JMLR: Workshop and Conference Proceedings 27:155–166, 2012 Workshop on Unsupervised and Transfer Learning

Transfer Learning for Auto-gating of Flow Cytometry Data

Gyemin Lee GYEMIN@EECS.UMICH.EDU
Lloyd Stoolman STOOLMAN@UMICH.EDU
Clayton Scott CSCOTT@EECS.UMICH.EDU
University of Michigan, Ann Arbor, MI, USA

Editor: I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver

Abstract

Flow cytometry is a technique for rapidly quantifying physical and chemical properties of large numbers of cells. In clinical applications, flow cytometry data must be manually "gated" to identify cell populations of interest. While several researchers have investigated statistical methods for automating this process, most of them falls under the framework of unsupervised learning and mixture model fitting. We view the problem as one of transfer learning, which can leverage existing datasets previously gated by experts to automatically gate a new flow cytometry dataset while accounting for biological variation. We illustrate our proposed method by automatically gating lymphocytes from peripheral blood samples.
Keywords: flow cytometry, automatic gating, transfer learning, low-density separation

1. Introduction

Flow cytometry is a technique widely used in many clinical and biomedical laboratories for rapid cell analysis (Shapiro, 1994). It plays an important role in the diagnosis of blood-related diseases such as acute or chronic leukemias and malignant lymphomas.

Mathematically, a flow cytometry data can be represented as $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is an attribute vector of the ith cell. The attributes include the cell's size (FS), granularity (SS) and expression levels of different antigens (CD45, CD3, CD4, etc.). The number of cells N can range from 10,000 to 1,000,000, and d is usually between 7-12. In clinical settings, each data corresponds to a particular patient, where the cells are typically drawn from a peripheral blood, lymph node, or bone marrow sample.

To make a diagnosis, a pathologist will use a computer to visualize different two-dimensional scatter plots of a flow cytometry data as in Fig. 1. These plots illustrate the presence of several clusters of cells within each dataset. They also illustrate the variation of measured data from one patient to another. This variation arises from both biological (e.g., health condition) and technical (e.g., instrument calibration) sources.

The pathologist will typically visualize a certain type of cell (e.g., lymphocytes in the diagnosis of leukemias) and diagnose based on its shape, range and other distributional characteristics. A necessary preprocessing is to label every cell as belonging to the cell type of interest or not, a process known as "gating." This amounts to assigning binary labels $y_i \in \{-1, 1\}, i = 1, \ldots, N$, to every cell. Fig. 1 indicates lymphocytes with an alternate color. Without gating, cells of other types will overlap with the targeted cell type in the scatter plots used for diagnosis. After gating, only the cells of interest are visualized.

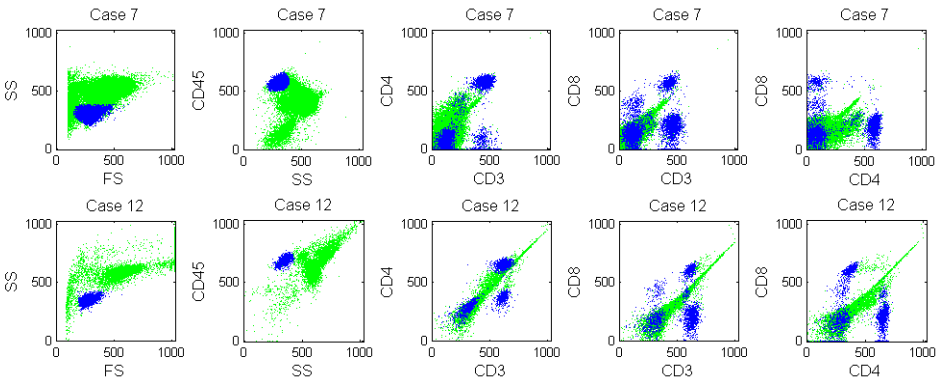

Figure 1: Clinicians analyze flow cytometry data using a series of scatter plots on attributes pairs. The distribution differs from patient to patient, and changes after treatments. Lymphocytes, a type of white blood cell, are marked dark/blue and others are marked bright/green. These were manually selected by a domain expert.

Unfortunately, in clinical settings gating is still performed manually. It is a labor-intensive task in which a pathologist visualizes the data from different two-dimensional scatter plots, and uses special software to draw a series of boundaries ("gates") to eliminate a portion of the cells outside of the desired type. The person performing gating must utilize specialized domain knowledge together with iterative refinement. Since modern clinical laboratories can see dozens of cases per day, it would be highly desirable to automate this process.

