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

We describe an algorithm to dynamically classify flow cytometry data samples into several classes based on their immunophenotypes. Flow cytometry data consists of fluorescence measurements of several proteins that characterize different cell types in blood or cultured cell lines. 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. We dynamically update the templates and their statistical parameters as new samples are classified, so that the new information is reflected in the classes. We use our dynamic classification algorithm to classify T cells that on stimulation with an antibody show increased abundance of the proteins SLP-76 and ZAP-70. These proteins are involved in a platform that assembles signaling proteins in the immune response. We also use the algorithm to show that variation in an immune subsystem between individuals is a larger effect than variation in multiple samples from one individual.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and Genetics*; G.2 [**Discrete Mathematics**]: Graph Algorithms

General Terms

Flow Cytometry, Immunology, Classification, Clustering, Templates, Template Tree

BCB '13, September 22 - 25, 2013, Washington, DC, USA

1. INTRODUCTION

We describe an algorithm to dynamically classify flow cytometry samples into several classes based on the cell populations present in the samples. Since we expect that most of the readers of this paper are unfamiliar with flow cytometry, we begin with a brief discussion of this technology before describing our contributions in more detail.

Flow cytometry (FC) is a platform for profiling the changes in the phenotype, functional response, or physiology, of cells due to various stimuli. The phenotype is measured in FC by quantifying the abundances of multiple protein complexes (called Clusters of Differentiation, CD). Antibodies conjugated with fluorophores are used to target marker proteins on the surface of or within 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 [23], 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 [19], to study how the immune system responds at multiple levels to the presence of a pathogen, to clinically diagnose diseases of the immune system [20], and to develop vaccines (e.g., HIV) [22].

In conventional FC practice, cell populations are identified by visualizing cells by a collection of two-dimensional scatter plots as shown in Fig. 8 (in Section 5). The reader will find it helpful to refer to this Figure now to follow the rest of the discussion. Fig. 8 shows how four types of immune cells are identified using five CD protein complexes. (In FC terminology, '+' and 'high' both indicate higher abundances of a marker, and '−' and 'low' indicate lower levels of it.) This approach of using a collection of biaxial plots assumes that the axes are orthogonal to each other; however, often there are correlations between these protein markers, and these are difficult to analyze using biaxial projections. To address this problem, many automated data clustering algorithms to identify cell populations in multi-dimensional 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 register cell populations in a large collection of samples, and to classify samples according to the presence, location in marker space, and shape of cell populations. The cell populations with their statistical descriptions in each sample are identified with a clustering algorithm, and we organize samples into a few classes using these descriptions. Mathematically each cell population in a sample is represented by a normal distribution, and a sample is represented by a mixture of normal distributions corresponding to different cell populations.

Samples belonging to the same class are summarized by a template, which is a summary of the sample's expression pattern [3, 11, 21]. The concept of cell 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. Each meta-cluster is formed by combining cell populations expressing similar phenotypes in different samples. Hence mathematically a meta-cluster is characterized by a normal distribution, with parameters computed from the distributions of the clusters included in it. Clusters in a meta-cluster represent the same type of cells and thus have overlapping distributions in the marker space.

A *template* is a collection of relatively homogeneous meta-clusters 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. Mathematically a template is characterized by a finite mixture of normal distributions.

We summarize these concepts in Table 1. Given the inter-sample variations due to innate biological variability 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, 21]. We show that the use of templates leads to efficient classification algorithms.

Terms	meaning
Cell population (cluster)	a group of cells expressing similar features, e.g., helper T cells, B cells
Sample	a collection of cell populations within a single biological sample
Meta-cluster	a set of biologically similar cell clusters from different samples
Template	a collection of meta-clusters from samples of same class

Table 1: Summary of terminology used in this paper.

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. Fig. 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 across the two children and merging the matched clusters into meta-clusters. For example, the internal node $T(S_1, S_2)$ in Fig. 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.

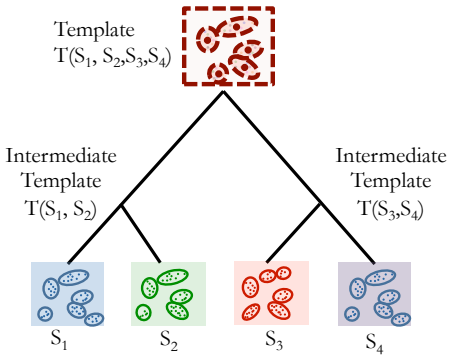

Figure 1: An example of a hierarchical template tree created from four hypothetical samples S_1, S_2, S_3 and S_4. A leaf node of the template tree represents a sample and an internal node represents a template created from its children.

