Flow Cytometry and Machine Learning

Patrick Cheng

Jan 10, 2017

Introduction:

In the United States, an estimated 168,520 cases of leukemia, lymphoma, and myeloma occurred in 2016. Leukemia is a form of “blood cancer” that is serious and oftentimes fatal. Compounding the difficulty of this disease is that patients often present to the doctor with symptoms and laboratory findings that are vague and non-specific. It is therefore important that the doctor follow up with flow cytometry, a diagnostic tool that is highly sensitive for leukemia.

Flow cytometry is a process in which a sample of thousands of cells are evaluated for the presence of certain proteins. The sample is mixed with a fluorescent molecule that tags only one unique protein at a time. The fluorescence level is then used as an indirect measurement of how much of that protein each cell is making.

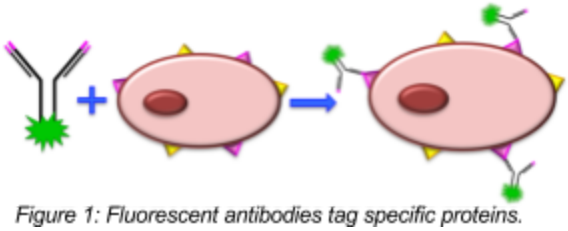


Figure 1: Fluorescent antibodies tag specific proteins.

Every kind of cell in the human body has a “signature” pattern of protein levels that does not differ greatly from person to person. This means that cancer cells are able to be detected by their aberrant flow readings.

Leukemia is a cancer that arises from corruption of the maturation process among one of several of the kinds of immune “white” cells. As an example, the flow cytometry for a normal developing white cell may gradually gain protein “CD38” fluorescence until it matures, at which point “CD34” protein levels drop precipitously. A finding of a large population of cells showing low fluorescence of both CD38 and CD34 (figure 2) is highly suspicious and may signal a mutant population of cells that would warrant further investigation.

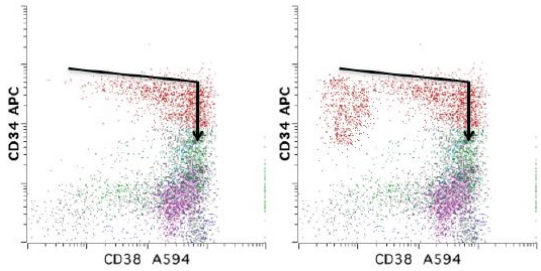


Figure 2: CD34 vs CD38. Healthy patient (left) vs. sick patient (right) shows varying distributions of developing white cells.

A typical flow cytometry readout may be time consuming for a pathologist to interpret for even the most simple cases. The first step in this process is manually “gating” for different cell populations – drawing boxes on 2d plots comparing 2 protein markers at a time. For a panel of 30 markers, to draw maximal information this may involve examining on the order of “30 choose 2” marker combinations. This process may additionally be inconsistent between users as it is dependent on personal gating technique.

My hypothesis is that this process can be automated with a high degree of sensitivity, using machine learning algorithms to locate cases of acute myeloid leukemia from flow cytometry readings. The cost savings standpoint alone is compelling; the cost of a pathologist is roughly $1 per minute, and the most

straightforward flow cytometry may take 5-10 minutes to read, for the 150,000 new cases of leukemia per year, an automated flow reader would save around $750,000 to $1,500,000 per year. This does not account for negative cases, periodic testing for relapse, and treatment monitoring, which in reality account for a very significant portion of flow cytometries.

The code requires Python 2.7, scikit-learn, pandas, and numpy. The Anaconda distribution is recommended.

Dataset:

Data is acquired from a publicly available flow cytometry repository at flowrepository.org. The dataset in use is a reference dataset for a commercial laboratory containing 359 people, including 43 patients suffering from acute myeloid leukemia (AML). The AML diagnoses were determined by manual gating methods.

The data was located in standardized .fcs form and had to be converted into .csv. The module to do this was located at: https://genepattern.broadinstitute.org/gp/pages/login.jsf and the code for the module is located at: https://github.com/genepattern.

Each patient has eight csv files representing eight physical tubes that were analyzed by the flow cytometer. Therefore, the dataset contains 2,872 csv files (359 patients by 8 tubes). Each patient's 8 tubes contains a different combination of protein markers. The combinations roughly correspond to characteristic profiles for different cell lineages, which may be helpful as cells may become cancerous at many points in differentiation or from many different cell lines. Therefore, a single tube may signal flagrantly leukemic while other tubes are relatively normal. Every tube has roughly 30,000 analyzed cells. Each data point represents fluorescent intensity which is an analog of protein concentration.

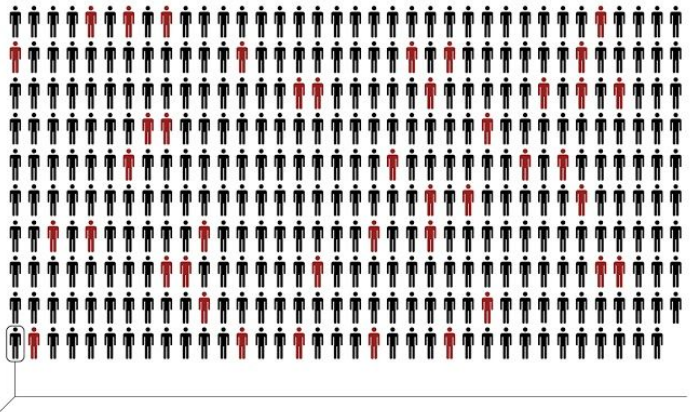
Examination reveals that fluorescence of FS (forward scatter) ranged from 0 to 205 and all other markers from 0 to 1015.

The eighth tube in each set is a negative control with no meaningful information, and will not be examined.

Exploring the data:

One of the interesting things about our dataset is that our data is found in three "tiers". We have 359 patients in the dataset, with 8 tubes each, containing 30,000 different cells each. The first question to answer becomes, how do we summarize or otherwise compare each tube's 30,000 cells as a "dataset within a dataset?"

