n_k$, is formed by merging a group of matched clusters or meta-clusters, $\{c_x^i \cup \texttt{mec}(c_x^i)\}$; here x ranges over the metaclusters in sample S_i or subtemplate represented by node v_i. The distribution parameters, (μ_z^k, Σ_z^k), of each of the newly formed meta-clusters c_z^k are estimated by the EM algorithm. The height of v_k is set to $D(v_i, v_j)$. The node v_k then becomes the parent of v_i and v_j, and the set of orphan nodes is updated by deleting v_i and v_j from it and including v_k. If there are orphan nodes remaining, then we return to the matching step, and otherwise, we terminate.

2.2 Computational complexity of a template-tree

Initially we need to compute $O(n^2)$ dissimilarities for each pair of samples. Then the algorithm iterates $n-1$ times in order to create $n-1$ internal nodes. Let $|\texttt{Orphan}_i|$ be the number of orphan nodes at the i^{th} iteration. Then we need to compute $|\texttt{Orphan}_i|$ dissimilarity computations and a merge operation at the i^{th} iteration. Therefore we need a total $O(n^2)$ dissimilarity computations and $O(n)$ merge operations. Let k be the maximum number of clusters or meta-clusters in any of the nodes of the template-tree. Then a dissimilarity computation takes $O(k^3 \log k)$ time whereas the merge operation takes $O(kp)$ time when distribution parameters of the meta-clusters are computed by maximum likelihood estimation. Therefore, the total time complexity of the algorithm is $O(n^2 k^3 \log k)$, which is $O(n^2)$ for bounded k.

2.3 Creating static templates from a template-tree

The height of an internal node in the template-tree is measured by the dissimilarity between its left and right children. By recursion, a template denoted with a relatively lower internal node represents a relatively homogeneous collection of samples and vice versa. Let a collection of samples belong to m classes. After building a template tree, we can cut

the tree at a suitable height so that m disjoint subtrees are produced. The root of each subtree represents a template of the samples placed in the leaves of that subtree. The class (label) of a template is determined by the label of the majority of the samples in the subtree rooted at the template. In the special case of all samples belonging to the same class, a single template is generated from the root of the template tree. However, if the number of classes m is not known *a priori*, we select m by the number of well-separated branches based on the relative heights of the subtrees. The roots of these well-separated subtrees represent the class-templates, where within-class variations (heights of the subtrees) are small relative to the between-class variations (heights of the ancestors of the subtrees).

2.4 Classifying new samples with static templates

When the data set consists of samples belonging to m classes, we build m templates, T_1, T_2, ..., T_m, where the i^{th} template T_i represents samples of the i^{th} class. When we obtain a new sample S, we compute the dissimilarity $D(S, T_i)$ between S and every template T_i. The new sample is predicted to belong to the class whose template it is most similar (least dissimilar) to:

$$i^* = \underset{1 \le i \le m}{\operatorname{argmin}} D(S, T_i), \quad \text{class}(S) = \text{class}(T_{i^*}). \tag{3}$$

If the sample's dissimilarity with the closest template is above a threshold, then it is not similar to any of the class templates, and we need to create a new class for this sample. We address this issue in the next subsection. The template based classification is very fast because we need to compare a new sample only with m templates instead of all the other samples. The time complexity of a classification is therefore $O(mk^3 \log k)$, which is $O(m)$ for bounded k. This is faster than classifying the sample from scratch, since the n^2 factor from the number of samples in the complexity is reduced to the number of templates, m.

3. BUILDING DYNAMIC TEMPLATES

3.1 The algorithm

The static template based classification method works well in practice, but it has two limitations. First, the algorithm needs to see all the samples in the training set before constructing the templates. However, frequently samples arrive sequentially or in batches, as for instance in a longitudinal study of an epidemic. Second and more important, the algorithm builds static templates since it does not update templates as new samples are classified. Therefore, future classification can not use the information gained from samples classified after the building of the static template tree.

To address the aforementioned limitations we update the templates dynamically. When a new sample arrives, we classify it *and* insert it into the current template-tree so that future samples can be classified on the updated templates. This approach also works when we do not have any training dataset to start with. In that case we build the template-tree as the samples are available in a dynamic fashion starting with empty templates. Consider an existing template-tree `TT` (possibly empty) with r as the root node. Note that r can be considered as the template of all samples in the leaves of the whole tree. In order to insert a new sample S in `TT`, we first create a singleton node v from S. If `TT` is an empty tree we make v the root of the template tree, and otherwise, we insert v into the tree `TT` by invoking the procedure `insert` shown in Figure 2 with r and v as the parameters.