Because of this need, several researchers have tackled the auto-gating problem. A recent survey on flow cytometry analysis revealed that more than 70% of studies focus on auto-gating techniques (Bashashati and Brinkman, 2009). However, the vast majority of approaches rely on a clustering/mixture modeling, using a parametric representation for each cell type (Chan et al., 2008; Lo et al., 2008; Pyne et al., 2009). The mixture modeling approach has a number of difficulties, however. One is that the clusters are typically not elliptical, meaning complex parametric models must be employed, such as skewed Gaussians, leading to challenging inference problems. Another limitation is that human intervention is necessary to interpret the clustering results and to select some of the clusters for the task. Finally, these algorithms are unsupervised, and do not fully leverage expert knowledge.

We propose to view auto-gating as a transfer learning problem. In particular, we assume that a collection of expert-gated datasets are available. Although different datasets have different distributions, there is enough similarity, (e.g., lymphocytes show low levels of SS while expressing high levels of CD45) that this expert knowledge can be transferred to the new data. Our approach is to train classifiers on expert-gated data, and to summarize these classifiers to form a baseline classifier. This baseline is then adapted to the new data by optimizing a "low-density separation" criterion. The transfer learning problem we study is, to our knowledge, a new one, although it has similarities to previously studied transfer learning problems, as well as multi-task learning. These connections are reviewed below.

2. Problem Setup

There are M labeled datasets $\mathcal{D}_m = \{(\mathbf{x}_{m,i}, y_{m,i})\}_{i=1}^{N_m}$, $m = 1, \cdots, M$, each a random sample from a distribution $\mathcal{P}_m$. $\mathcal{D}_m$ corresponds to the mth flow cytometry dataset and its labels are determined by experts. There is also an unlabeled dataset $\mathcal{T} = \{\mathbf{x}_{t,i}\}_{i=1}^{N_t}$, a random sample from a new distribution $\mathcal{P}_t$ corresponding to a new flow cytometry dataset. The labels $\{y_{t,i}\}_{i=1}^{N_t}$ are not observed. The goal is to assign labels $\{\widehat{y}_{t,i}\}_{i=1}^{N_t}$ to $\mathcal{T}$ so that the misclassification rate is minimized. All the distributions are different, but defined on the same space $\mathbb{R}^d \times \{-1, +1\}$.

3. Related Work

As a transfer learning problem, our problem is characterized by having multiple source domains (the expert-gated datasets), and a single target domain (the unlabeled dataset). Using the taxonomy of Pan and Yang (2010), our setting can be described as follows:

(1) the source and target domains are different, because the marginal distributions of $\mathbf{x}$ are different,

(2) the source and target tasks are different, because each dataset requires a different gating,

(3) there are no labeled examples in the target domain.

To the best of our knowledge, previous work has not addressed this combination of characteristics. Many previous works fall under the heading of *inductive transfer learning*, where at least a few labels are given for the target domain (Ando and Zhang, 2005; Rettinger et al., 2006). In *transductive transfer learning* (Arnold et al., 2007), and the related problems of sample selection bias and covariate shift, the source and target tasks are assumed to be the same.

Another closely related area is multi-task learning (Caruana, 1997; Evgeniou and Pontil, 2004). However, our problem contrasts to this line of studies in the sense that our ultimate goal is achieving high performance for the target task only, and not the source tasks.

Toedling et al. (2006) explore using support vector machines (SVMs) for flow cytometry data from multiple patients. They merge all the datasets to form a single large data, and build a classifier on this data. However, due to its size of the combined dataset, the training requires demanding computational and memory resources. Furthermore, this approach ignores the variability among multiple datasets and treats all the datasets as arising from the same distribution. This reduces the problem to standard single-task supervised learning.

4. Algorithm

We describe our approach to the problem. In this section, we show how our algorithm summarizes knowledge from the source data and adapts it to the new task.

4.1. Baseline Classifier for Summarizing Expert Knowledge

We suppose that the knowledge contained in the source tasks can be represented by a set of decision functions $f_1, \cdots, f_M$. The sign of a decision function f_m provides a class prediction

Algorithm 1 Baseline Classifier

Input: source task data $\mathcal{D}_m$ for $m = 1, \cdots, M$, regularization parameters $\{C_m\}_{m=1}^M$

1: **for** $m = 1$ **to** M **do**
2: $(\mathbf{w}_m, b_m) \leftarrow SVM(\mathcal{D}_m,\ C_m)$
3: **end for**
4: Robust Mean:
$(\mathbf{w}_0, b_0) \leftarrow$ *Algorithm 2* $(\{(\mathbf{w}_m, b_m)\}_m)$

Output: $(\mathbf{w}_0, b_0)$ or $f_0(\mathbf{x}) = \langle \mathbf{w}_0, \mathbf{x} \rangle + b_0$