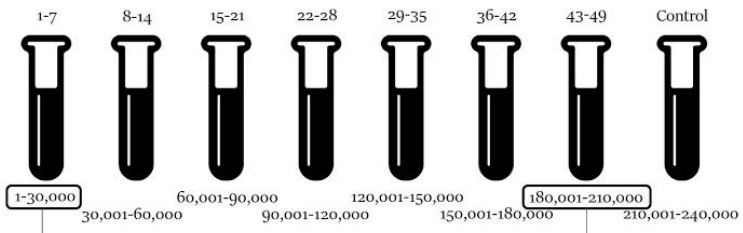
TIER 1
359 patients



TIER 2
8 tubes per patient

Patient #325

Marker numbers



Cell numbers

TIER 3
30,000 cells per tube

MARKER NAMES

FLUORESCENCE LEVELS (PROTEIN CONCENTRATION)

Tube 1

Cell	FS Lin:FS Lin	SS Log:SS Log	FL1 Log:IgG1-FITC	FL2 Log:IgG1-PE	FL3 Log:CD45-ECD	FL4 Log:IgG1-PC5	FL5 Log:IgG1-PC7
0	83.2	3.42	0.10	0.10	22.39	0.10	0.10
1	131.6	8.55	1.14	0.51	43.57	0.30	0.10
2	134.4	17.10	1.92	1.53	8.71	0.35	0.33
...	...	...	...	...	...	...	...
29997	65.8	2.12	0.36	0.10	26.33	0.10	0.10
29998	126.0	18.88	2.41	1.49	26.33	0.46	0.68
29999	104.4	22.80	0.92	0.71	16.64	0.10	0.10

Tube 7

Cell	FS Lin:FS Lin	SS Log:SS Log	FL1 Log:CD5-FITC	FL2 Log:CD19-PE	FL3 Log:CD45-ECD	FL4 Log:CD3-PC5	FL5 Log:CD10-PC7
180000	84.0	4.09	41.66	0.10	26.09	8.87	0.14
180001	130.8	21.80	1.92	1.42	5.66	0.58	0.18
180002	66.4	2.18	15.91	0.21	17.10	7.41	0.10
...	...	...	...	...	...	...	...
209997	204.6	33.57	3.48	2.41	48.54	1.23	1.10
209998	129.2	19.92	2.07	1.35	16.49	0.98	1.68
209999	156.0	25.17	2.16	0.89	11.10	0.20	0.50

Scatter plotting the first patient's first two markers results in the scatterplot in figure 3.

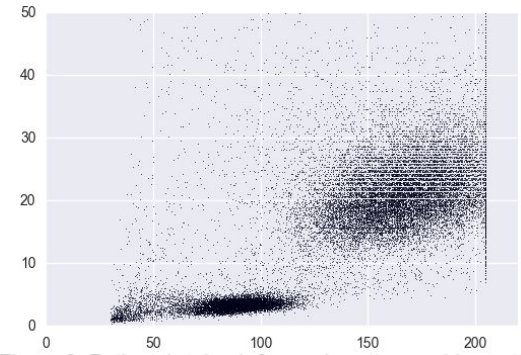


Figure 3: Patient 1, tube 1, forward scatter vs side scatter

With this visualization, a concept about cancerous cells must be explained. Cells in the body have their own programming language called DNA that determines every protein made and as a result nearly everything about the way it behaves, including how often the cell replicates. Discrete maloccurrences may happen as a result of toxic radicals or freak biochemical accidents that cause damage to the "code". When enough errors accumulate to bypass the body's innate correction mechanisms, the one malignant cell may begin to replicate unchecked.

As can be seen from the plot, cell populations are detected as concentrations of cells with quite some variation to them. A malignant cell population can show up to pathologists as a disturbance of the distributions of these concentrations, whether that be an extra population where there shouldn't be, a disturbed spatial distribution of a typical population, or a significantly upset ratio between populations. The plot also demonstrates a fair amount of typical "debris", which can represent dead cells, cells exploded or leaking due to the process of sample ascertainment, or simply stray, irrelevant cells. What is *not* typically evaluated is the malignancy status of individual cells, which is too difficult to differentiate from normal variance or cellular debris. Therefore, our summary of the cellular tier data will be made from the shape of *distributions* of the populations rather than direct evaluations of malignancy of individual cells.

A log 10 transformation is applied to attain better resolution. The same patient from above is shown on the left, compared with a leukemic patient on the right.

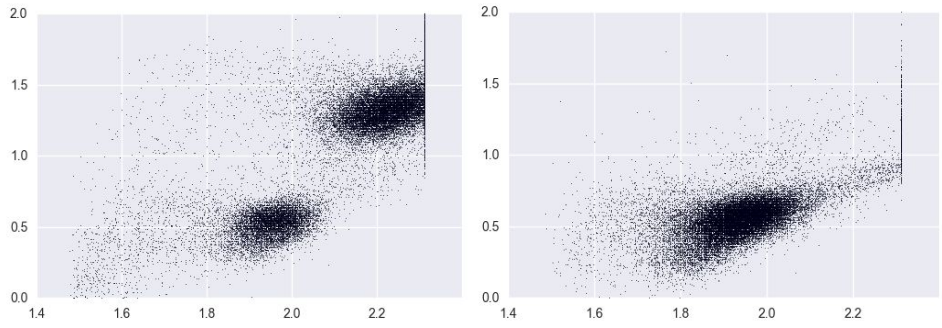


Figure 4: Patient 1, healthy (left) vs patient 9, leukemic (right), tube 1 forward scatter vs side scatter, with log 10 transformation

The log transformation spreads the data out much more evenly between a range of 0 to approximately 3, as the range of signal for these markers is 0 to 1015. There is also a very visible difference between the two patients. One would hope a machine learning algorithm could also tell the difference.