The procedure `insert` works in a recursive fashion. It follows a path from the root to a node (a leaf or internal node), to be identified by the algorithm, where the new node v is inserted. The procedure then backtracks by updating the mixtures of the internal nodes found in the return path back to the root. We consider four cases while inserting v in a subtree rooted at u. The cases are illustrated in Figure 3. In the first case u is a leaf node, and we create a new node w and make u and v the children of w. We create a template from the samples in the leaves u and v and save it in node w. In the other cases u is an internal node. Let u_l and u_r be the left and right children of u, respectively. We compute dissimilarities $D(u_l, u_r)$, $D(u_l, v)$ and $D(u_r, v)$ between each pair of nodes from u_l, u_r, and v. If $D(u_l, u_r)$ is the smallest among the three dissimilarities, then v cannot be inserted in a subtree rooted at u. Thus we create a new node w and make u and v the children of w. We create a new template from the template u and sample v, save it in node w and return. When $D(u_l, v)$ is the smallest dissimilarity, we insert v in a subtree rooted at u_l by calling the procedure `insert` with u_l, v as parameters. In this case the left subtree of u gets updated. Similarly, if $D(u_r, v)$ is the smallest then v is inserted in the right subtree rooted at u_r.

3.2 Computational complexity

To insert a new sample, we need to traverse a path starting from the root to a leaf or an internal node in a template tree. In the worst case the length of the traversed path is the height of the template-tree. Let n be the number of samples and h be the height of a template tree where $(n-1) \le h \le \log_2(n)$. The former equality holds when the tree is completely unbalanced (a chain) whereas the latter equality is satisfied when the tree is balanced. At each node in the traversed path we need to compute three dissimilarities and one update operation (when backtracking). A dissimilarity computation takes $O(k^3 \log k)$ time whereas an update operation takes $O(k^3 \log k) + O(kp)$ time when distribution parameters of the meta-clusters are computed by the maximum likelihood estimation. Thus the time complexity of inserting a sample in a template tree is $O(hk^3 \log k)$.

3.3 Classifying a sample

To classify a new sample S, we first insert it into the current template tree. The class of S is predicted to be the class of the template created from the subtree where S is inserted. At the time of insertion the template-tree is dynamically updated to reflect the information gained from the new sample. The dynamic template approach is especially useful in unsupervised classification where the class labels of the samples are not known in advance. In that case, the class-templates are created from the well-separated subtrees such that within-class variations (heights of the subtrees) are small relative to the between-class variations (heights of the ancestors of the subtrees). In this context, the algorithm is similar to the spirit of the hierarchical clustering algorithm UPGMA, with significant differences in the distance computation and management of the internal nodes. Furthermore,

```
1: procedure insert(u, v)                                   ▷ Insert leaf node v in the subtree rooted at u
2:     if u is a empty then                                 ▷ Inserting in an empty tree
3:         return v
4:     end if
5:     if u is a leaf then                                  ▷ Case 1
6:         w ← empty node,  w_l ← u,  w_r ← v,  x ← w
7:     else
8:         D ← min{D(u_l, u_r), D(u_l, v), D(u_r, v)}
9:         if D(u_l, u_r) = D then                          ▷ Case 2
10:            w ← empty node,  w_l ← u,  w_r ← v,  x ← w
11:        else if D(u_l, v) = D then                       ▷ Case 3
12:            u_l ← insert(u_l, v),  x ← u
13:        else                                             ▷ when D(u_r, v) = D,  Case 4
14:            u_r ← insert(u_r, v),  x ← u
15:        end if
16:    end if
17:    update node x by matching and merging meta-clusters from x_l and x_r   ▷ Update node
18:    height(x) = D(x_l, x_r)
19:    return x                                             ▷ Going up in the tree
20: end procedure
```

Figure 2: Inserting a leaf node (sample) v in a subtree rooted at u (template or sample)

