International Journal of Computing and Optimization
Vol. 4, 2017, no. 1, 53 - 89
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ijco.2017.755

Clustering via Support Vector Machine Boosting with Simulated Annealing

Stephen Winters-Hilt

Computer Science Department and Biology Department
Connecticut College
270 Mohegan Ave.
New London, CT 06320, USA
&
Meta Logos Inc.
124 White Birch Dr.
Guilford, CT 06437, USA

Abstract

SVMs are shown to be effective at clustering when used with metaheuristics to recover label information in a bootstrap learning process. An automated tuning solution for Support Vector Machine (SVM) classification and clustering methods is described. This is shown for two complemetary situations: (1) by implementing a simulated annealing with perturbation tuning procedure on the relabeler-SVM boosted clustering method (single-pass), where kernel and algorithmic parameters are already optimized both by brute force computational search and by genetic algorithm evolutionary search; and (2) by implementing a multi-pass, random restart, SVM clustering optimization analysis (with fixed kernel and algorithmic parameters). Informatics-based SVM kernels, introduced previously, are used. Tuning is done by using the SVM performance on training data to define a fitness function. Tuning metaheuristics offer an automated way to obtain a powerful SVM classifier, or clustering process, for a given dataset. The SVM discussion is interwoven with data analysis involving channel current data, with specific application to the signal processing associated with the nanopore transduction detector.

Keywords: SVM, Simulated Annealing, Clustering

1 Introduction

Clustering is an unsupervised learning problem whose goal is finding 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 [1, 2]. Clustering is a knowledge discovery process that has utility in almost any field. Some applications of clustering include search engine document classification [3], finding groups of customers with similar buying patterns [4], image clustering for computer forensics [5], and knowledge discovery in a general setting where feature or concept primitives themselves need to be identified [6].

Classification methods allow class identity to be 'learned' from a collection of training data, where the class identity is already known. A trained classfier, that has learned a rule for class discrimination, can be applied to similarly generated data for automated classification. Clustering methods allow class structure to be discovered from a collection of data when the class identity is not already known.

Application of informatics and machine learning can start with a variety of challenging initial signal processing stages involving signal (data instance) acquisition and related signal feature extraction. Once a collection of *numerical* signal features is chosen we've arrived at a signal representation, or data instance representation, as a feature vector, where training data consist of such a signal to feature vector mapping with the additional datum of the signal class. Regardless of the preprocessing or observational method, once the signals or data instances are represented as (numerical) feature vectors, the data instance feature vector information is in the form needed for the classification and clustering descriptions that follow [1,2,6].

The definition of cluster is whatever can be clustered… and we don't know what can be clustered until we've done it. This makes clustering even less well-defined than the legal notion of obscenity (according to the Supreme Court's"I know it when I see it" criterion [7]). Fortunately, many clustering problems are almost trivial (you do know it when you 'see' it, as shown in Fig. 1), with a simple clustering process sufficing to reliably capture the clusters of interest.

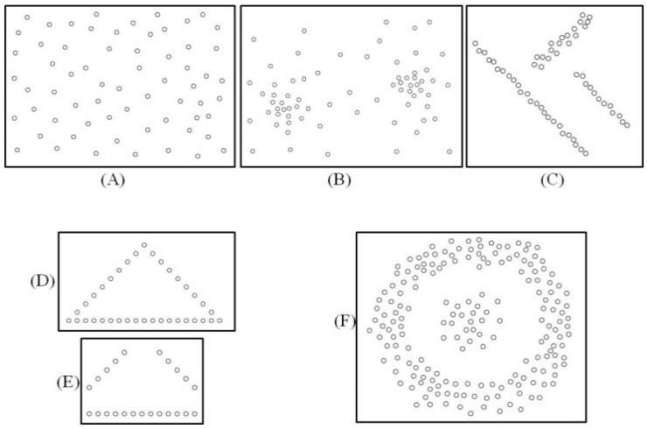


Fig. 1 What is a Cluster? (A) Homogenous, no structure; (B) Two obvious clusters; (C) Three obvious line-shaped clusters; (D) The line clusters joined as one cluster in a triangle; (E) The line clusters segmented as three line-clusters in a triangle (corners missing). (F) Shows two centered clusters, one central the other annular.