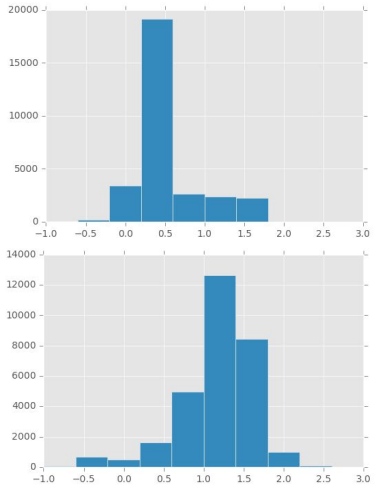


Figure 5: Univariate histogram of forward scatter, patient 1, healthy (top) vs patient 9, leukemic (bottom)

Tier 3: Cellular populations

As mentioned, a method of examining the distribution of populations is needed. One way to do this is by histogramming the data. For example, a count of all cells for a single marker would look like the figure at left. Again, we can see the overall shape of the histogram is markedly different between the same healthy and leukemic patients. The histogram bin count was set to 10; this parameter may be adjusted in the future (further addressed in Discussion).

The two histograms in figure 5 are made using univariate data only. For a bivariate histogram, the plot is divided into 10 by 10 boxes and the number of cells in each box becomes the histogram count (figure 6). For a 3-dimensional histogram, the boxes become 10 by 10 by 10 cubes, and for 4 dimensions, 10 x 10 x 10 x 10 hypercubes, and so on.

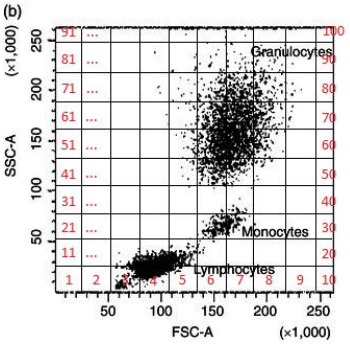


Figure 6: Construction of a bivariate histogram

Tier 2: Tube by tube comparison

So we have a way of characterizing the populations of each tube by analyzing the histogram of all 30,000 cells per tube. We can now continue to the next “tier”, in which we compare corresponding tubes for each patient (i.e., tube 1 against all other tube 1’s, etc.).

When analyzing data at the tube level, each patient will have 7 different predictions from each corresponding tube. While each patient can only be healthy or ill, each tube is an independent sample containing 30,000 different cells, measured for different proteins. Accordingly, each tube must be trained to its own machine learning model. Therefore, to make predictions for each patient we must consult 7 different models and integrate those predictions into a single overall prediction.

Machine learning:

To make a prediction for each tube, we can employ a machine learning algorithm. There are literally dozens of machine learning tools to choose from, so it’s important to choose one that suits our purposes. There are a few demands of the algorithm:

- Classifier, rather than regression, specifically for binary data (healthy or sick)
- Reasonably fast, preferably once trained can predict quickly

- Handles high numbers of “features”, especially with relatively low sample size (359)
- Flexible decision boundary, given the nature of biological data
- Resistant to overfitting (more on this in a minute)

Given these conditions, support vector machine (SVM) seems like a reasonable choice.

SVM is an algorithm that, given a dataset of pre-labeled data, will train on that data by seeking a margin separating the classes that is as wide as possible. The dimensionality of the marginal hyperplane depends on the dimensionality of the data; for example, for a dataset predicting blood pressure from two dimensions like age and weight, a separating hyperplane will be a one dimensional line (looks like figure 7). For data in three dimensions, the hyperplane will be a two dimensional plane, and so on. Generalized, for a dataset of n dimensions, the separating hyperplane will be n - 1 dimensions.

The hyperplane is calculated by finding the samples that lie closest and finding the hyperplane that maximizes the distance to these points as much as possible. In this sense, the points that have the most influence on the hyperplane are a small fraction of the total dataset, and we call them “support vectors.” (More in depth explanation of SVM in appendix A.)

One strength of SVM is that once trained onto the support vectors, classification of new samples requires only knowledge of the support vectors and not the entire dataset. This can be particularly useful for large datasets.

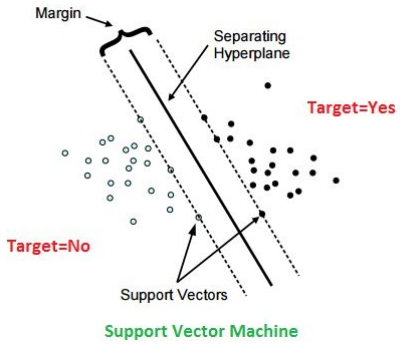


Figure 7: Bivariate SVM visualized

The other great strength of SVM is using what is called the “kernel trick” to map data into high dimensional space in the situation that the data is not immediately separable. This is best illustrated by the figure 8:

Untransformed Transformed

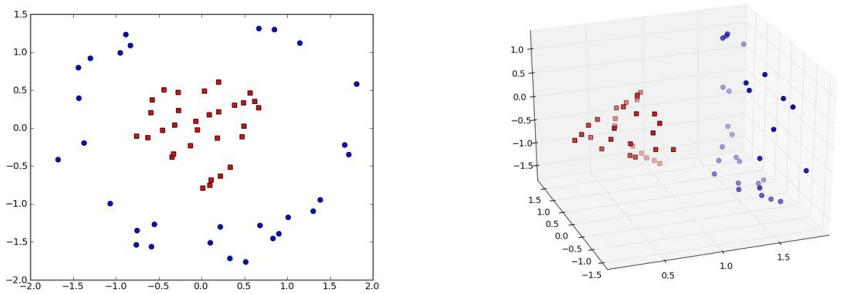


Figure 8: Kernel trick. Left, untransformed data vs right, data transformed into higher dimensional space

This is useful for datasets with non-linear relationships, as you can see. Particularly for biological data, choosing a kernel that can handle non-linear relationships can be very helpful.

Furthermore, the linear kernel in the sklearn module I chose has the added advantage of superior optimization. It is, simply, fast.

Regularization:

The last consideration for our machine learning algorithm is how we will prevent our model from overfitting our data. What is overfitting?

