**Unsupervised Learning: (using image segmentation techniques)**

The purpose for the unsupervised learning part is to separate the data into two clusters, where in each cluster, most of data come from a single source. The method employed here is based on Image segmentation algorithm, which asks for the user interaction.

1. **Measurement and initial approach**

Measurement Metrics: L-> true label, P-> predicted label

With this metric, the worst result is 0.5 accuracy. At the starting stage, a kmean++ is used with original high-dimension 4-mer matrix, F. The accuracy obtained is 0.60, and we obviously want to do something better than that.

To visualize the difficulty of the problem, we apply two visualization techniques, T-SNE (t-distributed stochastic embedding) and SIMLR (Single-cell Interpretation via Multi-kernel Learning) which is the state of the art technique for visualization and dimension reduction. The Fig 1 is 2d plots after reducing dimension of F, where color indicates the true labels. Hence from the plots, without knowing some of the true labels, the appropriate clustering is very difficult. But low dimensional data points has some internal structure that keeps them stay together, instead of mixed together, which might make this task impossible. Fig 2 shows the low dimension result based on Quality scores matrix Q.

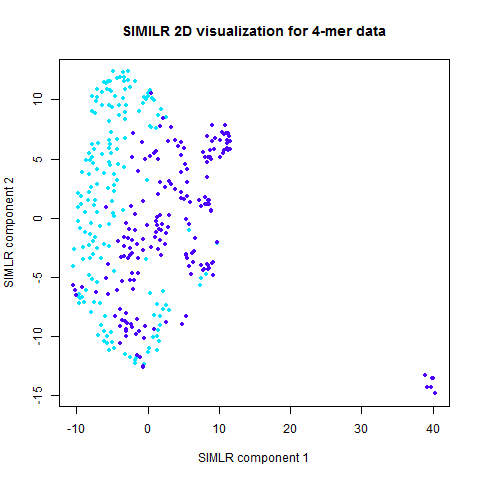
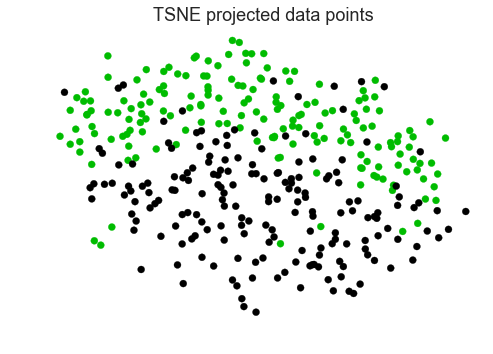


Fig 1

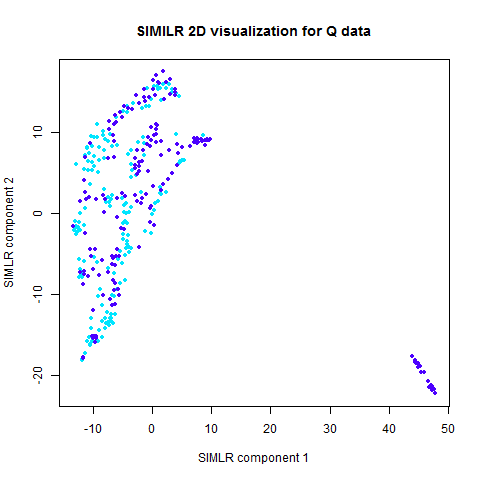
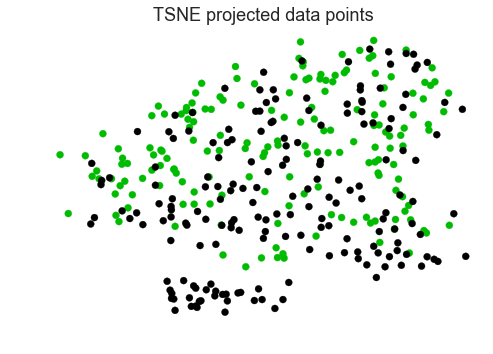
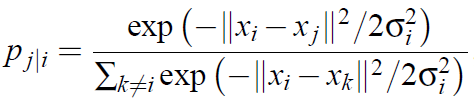
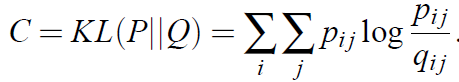
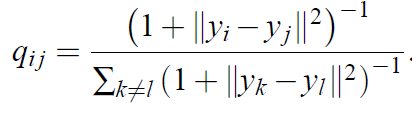


Fig 2

1. **Brief discussions on TSNE and SIMLR**

t-SNE is a dimension reduction algorithm developed from SNE with the aim to preserve as much of the important structure of the high dimension data inside the low dimension data. In this framework, each single point has a distribution of potential neighbors on all other points, defined as Pj|i which translates as probability that j is I’s neighbor, which means each data has its internal view about all other points.