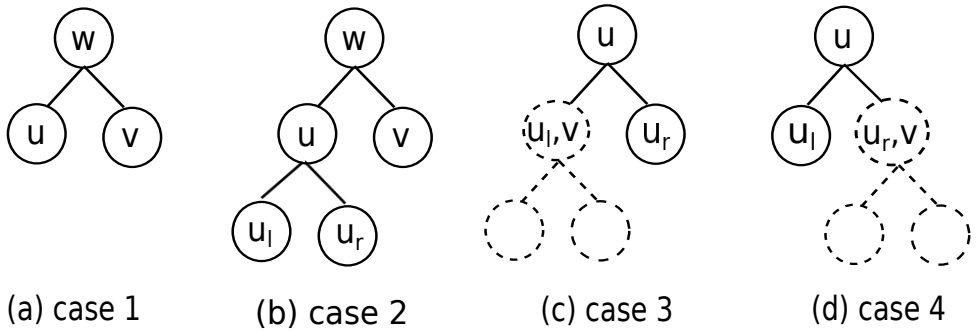

Figure 3: Four cases to consider when inserting a leaf node v into the subtree rooted at u. (a) case 1 (u is also a leaf): a new internal node w is created and is made the parent of u and v, (b) case 2 (u is a non-leaf and the left and right children of u are more similar to each other than to v): a new internal node w is created and is made the parent of u and v, (c) case 3 (u is a non-leaf and the left child u_l of u is more similar to v than to u_r): insert v into the subtree rooted at u_l by calling insert(u_l, v), and (d) case 4 (u is a non-leaf and the right child u_r of u is more similar to v than to u_l): insert v into the subtree rooted at u_r by calling insert(u_r, v). The dotted parts in Subfigure (c) and (d) are determined by the insert function in a recursive fashion.

when a new sample is highly dissimilar to every existing template, the algorithm automatically creates a new branch in the tree indicating a new class. This approach therefore has the ability to discover unknown classes from the incoming samples, which, for example, is very useful in detecting new stains of a disease.

How sensitive is the template tree to the order in which the samples are inserted? Recall that we compute the template tree by merging the most similar samples or sub-templates from the samples available in the training set. Our experience is that if the between-class variation is significantly higher than the within-class variation (as is the case in the two datasets studied in this paper), the classification accuracy is unaffected by the small differences in the subtrees of the template tree. We omit detailed results due to space limitations.

4. RESULTS

4.1 Classifying stimulation status of T cells

We reanalyze the T Cell Phosphorylation (TCP) data from Maier et al. to determine differences in phosphorylation events downstream of T cell receptor activation in naive and memory T cells [17]. For each of the 29 subjects in this study, whole blood was stained using labeled antibodies against CD4, CD45RA, SLP-76 (pY128), and ZAP-70 (pY292) protein markers before stimulation with an anti-CD3 antibody, and another aliquot underwent the same staining procedure five minutes after stimulation. During the stimulation anti-CD3 antibody binds with T Cell Receptors (TCR) and activates the T cells, initiating the adaptive immune response. The binding with TCR induces dramatic changes in gene expression and cell morphology, and induces the formation of a phosphorylation-dependent signaling network via multi-protein complexes. ZAP-70 is a kinase that phosphorylates tyrosine in a trans-membrane protein called LAT, and LAT

and SLP-76 are part of a platform that assembles the signaling proteins [5].

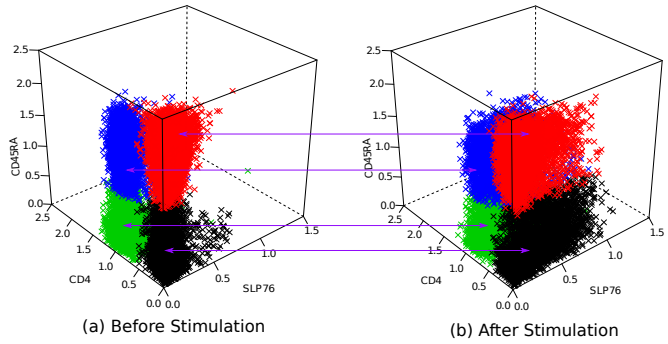

Figure 4: Three dimensional projections of marker expressions for a pair of samples from (a) before anti-CD3 stimulation and (b) after anti-CD3 stimulation. Each sample is clustered independently to identify cell populations. The clusters are then matched to show the effect of stimulation on different cell subsets. Identical colors indicate matched cell populations.

By using the clustering algorithm we have identified four cell clusters in each of the 58 samples (29 pairs). Two of these clusters represent memory ($CD4^+$ $CD45RA^{low}$) and naive ($CD4^+$ $CD45RA^{high}$) T cell subsets. Cell clusters are then matched across stimulation using the MEC algorithm to register the same cell type. The stimulated cells show increased levels of SLP-76 as expected. (We also see increased levels of ZAP-70, though it is not shown in the Fig.) For visualization purposes, we plot a three dimensional projection of a pair of samples in Figure 4 where four cell clusters are shown in different colors. We used same color to denote matched cell clusters. By comparing the matched clusters we can clearly see increased levels of SLP-76 and ZAP-70.