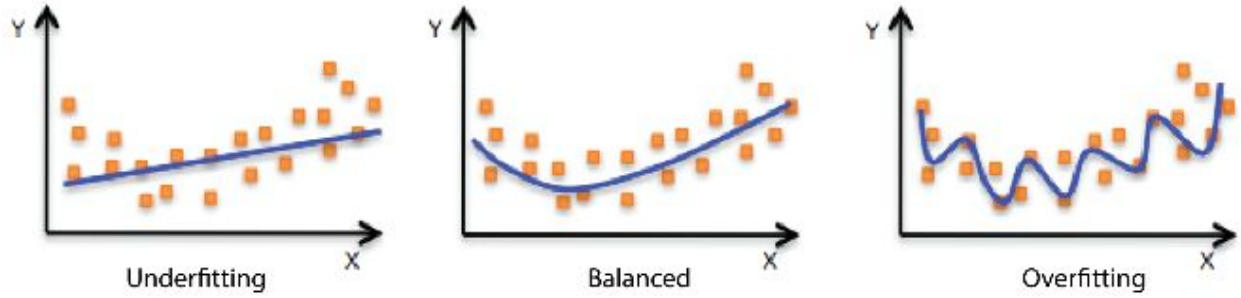


Figure 9: Underfitting vs appropriate model vs overfitting

Overfitting is fitting an overly complex model to your data such that it produces excellent training scores, but the model is not useful because it does not generalize to predict new data well. In figure 9, you can see the left figure is a regression line that is mathematically “correct”, but clearly does not fit the data well. The overfitted figure on the right may technically fit the available training data exceedingly well, but is overly complex and likely does not predict new data well. In most cases, the middle figure is the balance to strive for.

There are a few methods of reducing model overfit that we will employ. The first is called regularization, in which model complexity is mathematically “punished” by introducing a second term to the original SVM optimization problem, where ξ is the misclassification distance of sample i (0 if classified correctly). Essentially, SVM sums all misclassification distances, and then multiplies by a variable C that determines how severe the punishment is for a given amount of error. While it may seem intuitive that we want as little error as possible, remember that overfitting leads to overly complex models that do not generalize well. Figure 10 shows an example of where introducing tolerance via adjusting C and allowing a “softer” margin may lead to a more robust model, even if it leads to some training misclassifications.

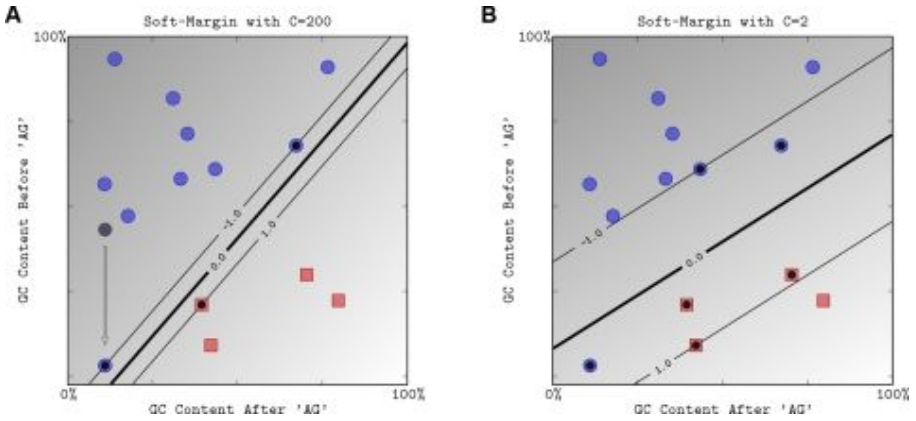


Figure 10: Soft margin regularization allows training misclassification but increases generalization.

Cross Validation:

Cross validation is another way of reducing overfitting. As overfitting arises from over-reliance on training data, we can chop our data up in several ways to present more testing data. This helps lend more power to our model by making it more generalizable.

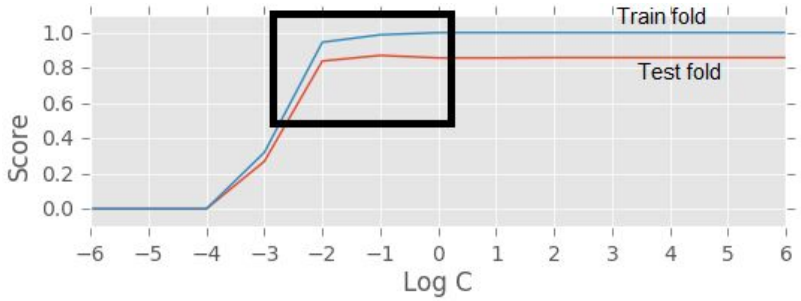
The cross validation method employed is 5-fold validation. While one fifth of the data is held out for the test fold at a time, the remaining data is trained on an optimal C value determined by iterating over several C values to find the one that is most appropriate for the model. The chosen regularization term is then used to make predictions on the test fold.

	mean_test_score	mean_train_score	param_C
0	0.000000	0.000000	1e-06
1	0.000000	0.000000	1e-05
2	0.000000	0.000000	0.0001
3	0.269091	0.320124	0.001
4	0.838870	0.945572	0.01
5	0.870866	0.988060	0.1
6	0.856564	1.000000	1
7	0.856564	1.000000	10
8	0.858422	1.000000	100
9	0.858422	1.000000	1000
10	0.858422	1.000000	10000
11	0.858422	1.000000	100000
12	0.858422	1.000000	1e+06

Figure 11: Sample SVM grid search for appropriate regularization term

In figure 11 a sample gridsearch can be seen. This is the stage at which we have generated a large dataset for a single tube of 359 patients by “n choose r” marker combinations and are selecting a C to use for the 72 patients held out as a test fold. The grid search is essentially brute force iteration over a range of C in a logspace of 10^-6 to 10^6.