Algorithm 2 Robust Mean and Covariance

Input: $(\mathbf{w}_m, b_m)$ for $m = 1, \cdots, M$

1: Concatenate: $\mathbf{u}_m \leftarrow [\mathbf{w}_m, b_m], \quad \forall m$
2: Initialize: $\boldsymbol{\mu} \leftarrow mean(\mathbf{u}_m), \quad \mathbf{C} \leftarrow cov(\mathbf{u}_m)$
3: **repeat**
4: $d_m \leftarrow \left((\mathbf{u}_m - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{u}_m - \boldsymbol{\mu})\right)^{1/2}$
5: $w_m \leftarrow \psi(d_m)/d_m$
6: Update: $\boldsymbol{\mu}^{new} \leftarrow \frac{\sum_m w_m \mathbf{u}_m}{\sum_m w_m}$
$\mathbf{C}^{new} \leftarrow \frac{\sum_m w_m^2 (\mathbf{u}_m - \boldsymbol{\mu}^{new})(\mathbf{u}_m - \boldsymbol{\mu}^{new})^T}{\sum_m w_m^2 - 1}$
7: **until** Stopping conditions are satisfied

Output: $\boldsymbol{\mu} = [\mathbf{w}_0, b_0], \quad \mathbf{C}_0 = \mathbf{C}(1:d, 1:d)$

of a data point $\mathbf{x}$: $\widehat{y} = sign(f_m(\mathbf{x}))$. Each f_m is separately learned from each of the M source datasets. Then these decision functions form the pool of knowledge.

In this work, we consider linear decision functions $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. Then f defines a hyperplane $\{\mathbf{x} : f(\mathbf{x}) = 0\}$ with a normal vector $\mathbf{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$. The SVM is among the most widely used methods for learning a linear classifier (Schölkopf and Smola, 2002). It finds a separating hyperplane based on the maximal margin principle. We use the SVM to fit a decision function f_m or a hyperplane $(\mathbf{w}_m, b_m)$ to the mth source data $\mathcal{D}_m$.

We devise a baseline classifier $f_0 = \langle \mathbf{w}_0, \mathbf{x} \rangle + b_0$ by letting $(\mathbf{w}_0, b_0)$ be the mean of $(\mathbf{w}_m, b_m)$. Instead of the simple mean, Algorithm 1 uses a robust mean to prevent f_0 from being unduly influenced by atypical variations among datasets. Algorithm 2 presents the robust estimator as formulated in Campbell (1980). Here ψ is a weight function corresponding to a robust loss, and we use the Huber loss function. Note that we also robustly estimate the covariance of the $\mathbf{w}_m$, which is used below in Section 4.2.3.

The learning of f_0 does not involve $\mathcal{T}$ at all. Thus, it is not expected to provide a good prediction for the target task. Next we describe a way to adapt this baseline classifier to the target data based on the low-density separation principle.

4.2. Transferring Knowledge to Target Task

Low-density separation is a concept used extensively in machine learning. This notion forms the basis of many algorithms in clustering analysis, semi-supervised classification, novelty detection and transductive learning. The underlying intuition is that the decision boundaries between clusters or classes should pass through regions where the marginal density of $\mathbf{x}$ is low. Thus, our approach is to adjust the hyperplane parameters so that it passes through a region where the marginal density of $\mathcal{T}$ is low.

4.2.1. Preprocessing

Instrument calibration often introduces different shifting and scaling to each flow cytometry data along coordinate axis. While a typical solution is aligning all datasets via some global d-dimensional shift/scale transformation, it is sufficient for our purposes to align datasets in the direction of the baseline normal vector $\mathbf{w}_0$. Specifically, for each dataset, we compute a kernel density estimate (KDE) of the projection onto $\mathbf{w}_0$. Then, we align the target data

Algorithm 3 Shift Compensation

Input: hyperplane $(\mathbf{w}, b)$, source task data $\{\mathcal{D}_m\}_{m=1}^{M}$, target task data $\mathcal{T}$

1: $z_{t,i} \leftarrow \langle \mathbf{w}, \mathbf{x}_{t,i} \rangle + b, \quad \forall i$
2: **for** $m = 1$ **to** M **do**
3: $\quad z_{m,i} \leftarrow \langle \mathbf{w}, \mathbf{x}_{m,i} \rangle + b, \quad \forall i$
4: $\quad e_m \leftarrow \arg\max_z \mathrm{KDE}(z, z_{t,i}) \star \mathrm{KDE}(z, z_{m,i})$
5: **end for**
6: $b \leftarrow b - median(e_m)$