The example of two clusters, clearly separated, in Fig. 1.B gives us the first idea on how to describe a cluster: the instances within a cluster should be notably nearer to each other than to instances of any other cluster. Two clusters as approximately parallel lines (Fig.1.C) are an example where this criterion will begin to fail. Focusing on two Gaussian (spherically symmetric) clusters, and data that can reasonably be modeled as such, we can describe separable clusters with mean μ and standard deviation σ, as those for which $\mu_1-\mu_2>\sigma_1+\sigma_2$ (see Fig. 2, related to the Davies-Bouldin clustering index [8]). This can be trivial for many nicely 'peaked' Gaussian clusters with $\mu>>\sigma$. This statistical relation on Gaussians is a comparable relation to the classical Rayleigh criterion from star resolution studies (important in classification of the star as a binary or not) and in later laser optics studies (see Fig. 2). In the case of laser optics, the resolution of two sources can be pushed beyond the Rayleigh limit by means of information pertaining to the individual photons that arrive. Information at the *individual* data instance or feature vector level is automatically the case with the classification and clustering methods that follow, so again the rough resolution-limit criteria for clusters and their separability can often be surpassed by the actual results, e.g., solutions in practice will offer better resolution than the Rayleigh limit or maximal Gaussian overlap limit (see Fig. 2), especially with higher-dimensional feaure vectors and kernel functions.

Distance measure is an important component of clustering algorithms. Either domain knowledge is used to choose an appropriate choice of distance formula, or a collection of distance measures must be tuned over. Euclidean distance is often used, for example, in a regularized form that gives rise to the Gaussian kernel, as will be described in the Background and in the Supplement. The choice of distance

measure, via choice of feature mapping kernel, can strongly affect the SVM convergence (if it even occurs), whether in classification applications or clustering applications.

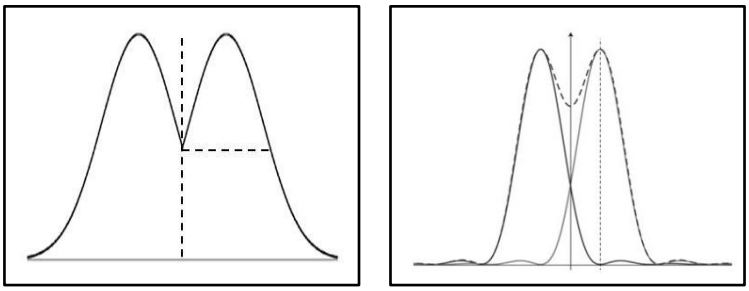


Fig. 2 Classical Cluster Separation Limits: Left: Gaussian overlap limit is set at theircontacting at their full width at half maximum (where FWHM = 2.355 std. dev.s). No closer than touching at half-height is approximately the same as being more than two standard deviations between the means, which is related to the Davies-Bouldin clustering index [8]. Right: the overlap limit for resolution according to the Rayleigh Criterion, where the first local minima of the diffraction pattern must be nearer to the maximum than the maximum of the other diffraction pattern.

Sometimes the training data in classification problems admits no fully separable solution. So, whether working on classification or clustering, a completely separable training set or cluster identity is not always a given. Assuming the notion of separability is satisfactorily worked out, however, there are still complications with notions of connectivity, as already indicated in Fig. 1.D where there is one cluster while the slightly smaller set in 1.E has three clusters. There are also complications due to clusters being embedded inside other clusters (Fig. 1F). Many of these problems are eliminated in practice by effectively mapping to higher dimensions (as occurs in all the kernel methods described in the Background) where embedding issues are trivially eliminated, and connectivity (topology) information retained.

2 Background

There are two main types of clustering: partition-based and agglomerative (hierarchical). The partition-based methods try to directly split the data instances into clusters, where the clusters can be disjoint or overlapping (the ‘fuzzy’ methods), with some cluster fitness measure to determine when clustering is complete. Partitioning can also be done in a model-based fashion, such as with mixtures of Gaussian, where expectation-maximation can be used to optimize the clustering solution. Partitioning can also be done in a Lagrangian optimization context (the SVM-based methods [1, 2, 9]).