10-fold cross validation was used during this process, resulting in an overall nested cross validation. One fifth of the total set is held out as previously mentioned, then the remaining ⅘ is again split into ⅕ and ⅘ for the purpose of grid searching C. For each C in the search, 10 models are trained and the results averaged in an effort to reduce variance. The reason for the nested split is to conserve the all-important



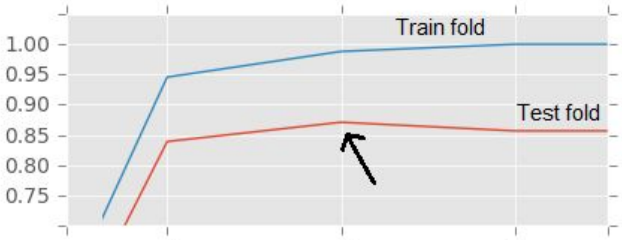


Figure 12: Plot of train and test scores for range of C values with selection at optimal fit C (arrow)

separation of training influence from the final test fold until the predictions are ready to actually be made. Using test data as part of the training fold is once again a way to cause overfit.

It is important to select a C value based on test score rather than validation score. In figure 12 you can see both test and training score plotted for each value of C. While it is tempting to choose values of C for which the training score was a perfect 1.0, that would be the definition of overfit. Instead, as you can see in the exploded section, there is a peak for the test score while the training score is still rising, and this is the optimal C that should be chosen. In this example C would be 10^-1, which would then be used for original test fold prediction. It should be noted that this particular shape of score plot is conserved through every SVM grid search.

Metrics:

One last thing before we run some data. We must define our evaluation method for our models (including in the grid search shown previously). The easiest and most accessible way to evaluate the value of any model is “accuracy”, defined as fraction of correct predictions over the total number of samples. Accuracy can be very misleading, however, especially in datasets where the true positive ratio is naturally very low. Imagine a scenario where 1% of the population has a common cold. If someone invented a “test” that simply said 100% of people were healthy, they would be 99% accurate! This must be avoided for the leukemia dataset in which there are only 43 positive samples to begin with.

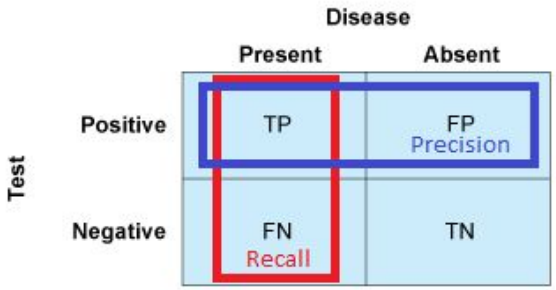


Figure 13: Avoiding misleading reporting for lopsided population ratios

A commonly used alternative is called the F-score, which is defined as the 2 times the harmonic mean of precision and recall. Precision is analogous to positive predictive value and recall is analogous to

sensitivity. If you look at the formulas for both (figure 13) you will notice that true negative results (the 99% that don't have colds) are not included in the calculation.

Running some data:

Let's run some data through SVM. We will feed a sample set composed of 359 samples, where each sample is a histogram that describes the shape of our data. I started with a bivariate histogram of forward scatter versus side scatter because it is a very basic combination familiar to pathologists.

Since we need to feed the data into the SVM in a form it can understand, we will "flatten" the 10 by 10 histogram into a line, in which the first feature is the lower left bin, the second feature is the bin to its right, and so on until all 100 bins are accounted for. After this is done we have a 359 x 100 matrix ready for analysis.

Running this matrix produces a F-score of 0.76 and a recall of 0.67. This means that 67% of the actual sick patients were detected. Obviously not ideal, but this was only one combination of two markers.

What if we include all combinations in the predictions? We can take the histogram of every combination of markers in each tube, and include them in the matrix. This means the input matrix will take the shape of 359 patient rows by "7 (total markers) choose 2" combinations by 10 by 10 histogram cells, or 359 rows x 2100 columns. We shall include all 7 tubes as well, so…

There is an important decision to be made about combining data from all the tubes (Tier 1: Patient level predictions!). As there will be 7 distinct predictions based on all markers in each tube, I choose to call it a positive overall prediction if any single tube is positive. The reason is that the likely use for this machine learning algorithm may be to screen for positive cases rather than going for straight accuracy or specificity. In this case, it is far worse to call a sick person healthy than it is to call a healthy person sick. In the latter case further investigation (which would likely be undertaken for any positive) would simply reveal it to be a false positive, whereas depending on a false negative would allow unchecked disease progression. Calling the overall prediction positive from any single positive marker combination is one way to make sure sensitivity is as high as possible.

Continuing on, if we run this new matrix through the SVM we can see that the F-score increases to 0.85 and recall increases to 0.95. Great progress! It appears there were still 12 false positives and 2 false negatives though. What can we do to make our model more accurate?

Three Dimensions:

At the outset of this project I wanted to examine the potential of looking at data in more dimensions that we as humans are used to perceiving. Barring Minority Report-style technology, we do not visualize data in three dimensions. What if we change our histogram to include a third marker? In fact, let's run the data through 1, 4, 5, 6, and 7 dimensions as well. We will examine all combinations of 3 markers in each tube, make a prediction based off each tube, then combine those predictions for one overall prediction just like for our 2 dimensional version; then we will repeat for combinations of 1, 4, 5, 6, and 7 markers at once as well (figure 14).

Dimensions	F-score	Recall	Precision	TP	FP	FN	Features	Median C	S.d. of C	Log s.d. Of C
1	0.81	0.95	0.71	41	17	2	72	1	16.82	1.23
2	0.85	0.95	0.76	41	13	2	1295	10	21.18	1.33
3	0.89	0.95	0.84	41	8	2	9188	10	0	0
4	0.96	0.95	0.97	41	1	2	25519	10	0	0
5	0.96	0.93	1	40	0	3	32384	10	0	0
6	0.95	0.95	0.95	41	2	2	19158	10	0	0
7	0.91	0.95	0.87	41	6	2	4294	10	25.23	1.41

Figure 14: Results for all numbers of concurrent markers considered. TP=True positive, FP = False positive, FN = False negative, s.d. = standard deviation