Besides their use in high-level visualization and cross-class comparisons, templates can be employed 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 organize them into a template tree data structure. A new sample is then predicted to come from a class whose template it is most similar to. In the dynamic classification approach, we update the templates and also the template tree, as new samples are classified.

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 immune cells in five healthy individuals. Five technical replicates were created from each subject, and we show that these replicates are correctly classified to the corresponding template for the individual, and that technical variations are relatively small when compared with the biological variation 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. An unclassified image is compared 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 a feature matching algorithm. Our approach has similarities to these methods in principle but differs from them significantly in how the templates are created, represented, and compared with incoming samples. In contrast to these approaches, we maintain a hierarchy of the training samples in order to use their relationships in future classification. 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 classification.

FC data is continuous, high-dimensional (each column corresponds to a fluorescence from a protein being measured), perturbed by Poisson and Gaussian noise, and provides information at the single cell level for millions of intact cells in a sample. Although microarray data is similar in that it is continuous and based on fluorescence, we cannot analyze FC data using established methods for analyzing microarray data. 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. In microarrays, the expression data is the average of a large number of cells in a population, whereas in FC, each cell is measured individually, and a distribution of the expression profile within a cellular population is observed. Hence the nature of the data, its pre-processing, statistical treatment, and algorithms for downstream analysis, are all substantially different for the two technologies.

2. BACKGROUND

2.1 Identifying cell populations

An FC sample measuring the abundances 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. This data needs to be pre-processed by spectral unmixing, gating to identify cell populations of interest (e.g., lymphocytes), and a variance-stabilizing transformation (commonly a generalized logarithm). Unmixing recovers the correct abundances of a protein marker by removing the contributions of other markers at the observed frequency of the fluorescence [18]. Stabilizing the variances of the cell populations, i.e., making the variance independent of the value of the mean, is a step needed for correct statistical analysis using ANOVA methods, but this will be discussed in another publication [4].

Each sample consists of several clusters of cells, with each cluster of cells expressing a single phenotype in the measured marker space. 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. 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.

2.2 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 triangle inequality. Nevertheless, it enables our goal of computing the dissimilarity between two probability distributions.

2.3 Dissimilarity between a pair of samples

In the model described in Section 2.1, 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 FC 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 `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. We select λ empirically by plotting the total number of unmatched clusters in all pairwise matchings of samples against λ, and finding a "knee" in the curve [2]. The cost of `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:

$$D(S_1, S_2) = \min_{\substack{\text{mixed edge} \\ \text{covers, mec}}} \frac{1}{2}\Big(\sum_{\substack{c_k^2 \in \texttt{mec}(c_j^1) \\ 1 \le j \le n_1}} d(c_j^1, c_k^2) + \sum_{\substack{c_j^1 \in \texttt{mec}(c_k^2) \\ 1 \le k \le n_2}} d(c_j^1, c_k^2)\Big), \quad (2)$$

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 or 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. Partition 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 typically small (fewer than fifty in experiments), and the dissimilarity between a pair of samples can be computed in seconds on a desktop computer.

3. CLASSIFYING SAMPLES WITH STATIC TEMPLATES

3.1 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 2.3.

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 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 consists of the 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 `Orphan` set with the mixed edge cover procedure described in Section 2.3.

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 meta-clusters 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.

3.2 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 totally we need $O(n^2)$ 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. Hence the time complexity of the algorithm is $O(n^2 k^3 \log k)$, which is $O(n^2)$ for bounded k.

3.3 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 by 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).

3.4 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 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.

4. BUILDING DYNAMIC TEMPLATES

4.1 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 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 Fig. 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 Fig. 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.

4.2 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 $O(hk^3 \log k)$.

4.3 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, 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 strains 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. We find 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 the results due to space limitations.