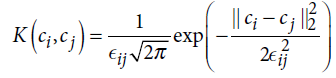
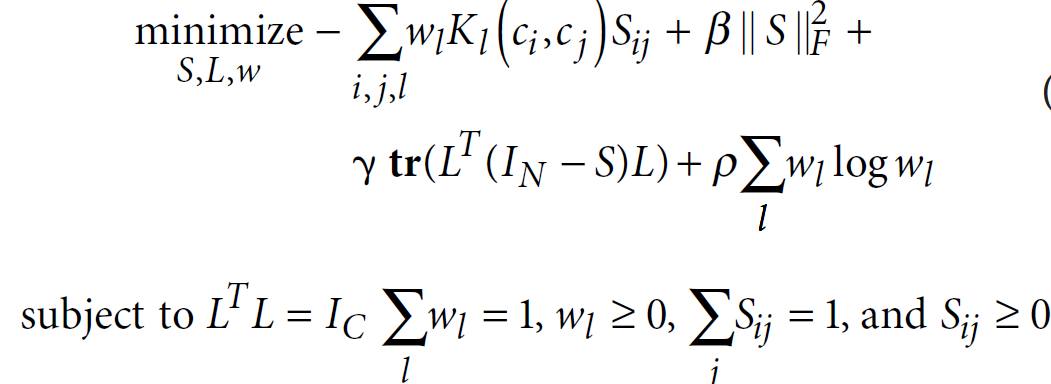
, , ,



In the above equation, P stands for high-dimensional joint probability, Q stands for low dimensional joint probability. The cost function is based on KL divergence between the high-d, hence minimizing the cost is equivalent to minimize the difference distribution difference between P and Q. This function is not convex, hence the method concludes several methods to improve the clustering, which avoids the “crowding problem”, which states that all the points get reduced to a small region, because there is not enough space to so many high-dimension points.

SIMLR is currently the state of the art dimension reduction technique. As the name implies, it uses multiple kernels to give the best estimate the dimension reduction result.

It tries to minimize the following cost function:



Where w is the weight vector for each kernel, L is a matrix to enforce the number of clusters (this result comes from spectral clustering theorem), and S is the similarity matrix.

Result from Kmeans++ with the above metrics:

t-SNE: 0.573

SIMLR: 0.615

1. **Our approach:**

Given that both algorithm gives poor clustering performances. We do not attempt to propose new algorithm like above. Instead, we try to solve an easier problem, that given some points has known label, and the task is to classify another unlabeled node. This problem formulation is like the supervised learning framework, but the number of known data is not required to be many.

Then we borrow some techniques from the image segmentation algorithm, because segmenting out an object from a complex background is ill-conditioned, and in natural images background is always complex. Hence human interaction is necessary in this case to assist the segmenting process. Fig 3 is one of the popular segmentation algorithm, called graphcut, which asks user to identify some fore-ground and back-ground points before segmentation.



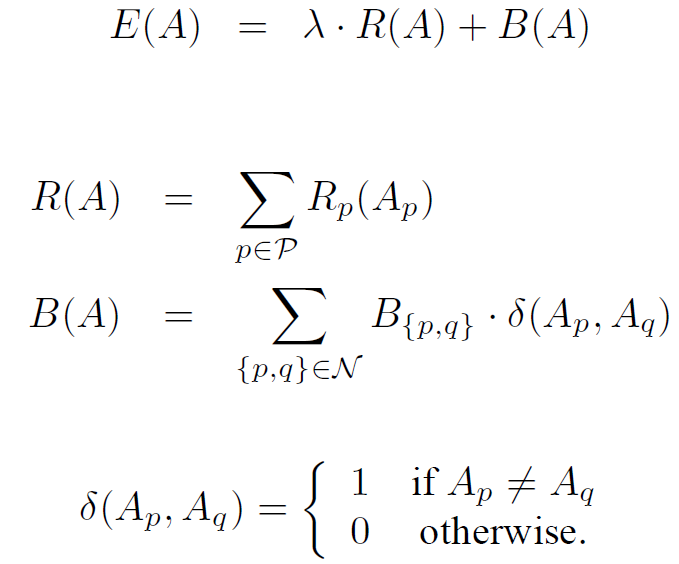


Fig 3

It is easier to start with original version, and then formulated our problem in the similar way.