All of the classification and clustering methods discussed in this paper rely on a notion of distance (or information divergence) between the feature vectors. Simple clustering rules can be devised based on criteria for maximal distances allowed to be considered in the same cluster, and minimal distances required to be considered separate from elements of some other cluster. An example of disjoint partitioning method is K means clustering (since this is used in comparisons a brief description is given in Sec. 2.1). An example of an overlapping partitioning method is robust kernel fuzzy clustering (in Sec. 2.2 for the same reason). An example of model-based partitioning is the aforementioned mixture of Gaussian methods (see Sec. 2.3). The SVM-based clustering methods (partitioning-based using Lagrangian optimization and bootstrap convergence [1, 2, 6, 10], and single-class SVM clustering [9]) are described in detail in Sec. 2.4 and Sec. 3.1.

The agglomerative or hierarchical methods produce a data-instance relationship tree in the process of doing their clustering. This is itself of interest in many cases, as in phylogentic tree construction in biology. But here more than in the other clustering methods is made all the more apparent the need for 'cluster and verify'. Previously we saw how we didn't know if a clustering solution existed until we found one. With the agglomerative approaches we will see that obtaining a clustering solution is not a problem, the problem is that the clustering is one of so many possibilities. The number of rooted binary trees with N leaves grows as a factorial, (2N-3)!/[2^(N-2)(N-3)!], so for N=3 have 1 tree, for N=5 have 105 trees, and for N=10 have 34,459,425 trees [11]. So the indicated tree solution, and subsequently the indicated clustering solution, is often for just one tree selected from millions or more. Again, a separate objective evaluation of cluster performance is needed before trusting the clustering solution indicated by any method. Even the strongest methods benefit from retry-handling for overall better clustering as will be seen in the Results that follow.

Classification methods are split into two classes: discriminative and generative. Discriminative methods are partition rule based, while generative methods can be hierarchical tree associated via Bayesian Nets, including Bayesian Classification and Naïve Bayes as special cases. Examples of discriminative methods include the classic Perceptron, neural nets (NNs) and SVMs. Some classification methods are very simple and 'local', like k-Nearest-Neighbors (kNN), which classifies according to the k nearest data instances with label information (usually combined according to a majority vote). The online learning of NNs allow them to have a generative mode of classification as well. Further details on general classification methods are discussed elsewhere [12-14]. In what follows we focus on classifier specifications for SVM, along with related SVM-based clustering specifications.

A classifier is typically a simple rule whereby a class determination can be made, such as a decision boundary. Fig. 3 shows labeled training data and a decision boundary with a margin region. Learning the decision rule, or a sufficiently good decision rule, especially if simple and elegant, is the implementation aspect of a

classifier, and can be difficult and time consuming. Even so, this is often manageable because at least there is data to 'learn from', e.g., supervised learning, with instances and their classifications (or 'labels'). Learning for classification can be done very effectively using generalized Support Vector Machines (SVMs), as will be described in what follows. With clustering efforts, or unsupervised learning, on the other hand, we don't have the label information during training. In what follows SVMs will also be shown to be incredibly effective at clustering when used with metaheuristics to recover label information in a bootstrap learning process. Implementation details will also be describe (Sec. 3.5) for distributed SVM training [15], and other speedup optimizations, allowing practical deployment, with the auto-tuning methods described here, of the generalized SVM classification and clustering methods.