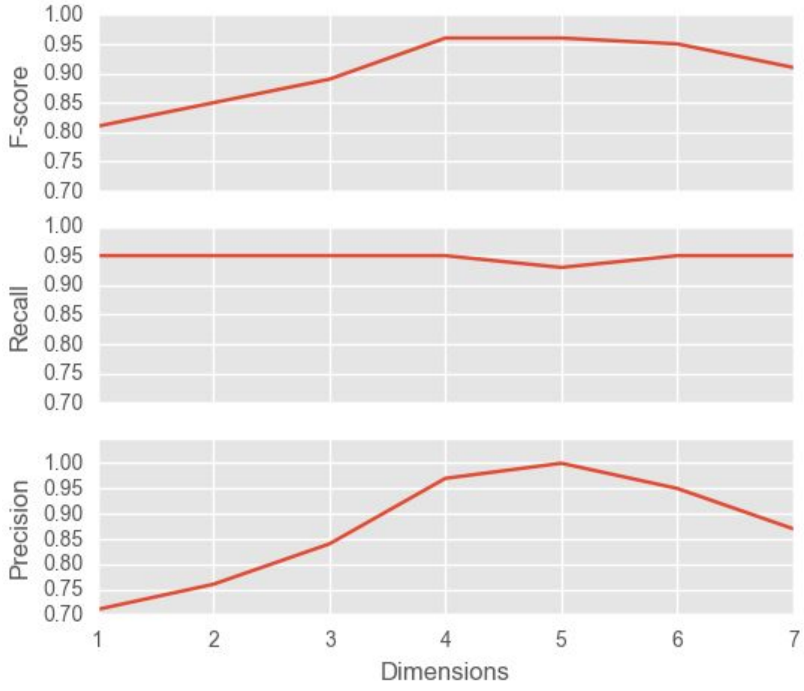


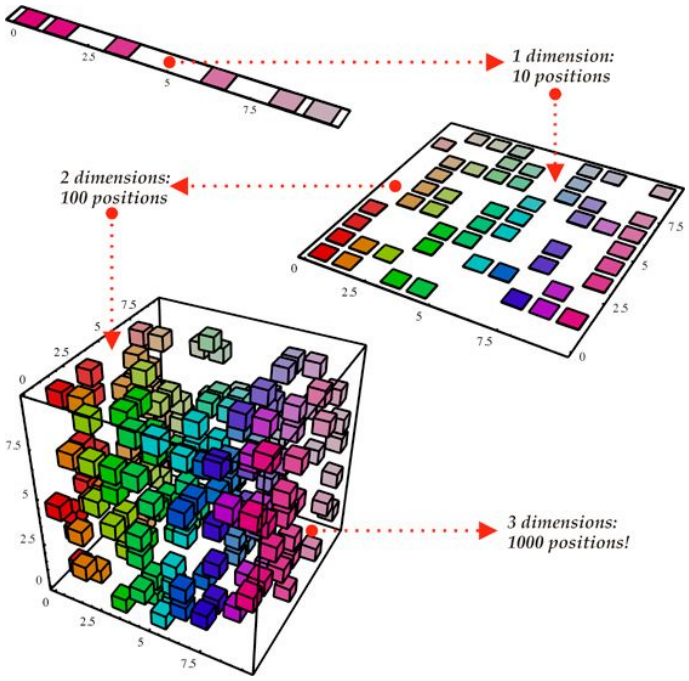
Figure 15: Plots of F-score, precision, recall, for all numbers of concurrent markers

It seems that positive cases are detected at 95% (recall), except for 93% at 5 marker. Otherwise, the model becomes more discerning of negative patients (precision) until the 5 marker combination level. Afterwards, precision begins to drop again. F-score, as a combination of recall and precision, tops out with precision at 4 and 5 concurrent markers.

In addition, our regularization term has stayed relatively consistent despite the feature space ballooning into several thousands. Reassuring.

Discussion:

Figure 16: Curse of dimensionality



Overall, our original question of if machine learning can predict patient disease status has been answered. Our SVM algorithm performs at a consistent rate of detecting 95% cases of leukemia. The fact that this is consistent despite how many concurrent markers you consider lends some validation to the current method of diagnosis using 1 and 2 dimensional plotting of markers.

Our algorithm's precision is higher as it reaches 5 dimensional histogramming, but then seems to fall off again. An explanation for this may be that more and more information is gained as you consider marker levels in combination with each other, but are fighting the "curse of dimensionality" the entire time (figure 16). That is, with every dimension added to your histogram, the "space" within which your data exists becomes exponentially larger. Perhaps the information gained from more markers "wins out" to begin with but then succumbs to dimensionality, giving rise to the peak at 4 and 5 concurrent markers. In any case, SVM does demonstrate robustness in spite of a sample size of 359 and a feature space expanding into the several thousands by comparison.

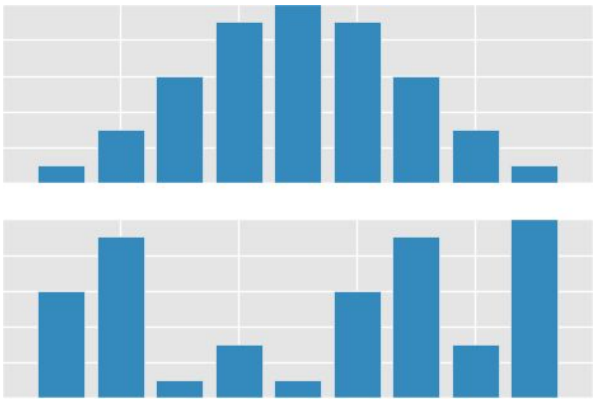


Figure 17: Weakness of histogram method is unordered feature space

That recall stays consistent may be due to the nature of our data; that the vast majority of positive cases are very readily separable by flow cytometry. Supporting this theory is that every model appears to have missed the same two cases: patients #6 and #101. Since the two cases are consistent against all 7 models, they may both have something in common that our model is not able to detect. One glaring weakness to our model is that to the SVM, two features residing next to each other in the input array are completely unrelated. For example, a histogram with a picturesque bell curve shape is identical to the same bell curve histogram but with the bins randomly rearranged (i.e. the two histograms in figure 17 are equivalent). It is possible that to detect positives in patients #6 and #101 are cases, this would have to be addressed.