**Image Segmentation Setting:**

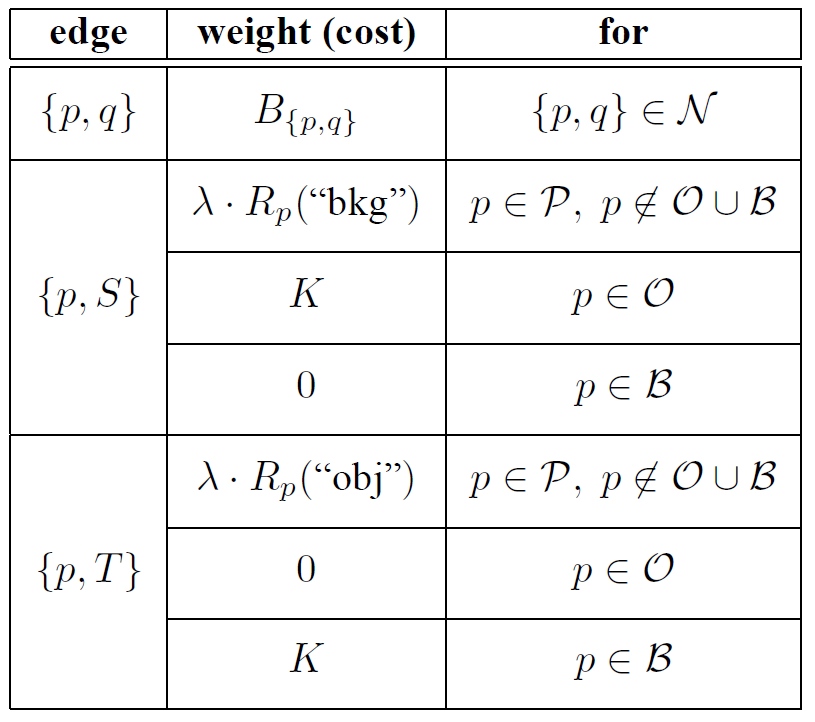
The graphcut algorithm tries to minimize the following cost function



Where the R(A) represents the data term which try to fit the optimal model for the measurement; B(A) represents the smoothness term, which comes from some prior knowledge.

A is a vector with each element being a node, and the node is either an object or a background; P denotes all pixels in an image; R is a metrics which measures the fitness of a node Ap if we know the category of the node; B is a metrics which measures the similarity between nodes.

The segmentation problem can also be viewed as the following form. Define a graph, G, with each pixel being a node. In the image setting, there is a natural definition for edges, which are the surrounding four other pixels. In addition, define a source node and a terminal node, and connect all pixels to those two nodes. See Fig 4 below for illustration.



