8-9-2006

External Support Vector Machine Clustering

Charlie McChesney
University of New Orleans

EXTERNAL SUPPORT VECTOR MACHINE CLUSTERING

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

by

Charles B. McChesney III

B.S. Computer Science University of New Orleans, 2004

August 2006

Table of Contents

Abstract. The external-Support Vector Machine (SVM) clustering algorithm clusters data vectors with no *a priori* knowledge of each vector's class. The algorithm works by first running a binary SVM against a data set, with each vector in the set randomly labeled, until the SVM converges. It then relabels data points that are mislabeled and a large distance from the SVM hyperplane. The SVM is then iteratively rerun followed by more label swapping until no more progress can be made. After this process, a high percentage of the previously unknown class labels of the data set will be known. With sub-cluster identification upon iterating the overall algorithm on the positive and negative clusters identified (until the clusters are no longer separable into sub-clusters), this method provides a way to cluster data sets without prior knowledge of the data's clustering characteristics, or the number of clusters.

I. Introduction

An ideal clustering algorithm can perfectly determine the clusters in a data set with no supervision regardless of the shape, noise-level, and similarity of the clusters in the set. At present, none of the popular clustering algorithms are ideal. Instead, some methods perform strongly on similar clusters, but weakly on arbitrarily shaped clusters. Other methods perform very well on non-overlapping clusters, but become unreliable when the data contains noise. And the best performing methods require supervision, often requiring the tuning of multiple dependant parameters by an experienced user.

In this paper, an External SVM clustering algorithm is presented. This algorithm provides an alternative method to the current clustering algorithms, which (when automated) requires no supervision, easily distinguishes arbitrarily shaped clusters, is resistant to noise, and separates clusters that strongly overlap. The performance of this algorithm is examined on real DNA hairpin data and two-dimensional artificial data, and is compared to the results of the other popular clustering methods.

II. Background

Support Vector Machines

SVM's are learning machines, which provide good generalization performance for classification tasks. They have been used for pattern recognition in many tasks including: isolated handwritten digit recognition, object recognition, speaker identification, face detection in images, and text categorization. This section will provide a brief introduction to the inner workings of SVM's. For a more detailed explanation of SVM's, see the papers of Scholkopf, Burges, and Vapnik of Bell Laboratories [12].

The goal of SVM's is, for a given learning task, with a given finite amount of training data, to achieve the best generalization performance by striking the right balance between the accuracy attained on the particular training set, and the ability of the machine to learn

the training set without error [12]. In comparison with neural net learning machines, which overfit easily based on local anomalies, SVM's use structural risk minimization to minimize the risk of overtraining and provide unique global solutions.

To begin, let us assume there is a learning problem with *l* observations. Each observation consists of a data vector, *xi*, and an associated label, *yi*. The task of the SVM is to learn the mapping of each x_i to each y_i. The function $f(x,\alpha)$, where α is an adjustable parameter, will determine this mapping. This function is deterministic, in that for a given *x*, and a choice of α, it will always give the same output. A particular choice of α generates a trained machine. The expected test error of a trained machine is therefore:

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x}, \alpha)| dP(\mathbf{x}, y) \qquad (1)$$

The goal of structural risk minimization is to reduce the expected risk of error. This is obtained through the following equation:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l}\right)} \qquad (2)$$

Here $R_{emp}(\alpha)$ is the empirical risk. The empirical risk is defined to be the measured mean error rate on the training set. The letter *h* is a non-negative integer called the Vapnik Chervonenkis (VC) dimension, and is a measure of the amount the machine can learn without error. $\eta$is a chosen loss parameter between 0 and 1. The second term on the right hand side is the "VC confidence". The above risk minimization equation says that given several learning machines, defined by $f(x,\alpha)$, we can calculate each machine's risk. This is the essential idea of structural risk minimization. Given a fixed family of learning machines for a particular task, we can then choose the machine that has the least risk.

Now, lets discuss the VC dimension. Given *l* points, and for each labeling, if a member of the set *{f(x)}* can be found which correctly assigns those labels, then that set of points is "shattered" by that set of functions. The VC dimension for the set of functions *{f(x)}* is defined as the maximum number of training points that can be shattered by *{f(x)}*. SVM's define a hyperplane which separate the data point's labels. On one side of the hyperplane labels will be positive (denoted by 1), and the other side negative (denoted by –1). We will now introduce a theorem.

THEOREM- *Consider some set of m points in* $\boldsymbol{R}^n$. *Choose any one of the points as origin. Then m points can be shattered by oriented hyperplanes if and only if the position vecors of the remaining points are linearly independent [12].*

Corollary: The VC dimension of the set of oriented hyperplanes in $\boldsymbol{R}^n$ is *n+1*, since we can only choose *n+1* points, and then choose one of the points as origin, such that the position vectors of the remaining *n* points are linearly independent, but can never choose *n+2* such points (since no *n+1* vectors in $\boldsymbol{R}^n$ can be linearly independent) [12].

(Note: A proof of this theorem and corollary can be found in [13].)

The VC confidence is a monotonic increasing function of *h*. This will be true for any value of *l*. Thus, given some selection of learning machines, one wants to choose the learning machine whose associated set of functions has a minimal VC dimension. This will lead to an upper bound on the actual error. In general, one wants to reduce the VC confidence. (Note: This only acts as a guide. Infinite VC dimension (capacity) does not guarantee poor performance).

Structural risk minimization (SRM) finds the subset of the chosen class of functions (for a training problem), so that the risk for that subset is minimized. This is done by introducing a "structure" which divides the entire class of functions into a set of method subsets. Structural risk minimization then consists of finding that subset of functions, which minimize the bound on the actual risk. This can be done by training a series of

machines, one for each subset, where for a given subset the goal of training is to minimize the empirical risk. Then take that trained machine in the series whose sum of empirical risk and VC confidence is minimal.

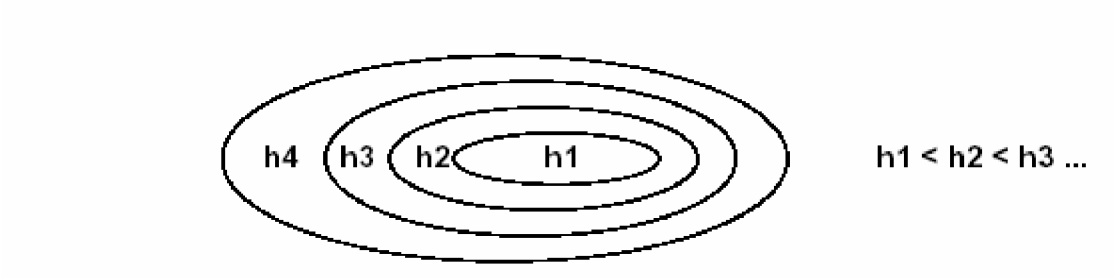


Figure 1. Nested subsets of functions, ordered by VC dimension.

Now we will illustrate how the hyperplane separates the different classes of data vectors. The following constraint determines the class for each vector.

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \tag{3}$$

This constraint is replaced by constraints on Lagrange multipliers, so that the training data appears in the form of dot products between vectors. This property allows generalization to the non-linear case. Below are the primal and dual Lagrangian.

$$L_P \equiv \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{4}$$

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{5}$$

The $\frac{1}{2}||W||^2$ term in (4) represents the structural risk minimization term corresponding to the hyperplane having the largest possible margin, and thus limiting the risk of error. L_P and L_D arise from the same objective function, but with different constraints. The

solution is found by minimizing L_P or maximizing L_D. Support vector training therefore amounts to minimizing L_P or maximizing L_D with respect to certain constraints. There is a Lagrange multiplier for every training point. In the solution, those points for which $\alpha_i > 0$ are called support vectors, and lie on one of the hyperplanes H_1 or H_2. All the other training points have $\alpha_i = 0$ and lie on H_1 or H_2, or lie on the positive or negative side of H_1 or H_2 *[12]*.

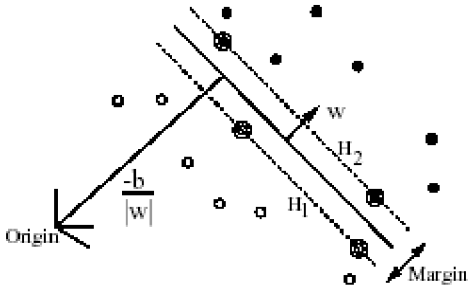


Figure 2. Linear separating hyperplane for the separating case. The support vectors are circled.

Now we will discuss the Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are satisfied at the solution of any constrained optimization problem, with any kind of constraints, provided the intersection of the set of feasible directions with the set of descent directions coincides with the intersection of the set of feasible directions for linearized constraints with the set of descent directions. The KKT conditions are analogous to the equations of motion that are obtained from the Lagrangian in classical mechanics. Solving the SVM problem is equivalent to finding the solutions of the KKT conditions. Below are KKT conditions corresponding to the Lagrangian shown in equation *(4)* (see [12] for further details).

$$\frac{\partial}{\partial w_\nu} L_P = w_\nu - \sum_i \alpha_i y_i x_{i\nu} = 0 \quad \nu = 1, \cdots, d \qquad (6)$$

$$\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i = 0 \tag{7}$$

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad i = 1, \cdots, l \tag{8}$$

$$\alpha_i \geq 0 \quad \forall i \tag{9}$$

from $\partial / \partial \alpha_i L_P = 0$, when α_i is restricted to be >=0.

$$\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0 \quad \forall i \tag{10}$$

this results from combining (8) and (9) under the conditions where their negative contribution to the Lagrangian must be reduced to zero for the Lagrangian L_P to be maximized.

Now we will explain how the kernel is used. We need to put $\mathbf{x}_i \cdot \mathbf{x}_j$ into the form of dot products. Mapping to some other (possibly infinite) dimensional Euclidean space would cause the training algorithm to only depend on the data through dot products in the Euclidean space, *H*. Now, if there were a kernel function *K*, such that $K(x_i,x_j)=F(x_i) \cdot F(x_j)$, only *K* would be needed in the training algorithm (*F()* corresponds to mapping x_i and x_j into *H*). The kernel allows for the mapping of the training data into kernel space.

The Gaussian kernel is:

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2} \tag{11}$$

For this kernel function, *H* is infinite dimensional, and allows production of a support vector machine which lives in infinite dimensional space, and it also allows the SVM to train roughly in the same time as the unmapped data. Since most SVM's function best with kernels that operate in infinite dimensional spaces, the use of the VC-dimension and

VC-confidence is marginalized in actual application, beyond its contribution to the original incorporation of SVM concepts.

Clustering

Clustering is an unsupervised learning problem, whose goal is finding a structure in a collection of unlabeled data. A cluster is a collection of objects, which are similar to the other objects in that collection and dissimilar to the objects belonging to the other clusters [3]. Clustering can be used in almost any field. Some applications of clustering include search engine document classification, finding groups of customers with similar buying patterns, image clustering for computer forensics, DNA hairpin classification, and knowledge discovery in a general setting, where feature or concept primitives themselves need to be identified.

Distance measure is an important component of clustering algorithms. If the data vectors are all in the same physical units, then a Euclidean distance metric can be used. In some cases where the units do not match or are some nominal unit, domain knowledge must be used to formulate a suitable distance formula. Still, in cases where Euclidean distance can be used, the choice of the mathematical formula used for the clustering distance measure will affect the clustering groupings.

The two basic types of clustering are hierarchical and partitional [6]. Hierarchical clustering finds new clusters using previously established clusters [8]. Partitional clustering can be further subdivided into exclusive, overlapping, and model-based clustering [2]. Exclusive clustering groups data points in an exclusive way, so that if a certain point belongs to one cluster, then it cannot be included in another cluster. Overlapping clustering uses fuzzy sets to cluster data, so that each point may belong to more than one cluster with different degrees of membership. Each point will be related to each cluster by its membership value. Finally, model-based clustering uses models for clusters and attempts to optimize the fit between the model and the data. Next, the four most used clustering algorithms representing each type will be introduced.

Hierarchical Clustering

"Bottom-up" hierachical clustering is based on the union between the two nearest cluster. Initially, all data vectors will represent their own cluster, and iteratively clusters will be joined by union to the nearest cluster, until the final desired cluster representations are reached. Hierarchical clustering can also be approached from the "top-down". In this approach, all data vectors are initially members of the same cluster. This cluster will be split into two clusters by optimal bisection, and then iteratively all new clusters will be split until the desired number of clusters is reached. The External-Relabel SVM clustering algorithm uses a variation of "top-down" hierarchal clustering for multiple cluster identification.

K-means Clustering

K-means clustering is an example of exclusive clustering. The procedure classifies a given data set through a certain number of clusters ("K" clusters are chosen a priori). The following steps represent the algorithm:

1. Randomly place K points into the data space. These points represent the initial cluster centroids.
2. Assign each data vector to the cluster that has the "closest" centroid.
3. When all the data vectors have been assigned, recalculate the position of the K centroids. This calculation is done making each centroid the minimizer of the distance-based objective function of its associated cluster.
4. Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the data objects into clusters for which the chosen Euclidean distance squared objective function is calculated.