Output: b

Algorithm 4 Bias Update

Input: hyperplane $(\mathbf{w}, b)$, target task data $\mathcal{T}$

1: Compute: $z_i \leftarrow \langle \mathbf{w}, \mathbf{x}_{t,i} \rangle + b, \quad \forall i$
2: Build a Grid: $s_i \leftarrow sort\ (z_i)$
3: **for** $j = 1$ **to** N_t **do**
4: $\quad c_j \leftarrow \sum_i \mathbb{I}\{\frac{|z_i - s_j|}{\|\mathbf{w}\|} < 1\}$
5: **end for**
6: $h \leftarrow kernel\ bandwidth\ (\{(s_j, c_j)\}_j)$
7: Smooth: $\widehat{p}(z) \leftarrow \sum_j c_j k_h(z, s_j)$
8: $z^* \leftarrow gradient\ descent\ (\widehat{p}(z), 0)$
9: $b^{new} \leftarrow b - z^*$

Output: b^{new} or $f_b(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b^{new}$

to each source data using maximum cross-correlation (denoted by $\star$ in Algorithm 3), and modify the baseline bias by the median of these shifts. This new bias b will serve as the initial bias when adapting the baseline to $\mathcal{T}$.

4.2.2. Varying Bias

We first describe adapting the bias variable to the unlabeled target data $\mathcal{T}$ based on low-density separation. The process is illustrated in Algorithm 4.

To assess whether a linear decision boundary is in a low density region, we count data points near the hyperplane. As the hyperplane moves, this number will be large in a high density region and small in a low density region. In particular, we define a margin, as in SVMs, to be a region of a fixed distance from a hyperplane, say Δ, and count data points within this margin. We use $\Delta = 1$. Given a hyperplane $(\mathbf{w}, b)$, basic linear algebra shows that $\frac{\langle \mathbf{w}, \mathbf{x} \rangle + b}{\|\mathbf{w}\|}$ is the signed distance from $\mathbf{x}$ to the hyperplane. Hence, computing

$$\sum_i \mathbb{I}\{\frac{|\langle \mathbf{w}, \mathbf{x}_{t,i} \rangle + b|}{\|\mathbf{w}\|} < \Delta\}$$

over a range of b followed by locating a minimizer near the baseline hyperplane gives the desired solution. Algorithm 4 implements this on a grid of biases $\{s_j\}$ and builds $\sum_j c_j\, \delta(z - s_j)$ where δ is the Dirac delta. The grid points and the counts at each grid point are denoted by s_j and c_j.

Before searching for the minimizing bias, we smooth these counts over the grid by convolving with a Gaussian kernel $k_h(z, z') = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{|z - z'|^2}{2h^2}\right)$. The bandwidth h controls the smoothness of the kernel. This operation yields a smooth function $\widehat{p}(z) = \sum_j c_j k_h(z, s_j)$. Running a gradient descent algorithm on this smoothed function $\widehat{p}(z)$ returns a local minimum near 0 if initialized at 0 (the second parameter in Line 8).

To facilitate a streamlined process for practical use, we automatically select the kernel bandwidth h as shown in Algorithm 5. This kernel choice is motivated from the rule of thumb for kernel density estimation suggested in Silverman (1986).

Algorithm 5 Kernel Bandwidth

Input: grid points and counts $\{(s_k, c_k)\}$

$N \leftarrow \sum_k c_k$

$\overline{s} \leftarrow \frac{1}{N}\sum_k s_k c_k$

$\widehat{\sigma} \leftarrow \left(\frac{1}{N-1}\sum_k c_k (s_k - \overline{s})^2\right)^{1/2}$

$h \leftarrow 0.9 \cdot \widehat{\sigma} \cdot N^{-1/5}$

Output: h

Algorithm 6 Normal Vector Update

Input: hyperplane $(\mathbf{w}, b)$, direction of change $\mathbf{v}_t$, target task data $\mathcal{T}$

for $a_k = -0.5$ **to** 0.5 **step** 0.01 **do**

  $\mathbf{w}_k \leftarrow \mathbf{w} + a_k \mathbf{v}_t$

  $c_k \leftarrow \sum_i \mathbb{I}\{\left|\frac{\langle \mathbf{w}_k, \mathbf{x}_{t,i}\rangle + b}{\|\mathbf{w}_k\|}\right| < 1\}$

end for

$h \leftarrow$ *kernel bandwidth* $(\{(a_k, c_k)\}_k)$

Smooth: $g(a) \leftarrow \sum_k c_k k_h(a, a_k)$

$a_t \leftarrow$ *gradient descent* $(g(a), 0)$

$\mathbf{w}^{new} \leftarrow \mathbf{w} + a_t \mathbf{v}_t$

Output: $\mathbf{w}^{new}$

Algorithm 7 Set Estimation based on Low-Density Separation

Input: source task data $\{\mathcal{D}_m\}_{m=1}^M$, target task data $\mathcal{T}$, regularization parameters $\{C_m\}_{m=1}^M$

for $m = 1$ **to** M **do**

  $(\mathbf{w}_m, b_m) \leftarrow SVM(\mathcal{D}_m, C_m)$

end for

Initialize:

  $((\mathbf{w}_0, b_0), \mathbf{C}_0) \leftarrow$ *Alg* 2 $(\{(\mathbf{w}_m, b_m)\}_m)$

  $\mathbf{v}_0 \leftarrow eig(\mathbf{C}_0)$

Normalize:

  $\mathbf{w}_t \leftarrow \mathbf{w}_0/\|\mathbf{w}_0\|, \qquad b_t \leftarrow b_0/\|\mathbf{w}_0\|$

  $\mathbf{v}_t \leftarrow$ *orthonormalize* $\mathbf{v}_0$ *with respect to* $\mathbf{w}_0$

Compensate Shift:

  $b_t \leftarrow$ *Alg* 3 $(\mathbf{w}_t, b_t, \{\mathcal{D}_m\}, \mathcal{T})$

Update Bias:

  $b_t \leftarrow$ *Alg* 4 $(\mathbf{w}_t, b_t, \mathcal{T})$

repeat

  Update Normal Vector:

    $\mathbf{w}_t \leftarrow$ *Alg* 6 $(\mathbf{w}_t, b_t, \mathbf{v}_t, \mathcal{T})$

  Update Bias:

    $b_t \leftarrow$ *Alg* 4 $(\mathbf{w}_t, b_t, \mathcal{T})$

until Stopping conditions are satisfied

Output: $(\mathbf{w}_t, b_t)$ or $f_t = \langle \mathbf{w}_t, \mathbf{x}\rangle + b_t$

4.2.3. Varying Normal Vector

We can also adjust the normal vector of a hyperplane. Given a normal vector $\mathbf{w}$, we update the normal vector by $\mathbf{w}^{new} = \mathbf{w} + a_t \mathbf{v}_t$ where $\mathbf{v}_t$ is the direction of change and a_t is the amount of the change. Thus, the new normal vector is from an affine space spanned by $\mathbf{v}_t$.

Now we explain in detail the ways of choosing $\mathbf{v}_t$ and a_t. We find a direction of change from the covariance matrix of the normal vectors $\mathbf{w}_1, \cdots, \mathbf{w}_M$ obtained from Algorithm 2. We choose the first principal eigenvector for $\mathbf{v}_t$ after making it orthogonal to $\mathbf{w}_0$, the baseline normal vector, because changes in the direction of $\mathbf{w}_0$ do not affect the decision boundary.

To determine the amount of change a_t, we proceed similarly to the method used to update the bias. We count the number of data points inside the margin as the normal vector varies by a regular increment in the direction of $\mathbf{v}_t$. Convolving with a Gaussian kernel smooths these quantities over the range of variation. Then a gradient descent algorithm can spot a_t that leads to a low density solution near the baseline hyperplane. Algorithm 6 summarizes this process.

4.2.4. Putting It All Together

Once the normal vector is updated, we build a new hyperplane by combining it with an updated bias so that the hyperplane accords to a low density region of $\mathcal{T}$. Algorithm 7 outlines the overall scheme. In the algorithm, one can repeatedly update the bias and the

normal vector. A simple method is running a fixed number of times. In our experience, one round of iteration was sufficient for good solutions.

Although the presented algorithm limits the varying direction of normal vector to a single vector $\mathbf{v}_t$, we can generalize this to multiple directions. To do this, more than one eigenvector can be chosen in Step 5 of Algorithm 7. Then the Gram-Schmidt process (Golub and Van Loan, 1996) generates a set of orthonormal vectors that spans a subspace for a new normal vector. The counting in-margin points in Algorithm 6 can be extended to a multivariate grid with little difficulty.

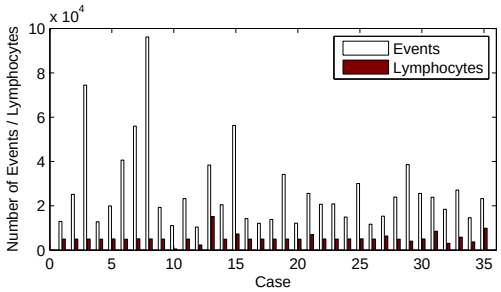

Figure 2: Number of total events and lymphocytes in each flow cytometry dataset.

5. Experiments

We demonstrate the proposed methods on clinical flow cytometry data. Specifically, we apply them to detect lymphocytes from peripheral blood samples. In diagnosing diseases such as leukemias, identifying these cells is the first step in most clinical analysis. Since the gating tools for this task are still primitive and unsatisfactory, a streamlined automatic gating procedure will be highly valuable in practice.