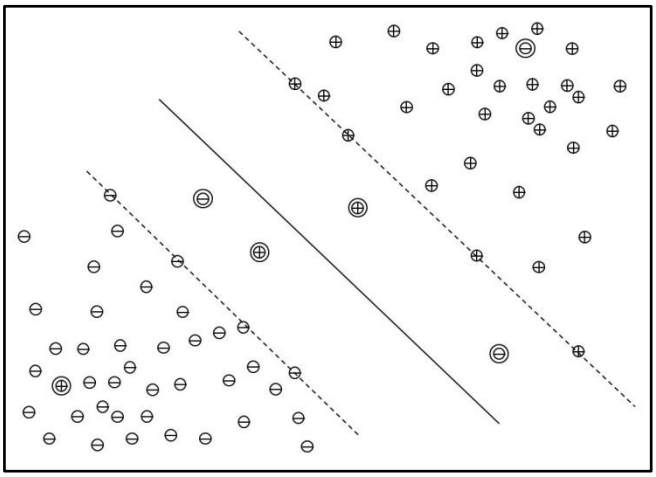


Fig. 3. Decision boundary (solid line); with margin (region between dotted lines). Instances are indicated as positive class (+) or negative class (–), where misclassified data on wrong side of hyperplane (or penalized if in margin), is allowed by incurring a penalty in the optimization process. The misclassified or partly penalized margin-region instances are shown with double walled circles.

In setting up an SVM Classifier one must have training data in the form of feature vectors, where all of the feature vectors are the same length. One typically needs to specify a choice of kernel and kernel parameter (and possibly other parameters), and therein lies the rub. The SVM may not converge with your specification. SVMs have a surprising amount of practical functionality, however, as will be shown. It is fairly easy to tune SVMs, in many cases, by simply using a default set of kernel's and parameter ranges. There is more robust performance, however, with more sophisticated tuning. In the SVM applications that attempt to bootstrap a clustering solution, there appears to be more sensitivity to kernel and kernel parameter overall. More sophisticated tuning methods are, thus, strongly indicated for use in the more challenging SVM applications.

The SVM-based clustering method that will be described here makes use of the SVM-classifier convergence process. Single-convergence initialized clustering methods that use label-flipping after each SVM convergence [1], will be described in the Background Section. The single-convergence methods outperform the non-SVM based methods on the test sets considered. In examining the clustering failures (albeit fewer than with parameterized methods), there appears to be room for improvement. Efforts to handle this with more sophisticated tuning have met with initial success, as will be related in the Results, where perturbed single-convergence processes and multiple convergence processes are examined and scored according to a post-processing sum-of-squared-error (SSE) criterion, where a minimal SSE is sought, to reliably obtain very well-tuned strong SVM clustering performance (further details in Methods).

Use of SVMs for clustering (unsupervised learning) is possible in a number of different ways. As with the multiclass SVM discriminator generalizations, the strong performance of the binary SVM enables SVM-External as well as SVM-Internal approaches to clustering [1, 2 ,6]. Non-parametric SVM-based clustering methods may allow for much improved performance over parametric approaches, particularly since they can apparently be designed to inherit the strengths of their supervised SVM counterparts as will be shown in the data analyzed here. The 'external' SVM clustering algorithm, described in detail in the Methods and Results, clusters data vectors with no *a priori* knowledge of each vector's class.

In application to channel current signal analysis, also briefly described in what follows (with brief details regarding experimental data acquisition provided in the Methods and Supplement), there is generally an abundance of data available (if not, the experimenter can usually just take more samples and make it so). In this situation it is appropriate to seek a method good at both classifying data and evaluating a confidence in the classifications given. In this way, data that is low confidence can simply be dropped. The structural risk minimization at the heart of the SVM method's robustness *also provides a strong confidence measure*. For this reason, SVM's are the classification method of choice for channel current analysis, as they have excellent performance at 0% data drop, and as weak data is allowed to be dropped, the SVM-based approaches become remarkably accurate [16] (see Fig.s in Supplement).

The ability to do fast SVM training (with distributed chunking [15] and, possibly, GPU enhancements [17]) means that ***online*** SVM learning can be managed in a brute-force fashion, with re-tuning on kernels periodically, and directly re-training on a moving window of data. Note: the applications of the SVM methods not only include classification and clustering, but also impact feature extraction and identification in HMM-based methods, using an HMM/SVM vectorization/ classification boost [18], among other things.