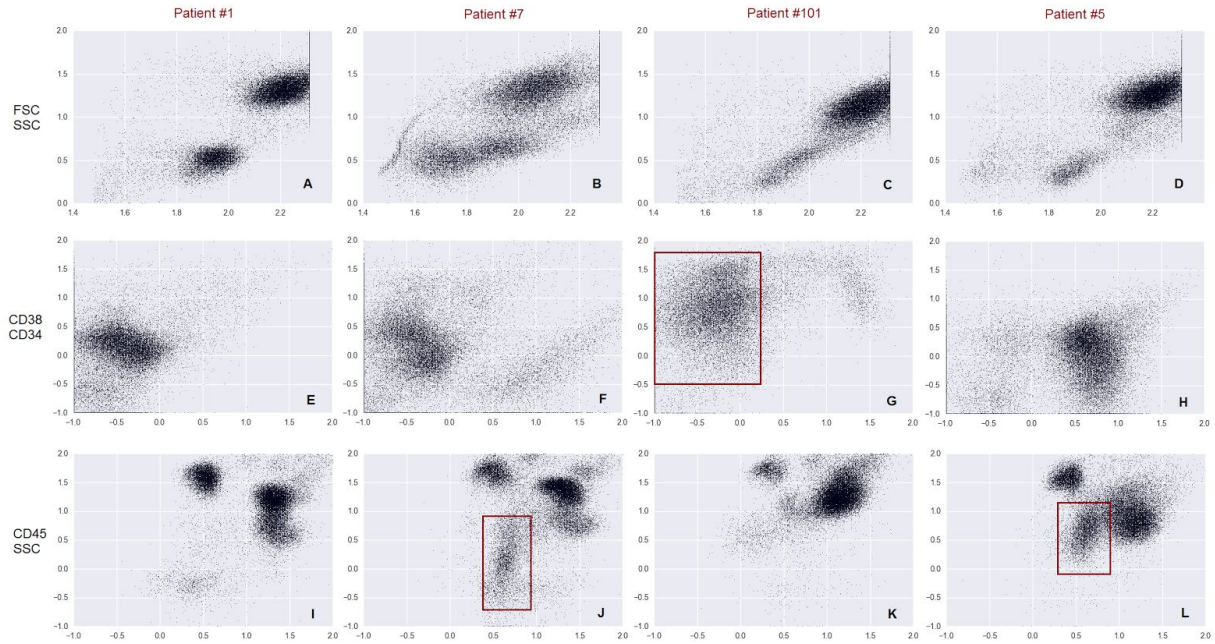


Figure 18: Closer examination of common misclassifications. Column 1- patient 1, healthy. Column 2, 3- patients 7 and 101, respectively (leukemic but misclassified by every model). Column 4, patient 9, leukemic but classified correctly. Row 1- FSC vs SSC, row 2- CD34 vs CD38, row 3- CD45 vs SSC.

It is also possible that these two cases represent difficult to detect “edge” cases. Preliminary investigation of these two cases suggests that there may be some separability as suggested by the plots in figure 18. The rows are matching plots that are typically one of the first inspected by pathologists during routine evaluation. The red boxes indicate some of the suspicious populations that would certainly increase a human’s degree of suspicion. Again, this may be a result of inherent weakness of the histogram method. Future study may include a different way of summarizing the cell population data, exploring alternate methods such as K-means.

Another weakness that may be responsible, to be addressed in the future, includes the relative weighting of marker combinations in diagnosis. Some markers have more predictive value than others; certainly the ones plotted in the figure are generally significant. It is possible that the signal from these combinations is being diluted by opposing or neutral signals from other markers. This is an area that a tree boosting algorithm such as XG Boost may certainly help.

Despite the weaknesses, it is somewhat telling that the precision and overall F-score of our model continue to rise as we approach 5 concurrent markers, a data space unable to be perceived by humans. It

suggests that extracting maximal information from the data may certainly involve viewing the data from more than two dimensions, as was hypothesized.

Furthermore, the histogram bin count employed for summarizing cellular data may be optimized in the future. In fact, a range of bin values was investigated for 2 dimensional data, revealing that the test f-score rose significantly from 10 bins to 300, after which it plateaued onwards past 1000 bins. 10 bins was chosen not for accuracy but for memory constraints; for every dimension added the data space and hence possible bins increases exponentially by the number of bins. However, it is not a given that more bins is better in high dimension space, as the histogram method suffers especially from the curse of dimensionality as discussed previously.

As of now, this model can be used to help screen cases, as for 5 markers there was a perfect record of 0 false positives. That is, if a positive is predicted, we can be very certain the case needs more attention. Likewise, once refined to detect 100% of actual ill cases, it may serve as a negative screen; if the test predicts negative, we can safely say the patient is healthy and the diagnosis of leukemia can be ruled out. Lastly, this model is able to lend support to individual diagnoses as it uses information from high dimensional space not perceivable by humans.

Summary:

It was shown that using machine learning algorithms is a promising way to save time and energy to accurately read flow cytometry.