5. RESULTS

```
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)

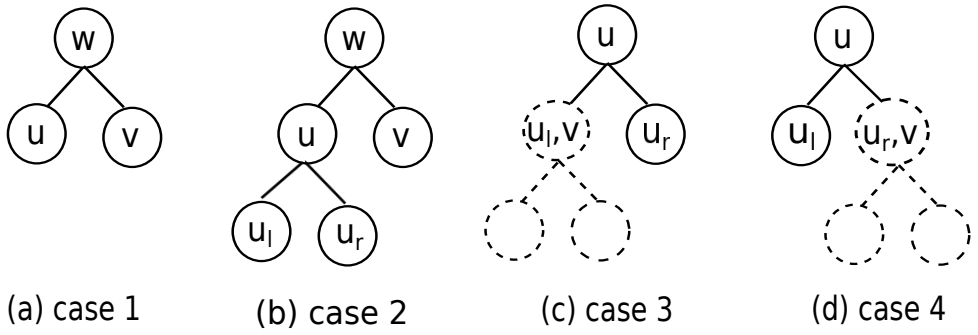

Figure 3: Four cases to consider when inserting a leaf node v into the subtree rooted at u. (a) case 1 (u is also a leaf): a new internal node w is created and is made the parent of u and v, (b) case 2 (u is a non-leaf and the left and right children of u are more similar to each other than to v): a 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 the right child 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 the left child 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.

5.1 Classifying stimulation status of T cells

We analyze a T cell phosphorylation (TCP) data set from Maier et al. [17] to determine differences in phosphorylation events downstream of T cell receptor activation in naive and memory T cells. 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].

By using the clustering algorithm we have identified four cell populations in each of the 58 samples (29 pairs). Two of these populations represent memory ($CD4^+$ $CD45RA^{low}$) and naive ($CD4^+$ $CD45RA^{high}$) T cell subsets. (Recall that '+' and 'high' indicate higher abundances of a marker, and '−' and 'low' indicate lower levels of it.) Cell populations 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. For visualization purposes, we plot a three dimensional projection of a pair of samples in Fig. 4 where four cell clusters are shown in different colors. We used the same color to denote matched cell populations. By comparing the matched clusters we can clearly see increased levels of SLP-76 (and ZAP-70, although this is not included in the Fig.)

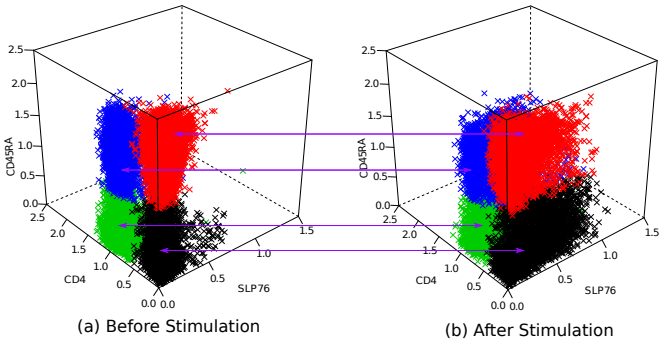

Figure 4: Three dimensional projections of marker expressions for a pair of samples from (a) before anti-CD3 stimulation and (b) after anti-CD3 stimulation. Each sample is clustered independently to identify cell populations. The clusters are then matched to show the effect of stimulation on different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

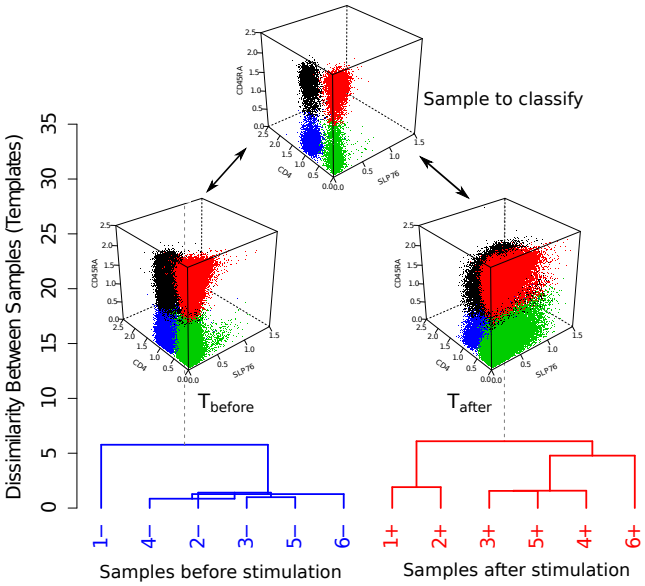

Figure 5: Sample classification based on static templates. The HM&M algorithm creates two templates, T_{before} for before-stimulation and T_{after} for after stimulation classes, from six pairs of samples in the TCP dataset. A new sample is compared with the templates, and is classified with the template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

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 Fig. 5 where the training set contains six pairs of samples. (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 (based on the relative heights of the subtrees) and create two templates T_{before} and T_{after} 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 Fig. 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.