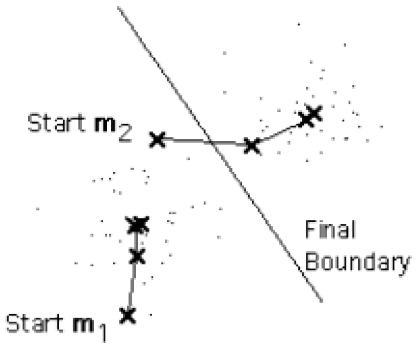


Figure 3. Illustrates the k-means algorithm steps, from initial centroid placement to final cluster definitions.

In general, the algorithm aims to minimize some *objective function.* The objective function used in k-means algorithms is the squared distance function, which consists of a double summation on $||x_{ij}-c_j||^2$. Here $||x_{ij}-c_j||^2$ is the chosen distance measure between a data vector x_{ij} and the cluster centroid c_j. This objective function is an indicator of the variance of the n data vectors from their respective cluster centroids.

There are several problems with the K-means algorithm. First of all, since the number of clusters must be known a priori, the algorithm by itself cannot solve problems where the number of clusters is unknown. Next, the algorithm does not always find the most optimal clusters because the objective function used does not always use the best distance metric for the problem. Finally, the location of the initial random cluster centroids can heavily influence the resulting clusters, creating the probability that the final clusters will be trapped in a local minima of the objective function.

Kernel K-means Clustering

Kernel K-means is a K-means clustering algorithm, in which a kernel is used to map the data into a higher dimensional space. Kernel functions perform a non-linear transformation of the data by mapping it into a space where the separability of the data is increased [5]. The distance measure, $||x_{ij}-c_j||\text{^}2$, in the objective function of the K-means algorithm is therefore transformed by some kernel function [11]. In this paper, the Kernel K-means step is used to identify roughly identify each data vector's cluster before

the data vectors are run through an SVM, which improves the clustering accuracy by dropping data with weak clustering scores.

Robust Kernel Fuzzy Clustering

Robust Kernel Fuzzy clustering is an example of overlapping clustering, which allows each data vector to belong to more than one cluster. It is an alternative to the popular Fuzzy C-Means clustering, which is an overlapping clustering method developed by Dunn and extended by Bezdek [6]. The Robust Kernel Fuzzy clustering method adopts the fuzzy partition matrix in its objective function (from Fuzzy C-Means clustering), $x_{ij}||\phi_i - r_j^{\phi}||^2$. The fuzzy partition matrix, x_{ij}, allows data points to have membership values in each of the clusters [9]. In contrast to the Fuzzy C-Means method, the Robust Kernel Fuzzy clustering uses a kernel, which is incorporated to allow the method to recognize arbitrarily shaped clusters [10]. The method gains robustness through the modification of the Euclidean distance formula in its objective function from $\|d_i - r_j\|^2$ to $1 - e^{-\gamma\|d_i - r_j\|^2}$ [9]. This clustering algorithm is similar to the K-means algorithm, in that it aims to minimize its objective function when determining a data vector's cluster membership. It is different, because it includes a fuzzy partition matrix, which scores each data vector's membership value with every cluster in the problem. The Robust Kernel Fuzzy clustering algorithm implemented for this paper is described in the Methods section.

Mixture of Gaussians

Mixture of Gaussians is an example of model-based clustering. In this approach, clusters are considered Gaussian distributions centered on their centroids. The Expectation Maximization (EM) [4] algorithm is used to find the Gaussian distributions, which model the data. In the clustering process used in this paper, Gaussian distributions model clusters in both the preprocessing (Kernel K-means) and External-Relabel SVM clustering phases.

DNA Hairpin Data

The DNA hairpin data clustered in this paper is created by running raw data through a tFSA/HMM, which creates a data set that contains 151 feature vectors for each element. A description of the raw data can be found in Appendix E of [15].

III. Methods

Single Class SVM Clustering

The Single Class SVM clustering method was developed by Vapnik [14]. It is able to perform multi-cluster separation in a single SVM run, by enclosing clusters in hyperspheres. A hypersphere is similar to a hyperplane in that it is a boundary, in possibly infinite dimensional kernel space, which separates data vectors of different classifications. It is different than a hyperplane because, instead of only separating the data vectors with different classifications, it surrounds the data allowing for more than 2 classes to be defined. For clusters where there is not total separation between the respective members, a drop zone must be added to split the clusters. The following paragraphs describing this clustering method are from our recent publication on SVM classification and clustering in the MCBIOS proceedings in BMC Bioinformatics [1].

Let $\{x_i\}$ be a data set of 'N' points in R^d. Using a non-linear transformation φ, we transform 'x' to some high-dimensional space called Kernel space and look for the smallest enclosing sphere of radius 'R'. Hence we have: $\|\varphi(x_j) - a\|^2 \leq R^2$ for all j = 1,…,N; where 'a' is the center of the sphere. Soft constraints are incorporated by adding slack variables 'ζ_j':

$$\|\varphi(x_j) - a\|^2 \leq R^2 + \zeta_j \text{ for all } j = 1,\ldots,N$$
$$\text{Subject to: } \zeta_j \geq 0 \qquad (12)$$

We introduce the Lagrangian as:

$$L = R^2 - \sum_j \beta_j (R^2 + \zeta_j - \|\varphi(x_j) - a\|^2) - \sum_j \zeta_j \mu_j + C \sum_j \zeta_j$$

Subject to: $\beta_j \geq 0$, $\mu_j \geq 0$, *(13)*

where C is the cost for outliers and hence $C\sum_j\zeta_j$ is a penalty term. Setting to zero the derivative of 'L' w.r.t. R, a and ζ we have: $\sum_j\beta_j = 1$; $a = \sum_j\beta_j\varphi(x_j)$; and $\beta_j = C - \mu_j$.

Substituting the above equations into the Lagrangian, we have the dual formalism as:

$W = 1 - \sum_{i,j}\beta_i\beta_jK_{ij}$ where $0 \leq \beta_i \leq C$; $K_{ij} = \exp(-||x_i - x_j||^2/2\sigma^2)$

Subject to: $\sum_i\beta_i = 1$ *(14)*

By KKT conditions we have: $\zeta_j\mu_j = 0$ and $\beta_j(R^2 + \zeta_j - ||\varphi(x_j) - a||^2) = 0$.

In the kernel space of a data point 'x_j' if $\zeta_j > 0$, then $\beta_j = C$ and hence it lies outside of the sphere i.e. $R^2 < ||\varphi(x_j) - a||^2$. This point becomes a bounded support vector or BSV. Similarly if $\zeta_j = 0$, and $0 < \beta_j < C$, then it lies on the surface of the sphere i.e. $R^2 = ||\varphi(x_j) - a||^2$. This point becomes a support vector or SV. If $\zeta_j = 0$, and $\beta_j = 0$, then $R^2 > ||\varphi(x_j) - a||^2$ and hence this point is enclosed within the sphere.

Kernel K-Means Clustering

The data is first mapped into kernel space. The following absdiff kernel is used:

$$\exp(-1/2(\sum|x_i-x_j|) / 2\sigma^2) \quad (15)$$

(Note: For more information on the absdiff kernel, check out Appendix A.)

Next the K-Means algorithm is performed on the data in kernel space.

The K-Means implementation used in this paper is very similar to the K-Means algorithm described in the background section, with a couple of exceptions. First of all, our implementation uses the constraint that every problem always has at most two clusters (one cluster can also be obtained when the data cannot be separated into two clusters). Further cluster identification is obtained through iteratively clustering the results of prior clustering runs. The other difference involves the definition of the initial clusters as

defined in the first step of the algorithm. Instead of randomly picking a cluster centroid in the data space explicitly, the initial centroids are defined by the initial random labeling of the data vectors. Each vector is randomly labeled with a 50% chance of being positive and a 50% chance of being negative, and the two centroids are defined as the barycenter of their respective data vectors. Hence the initial cluster centroids are implicitly random, since their locations are determined by the random locations of their members.

Robust Kernel Fuzzy Clustering

The Robust Kernel Fuzzy clustering method aims to perform better than the Kernel K-Means algorithm, through providing more resistance to noisy data through changing the Euclidean distance formula used in the objective function from $\|d_i - r_j\|^2$ to $1 - e^{-\gamma\|d_i - r_j\|^2}$ [9]. Data can be dropped by setting a minimum membership score requirement. Then, using the fuzzy partition matrix, the data points that do not meet the minimum score requirement for any clusters are dropped. This paper implements the algorithm described in [9] exactly. For a more thorough description of the algorithm, see [9].

External-Drop SVM Clustering

The External-Drop SVM clustering method improves the accuracy on a data set that has already been split into two clusters. In this paper, we use this method after the Kernel K-Means algorithm has separated a set of data into clusters. The External-Drop SVM then acts as a filter, dropping all data that isn't strongly identified with either cluster, and thus improving the cluster identifications. The SVM can then be iteratively rerun, defining a more accurate hyperplane for clustering the current data, and for cluster classification of new similar data. The, weaker linear kernel, $\mathbf{x}_i \bullet \mathbf{x}_j$, is used in the SVM, because it does not overfit the data in this circumstance. The algorithm is as follows:

1. After the data is labeled by the Kernel K-Means clustering step, it is run through a binary SVM.

2. All the data vectors, which are identified as support vectors (bounded and unbounded) by the SVM, are dropped.
3. The SVM is then rerun on the updated data set (updated data set = original data set – dropped data).
4. Repeat steps 2 and 3 until an acceptable amount of data has been dropped.

External-Relabel SVM Clustering

The External-Relabel SVM clustering algorithm clusters data by relabeling the data vectors, which are strongly mislabeled. Strongly mislabeled data are vectors, which are a large distance away from the hyperplane, relative to the other mislabeled data. The following is a simple step-by-step description of the basic algorithm used for External-Relabel SVM-clustering:

1. Start with a set of data vectors.
2. Randomly label each vector in the set as positive or negative.
3. Run the SVM on the randomly labeled data set until convergence is obtained (random relabeling is needed if prior random label scheme does not allow for convergence).
4. After initial convergence is obtained for the randomly labeled data set, relabel the misclassified data vectors, which have confidence factor values greater than some threshold (vectors with larger confidence factor values are farther away from the hyperplane).
5. Rerun the SVM on the newly relabeled data set.
6. Continue relabeling and rerunning SVM until no vectors in the data set are misclassified.

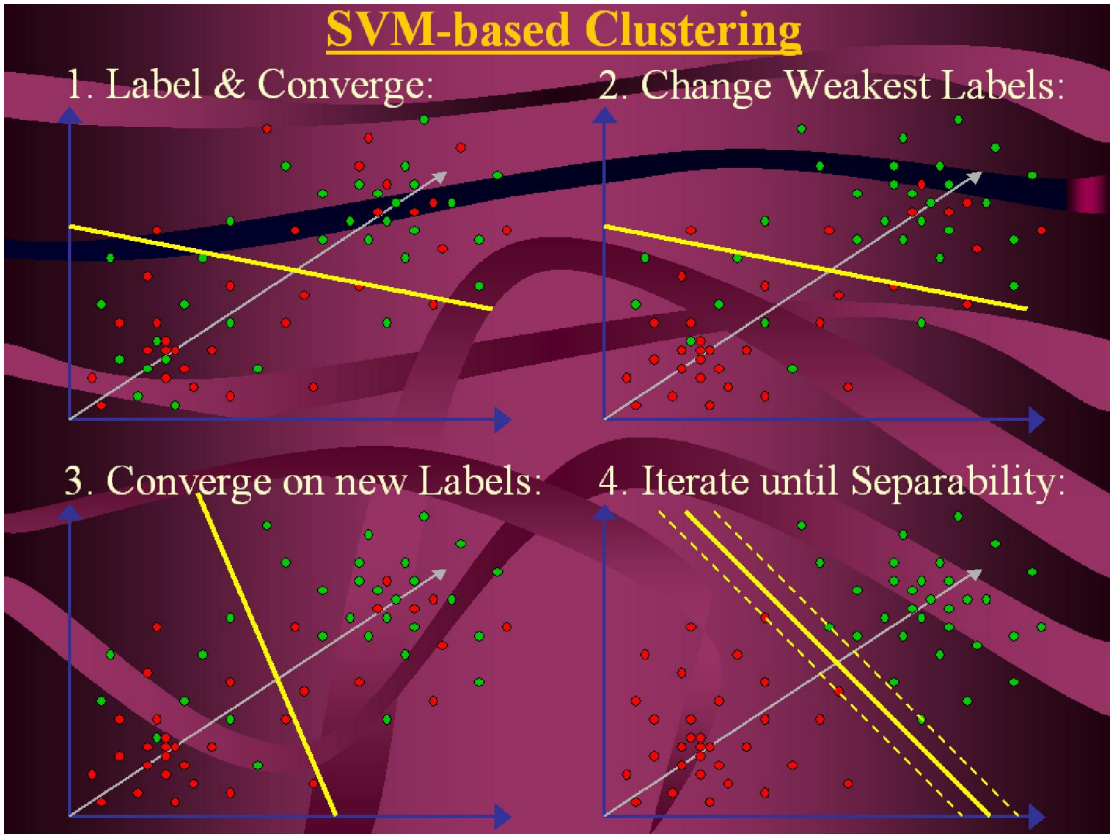


Figure 4. Illustrates the External-Relabel SVM clustering process.