For the experiments, peripheral blood sample datasets were obtained from 35 normal patients. These datasets are provided by the Department of Pathology at the University of Michigan. The number of events in a dataset ranges from $10,000$ to $100,000$ with a varying portion of lymphocytes among them (see Fig. 2). An event in a dataset has six attributes (FS, SS, CD45, CD4, CD8 and CD3) and a corresponding binary label ($+1$ for lymphocytes and -1 for others) from the manual gates set by experts (see Fig. 1).

For the experiments, we adopt a leave-one-out setting: choose a dataset as a target task $\mathcal{T}$, hide its labels, and treat the other datasets as source tasks $\mathcal{D}_m$. Each $\mathcal{D}_m$ constitutes a binary classification problem with the goal of predicting the correct labels.

On each source data $\mathcal{D}_m$, we trained a SVM classifier f_m. We used LIBSVM package (Chang and Lin, 2001) with the default setting ($C_m = 1$). Then we applied the algorithms described in Section 4 and evaluated the prediction accuracy on the unlabeled target data $\mathcal{T}$. The considered transfer learning algorithms are:

- f_0 : baseline classifier with no adaptation, referred to as "baseline."
- f_b : classifier adapted to $\mathcal{T}$ by varying the bias-only, referred to as "bias."
- f_t : classifier adapted to $\mathcal{T}$ by varying both the direction and the bias, referred to as "dir. and bias."

In addition to the above classifiers, we compared the error rates from the following classifiers as points of reference:

- `Pooling` : A SVM is trained after merging all source data as in Toedling et al. (2006).

- `Transductive` : A transductive SVM is also trained on the merged source data and the target data using SVM-light package (Joachims, 1999).
- f_m for $m = 1, \cdots, M$: Each f_m learned from a source data $\mathcal{D}_m$ is applied straight to $\mathcal{T}$. This emulates a supervised learning setup with a train sample $\mathcal{D}_m$ and a test sample $\mathcal{T}$ while implicitly assuming $\mathcal{D}_m$ and $\mathcal{T}$ are drawn from the same distribution. A box plot in Fig. 3 displays the range of results with some '+' indicating extreme values. Table 1 numbers f_m^{Best}, the best of the 34 error rates.
- `Oracle` : We also applied the standard SVM with the true labels of $\mathcal{T}$. Its performance is computed by 5-fold cross validation. This quantity is what we can expect when a sufficient amount of labeled data are available for the target task.

For each dataset, we repeated these and reported their results in Fig. 3 and Table 1.

As can be seen in the figure and table, applying one of the f_m to the target task result in a wide range of accuracy. A classifier performing well on a dataset can perform poorly on other datasets due to relative difficulty of a task, dataset shift, or dissimilarity between tasks.

The `Pooling` performs poorly on many datasets. The merging step of the `Pooling` makes the classification problem more difficult. Even if classes are well-separated in each dataset, the separation will be lost in the merged dataset. Additionally, the classifier from `Pooling` can be biased toward larger source data. The optimization algorithm might also terminate prematurely before it converges to an optimal solution because the merged dataset is very large. Therefore, the `Pooling` can perform poorly, sometimes even worse than the worst f_m.

Because `Transductive` merges all the labeled and unlabeled datasets for training, it shares the similar problems as the `Pooling`. Moreover, the objective function is non-convex, and the solutions from an optimization algorithm are often suboptimal. The obtained results are very high error rates and low gating quality as shown in the table.

The baseline classifier f_0 typically improves when we adapt f_0 by changing the bias variable in most cases except Case 14 and Case 23. They further improve by adaptively varying both the direction and the bias. The differences among the f_m^{Best}, `Oracle` and f_t are very small. This reveals that our strategy can successfully replicate what experts do in the field without labeled training set for the target task.

6. Conclusion

We cast flow cytometry auto-gating as a novel kind of transfer learning problem. By combining existing ideas from transfer learning, together with a low-density separation criterion for class separation, our approach can leverage expert-gated datasets for the automatic gating of a new unlabeled dataset.

Although linear classifiers are sufficient to gate lymphocytes in peripheral blood, non-linear classifiers may be necessary for other kinds of auto-gating. For example, a bone marrow sample contains cells of whole range of developmental stages and is known to be more difficult to gate. Depending on diseases being screened for, other types of cells need to be separated or the separated lymphocytes need to be further gated. Our approach accommodates the incorporation of inner-product kernels, which may offer a solution to

such problems. It is also quite likely that several other strategies from the transfer learning literature can be adapted to this problem.