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 stimulated before the experiment, and discuss their properties further in Sec. 5.1.2.

5.1.1 *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 Sec. 4. The complete template tree after inserting all samples is shown in Fig. 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 denoting 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 Fig. 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 Fig. 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

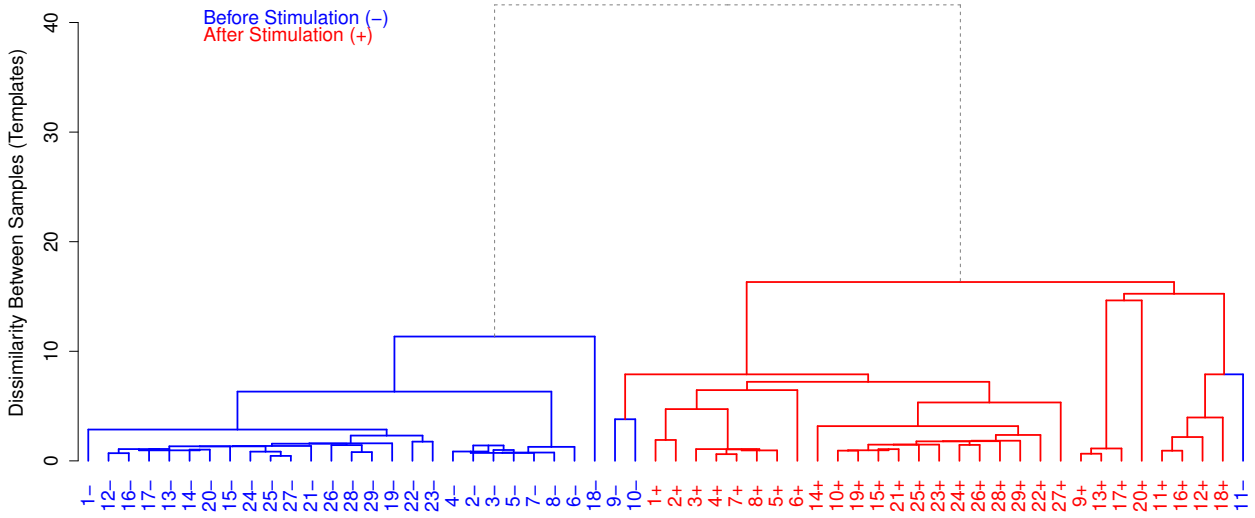

Figure 6: A dynamic template tree created incrementally by adding the samples in the TCP dataset one after another. Minus and plus signs are appended to the subject number to indicate pre- and post-stimulation samples. Pre-stimulation samples are in blue, and post-stimulation samples are red. The height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

with static templates, we observe that all samples except 9-, 10- and 11-, are correctly classified.

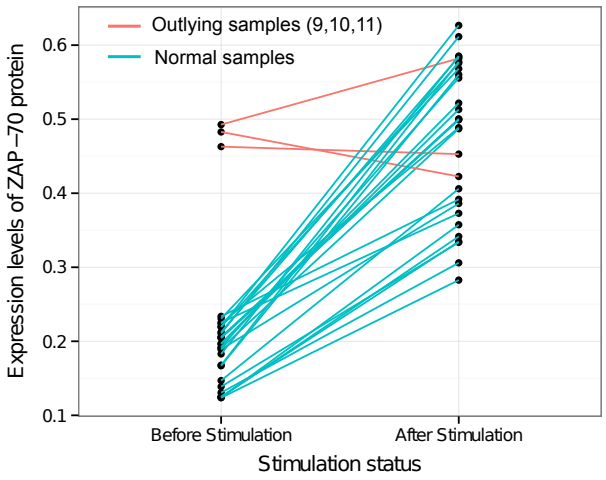

Figure 7: Levels of SLP-76 expression in each pair of samples (joined by a line) from the TCP dataset. For most samples, SLP-76 levels increase after the anti-CD3 stimulation. However, the three samples $\{9, 10, 11\}$ have high levels of SLP-76 protein in their pre-stimulation states, and do not show the usual increase after stimulation.

5.1.2 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 can explain this anomaly by plotting the average expression levels of SLP-76 protein for each pair of samples in Figure 7. The three outlying samples $\{9, 10, 11\}$ have much higher levels of SLP-76 than the remaining samples in their pre-stimulation states. On stimulation, two of these samples have decreased levels of SLP-76, while one shows an increase. Consequently these samples are classified with the post-stimulations samples by the dynamic templates-based classification algorithm presented here.

5.2 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 at Purdue's Bindley Biosciences Center. 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 unmixed in the pre-processing step, and lymphocytes ($CD45^+$) were identified using forward and side scatter data in the gating step. We then identified cell populations in each sample using the k-means algorithm, 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 ($CD3^-CD19^-$). Fig. 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.

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 tree by using the procedure described in Sec. 4. The complete template tree is shown in Fig. 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