Relabeling Methods

The relabeling function is implemented in several different overriding ways. For the results of this paper all misclassified vectors were flipped according to the following pseudo-code:

For positive vectors that are mislabeled,

```
if(abs(Ci)>abs(sum(Ci...Cj)/j)) {
        flip label;
}
```

(16)

Where C_i is the confidence factor value for the mislabeled vector,
and there are j mislabled positive vectors.

For negative vectors that are mislabeled,

```
if(Ci>(sum(Ci-Cj)/j)) {
        flip label;
}
```
(17)

Where C_i is the confidence factor value for the mislabled vector,
and there are j mislabled negative vectors.

It is important to note that this scheme considers the positive and negative mislabeled vectors as disjoint sets. This is done because the hyperplane created may be skewed to either the positive or negative side (confidence values of the positive misclassified vectors may be much higher than the confidence values of the negative misclassified vectors and vice versa). By separating the positives from the negatives, the scheme avoids improperly exploiting this imbalance.

Next, the simplest scheme used is the tui(textual user interface) scheme. In this scheme, statistics describing the misclassified vectors are output to the user. The statistics used are the value of the farthest outlier(farthest away from hyperplane), the value of the closest outlier(closest to hyperplane), the average value, the standard deviation, and the number of vectors. This information is then used to make a decision about which vectors should be flipped. The user enters a threshold value at which all misclassified vectors with greater confidence factor values are relabeled. The flipping conditions are:

Separately for the positive and negative vector sets that are mislabeled,

```
for(Ci...Cj) {
        if(abs(Ci)>abs(Cv)) {
                flip label;
        }
}
```
(18)

Where C_i is the confidence factor value for the mislabeled vector,
there are j mislabeled vectors, and C_v is user defined threshold value.

Another promising scheme relabels misclassified vectors in the same way as above except it flips a percentage of the worstly misclassified vectors. As before, the positive and negative vectors must be separated. The pseudo-code is:

For vectors that are mislabeled,

```
@sorted = descending_sort(abs(Ci...Cj));
n=total_vectors*percent_to_flip;
for(@sorted) {
        flip label of sorted[0]...sorted[n-1];
}
```

(19)

Where C_i is the confidence factor value for the mislabeled vector, there are j mislabeled vectors, and n is the total misclassified positive vectors multiplied by the percentage of vectors to flip. Process must be done separately for the positive and negative misclassified vector sets.

The advantage to this method is greater control over the number of vectors relabeled on each iteration. The percentage of misclassified vectors provides the user with a moer controllable threshold parameter.

SVM Parameters

The SVM Background section explains SVM parameterization in greater detail. This section will talk about the different svm parameters that were tuned for clustering purposes.

Implementations

The W-H SMO and Keerthi's SVM implementation are both used. The W-H SMO, which is a modification of the Platt-SMO [15]. The Keerthi implementation is different

from Platt based implementations in that it only chooses one alpha in each optimization step, while the other alpha is automatically specified. All other SMO implementations have to choose two alphas for optimization.

Kernels

Abs-diff, Gaussian, and Linear kernels are all used (see Appendix A). The Gaussian kernel is used for the External-SVM relabeling method. The Abs-Diff kernel is used in the Kernel K-Means implementation. The Linear kernel is used for dropping data in the External-SVM drop method.

Sigma(squared)

The choice of the σ^2 value is very important for finding initial convergence and achieving accurate clustering results. The σ^2 value must be tuned for the kernel to properly fit the data. This is illustrated in the results section.

Allowed KKT Violators

For the External-Relabel SVM clustering method, KKT violators must often be allowed to obtain initial convergence, because the random labeling scheme is likely to cause violators. However increasing the allowed KKT violators can reduce the accuracy of the hyperplane. Also, for this clustering method, if the SVM converges and creates a hyperplane that is inaccurate enough, the relabeling may cause the svm to fail to converge on the next iteration. If this happens, convergence may never be obtained again, so the program must be halted and restarted. So tuning may be needed on the number of violators to find a number that is large enough to allow convergence, but small enough to not strongly affect accuracy.

IV. Results

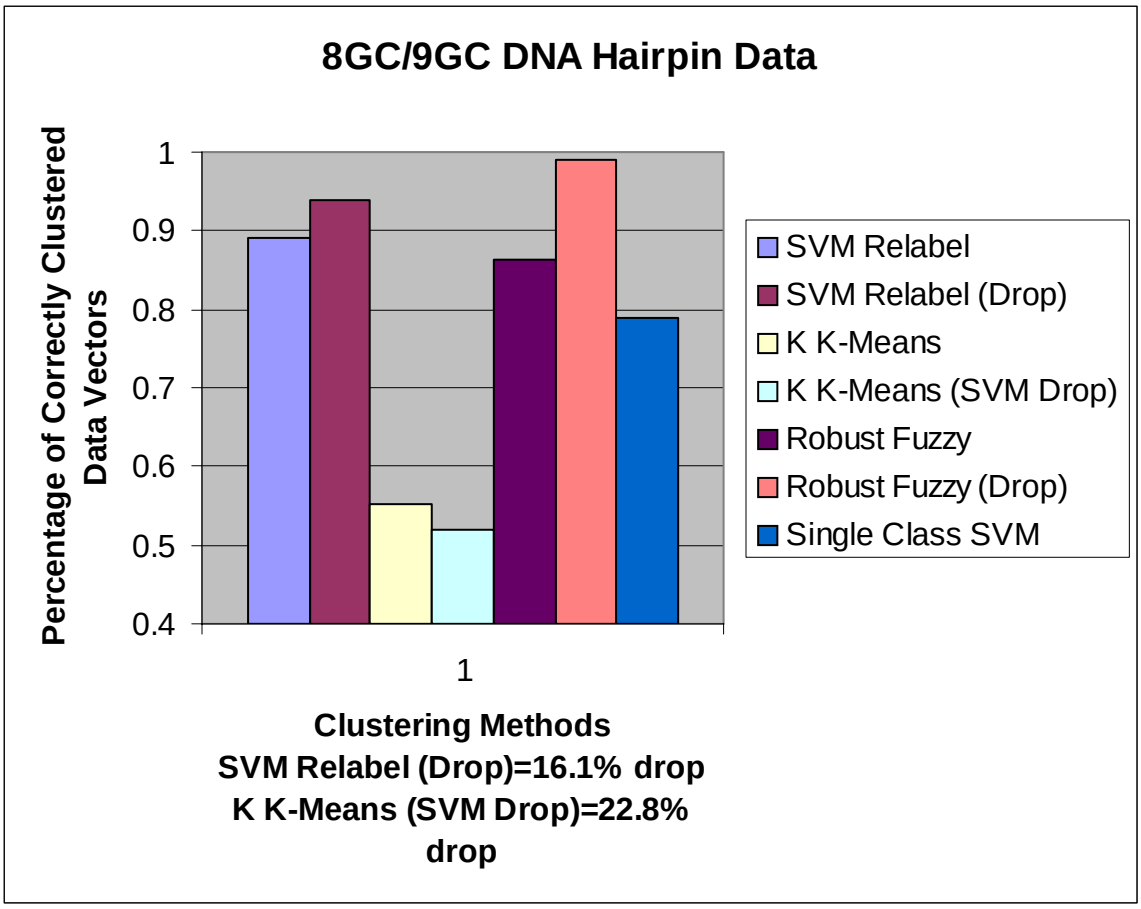


Figure 5. Accuracy comparison of each method's run on the 8GC/9GC DNA hairpin data.

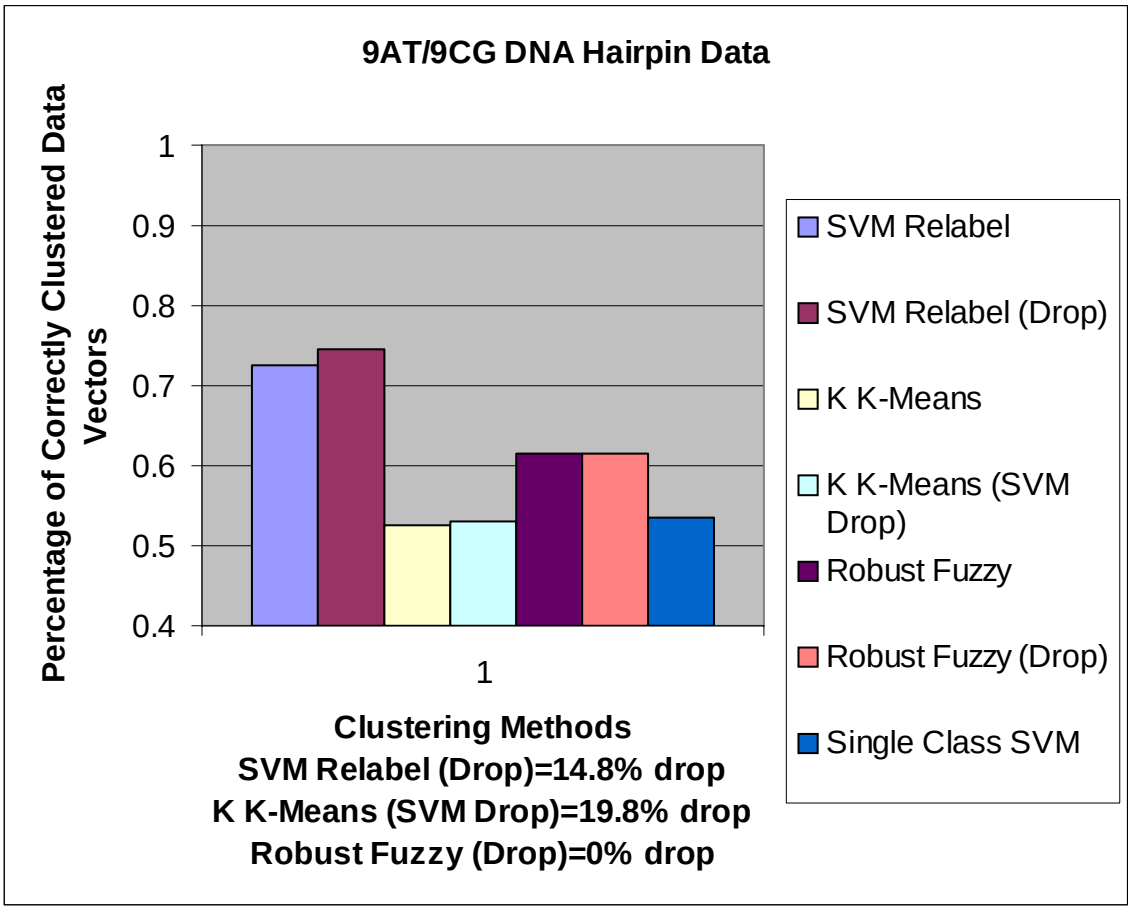


Figure 6. Accuracy comparison of each method's run on the 9AT/9CG DNA hairpin data.

Note: This is a much more challenging cluster separation problem than the data examined in Figure 5.