The 58 samples in this study group nicely into two distinct classes: pre-stimulation and post-stimulation. For a pair of samples from the i^{th} subject, we denote the unstimulated sample by $i-$ and the stimulated sample by $i+$. We first apply the static template-based classification to demonstrate how the classification works for this dataset. We divide the samples into a training set and a test set. By using the HM&M algorithm, we build a template-tree from the training set and create two class templates from the left and right children of the root. An example is shown in Figure 5 where the training set contains 6 pairs of samples. (Again, the plots of the three proteins are shown for visualization only; classification is performed using all of the protein markers.)

We cut the tree beneath the root and create two templates T_{pre} and T_{post} for the two classes of samples. The i^{th} sample S_i in the test set is predicted to come from the pre-stimulation class when S_i is more similar to T_{before} than to T_{after} (i.e., $D(S_i, T_{\text{before}}) < D(S_i, T_{\text{after}})$), and otherwise, from the post-stimulation class. In Figure 5, we show a sample from the test set above the two templates. The algorithm correctly classifies it as a pre-stimulation sample, and the correctness of the classification can be seen from the visual inspection of the sample, since it looks similar to the pre-stimulation template and does not show a phosphorylation shift in SLP-76.

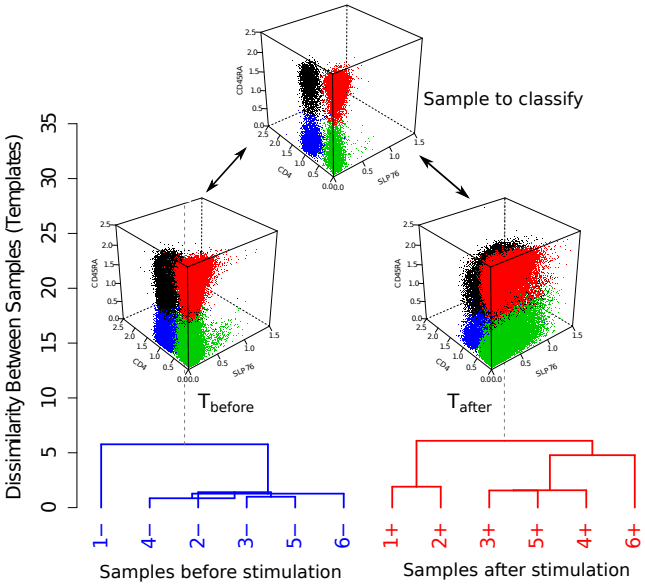

Figure 5: Sample classification based on static templates. The HM&M algorithm creates two template, T_{before} for before-stimulation and T_{after} for after stimulation classes, from 6 pairs of samples in the TCP dataset. A new sample is compared with the templates, and classified with the template it is most similar to.

We study the accuracy of the template based classifier by using cross-validation for this dataset. At each stage of the cross-validation, we create a test set from ten samples and a training set from the remaining 48 samples. After creating templates from the training set, we predict the class of each sample in the test set by comparing it with the templates. A sample is considered to be misclassified when the predicted class is different from the actual class. We repeat this process 58 times for different collections of training and test sets and compute the fraction of misclassified samples. We observed that three pre-stimulation samples, 9-, 10- and, 11-, were consistently classified with the after-stimulation class whenever they were present in the test set. No other sample is classified into a different class in the cross-validation. We consider these three samples as outliers, show that they are likely to have been pre-stimulated, and discuss their properties further in Section 4.1.2.

4.1.1 Classification with dynamic templates

In order to demonstrate our classification approach based on dynamic templates, we build a template-tree incrementally from the samples in the TCP dataset. We start with an empty tree and insert the samples one after another into the current tree by using the procedure described in Section 3. The complete template tree after inserting all samples is shown in Figure 6. In this tree, we draw a subtree consisting of samples from pre-stimulation in blue and from post-stimulation in red. Aside from three outlying samples, all samples create two well-separated branches of the root de-

noting the pre- and post-stimulation templates. The height of an internal node in a template-tree is measured by the dissimilarity between the pair of samples (templates) denoted by the left and right children of the internal node. The height of the root in Figure 6 is more than twice of the height of any other node. Hence the algorithm successfully identifies two templates with small within-template deviation while maintaining a clear separation between them.