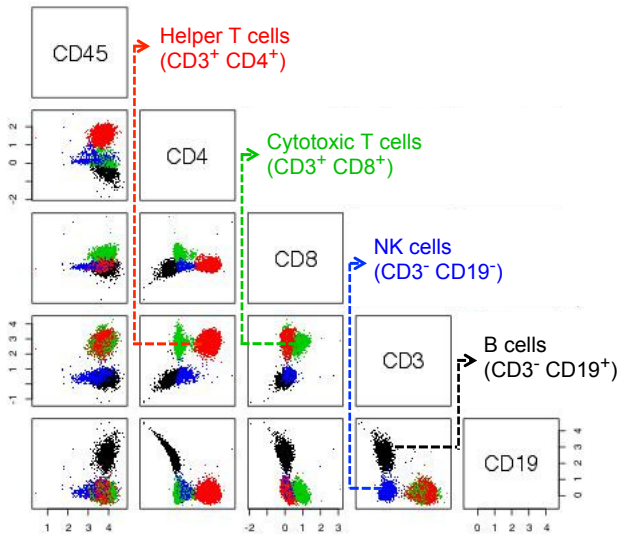

Figure 8: Bivariate projections of four clusters in a sample from the healthy donor dataset. Each cell cluster is CD45$^+$ since we pre-selected lymphocytes on the forward and side scatter channels. Four cell types are shown in red (helper T cells), green (cytotoxic T cells), blue (natural killer cells)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

the five branches.

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 innate biological variability in the healthy subjects. In this dataset we observe more natural biological variation than the technical and instrumental variations. We can visualize these variations with a heatplot shown in Fig. 10. Here the color of a square indicates the dissimilarity of a pair of samples, with a square drawn in a lighter shade when a pair of samples is similar, and in a darker shade 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 subject stay together in the template-tree in Fig. 9.

We observe a gradual increase of color intensity from left to right and from top to bottom in the off diagonal squares. Such a color pattern suggests that subjects located further away from each other (for example, A and B) have more dissimilar immune profiles than subjects adjacent to each other (for example, A and D).

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 Fig. 9 can be considered as a template for healthy individuals. This combined template represents a healthy immune profile by preserving the common features of healthy individuals and by removing between subject variations. This 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.

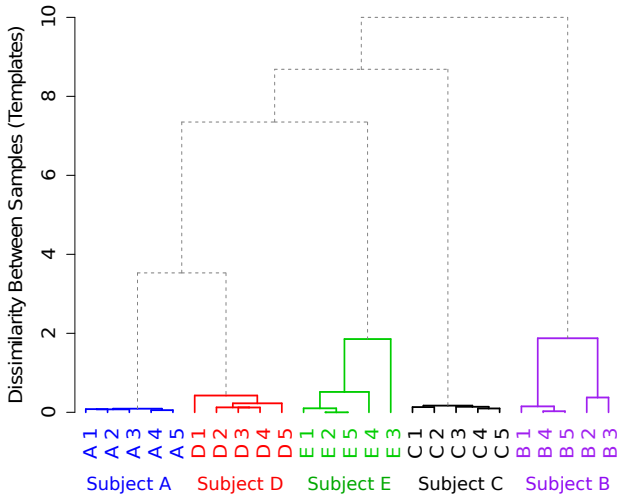

Figure 9: A dynamic template tree created incrementally from samples in the healthy donor dataset. The algorithm identifies five well separated branches denoting templates for the five subjects. Subtrees consisting of replicates from each of the five subjects are shown in five different colors.

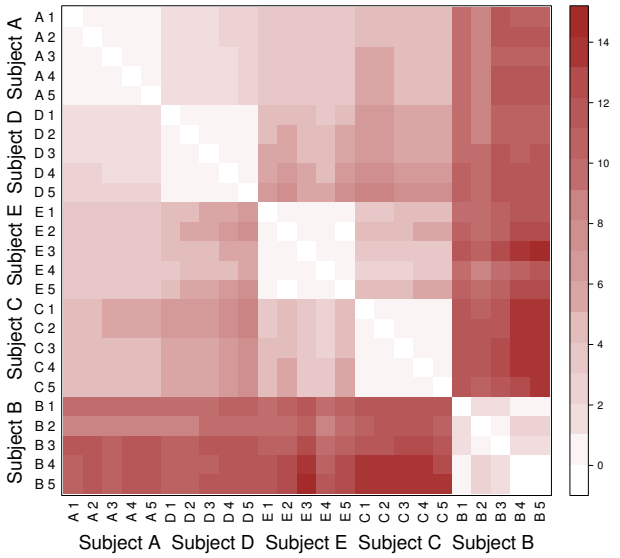

Figure 10: Heat plot showing the dissimilarity of samples in the healthy donor dataset.

6. CONCLUSIONS

We developed template-based classification methods for flow cytometry samples displaying a combination of different immunophenotypes. A template built from samples of a class provides a concise 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 maintain a hierarchy of the samples in a template tree such that samples can be analyzed in higher resolution as needed.