The Background sub-sections that follow provide further details on K means clustering and its variants (e.g., disjoint-cluster based methods) in Sec. 2.1, and details on overlapping-cluster based methods, such as Fuzzy Clustering, in Sec. 2.2. Sec. 2.3 briefly describes the popular model-based method of Mixtures of Gaussians and the single central cluster identification method that uses an SVM with a separating hypersphere (not hyperplane) boundary. Sec. 2.4 (and material placed in the Supplement Section) provides a review of the standard SVM formulations and the variations implemented here. In Sec. 2.4, background is provided on the multiclass SVM method and how it handles boundary support vectors and outliers, where the SV handling in the classification algorithm [6] suggests the direct label handling used in the SVM-based clustering process that is described in the Methods (a form of boosting since the training set is thereby altered). Specific details on SVM training via chunking (thereby extending the practical use of SVMs to Big Data settings) are provided in the Methods. The distributed SVM processing discussion is outside the scope of this paper, however, so is mainly described elsewhere [15]. Sec. 2.5 briefly describes the SVM tuning task (and how it is impacted by choice of kernel, algorithmic variants, chunking implementations, and choice of clustering implementation). Sec. 2.5 describes how search metaheuristics are used to perform the tuning task, with particular detail on the simulated annealing type of tuning optimization that is used in this paper.

2.1 K means and Kernel K means

K-means clustering is an example of disjoint or exclusive clustering. The procedure classifies a data set as a certain number of clusters, "K", that is chosen a priori. The algorithmic steps:

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. Closeness is often defined using the Euclidean distance, although any metric could be used. Information diverences, such as the symmetrized relative entropy, can provide a measure of closeness as well.

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 (which will be their mean).

4. Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the data objects into clusters.

There are several problems with the K-means algorithm. Since the number of clusters must be specified a priori, the algorithm by itself cannot solve problems

where the number of clusters is unknown (i.e., need further handling with repeated runs over a range of K values). Next, the algorithm does not always find the optimal clusters because the objective function used does not always use an appropriate distance metric for the problem or that data clusters aren't sufficiently spherically symmetric. Finally, the location of the initial random cluster centroids can heavily influence the clustering solution, creating the possibility that the final cluster identification will be trapped in local minima of the objective function.
Kernel K-means is a K-means clustering algorithm in which the data is mapped into a higher dimensional space. The mapping, and related kernel function, performs a non-linear transformation of the data by mapping it into a space where the separability of the data is increased and notably improved clustering solutions are often found [12-14].

2.2 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 a kernel extension to Fuzzy C-Means clustering [19]. The Robust Kernel Fuzzy clustering method uses a fuzzy partition matrix in its objective function (from Fuzzy C-Means clustering). The fuzzy partition matrix allows data points to have membership values in each of the clusters [20]. In contrast to the Fuzzy C-Means method, the Robust Kernel Fuzzy clustering uses a kernel, which enables recognition of arbitrarily shaped clusters [21, 22]. The method gains robustness through the modification of the Euclidean distance formula in its objective function [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 (giving rise to the same local minima weakness). It is different in that 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 method aims to perform better than the Kernel KMeans algorithm, through providing more resistance to noisy data [20-22]. 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.

2.3 Mixtures of Gaussians and Single Class SVM Clustering

Mixtures 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) [6, 14] algorithm is used to find the Gaussian distributions, which model the data. In the clustering process used in this paper, Gaussian distributions also model clusters in both the preprocessing (Kernel K-means) and External-Relabel SVM clustering phases (see Methods).

The Single Class SVM clustering method [9] identifies a single central data instance cluster, separate from 'other' data instances, including outliers. It does this

using a single SVM run with central cluster boundary given by an enclosing hyper*sphere*. A hypersphere is similar to a hyperplane in that it is a boundary in D-dimensional space which separates data instances. It is different than a hyperplane because, instead of separating the data vectors with different classifications, it surrounds a centralized 'cluster' of data instances inside a hypersphere for single cluster identification (with separation from outliers and non-central clusters).