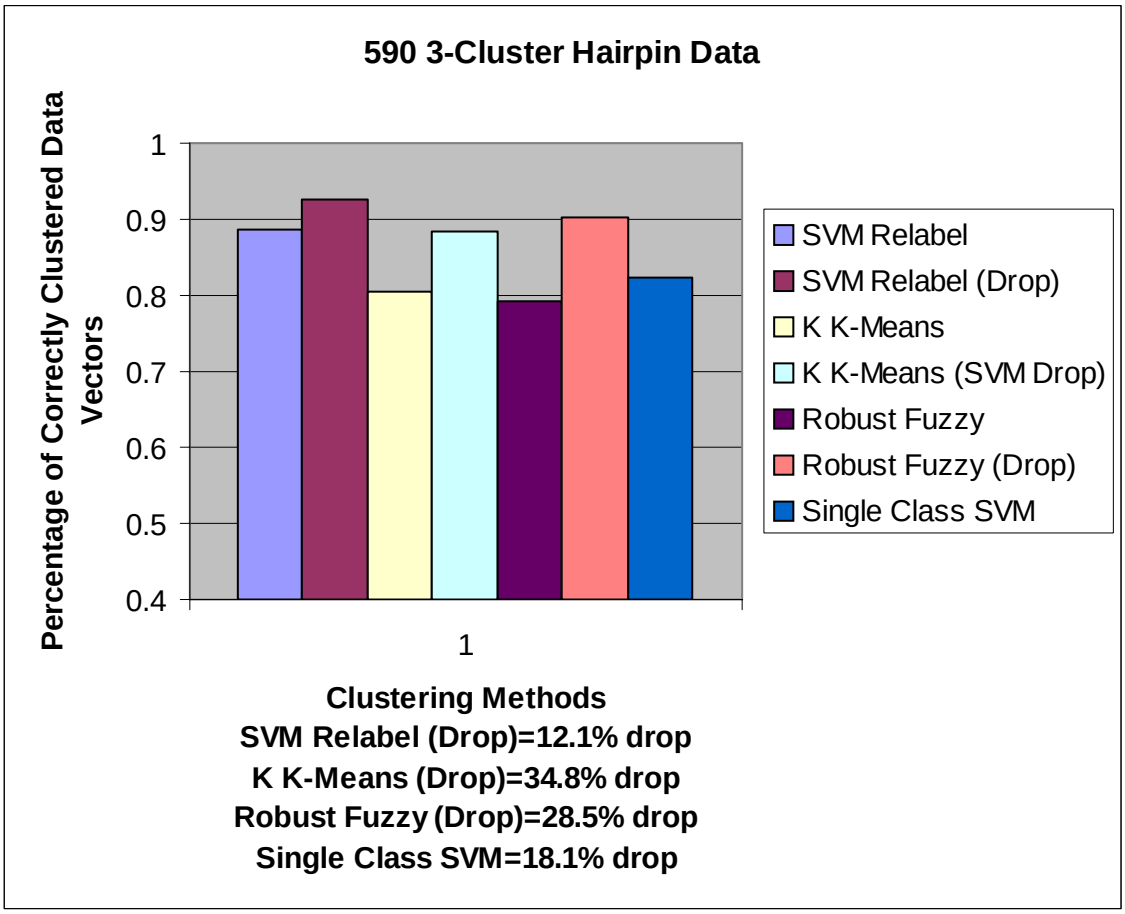


Figure 7. Accuracy comparison of each method's run on the first iteration for the multi-clustering on the 592/595/597 DNA hairpin data.

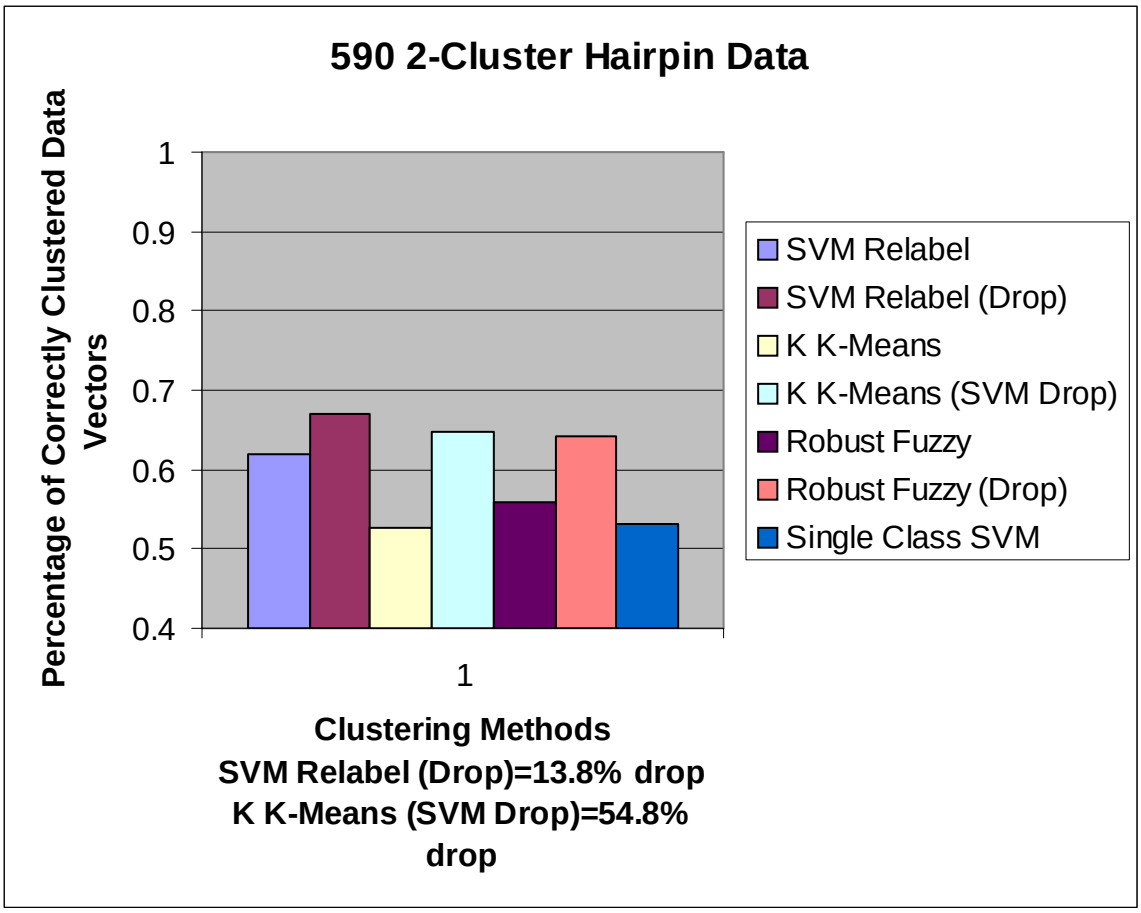


Figure 8. Accuracy comparison of each method's run on the second iteration for the multi-clustering on the 592/595 DNA hairpin data.

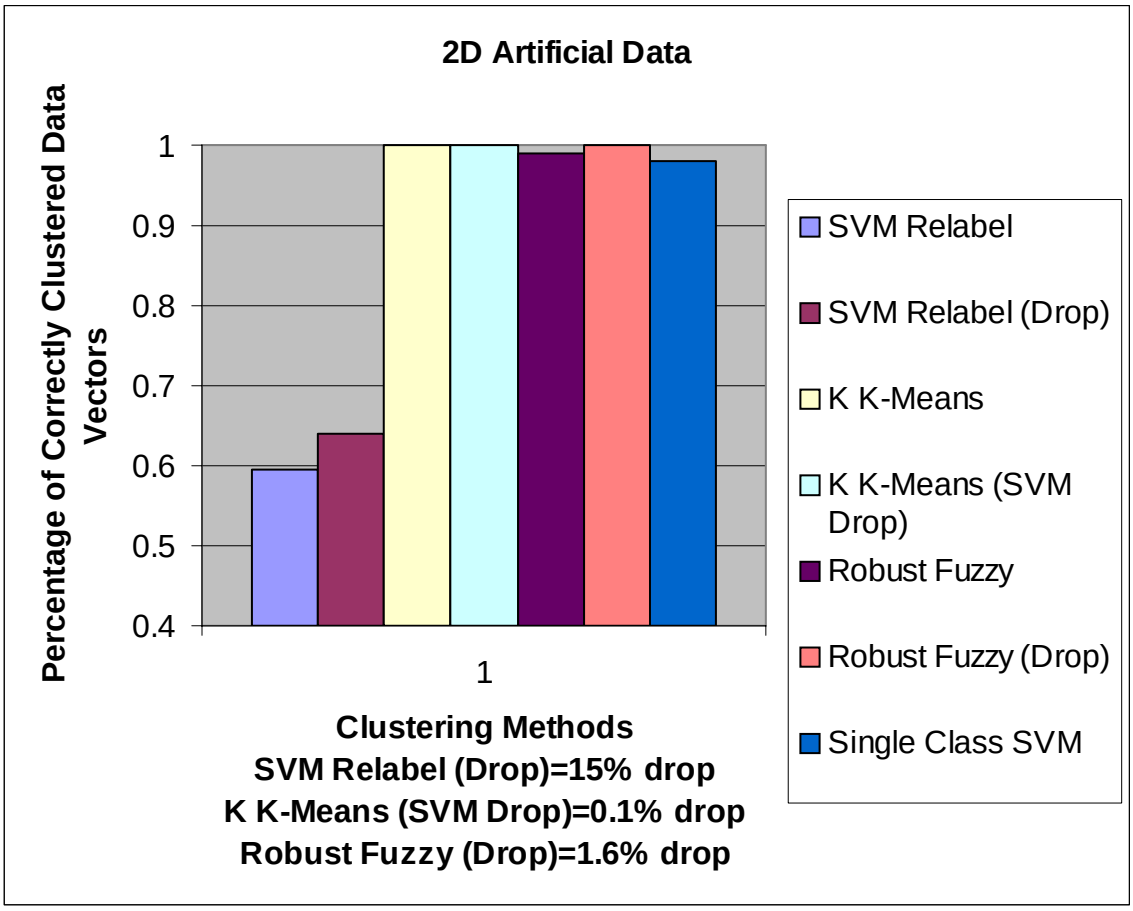


Figure 9. Accuracy comparison of each method's run on the 2D artificial data.

Note: This two-dimensional data contains two clusters with equal total width, completely separated by 10 times this width in the first dimension. This is a trivial problem for other clustering methods, but can cause the SVM Relabel method to enter deadlock if the initial chosen random labeling scheme is poor. This data result highlights a key difficulty with the current SVM Relabel method. Currently, several approaches are being researched, most notably using the Kernel K-Means as a preprocessing step, which creates a non-random initial labeling scheme, preventing the deadlock scenario Such preprocessing should reduce or remove the possibility of this situation occurring.

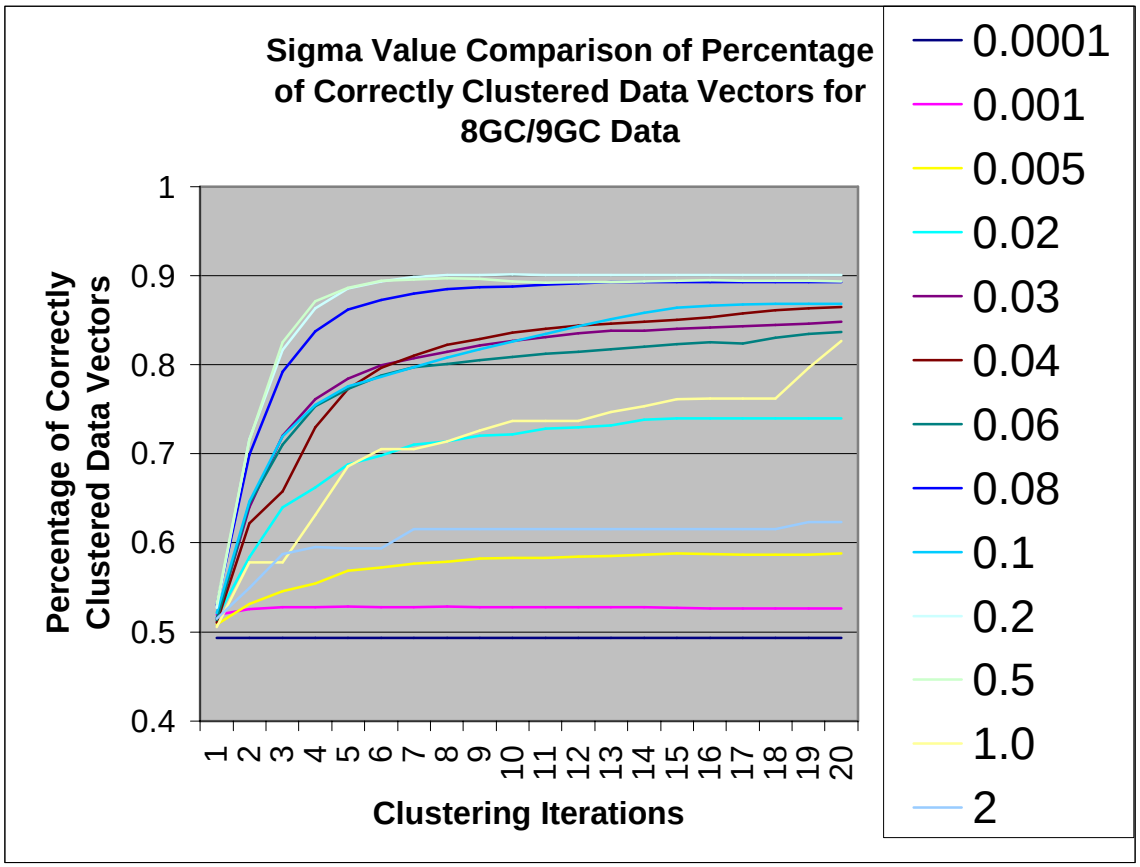


Figure 10. Accuracy comparison for External-Relabel SVM clustering runs on the 8GC/9GC DNA hairpin data using different σ^2 values.