Biological and technical variation pose challenges for the analysis of many types of biomedical data. Typically one or both types of variation is accounted for by performing task-independent "normalization" prior to analysis. Our approach to flow cytometry auto-gating can be viewed as a task-dependent approach. The application of transfer learning to overcome biological and/or technical variations in other kinds of biomedical data is an interesting problem for future work.

Table 1: The error rates (%) of various classifiers on each flow cytometry dataset, with the other 34 treated as labeled datasets. The results from f_t adapted to the unlabeled target data are comparable to `Oracle` trained on labeled target data, and make less errors than `Pooling`.

Case	Trans	Pool	f_0	f_b	f_t	f_m^B	Oracle
1	44.64	38.65	2.91	3.05	3.17	2.87	2.79
2	70.66	20.27	5.44	2.09	2.10	2.06	1.71
3	18.46	2.05	1.66	1.00	0.94	0.91	0.74
4	19.27	2.93	2.62	2.54	2.67	2.44	2.56
5	34.10	5.06	1.50	1.40	1.44	1.41	1.58
6	31.90	1.60	1.84	1.60	1.80	1.62	1.56
7	90.82	7.00	0.91	0.82	0.77	0.80	0.79
8	89.28	2.44	0.65	0.60	0.50	0.52	0.47
9	16.92	8.31	2.19	1.91	1.83	1.78	1.71
10	37.14	26.65	2.16	1.09	1.09	1.03	1.03
11	73.47	2.67	5.11	1.86	1.86	1.77	1.79
12	65.48	21.89	6.69	1.60	1.63	1.78	1.54
13	58.44	39.44	1.69	1.63	1.65	1.59	1.64
14	75.60	3.67	2.29	3.55	0.87	0.71	0.81
15	32.68	5.90	1.78	1.16	1.22	1.11	1.11
16	64.41	4.34	3.79	3.19	3.23	2.82	2.83
17	56.92	7.70	2.75	3.49	3.51	2.47	2.44
18	63.88	2.53	1.86	1.64	1.67	1.59	1.60
19	84.48	8.25	3.44	3.45	3.14	2.46	2.29
20	31.01	3.03	4.48	2.39	2.37	2.56	2.45
21	63.61	10.14	7.71	6.28	6.30	5.64	5.08
22	18.69	4.16	1.60	1.81	1.82	1.54	1.42
23	75.95	21.73	2.89	7.51	1.58	1.61	1.43
24	29.89	2.79	2.41	2.06	2.06	1.91	1.89
25	6.57	1.98	2.22	2.25	2.32	2.04	1.47
26	56.89	1.55	2.13	1.82	1.83	1.42	1.39
27	56.83	11.34	11.22	9.02	9.18	8.17	8.72
28	53.73	2.21	1.68	2.23	2.17	1.56	1.48
29	80.58	9.19	1.06	0.96	0.97	0.77	0.73
30	14.20	7.80	1.25	1.24	1.25	1.25	1.24
31	61.83	16.08	13.46	4.57	4.59	4.80	4.45
32	80.56	20.39	12.66	2.62	2.62	2.72	2.21
33	75.87	5.57	4.58	5.74	5.74	2.28	1.77
34	20.82	4.66	2.10	1.90	1.93	1.79	1.80
35	55.84	9.33	6.68	5.46	5.49	5.56	5.59
avg	51.76	9.80	3.70	2.73	2.49	2.21	2.12
std err	4.12	1.68	0.54	0.33	0.30	0.26	0.27