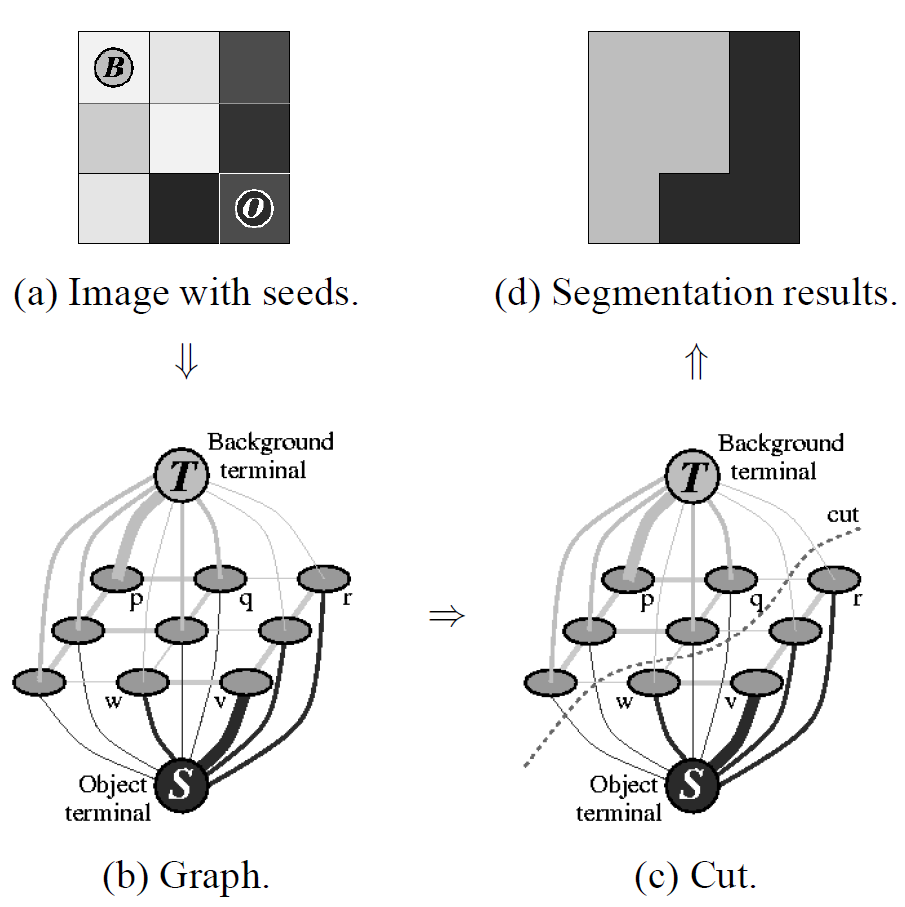


Fig 4

It can be shown that the solution of max-flow min-cut algorithm to the above graph is the solution to the previous cost function.

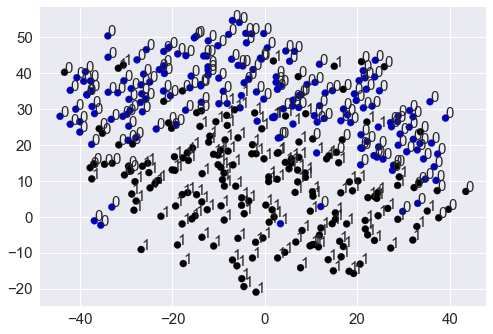
**Our Setting:**

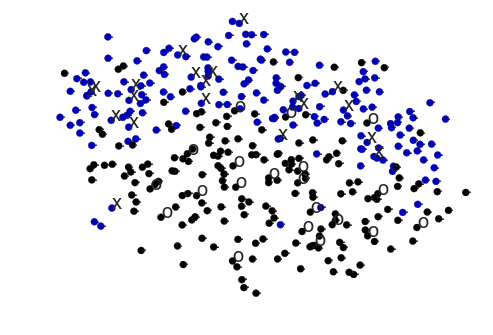
Since there is not a natural edge definition in the general graph, we define the neighbor edge, n-link, to be the neighbor probability in t-SNE section and data used here is high dimensional data. To reduce, the computation and noise, we only take highest 20 probable neighbor for each node. Hence the function B(p,q) has been defined. For data fitness part, Rp(), it is unreasonable to fit a histogram in high-dimension data due to curse of dimensionality and we don’t have enough data. Therefore R(p) is built from dimension reduced data, i.e. compute a histogram based on labelled object/ background data, and when encountered a new data, we fit the location info into that histogram and generate the scores.

**Result:**

Accuracy: 87.7%

Interesting Features:

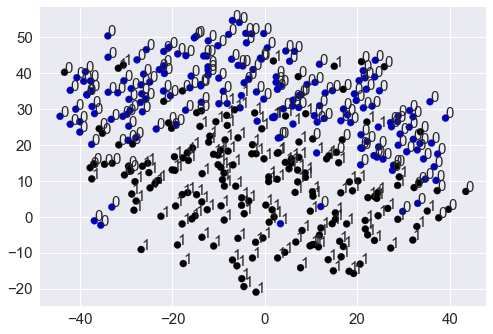




**Potential problem**

The current algorithm depends heavily on the choice of labelled point. Some reasonable choice of nodes might lead to bad performance, but currently there isn’t a decision rule for it. I am considering to use the max flow score to make the judgement, but it is left to the future.

Another simple methods for this task is to simply perform Kmeans, the result: 0.847



Evaluation if some points are identified incorrectly, given foreground background each 20 points

|  |  |  |
| --- | --- | --- |
|  | Graphcut | Kmeans |
| 0 mistake | 87.7% | 84.7% |
| 1 mistake | 86.7% | 81% |
| 2 mistake | 84.9% | 77% |