Note: The σ^2 value of 0.2 performs optimally, with a stable fall-off in performance as σ^2 is moved off of this value. The σ^2 choices are pushed to the extremes 0.0001 and 2.0 to show how poor choices eventually lead to complete failure (a percentage correct of 50% is the equivalent of random guessing in such sets where the clusters have an equal number of members).

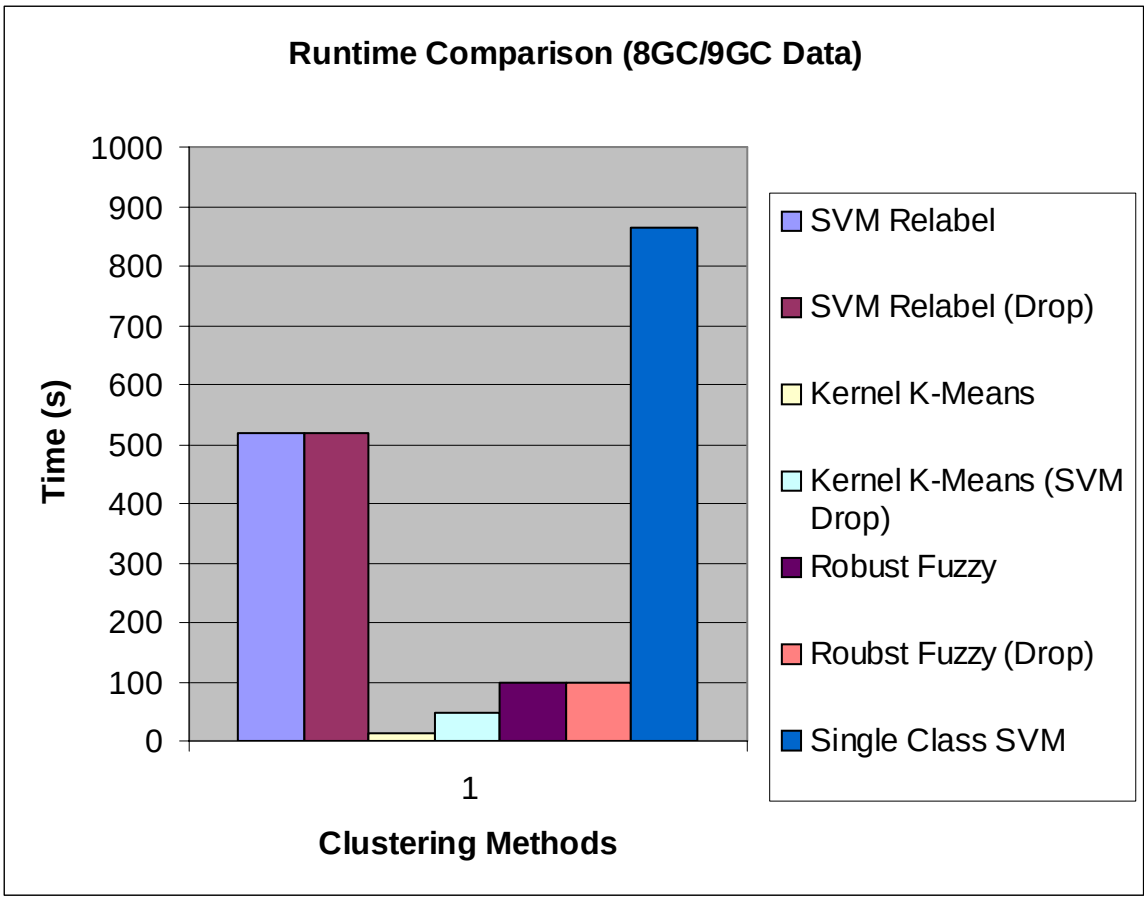


Figure 11. Runtime comparison of all clustering methods on the 8GC/9GC DNA hairpin data.

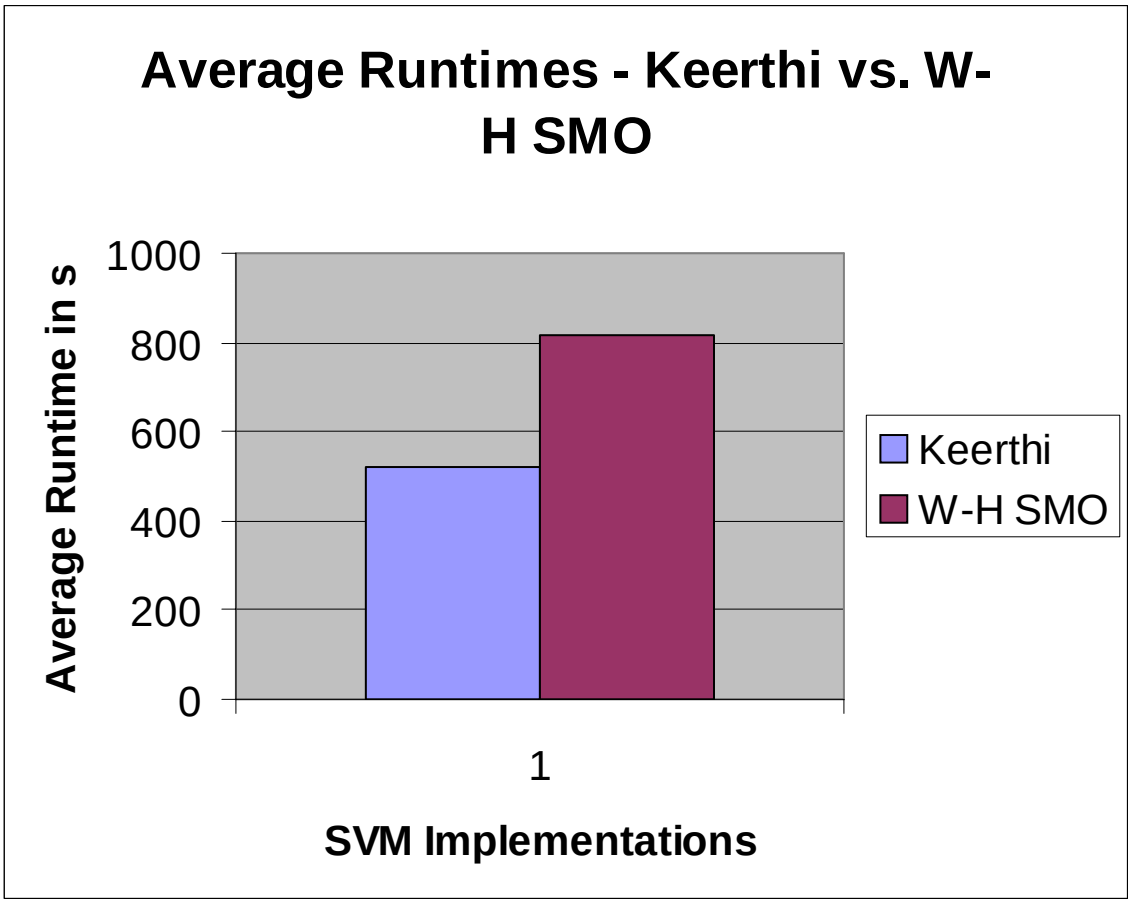


Figure 12. External-Relabel SVM clustering runtime comparison between the Keerthi and W-H SMO implementations.

V. Discussion

External-Relabel SVM Clustering

The External-Relabel SVM Clustering method performed very well on the real DNA hairpin data, but poorly on the artificial two-dimensional data. The poor performance on the artificial 2D data may be because there isn't enough information in the two dimensions for the SVM to obtain enough statistical evidence to accurately define the clusters. This can cause the External-Relabel Clustering method to enter deadlock if the initial chosen random labeling scheme is poor

The results from figure 5 show that this method successfully clustered the 8GC/9GC DNA hairpin data with almost 95% accuracy when including a drop zone. This is an excellent result on real data, because the 8GC and 9GC data are dissimilar with little overlap, and so should be split into two clearly separate clusters. Only the Robust Fuzzy Clustering method had accuracy as good as the External-Relabel SVM clustering on this data set.

Next, the results of the binary clustering runs on the 9CG/9AT DNA hairpin data are shown in figure 6. As opposed to the 8GC/9GC data set, the 9CG and 9AT data vectors are very similar, with a great deal of overlap. The graphs illustrating the results show that the data vectors only separate into clusters at around 72% accuracy. In this case, it was not surprising that the data vectors did not separate as accurately into well-defined clusters. This is because, even in the kernel space, the similarity between the data causes some of the outlier vectors of each class to overlap. The low confidence data may be "common mode" noise of the two data classes, which the SVM kernel fails to disambiguate because they fall too close to the hyperplane (these points can be removed by dropping all the points within a large enough range from the hyperplane), or just data representing weak properties of their classification. In SVM classification, such low confidence data have been shown to diminish the accuracy of the hyperplane separating the classifications [1]. Dropping 15% of the low confidence data causes an immediate

3% improvement in accuracy. It should be noted that this clustering method's performance was clearly superior to any other method on this data set. This suggests that the External-Relabel SVM clustering may be able to cluster complex, overlapping data sets better than the other methods.

The initial sub-clustering results for the binary clustering runs on the 592/595/597 DNA hairpin data are demonstrated in figure 7. Here, the 592 and 595 data sets form a separate cluster from the 597 data set with almost 90% accuracy (an accuracy of 93% is obtained when dropping 12% of the data). The SVM Relabel accuracy for this case is higher than all the other methods. This is a promising first step, since the 592 and 595 data sets are similar, while the 597 data set is not. Next, the sub-clustering algorithm will iteratively perform the same External-Relabel SVM clustering algorithm on the results of the previous iteration (the 592/595 and the 597 clusters). The results are shown in figure 8. When the algorithm tries to separate the 597 cluster by itself, two clusters are produced with an accuracy of about 50%. This is an indication that no improvement could be made from the initial random labeling, which accurately suggests that this data set contains only one cluster (which is true). The next iteration, which attempts to separate the 592 and 595 data sets, only results in clusters that are about 67% accurate. This result is analogous to the 9CG/9AT results, as the 592/595 data sets' features are also very similar. The accuracy of this outcome is higher for the SVM Relabel method than all the other methods, which again highlights the strength of this clustering method in separating complex and similar data sets. This process also illustrates how multi-cluster classification is done through iterating on the binary clustering method.

The last data set tested illustrates a key weakness with the External-Relabel SVM Clustering method when dealing with trivially separable data. The results are illustrated in figure 9. The two-dimensional artificial data set consists of clusters that have a variance of four units centered around a common mean. Each cluster's barycenter, which is the mean of all the data points in the cluster, is separated by twenty units in one dimension. The clusters unmistakably do not overlap, with the closest points between the two clusters at least four times the variance of the clusters apart. Human beings can

visualize this data set as clearly being in two clusters, yet this method was barely able to separate the clusters with 60% accuracy. Furthermore, all the other clustering methods were able to separate these clusters with close to 100% accuracy. The failure of this method in clustering such an uncomplicated data set can easily be prevented with simple pre-processing steps. Rather than use an ad-hoc approach, it appears best to preprocess with some a method such as Kernel K-Means as discussed in Figure 11.

Now the various parameters used in the External-Relabel SVM Clustering method will be discussed. This method relies on the following parameters: allowed KKT violators, choice of σ^2 value, and its random labeling scheme. The importance of these parameters will be discussed below.

The allowed KKT violators did not affect the chance of convergence or control the initial accuracy of the hyperplane as expected. Instead allowing the minimum KKT violators (no violators) usually did not stop initial convergence, and did not guarantee that the initial hyperplane was accurate enough for good clustering results. If using more than the minimum KKT violators provides no benefit in all cases (as has been observed so far), this parameter can be removed.

The value of the sigma squared parameter heavily affected the clustering results, indicating a tuning on this parameter is necessary. Results depicting the importance of the sigma squared are shown in figure 10. It can be seem that with the right choice of the sigma squared tuning parameter, the data set's clusters are identified with over 90% accuracy, and with the wrong sigma squared the accuracy is only 50%. Note, that the sigma squared value must be tuned so that the Gaussian kernel fits the data well.

The random labeling scheme may be the parameter that has the greatest affect on this method, because it determines the location of the initial hyperplane, and is the hardest to tune. Since this method decides the final clusters by iteratively improving on its hyperplane, the final result is heavily dependant on its initial hyperplane. One general rule to follow is that the SVM must consider some of the vectors as mislabeled, because

if all labels are deemed correct, then no relabeling can be done. This situation allows for no correction of the initial hyperplane, and gives final clustering results that are random. When using this method, if too few vectors are considered mislabeled by the initial hyperplane, the vectors should be randomly relabeled. The SVM can be then run again on the data set with the new labels, hopefully creating a better initial hyperplane. Creating an improvable initial hyperplane is the most challenging problem to the External-Relabel SVM Clustering method. More testing to determine the characteristics of an acceptable initial hyperplane needs to be done. This can be directly resolved in hybrid approaches, as mentioned previously, and is an area of future research.