A new sample S is inserted into the existing template-tree by following the procedure described in Figure 2. At the time of insertion the template-tree is dynamically updated to reflect the information gained from the new sample. After insertion, the position of S in the tree determines its predicted class. We classify S as a pre-stimulation sample when it is placed in the left (blue) subtree and otherwise, classify it as a post-stimulation sample. Similar to the classification with static templates, we observe that all samples except 9-, 10- and 11-, are correctly classified.

4.1.2 *Outlying samples*

Now we discuss the three outlying pre-stimulation samples, 9-, 10- and 11-, which are consistently classified with the post-stimulation samples by both the static and dynamic classification algorithms. We show the dissimilarity between every pair of samples in the dataset in a level plot in Figure 7, where a square is drawn in a lighter shade when a pair of samples is similar, and in darker shade when a pair of samples is highly dissimilar. We observe that most squares in the top-left and bottom-right quadrants are in light colors indicating similarity among samples within pre- and post-stimulation classes. However, three pre-stimulation samples, 9-, 10- and 11-, are more similar to the post-stimulation samples than to the pre-stimulation samples, suggesting that these samples were in a stimulated state prior to the experiment, and had higher levels of the SLP-76 and ZAP-70 proteins. Consequently they are classified with the post-stimulations samples by the dynamic templates-based classification algorithm presented here. Thus it is clear that the classification algorithm is working correctly in this instance with the data input to it.

4.2 Classifying replicated samples from individuals

We further validated our approach by a "biological simulation" where peripheral blood mononuclear cells (PBMC) were collected from five healthy subjects, and each sample was divided into five parts and analyzed through a flow cytometer. Thus we have five technical replicates for each subject, and each replicate was stained using labeled antibodies against CD45, CD3, CD4, CD8, and CD19 protein markers. Each sample was compensated for spectral overlap across channels, and lymphocytes ($CD45^+$) were identified using forward and side scatter data. We then use a cell population identification algorithm for each sample separately, and identified four cell types denoting (a) helper T cells ($CD3^+CD4^+$), (b) cytotoxic T cells ($CD3^+CD8^+$), (c) B cells ($CD3^-CD19^+$), and (d) natural killer cells ($CD3^-CD19^-$). Figure 8 shows the bivariate projections of the four clusters in a representative sample where different colors are used to denote the cell types. Note that every one of these cell types is $CD45^+$ because we pre-selected lymphocytes.

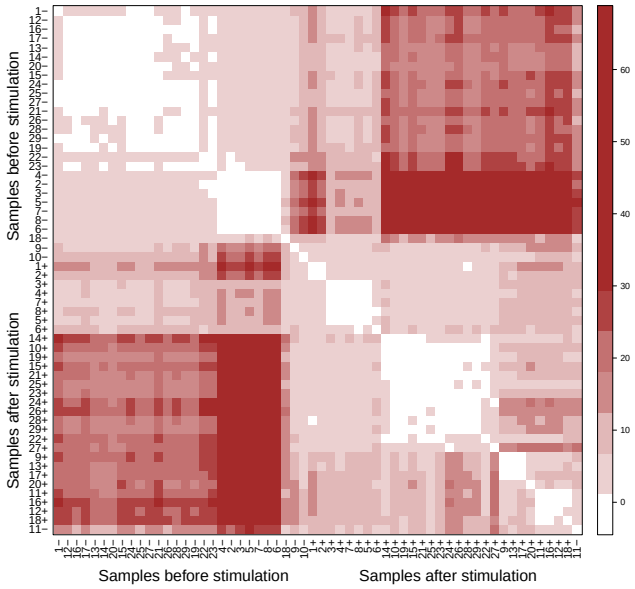

Figure 7: The level plot showing the dissimilarity between every pair of samples in the TCP dataset where the color of a square corresponds to the dissimilarity of a pair of samples. A square is drawn in a light shade when the pair of samples is similar, and in a dark shade when the pair of samples is highly dissimilar. The plot is symmetric about the main diagonal.