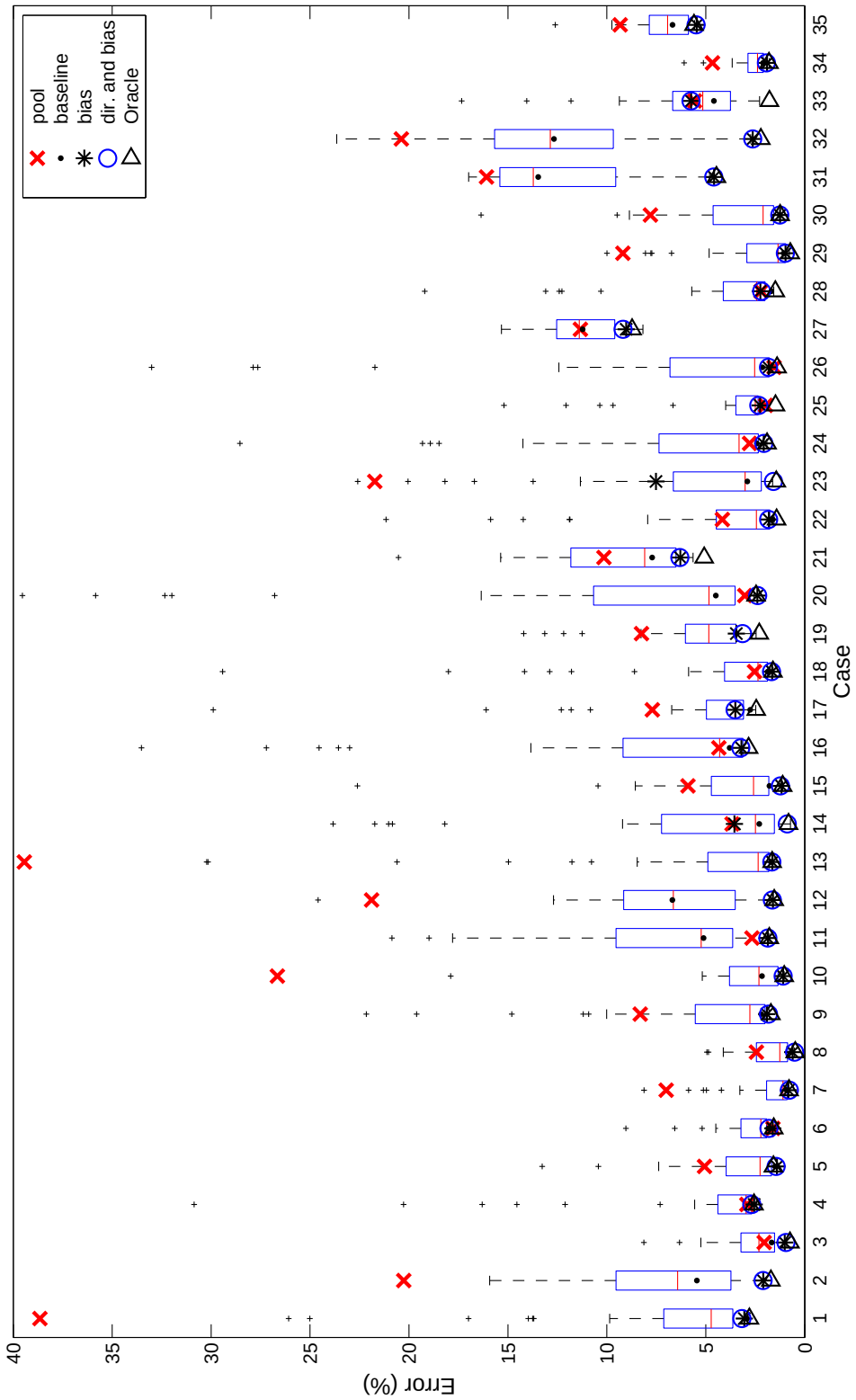

Figure 3: The error rates of various classifiers on each unlabeled dataset. The box plot corresponds to the range of results when one of f_m is applied to $\mathcal{T}$. '+' marks an extreme value that deviates from the others. The results from f_t, `Oracle` and the best of f_m are usually indistinguishable.

References

R. K. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 05)*, pages 1–9, 2005.

A. Arnold, R. Nallapati, and W.W. Cohen. A comparative study of methods for transductive transfer learning. *Seventh IEEE International Conference on Data Mining Workshops*, pages 77–82, 2007.

A. Bashashati and R. R. Brinkman. A survey of flow cytometry data analysis methods. *Advances in Bioinformatics*, 2009:Article ID 584603, 2009. doi: 10.1155/2009/584603.

N. A. Campbell. Robust procedures in multivariate analysis I: Robust covariance estimation. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29:231–237, 1980.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

C. Chan, F. Feng, J. Ottinger, D. Foster, M. West, and T.B. Kepler. Statistical mixture modeling for cell subtype identification in flow cytometry. *Cytometry Part A*, 73:693–701, 2008.

C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

T. Evgeniou and M. Pontil. Regularized multi–task learning. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD 04)*, pages 109–117, 2004.

G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

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

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

S. Pyne, X. Hu, K. Wang, E. Rossin, T. Lin, L. M. Maier, C. Baecher-Allan, G. J. McLachlan, P. Tamayo, D. A. Hafler, P. L. De Jager, and J. P. Mesirov. Automated high-dimensional flow cytometric data analysis. *PNAS*, 106:8519–8524, 2009.

A. Rettinger, M. Zinkevich, and M. Bowling. Boosting expert ensembles for rapid concept recall. *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 06)*, 1:464–469, 2006.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

H. Shapiro. *Practical Flow Cytometry*. Wiley-Liss, 3rd edition, 1994.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

J. Toedling, P. Rhein, R. Ratei, L. Karawajew, and R. Spang. Automated in-silico detection of cell populations in flow cytometry readouts and its application to leukemia disease monitoring. *BMC Bioinformatics*, 7:282, 2006.