W-H SMO vs. Keerthi

Both the W-H SMO (an variation of Platt's SMO) and Keerthi's SVM implementations were tested in the External-Relabel SVM Clustering method. Both implementations provided similar accuracy. The Keerthi implementation always converges, partly accounting for a slightly faster runtime. Figure 12 shows that the Keerthi implementation runs about 50% faster than the W-H SMO implementation. The W-H SMO may be the safer option, since it will reject random labeling schemes that don't converge.

Kernel K-means Clustering

The Kernel K-Means Clustering method performed extremely well on the two-dimensional artificial data, but only moderately well on the real DNA hairpin data. This method has difficulty differentiating some data sets with overlapping members.

This method was unable to separate either the 8GC/9GC or 9CG/9AT DNA hairpin data, assigning only approximately 55% of the data vectors to the correct cluster [Figures 5 and 6]. These are the worst results of all the methods for these data sets. This result highlights the inability of the K-Means method to differentiate complex clusters that have overlapping points.

For the initial multi-clustering iteration, the Kernel K-Means method respectably split slightly over 80% of the data vectors into their correct clusters. With the addition of a relatively large 35% drop, the accuracy reaches nearly 90% [Figure 7]. On the next iteration, the two-cluster data set can only be separated with slightly over 50% accuracy. With a very large 60% drop, the clustering accuracy reaches nearly 65% [Figure 8]. The results from the initial iteration suggest that the Kernel K-Means method has the ability to cluster some complex data sets, which only slightly overlap.

Next, from figure 9, the Kernel K-Means method separates two-dimensional artificial data set into two perfect clusters with 100% accuracy. This result highlights the strength of this method: splitting simple, clearly defined clusters that have zero overlap.

Next, the implementation decisions and parameters will be discussed. Both the absdiff and Gaussian kernels were implemented for this Kernel K-Means Clustering method. The absdiff kernel performed better than the Gaussian on the DNA hairpin data, using the Gaussian kernel resulted in higher accuracy on the two-dimensional artificial data.

The SVM Drop uses a linear kernel because Gaussian and absdiff overfit, while the linear kernel provides room for drop (k+1 VC dimension). Relabeling isn't done because only points close to the hyperplane would be relabeled, resulting in no improvement in clusters (furthermore, these points that are close to the hyperplane will be dropped anyway). The bounded and unbounded support vectors provide a reference for dropping data (bounded and unbounded support vectors are between 1 and –1 confidence factor value).

Robust Fuzzy Clustering

The Robust Fuzzy Clustering method obtained good results on most of the artificial and DNA hairpin data sets; however, on the DNA hairpin data that closely overlaps, this method was unable to accurately identify the clusters as well as the External-Relabel SVM method.

This method's clustering of the 8GC/9GC DNA hairpin data shows how Robust Fuzzy Clustering has no problem clustering non-artificial and complex data types. Without dropping any data, the two clusters were separated with over 85% accuracy. When including a 23% drop, the accuracy increases to 99% [Figure 5]. This data set contains data that doesn't greatly overlap, so this result doesn't illustrate how well Robust Fuzzy Clustering performs clustering very similar data sets. The next result will handle that case.

The results of the 9CG/9AT clustering demonstrate the most difficult type of data for this method to cluster. These clusters are split with only slightly over 60% accuracy [Figure 6]. While the accuracy of this result is only less than the External-Relabel SVM method, it is still rather low.

For the initial multi-cluster iteration, the Robust Fuzzy method clusters the data with almost 80% accuracy without any drop. With a 29% drop, the accuracy rises to slightly over 90% [Figure 7]. The accuracy of these results are a little less than the External-Relabel SVM method, and about the same as the Kernel K-Means method. After the next iteration, this method is able to split the 592/595 data set with less than 60% accuracy with zero drop, and near 65% accuracy with a 20% drop [Figure 8]. Again these results are similar to the Kernel K-Means results, and slightly less accurate than the External-Relabel SVM results.

The two-dimensional artificial data is accurately clustered by the Robust Fuzzy method. The accuracy is at 99% with no drop, and reaches 100% with a less than 2% drop [Figure 9]. This method has no trouble clustering simple non-overlapping data sets.

There are three tunable parameters used in Robust Fuzzy clustering. Explain Beta, Gamma, and σ^2. The Beta parameter can be defined as the weight of the entropic factor. Gamma is the width while computing the distance between any two data points. And similarly to the other methods, the σ^2 parameter is the variance of the Gaussian kernel.

Finding the right tuning can be difficult because the tuning of each of these parameters relies heavily on the tuning of the other two parameters.

There is also a tunable drop cutoff that determines the membership score necessary for each point to be deemed a member of a cluster. For example, with a drop cutoff of 80%, each point must score at least 80% for membership to one of the clusters or that point is dropped.

Single Class SVM

The Single Class SVM method was able to cluster both the artificial data and the DNA hairpin data moderately well. One advantage of this method is it does not have to do multi-cluster clustering through iterative binary clustering runs. Instead, this method is able to do multi-clustering on its initial run. This method also requires a drop to separate clusters.

This method clusters the 8GC/9GC data set with almost 80% accuracy, with a 29% drop [Figure 5]. This result is over 10% less accurate than the External-Relabel and Robust Fuzzy clustering methods, but is about 25% more accurate than the Kernel K-Means method. This demonstrates that this method has the ability to give moderate clustering results on complex, slightly overlapping data sets.

The Single-Class SVM method was only able to cluster the 9CG/9AT data with about 55% accuracy, with a 36% drop [Figure 6]. This is less accurate than the External-Relabel and Robust Fuzzy methods, and slightly more accurate than the Kernel K-Means method. A weakness of this method is its inability to accurately cluster this type of data.

The three cluster 590's data set is clustered with a little over 80% accuracy with an 18% drop, by the Single-Class SVM method [Figure 7]. This method has the ability to identify multiple clusters with one run; however, for this data set the method only found two clusters. Since only two clusters were found, an iterative step was performed on the 592/595 data. This step only resulted in cluster separation with about a 55% accuracy,

when dropping 24% of the data [Figure 8]. For this data set, the Single-Class SVM method actually performed slightly worse than the Kernel K-Means method, and again moderately worse than the External-Relabel SVM and Robust Fuzzy methods.

This method accurately clustered the two-dimensional artificial data with 98% accuracy [Figure 9]. This method can strongly cluster simple, non-overlapping data sets.

The two parameters used in this method are the drop zone percentage and σ^2 value. The maximum drop zone parameter defines the maximum number of data vectors that can be dropped. The drop zone must be large enough that the method can separate the data clusters, but not too large as too drop unnecessary data. The σ^2 value is non-dependant on the drop zone parameter, and must be fit to the data just as is done in the External-SVM clustering method. An advantage to this method is that the tuning is straightforward.

Runtimes

Figure 11 shows the runtimes for each method on the 8GC/9GC DNA hairpin data. The runtimes for each method vary depending on the choice of tuning parameters. Generally, more tuned in parameters result in faster runtimes. The figure presents the runtimes for each method with the best-found tuning parameters, and thus represents relatively fast runtimes. As can be immediately seen, the External-Relabel SVM and Single Class SVM have much longer runtimes than the other methods. This is because the SVM is more algorithmically intensive than the other methods. The Kernel K-Means method has the shortest runtime. This was expected, because the logic behind the algorithm is the simplest.

VI. Conclusion

This thesis presents a novel External SVM based clustering algorithm. Other current, popular clustering algorithms have also been implemented. The results of each algorithm

are compared to the results of the External SVM based clustering algorithm. Data sets with different levels of complexity and similarity are tested, highlighting the strengths and weaknesses of the External SVM based clustering algorithm compared to the other algorithms. Current testing indicates that the new External SVM based algorithm performs excellently in separating complex data sets with a high degree of overlapping features. Future work will include testing all the algorithms on high dimensional artificial data sets, with various, exactly controlled, levels of overlapping features.

VII. References

1. Winters-Hilt, S. Yelundur, A. McChesney, C. Landry, M. "Support Vector Machine Implementations for Classification and Clustering". *BMC Bioinformatics 2006, 7 Suppl 2:S14.*

2. Tariq Rashid. "Clustering of Fuzzy Image Features" *MSc Thesis, http://www.cs.bris.ac.uk/home/tr1690/, 2003.*

3. Zaiane, Osmar R. "Principles of Knowledge Discovery in Databases – Chapter 8: Data Clustering". *http://www.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter8/index.html*

4. Dempster, Arthur. Laird, Nan. Rubin, Donald. "Maximum Likelihood from Incomplete Data Via the EM Algorithm". *Journal of the Royal Statistics Society, Series B, 39(1):1-38, 1977.*

5. Zhang, Rong. Rudnicky, Alexander I. "A Large Scale Clustering Scheme for Kernel K-Means". *Preprint at School of Computer Science, Carnegie Mellon University.*

6. Wilson, H.G. Boots, B. Millward, A. A. "A comparison of hierarchical and partitional clustering techniques for multispectral image classification". *Geoscience and Remote Sensing Symposium, 2002. IGARSS '02, 2002 IEEE International.*

7. Bezdek, J. "Pattern Recognition with Fuzzy Objective Function Algorithms". *Plenum Press, USA, 1981.*

8. Chipman, H. Tibshirani, R. "Hybrid hierarchical clustering with applications to microarray data". *Biostatistics Advance Access, 2005.*

9. Du, W. Inoue, K. Urahama, K. "Robust Kernel Fuzzy Clustering". *Kyushu University, Fukuoka-shi, 81508540 Japan.*

10. Girolami, M. "Mercer kernel-based clustering in feature space". *IEEE Trans, Neural Netw. 13 (2002) 780-784.*

11. Kim, D. Lee, D. Lee, K. "Evaluation of the performance of clustering algorithms in kernel-based feature space". *Patt. Recog. 38 (2004) 607-6-11.*
References

12. Burges, Christopher J.C. Bell Laboratories, Lucent Technologies. Technical Report: "A Tutorial on Support Vector Machines for Pattern Recognition".

13. Anthony, M. and Biggs, N. Pac learning and neural networks In "The Handbook of Brain Theory and Neural Networks", pages 647-697, 1995.

14. V. Vapnik. "Statistical Learning Theory". Wiley, NY, 1998.

15. Winters-Hilt S, "Machine Learning Methods for Channel Current Cheminformatics, Biophysical Analysis, and Bioinformatics", UCSC PhD Thesis, March 2003.

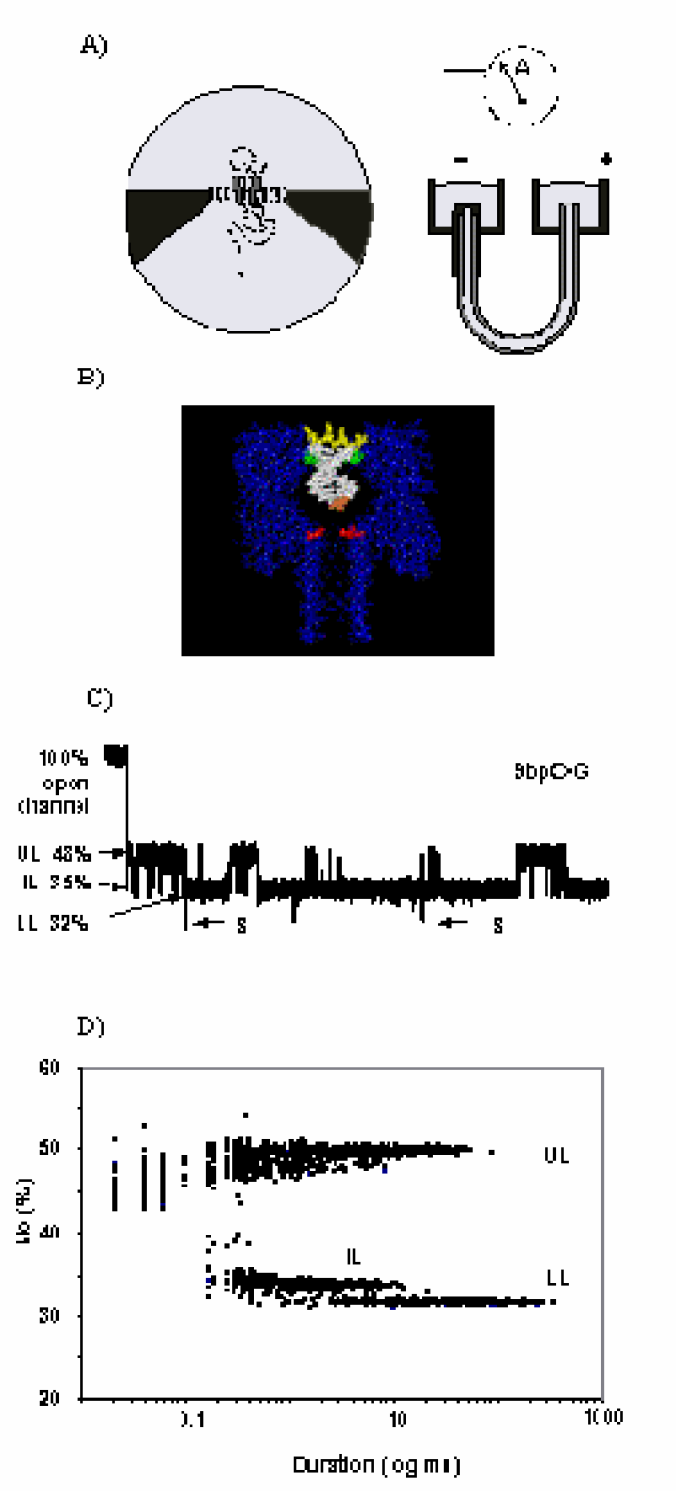
Appendix A Blockade Kinetics [15]

The results on blockade kinetics provided here were published in [Vercoutere et al, 2003]. Further progress on analysis of the single molecule biophysics is described in the Sec. 5.3.3 and Sec. 5.4.3.