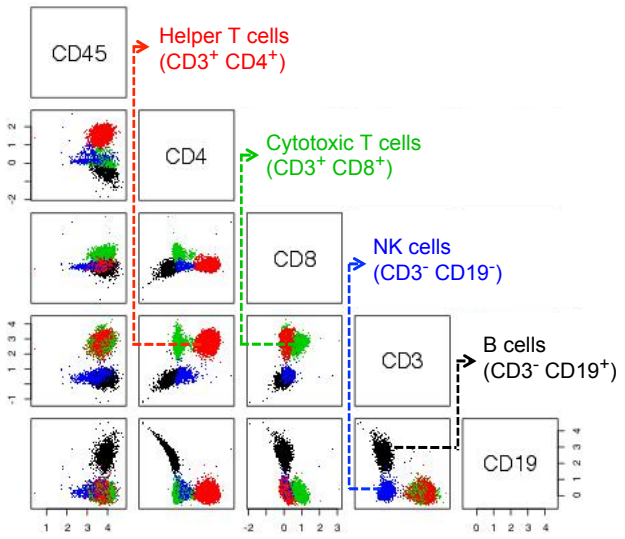

Figure 8: Bivariate projections of four clusters in a sample from the healthy donor dataset. Each cell cluster is $CD45^+$ since we pre-selected lymphocytes on the forward and side scatter channels. Four cell types are shown in red (helper T cells), green (cytotoxic T cells), blue (natural killer cells) and black (B cells).

After clustering each sample, we build a template tree from the 25 samples in this dataset. We start with an empty tree and insert the samples sequentially into the current

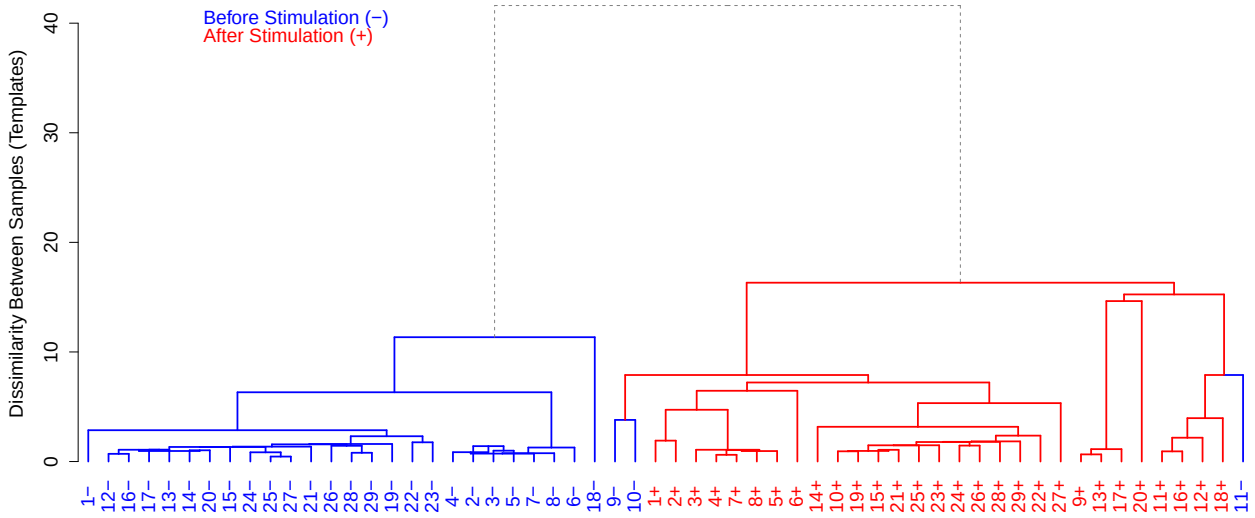

Figure 6: A dynamic template tree created incrementally by adding the samples in the TCP dataset one after another. Minus and plus signs are appended with the subject number in the to indicate samples from pre- and post-stimulation. A subtree consisting of samples from pre-stimulation and post-stimulation classes are shown in blue and red colors. The height of a node measures the dissimilarity between the left and right children, whereas the horizontal placement of a sample is arbitrary.

tree by using the procedure described in Section 3. The complete template tree is shown in Figure 9, where we see that samples from the five subjects create five well-separated branches. We can therefore construct five templates from the roots of the five well separated branches where each template denotes a summary representation of all samples from a subject.

Intuitively we expect samples from each subject to be classified together. Here, the within-subject variations among five replicates of a subject come from the technical variations in flow cytometry sample preparation and measurement, whereas between-subject variations come from the natural biological variations in the healthy subjects. In this dataset we observe more natural biological variation than the technical and instrumental variations. We visualize these variations with a levelplot in Figure 10. where the color of a square indicates the dissimilarity of a pair of samples. A square is drawn in a light shade when a pair of samples is similar, and with a dark shade r when they are highly dissimilar. We observe that samples from a subject are always more similar to each other (5×5 squares along the diagonal with light colors) than they are across subjects (off diagonal squares). For this reason, samples from a subjects stay together in the template-tree in Figure 9.