The major contribution in this paper is a dynamic algorithm to construct and update the templates, and build and maintain the template tree, when the samples arrive continuously over a period of time. As new samples come in, the templates are dynamically updated to reflect the information gained from them. This is a desirable property in dynamic situations, as in the course of an epidemic, when new samples are being collected and analyzed. Another context where the dynamic classification 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 need to be shared among the hospitals to create a global profile of the immune system, thus avoiding issues with privacy of clinical data.

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 and analyzed.

Acknowledgments

This work was partially supported by NIH grant 1R21EB015707-01, NSF grant CCF-1218916, DOE grant FC02-08ER25864, and an IBM Fellowship. SP thanks DBT, MoS&PI and DST, India, for support.

7. REFERENCES

[1] N. Aghaeepour, G. Finak, H. Hoos, et al. Critical assessment of automated flow cytometry data analysis techniques. *Nat. Meth.*, 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. *LNCS*, 6293:162–175, 2010, (WABI 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. *In preparation*, 2013.

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

[6] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(10):1042–1052, 1993.

[7] S. D. Connell and A. K. Jain. Template-based online character recognition. *Patt. Recog.*, 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. *IEEE Trans. Audio Speech Lang. Proc.*, 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 *Proc. Interspeech*, pp. 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. *Inf. Proc. Lett.*, 82(3):159–164, 2002.

[13] M. Halkidi and M. Vazirgiannis. Clustering validity assessment: Finding the optimal partitioning of a data set. In *ICDM 2001*, pp. 187–194, 2001.

[14] T. Lin. Robust mixture modeling using multivariate skew t distributions. *Stat. Comput.*, 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. *PNAS*, 104(47):18607–18612, 2007.

[18] D. Novo, G. Gregori, and B. Rajwa. Generalized unmixing model for multispectral flow cytometry utilizing nonsquare compensation matrices. *Cytometry Part A*, 83:508-520, 2013.

[19] S. P. Perfetto, P. K. Chattopadhyay, and M. Roederer. Seventeen-colour flow cytometry: unravelling the immune system. *Nat. Rev. Imm.*, 4(8):648–655, 2004.

[20] J. M. Peters and M. Q. Ansari. Multiparameter flow cytometry in the diagnosis and management of acute leukemia. *Arch. Path. Lab. Med.*, 135(1):44–54, 2011.

[21] S. Pyne, X. Hu, K. Wang, et al. Automated high-dimensional flow cytometric data analysis. *PNAS*, 106(21):8519–8524, 2009.

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

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