Making use of results from Sec. 4.2.2, four conductance states are defined: i) an intermediate level (IL) that initiates all 9 bp events; ii) an upper conductance level (UL); iii) a lower conductance level (LL) that must be preceded by the upper level; and iv) spikes down from the lower level that may indicate close proximity of the terminal base pair to the pore limiting aperture.

Terminal base-pair identity can be determined by kinetic analysis of dwell times in these conductance states. In particular, average dwell time in the lower conductance level (LL in Fig. E.1b-c) and the frequency of downward current spikes (S in Fig. E.1b) are highly dependent upon the presence of a stable base pair in the ninth position.

Figure A.1 (shown on next page)**.** A prototype nanopore device based on the α–hemolysin channel. a) Diagram of the prototype device. One α-hemolysin channel is intercalated in a horizontal diphytanoyl phosphatidylcholine bilayer. The bilayer is supported on a 20 micron diameter conical aperture at the end of a U-shaped Teflon tube. The tube connects two 70 μl volume baths filled with 1M KCl buffered at pH 8.0 in 10 mM HEPES/KOH. Voltage is applied between two Ag-AgCl electrodes that are connected to an Axon 200B amplifier. b) Two dimensional diagram of a 9bp hairpin captured in the pore vestibule. The stick figure in blue is a two dimensional section of the α-hemolysin pore derived from X-ray crystallographic data [Song *et al.*, 1996]. A ring of lysines that circumscribe a 1.5-nm-limiting aperture of the channel pore is highlighted in red. A ring of threonines that circumscribe the narrowest, 2.3-nm-diameter section of the pore mouth is highlighted in green. In our working model, the four dT hairpin loop (yellow) is perched on this narrow ring of threonines, suspending the duplex stem in the pore vestibule. The terminal base-pair (brown) dangles near the limiting aperture. The structure of the 9bp hairpin shown here was rendered to scale using WebLab ViewerPro. c) Representative blockade of ionic current caused by a 9bp DNA hairpin (9bp(GT/CA). Open channel current (I_o) is typically 120 pA at 120 mV and 23.0 °C. Here it is expressed as 100% current. Capture of a DNA hairpin causes a rapid decrease to a residual current I, expressed as a percent of the open channel current. Typically, 9bp hairpins cause the residual current to transition between four levels: an upper conductance level (UL), an intermediate level (IL), a lower level (LL), and a transient downward spike (S) . d) A two dimensional plot of log duration vs. amplitude for UL, IL, and LL conductance states.


A)
A
-
+
B)
C)
100%
open
channel
UL 48%
IL 35%
LL 32%
S
S
9bpC>G
D)
60
50
40
30
20
Io (%)
UL
IL
LL
0.1
10
1000
Duration (log ms)

This is illustrated in Fig. E.2 where neither a 5′ dC dangling nucleotide nor a 3′ dG dangling nucleotide alone stabilized ionic current in the lower level (I/Io = 32%), whereas both nucleotides together (the C•G pair) did so. We reasoned that the presence of two nucleotides alone at the terminus of the hairpin stem might account for this current stabilization. However, two weakly paired thymine bases at the blunt end terminus of a 9bp hairpin stem resulted in an unstable blockade signature (Fig. E.2). In practice, the lower conductance level has the added advantage that transitions to UL are stochastic, and that one first order exponential can be fit to the dwell time distribution giving a time constant (τ_{LL}) in the millisecond range.

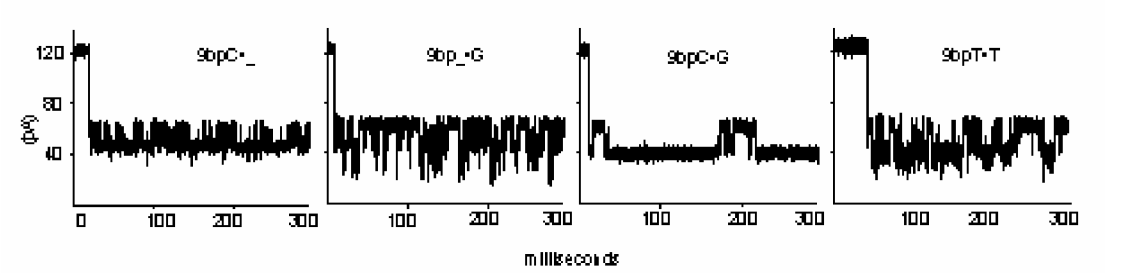


Figure A.2 Comparison of blockade signatures caused by DNA hairpins with dangling and blunt ends. All hairpins were built onto a core 8bp DNA hairpin with the primary sequence 5′-TTCGAACGTTTTCGTTCGAA-3′. 9bp(CT/-A) shows a blockade signature caused by a hairpin with a dangling 5′-C nucleotide. 9bp(-T/GA) shows a blockade signature caused by a dangling 3′-G nucleotide. 9bp(CT/GA) shows a blockade signature for a hairpin in which both terminal nucleotides are present forming a 5′-C•G-3′ terminal Watson-Crick base-pair. 9bp(TT/TA) shows a typical blockade signature for a blunt-ended 9bp hairpin in which the terminal 5′-T•T-3′ pair is weakly associated. Experimental conditions are described under *Methods*.

The sensitivity of the lower level conductance state to Watson-Crick base-pair identity was tested by measuring τ_{LL} and spike frequency for the four 9 bp hairpins whose blockade signatures are illustrated in Fig. E.3. Dwell time histograms for the lower conductance state caused by 9bpG•C and by 9bpT•A are shown in Fig. E.4. First-order exponentials fit to similar histograms for all four permutations of Watson-Crick base-pair terminal ends reveal τ_{LL} values ranging from 160 ms to 7 ms in the order 9bpG•C > 9bpC•G > 9bpA•T > 9bpT•A (Table E.1). The reverse order is observed for the spike frequency ranging from 4 spikes s^{-1} (9bpG•C) to 82 spikes s^{-1} (9bpT•A). Thus, two kinetic parameters can be used to discriminate among Watson-Crick base pairs on single DNA molecules, confirming the pattern recognition results established previously.

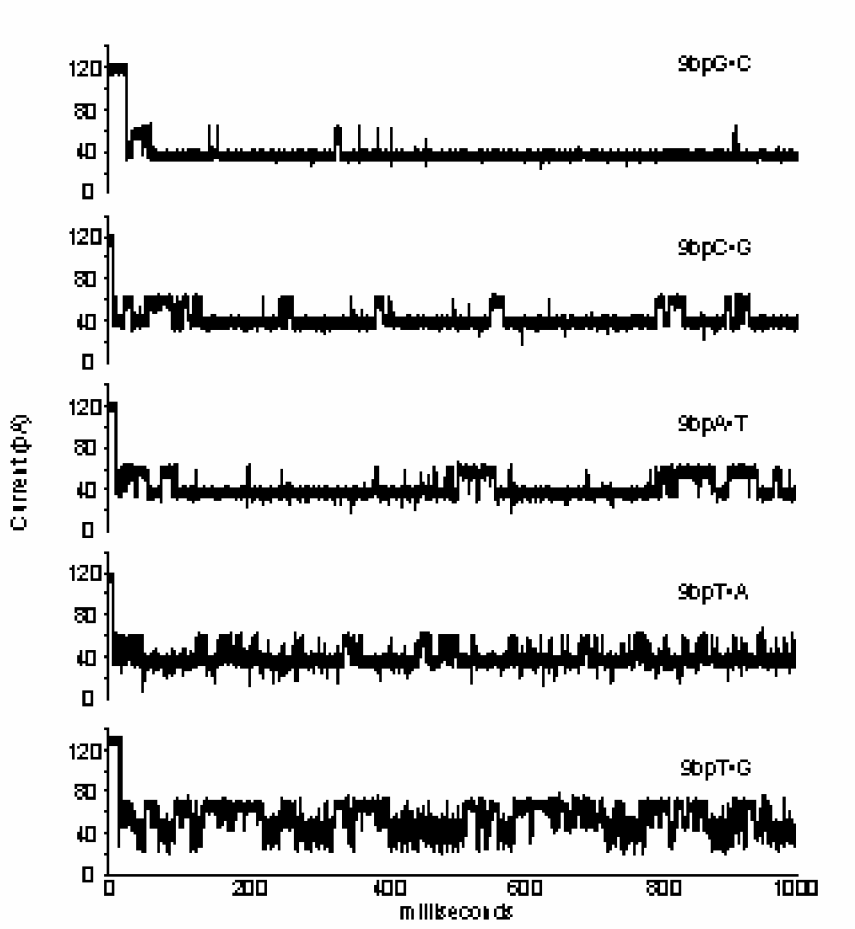


Figure A.3 Blockade of the α–hemolysin pore by 9bp DNA hairpins in which the terminal base pair is varied. Blockade events were acquired at 120 mV applied potential and 23.0 °C (see *Methods*). Each signature shown is caused by a single hairpin molecule captured in the pore vestibule, and is representative of several thousand single molecule events.

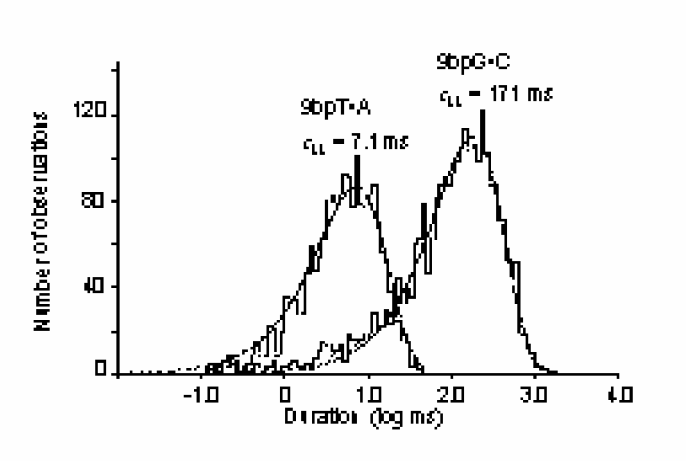


Figure E.4 **Dwell time histograms for lower level (LL) blockade events. Duration measurements were plotted in semi-log frequency histograms with 20 bins per decade. At least 1000 measurements of duration were used for each plot. To determine the probability density function and the average event lifetime, τ_{LL}, curves were fit to each histogram using the Levenberg-Marquardt method. 9bpT•A is the standard 9bp hairpin with a 5′-T•A-3′ terminus, and 9bpG•C is a 9bp hairpin with a 5′-G•C-3′ terminus.**

One of the more difficult base-pairs to recognize using conventional hybridization assays is a terminal mismatch, in particular a TG wobble pair. We tested the sensitivity of the nanopore to this mismatch by comparing blockade signatures caused by a hairpin composed of the sequence 9bpT•G with blockade signatures caused by the perfectly complementary sequences 9bpC•G and 9bpT•A (Fig. E.3). In this experiment, all individual blockades that exhibited the characteristic four current level signature could be identified as one of these molecules. Quantitative examination of the data revealed that spike frequency was the key diagnostic parameter. That is, there was a significant difference between spike frequencies caused by each of the three termini, i.e. 12 spikes s^{-1} (9bpC•G), 82 spikes s^{-1} (9bpT•A), and 1400 spikes s^{-1} (9bpT•G) (Table E.1). In

contrast, τ_{LL} values were statistically different between 9bpT•G and 9bpC•G termini, but not between 9bpT•G and 9bpT•A termini (Table E.1). It appears that τ_{LL} values plateau in the low millisecond time-range.