The observations from the healthy donor dataset confirm that we can build immune profiles for individuals despite within-subject variations from technical replicates. Additionally, the five templates from the five subjects create another level of hierarchy and the root of the tree in Figure 9 is considered as a template from all healthy individuals. This combined template represents a healthy immune profile by preserving the common features of healthy individuals and by removing between subject variations. Such a healthy template can be compared against templates created from diseased samples in order to diagnose diseases and to perform comparative study of healthy and diseased immune profiles.

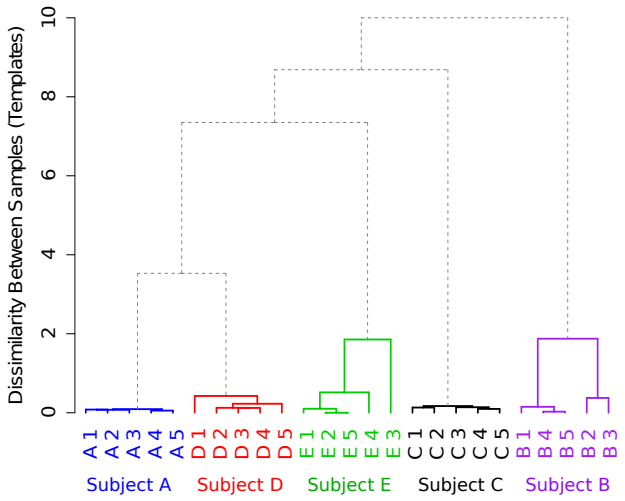

Figure 9: A dynamic template tree created incrementally from samples in the healthy donor dataset. The algorithm identifies five well separated branches denoting templates for the five subjects. Subtrees consisting of replicates from five subjects are shown in five different colors. The height of a node measures the dissimilarity between its left and right children.

5. CONCLUSIONS

We developed template based classification methods for flow cytometry samples displaying different immunophenotypes. A template built from samples of a class provides a concise

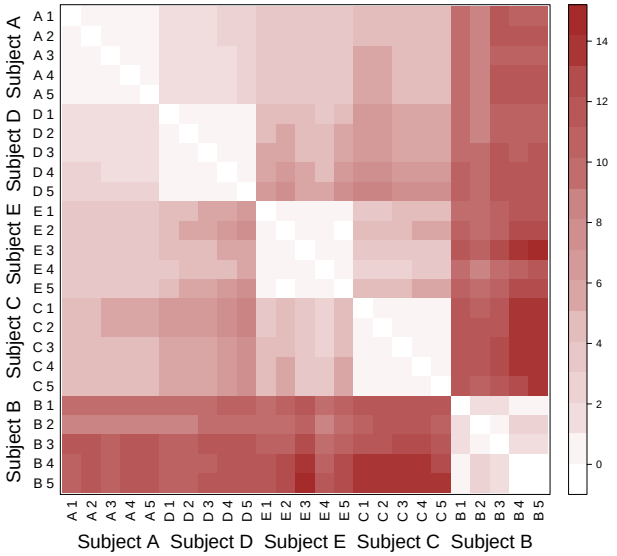

Figure 10: Level plot showing the dissimilarity between every pair of samples in the healthy donor dataset, where the color of a square corresponds to the dissimilarity between a pair of samples. A square is denoted with a light shade when the pair of samples is similar and with a dark shade when the pair of samples is highly dissimilar. The plot is symmetric about the diagonal.

description of the class by emphasizing the key characteristics while masking statistical noise and low-level details, and thus helps to measure overall changes in cell populations across different conditions. By moving beyond sample-specific variations, the templates act as the blueprints for different classes and can be used to classify future samples to different classes in a more relevant parameter space. It is also more efficient to classify a sample using templates rather than all of the previously seen samples. We also maintain a hierarchy of the samples in a template tree such that samples can be analyzed in higher resolution whenever necessary. As new samples come in, the templates are dynamically updated to reflect the information gained from them. This is a desirable property in dynamic siutations, as in the course of an epidemic, when new samples are being collected and analyzed.