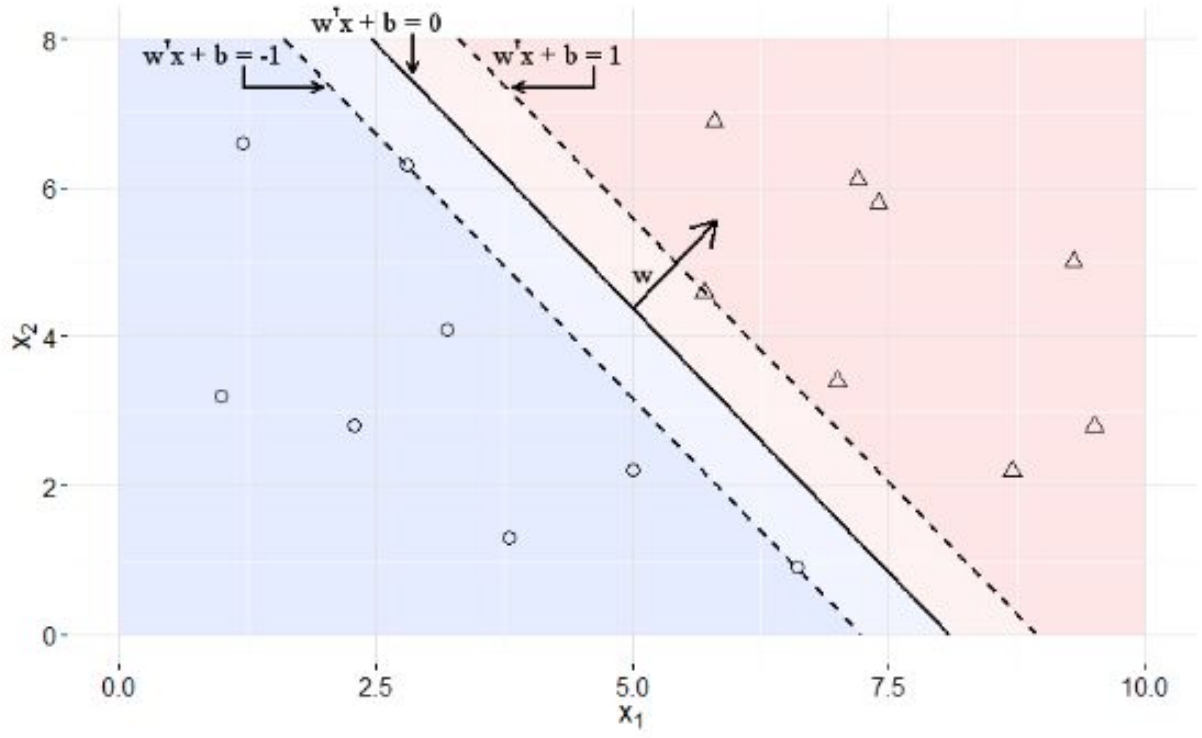
One of the main questions pertaining to the task of flow automation was of the fundamental structure of the dataset. Despite patients being only healthy or sick, there were 7 independent sub-samples per patient that required separate models each. Once predictions were made for each tube, a method of integrating the 7 models into an ensemble predictor was needed.

The main focus of this project, however, was making predictions for each tube. There were 30,000 different cells per tube, and being biological data, had no guarantee of linear relationships, pre-defined distribution, or otherwise. As such, histogramming was chosen as a method of feature engineering to be descriptive of cell distributions rather than individual samples themselves.

The SVM machine learning algorithm was chosen for its specific properties, and it was trained and predicted on histogramming data with regularization and 5-fold cross validation. Predictions for every patient and every tube were made, and a single overall prediction for each patient was created. This was repeated for increasingly many combinations of markers as a way of exploring the potential of computation to surpass the current 2-dimensional plotting pathologists currently use.

The results demonstrated that flow automation already has potential for practical applications. When considering 5 markers at once (which is beyond human perception), the model showed scores indicative of a perfect screening test. Meanwhile, sensitivity for the test were extremely promising for utility as a negative screen and diagnostic tool in general.

APPENDIX A - The Math Behind SVM



For a set of two binarily labeled sets of data, there exists a “hyperplane” in between the sets that separates the two. For simplicity, we can consider two dimensional data first, for which the hyperplane is a line. Surrounding the line on either side are two halves of the boundary zone that contains no data. There are many possible lines to choose (actually, usually infinite), but the goal of the support vector machine is to find the line for which this boundary is as wide as possible.

The general equation for the line or hyperplane is: $w_T x + b = 0$ where w is called the weight vector perpendicular to the margin and b is some constant to move the line away from the origin. We may label positive examples +1 and negative examples -1 such that positive samples are all $w_T x + b \geq 1$ and $w_T x + b \leq -1$.

Consider two points as close as possible to each other along these boundary lines $w_T x_+ + b = 1$ and $w_T x_- + b = -1$. The magnitude of the vector between these two points is equivalent to the width of the boundary, and furthermore, the line is parallel to the weight vector w. The difference between these equations gives $w_T(x_+ - x_-) = 2$. Since w and the vector between the points are parallel, this is also equivalent to $||w_T|| * ||(x_+ - x_-)|| = 2$. Divide by w to get the equation for the width of the margin: $||(x_+ - x_-)|| = 2 / ||w_T||$. We then see that we maximize the boundary by minimizing w.

You can convert this into a quadratic programming equation: subject to the constraints of: $0 \leq \alpha_i$ and $y \sum \alpha_i y_i = 0$. The terms α can best be thought of as weights for each sample, and it turns out that once solved, most α are equal to zero. The points with non-zero α are those that lie close to the boundary that contribute the most to its definition. These points are what are called the support vectors.

References and further reading:

Wood BL. Myeloid malignancies: myelodysplastic syndromes, myeloproliferative disorders, and acute myeloid leukemia. *Clin Lab Med.* Sep 2007; 27(3):551-575, vii.

Cherian, Sindhu, M.D. Flow Cytometry for Acute Myeloid Leukemia. https://www.cytometry.org/public/educational_presentations/Cherian.pdf.

Aghaeepour, N., Finak, G. Critical assessment of automated flow cytometry data analysis techniques. FlowCAP Consortium. January 10, 2013.

Figures:

Binary DNA:
https://3c1703fe8d.site.internapcdn.net/newman/gfx/news/hires/2016/fromlivingco.jpg

Fluorescent flow antibody:
http://www.signalsblog.ca/inside-a-cancer-stem-cell-researchers-toolbox-csc-markers-flow-cytometry/

2d SVM picture: http://dni-institute.in/blogs/building-predictive-model-using-svm-and-r/

SVM kernel trick:
https://www.quora.com/Support-Vector-Machines-How-does-going-to-higher-dimension-help-data-get-linearly-separable-which-was-non-linearly-separable-in-actual-dimension

Overfitting: http://docs.aws.amazon.com/machine-learning/latest/dg/images/mlconcepts_image5.png

Regularization:https://openi.nlm.nih.gov/detailedresult.php?img=PMC2547983_pcbi.1000173.g003&req=4

Curse of dimensionality: https://haifengl.files.wordpress.com/2016/01/cursedimensionality.jpg