Identity	τ_{LL} ms	Spike frequency s^{-1}	$\Delta\Delta G^{\circ}_{23}$ kcal/mol
9bpG•C	160 ± 23	4 ± 1	-1.9
9bpC•G	50 ± 4	12 ± 4	-1.8
9bpA•T	43 ± 5	34 ± 10	-1.2
9bpT•A	7 ± 1	91 ± 47	-1.3
9bpT•G	6 ± 2	1300 ± 400	-0.3

Table A.1 Comparison between single DNA hairpin kinetic parameters and $\Delta\Delta G^{\circ}$ for terminal base-pairs. $\Delta\Delta G^{\circ}_{23}$ values are the difference between calculated ΔG° of duplex formation for 9bp DNA hairpins and calculated ΔG° of duplex formation for core 8bp hairpins that lack the terminal base-pair. Calculations assumed 23.0 C and 1M KCl. They were performed using Mfold (http://bioinfo.math.rpi.edu/~mfold/dna/form1.cgi) which is based on data from SantaLucia (2). Spike frequency and τ_{LL} values are means ± standard errors for at least three experiments using different individual channels.

Do the rankings of spike frequency and τ_{LL} correlate with conventional estimates of terminal base-pair stability? Table E.1 lists free energy values for terminal base pairs ($\Delta\Delta G^{\circ}_{23}$) calculated using the online computational tool 'Mfold' (http://bioinfo.math.rpi.edu/~mfold/dna/form1.cgi) which is based on a nearest neighbor model of duplex stability [SantaLucia, 1998]. This model is particularly strong because it considers data from seven independent studies [Breslauer *et al.*, 1986; Doktycz *et al.*,

1992; Delcourt and Blake, 1991; Gotoh and Tgashira, 1981; SantaLucia *et al.*, 1996; Sugimoto *et al.*, 1996; Vologodskii *et al.*, 1984]. In Table E.1, the $\Delta\Delta G^{\circ}_{23}$ values are the difference between the free energy of duplex formation for a given 9bp hairpin and the free energy of duplex formation of a common 8bp core hairpin sequence at 23 degrees C. Among Watson-Crick base pairs, $\Delta\Delta G^{\circ}_{23}$ values ranged from –1.9 kcal/mol for 9bpG•C to –1.2 kcal/mol for 9bpA•T. $\Delta\Delta G^{\circ}_{23}$ for the T•G wobble pair was calculated to be –0.3 kcal/mol. In general, the rank of spike frequency and τ_{LL} correlated with $\Delta\Delta G^{\circ}_{23}$, however the correlation is imperfect in that the expected order of 9bpT•A and 9bpA•T was reversed. Possible explanations for this discrepancy include uncertainty surrounding the predicted stability of terminal 5′-A•T-3′ and 5′-T•A-3′ pairs [SantaLucia, 1998; SantaLucia *et al.*, 1996], and limits on the precision of optical melting curves that underlie the free energy calculations. We note that the calculated $\Delta\Delta G^{\circ}_{23}$ values for the 9bpA•T and 9bpT•A termini differed by only 0.1 kcal/mol (Table E.1), which is less than the 5% precision given for Mfold. We also note that τ_{LL} and spike frequency may be influenced by interactions between the duplex termini and amino acids in the vestibule wall. The magnitude of these effects could be sequence dependent, thus altering the stability ranking in the nanopore assay relative to a bulk solution assay.

Hydrogen bond and base stacking each influence τ_{LL} and spike frequency. Having established a general correlation between the nanopore data and classical measures of base-pair stability, we determined if non-covalent forces that contribute to DNA duplex stability could be detected by the nanopore. Forces that stabilize DNA duplexes include hydrogen bonding between bases, and base stacking. Forces that destabilize DNA

duplexes include hydrogen bonding between water molecules and nucleotide bases, and electrostatic repulsion between phosphodiester anions in the DNA backbone. Steric effects may stabilize or destabilize the duplex depending upon sequence context [Kool, 2001].

Initial inspection of the data in Table E.1 suggests that hydrogen bonding plays a significant role in spike frequency and τ_{LL}. That is, terminal base pairs that are known to form three hydrogen bonds when paired (5′-G•C-3′ and 5′-C•G-3′) have lesser spike frequencies and greater τ_{LL} values than do terminal base pairs that form two hydrogen bonds when paired (5′-A•T-3′, 5′-T•A-3′, and 5′-T•G-3′). To directly test the influence of hydrogen bonding on these kinetic parameters, we compared current signatures caused by 9bpT•A with those caused by a hairpin with a difluorotoluene (F)/adenine terminus (9bpF•A). Difluorotoluene is a good structural mimic of thymine [Kool, 2001; Guckian and Kool, 1998] that is recognized by DNA polymerases despite the presumed absence of hydrogen bonding to paired adenines [Morales and Kool, 2000; Moran *et al.*, 1997]. Blockade current signatures are illustrated in Fig. E.5. Reduction of hydrogen bonding by the T•A→F•A substitution causes destabilization of the lower conductance state reflected in a decreased average dwell time in that state (τ_{LL}=2 ms) and an increased spike frequency (290 ± 10 s^{-1}) relative to the T•A control (Table E.1).

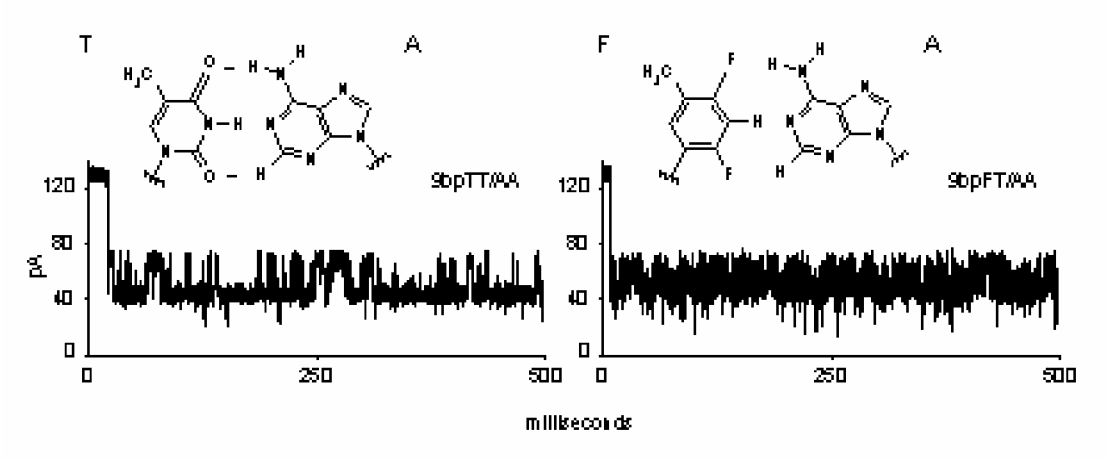
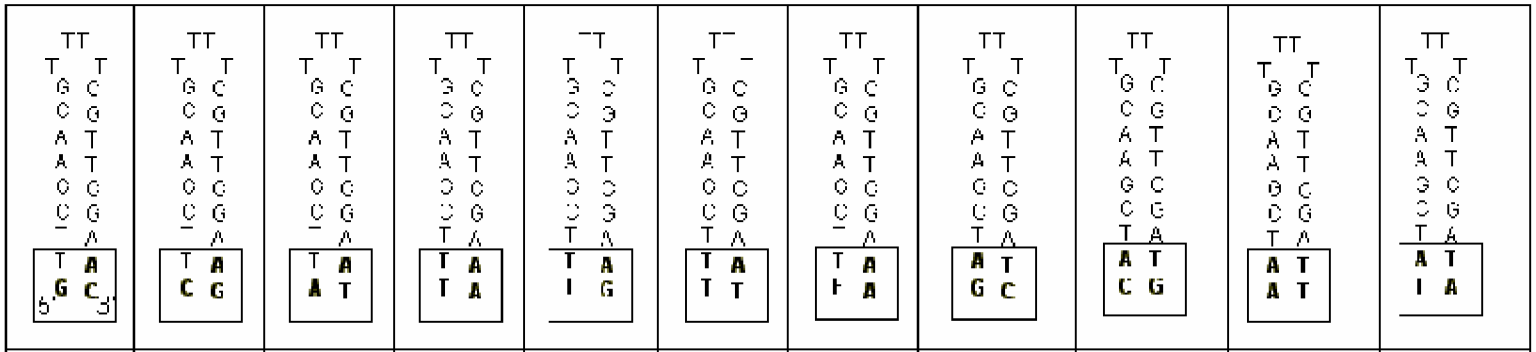


Figure E.5 **Effect of difluorotoluene (F) substitution for thymine (T) on blockades caused by 9bp hairpins. The blockade signature at left is caused by a 9bp hairpin with a 5′-T•A-3′ terminus (9bpT•A in Table E.2). The blockade signature at right is caused by a nearly identical 9bp hairpin in which the 5′ thymine is replaced by difluorotoluene (9bpF•A in Table E.2) giving a 5′-F•A-3′ terminus that lacks hydrogen bonds. The blockade signatures shown are representative of thousands of single molecule events acquired under standard conditions (see *Methods*).**

The data in Table E.1 also indicate that orientation of the bases in the terminal pair influences spike frequency and τ_{LL}. That is, flipping the terminal base pair so that a purine is on the 5′ side and a pyrimidine is on the 3′ side (9bpC•G→9bpG•C and 9bpT•A→9bpA•T) consistently increased τ_{LL} and decreased spike frequency. Among Watson-Crick base-pairs, the size of this effect equals or exceeds the effect of increasing hydrogen bond number (Table E.1). Classical thermodynamic studies suggest two possible explanations: i) stacking forces with the neighboring base-pair are altered when the terminal base-pair is flipped [SantaLucia, 1998; SantaLucia *et al.*, 1996]; and ii) stacking of bases at the 5′ position of a duplex can be different from those at the 3′

position independent of the neighboring base pair [Bommarito *et al.*, 2000]. A third possible explanation is that the terminal nucleotide on the 3′ or 5′ side of the duplex stem interacts with an amino acid in the vestibule wall and that this interaction is nucleotide dependent.

To test the first explanation, we compared τ_{LL} for the standard 9bp hairpins containing the four possible Watson-Crick termini (Table E.2 at left) with their counterparts in which the penultimate TA base pair was flipped, i.e. hairpins 9bp(AT)T•A, 9bp(AT)A•T, 9bp(AT)C•G and 9bp(AT)G•C at right in Table E.2. 9bpT•A was the least stable of the original sequences with τ_{LL} equal to 7 ms. (Note. Appendix A was taken from Appendix E of [15]).



9bpG•C	9bpC•G	9bpA•T	9bpT•A	9bpT•G	9bpT•T	9bpF•A	9bp(AT)G•C	9bp(AT)C•G	9bp(AT)A•T	9bp(AT)T•A

Table E.1 DNA hairpins used in this study. Primary sequence reads from 5' end at bottom left to 3' end at bottom right. Each hairpin has a 9 base-pair-long stem, and a four dT loop. The terminal base-pair and its nearest neighbor are highlighted by a box. These are the base-pairs in closest proximity to the pore limiting aperture when a given hairpin is captured in the α-hemolysin vestibule.

By making the penultimate base-pair substitution 9bp(TA)T•A→9bp(AT)T•A, τ_{LL} was increased about three-fold to 20 ms (Table E.3). Conversely, 9bpA•T was the most stable of the thymidine/adenine termini with τ_{LL} equal to 43 ms. By making the same alteration of the neighboring base pair as in the previous experiment, 9bp(TA)A•T→9bp(AT)A•T, τ_{LL} was decreased to 30 ms. Thus, stacking against the neighboring base pair did account for some of the stability difference associated with orientation of the thymine/adenine termini. The independent effect of placing adenine at the 5′ position was also important by inference. For the guanine/cytosine termini, the outcome was different (Table E.3). In those cases, flipping penultimate base pairs did not significantly influence τ_{LL}. Thus, the three-fold difference in τ_{LL} for 5′-G•C-3′ versus 5′-C•G-3′ is due to an end-specific effect independent of the neighboring base pair.

	Penultimate Base-Pair	
	5′-T•A-3′	5′-A•T-3′
Terminal Base-Pair	τ_{LL} *in milliseconds* ± *S.E.*	
5′-T•A-3′	7 ± 1	20± 4
5′-A•T-3′	43 ± 5	30± 6
5′-G•C-3′	160 ± 23	210 ± 90
5′-C•G-3′	50 ± 4	66 ± 20

Table E.3 Effect of penultimate base-pair orientation on τ_{LL} for 9bp hairpins with different Watson-Crick base-pair termini and orientations. Values shown represent means ± standard errors for at least three different individual channels. Experimental conditions are described under *Methods*.

Vita

Charles McChesney was born in Baton Rouge, LA and received his B.S. of Computer Science from the University of New Orleans. He works as a software developer at the Space and Naval Warfare Center in New Orleans.