In continuing work, we plan to investigate the use of networks instead of trees to organize the templates, similar in spirit to the use of networks rather than trees in phylogenetics [10]. Another issue is that the combinatorial dissimilarity measure between two samples is not a metric, and when the dissimilarity is extended to two templates, this value does not monotonically increase in the hierarchical matching and merging algorithm. Finally, dynamic classification is a critical step towards characterizing diverse states of the human immune system from big datasets of samples collected at geographically distributed laboratories, e.g., the Human Immunology Project Consortium (`www.immuneprofiling.org`). Our work makes it possible to summarize the data from each laboratory using templates for each class, and then to merge the templates and template trees across various laboratories, as the data is being continuously collected, summarized and analyzed.

Acknowledgments

This work was partially supported by NIH grant IR21EB015707, NSF grant CCF-1218916, DOE grant FC02-08ER25864, and an IBM Fellowship to Ariful Azad. SP thanks DBT, MoS&PI and DST, India, for support.

6. REFERENCES

[1] N. Aghaeepour, G. Finak, H. Hoos, et al. Critical assessment of automated flow cytometry data analysis techniques. *Nature Methods*, 10(3):228–238, 2013.

[2] A. Azad, J. Langguth, Y. Fang, A. Qi, and A. Pothen. Identifying rare cell populations in comparative flow cytometry. *Lecture Notes in Computer Science*, 6293:162–175, 2010.

[3] A. Azad, S. Pyne, and A. Pothen. Matching phosphorylation response patterns of antigen-receptor-stimulated T cells via flow cytometry. *BMC Bioinformatics*, 13(Suppl 2):S10, 2012.

[4] A. Azad, B. Rajwa, and A. Pothen. Homogeneous meta-clustering in flow cytometry by variance stabilization. *Manuscript under preparation*, 2013.

[5] C. Briockmeyer, W. Paster, D. Pepper, et al. T Cell Receptor (TCR)-induced Tyrosine phosphorylation dynamics identifies THEMIS as a new TCR signalosome component. *Journal of Biological Chemistry*, 286(9):7535–7547, 2011.

[6] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(10):1042–1052, 1993.

[7] S. D. Connell and A. K. Jain. Template-based online character recognition. *Pattern Recognition*, 34(1):1–14, 2001.

[8] M. De Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. Van Compernolle. Template-based continuous speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(4):1377–1390, 2007.

[9] L. Deng, H. Strik, et al. Structure-based and template-based automatic speech recognition-comparing parametric and nonparametric approaches. In *Proceedings of Interspeech*, pages 898–901, 2007.

[10] A. Dress, K. T. Huber, J. Koolen, V. Moulton, and A. Spillner. *Basic Phylogenetic Combinatorics*. Cambridge University Press, 2012.

[11] G. Finak, J. Perez, A. Weng, and R. Gottardo. Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics*, 11(1):546, 2010.

[12] D. Gusfield. Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82(3):159–164, 2002.

[13] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: Finding the optimal partitioning of a data set. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 187–194. IEEE, 2001.

[14] T. Lin. Robust mixture modeling using multivariate skew t distributions. *Statistics and Computing*, 20(3):343–356, 2010.

[15] K. Lo, R. Brinkman, and R. Gottardo. Automated gating of flow cytometry data via robust model-based clustering. *Cytometry Part A*, 73(4):321–332, 2008.

[16] E. Lugli, M. Roederer, and A. Cossarizza. Data analysis in flow cytometry: The future just started. *Cytometry Part A*, 77(7):705–713, 2010.

[17] L. Maier, D. Anderson, P. De Jager, L. Wicker, and D. Hafler. Allelic variant in ctla4 alters T cell phosphorylation patterns. *Proceedings of the National Academy of Sciences*, 104(47):18607–18612, 2007.

[18] S. P. Perfetto, P. K. Chattopadhyay, and M. Roederer. Seventeen-colour flow cytometry: unravelling the immune system. *Nature Reviews Immunology*, 4(8):648–655, 2004.

[19] J. M. Peters and M. Q. Ansari. Multiparameter flow cytometry in the diagnosis and management of acute leukemia. *Archives of Pathology & Laboratory Medicine*, 135(1):44–54, 2011.

[20] S. Pyne, X. Hu, K. Wang, et al. Automated high-dimensional flow cytometric data analysis. *Proceedings of the National Academy of Sciences*, 106(21):8519–8524, 2009.

[21] R. A. Seder, P. A. Darrah, and M. Roederer. T-cell quality in memory and protection: implications for vaccine design. *Nature Reviews Immunology*, 8(4):247–258, 2008.

[22] H. M. Shapiro. *Practical Flow Cytometry*. Wiley-Liss, 2005.