2.4 SVM Review

In Fig. 3 a decision boundary is shown as the solid line for a 2-D domain, where separating hyperplane generalizations to planes in 3-D and hyperplanes in higher dimensional domains are also possible (within any orientable hyperplane manifold). The notion of a separating hyperplane is not unique to the SVM approach, but it is with use of further Structural Risk Minimization (SRM) constraints via maximizing a margin around the decision hyperplane, where there are constrained to be low numbers of training instances in the margin region (zero if fully separable). The margin is shown as the region around the decision boundary in Fig. 3 that is between the dotted lines on either side of the decision hypersurface. Generalizations to compact [9] or multiple decisions surfaces [1, 23-25] are also possible. The standard binary SVM optimization problem is described in the Supplement, along with the Lagrangian formulation and a brief description of the kernel generalations that are used that are based on informatics and statistical physics stability criteria [1, 6, 16]. The multiclass SVM implementation is decribed next, with particular attention to BSV handling during the learning process as this will then be suggestive of how to proceed with the more direct label-flipping method described in the Methods for the SVM-based clustering used in this paper.

In the multiclass SVMformulation used previously [23-25], there are 'k' classes and hence 'k' linear decision functions – a description of that approach is given here, but with decoupling modifications to enable a new type of multiclass discriminator [1,6] . The role of BSV handling in this formalism is very clear, and provides an automatic parameter selection (no tuning required) for BSV handling. For a given input 'x', the output vector corresponds to the output from each of these decision functions. The class of the largest element of the output vector gives the class of 'x'. Each decision function is given by: $f_m(x) = w_m.x + b_m$ for all m = (1,2,…,k). If y_i is the class of the input x_i, then for each input data point, the misclassification error is defined as follows: $\max_m\{f_m(x_i) + 1 - \delta_i^m\} - f_{yi}(x_i)$, where δ_i^m is 1 if $m = y_i$ and 0 if $m \neq y_i$. We add the slack variable ζ_i where $\zeta_i \geq 0$ for all i that is proportional to the misclassification error: $\max_m\{f_m(x_i) + 1 - \delta_i^m\} - f_{yi}(x_i) = \zeta_i$, hence $f_{yi}(x_i) - f_m(x_i) + \delta_i^m \geq 1 - \zeta_i$ for all i, m. To minimize this classification error and maximize the distance between the hyper-planes (Structural Risk Minimization) we have the following formulation:

Minimize: $\sum_i\zeta_i + \beta(1/2)\sum_m w_m^T w_m + (1/2)\sum_m b_m^2$,
where $\beta > 0$ is defined as a regularization constant.

Constraint: $w_{yi}.x_i + b_{yi} - w_m.x_i - b_m - 1 + \zeta_i + \delta_i^m \geq 0$ for all i,m

Note: the term **(1/2)** $\sum_m \mathbf{b_m^2}$ is added for de-coupling, $1/\beta = C$, and $m = y_i$ in the above constraint is consistent with $\zeta_i \geq 0$. The Lagrangian is:

$L(w,b,\zeta) = \sum_i \zeta_i + \beta(1/2)\sum_m w_m^T w_m + (1/2)\sum_m b_m^2 - \sum_i \sum_m \alpha_i^m (w_{yi}.x_i + b_{yi} - w_m.x_i - b_m - 1 + \zeta_i + \delta_i^m)$,

where all α_i^ms are positive Lagrange multipliers. Now taking partial derivatives of the Lagrangian and equating them to zero (Saddle Point solution): $\partial L/\partial \zeta_i = 1 - \sum_m \alpha_i^m = 0$. This implies that $\sum_m \alpha_i^m = 1$ for all i. $\partial L/\partial b_m = b_m + \sum_i \alpha_i^m - \sum_i \delta_i^m = 0$ for all m. Hence $b_m = \sum_i (\delta_i^m - \alpha_i^m)$. Similarly: $\partial L/\partial w_m = \beta w_m + \sum_i \alpha_i^m x_i - \sum_i \delta_i^m x_i = 0$ for all m. Hence $w_m = (1/\beta)[\sum_i (\delta_i^m - \alpha_i^m) x_i]$ Substituting the above equations into the Lagrangian and after simplification reduces into the dual formalism:

Maximize: $-\frac{1}{2}\sum_{i,j}\sum_m (\delta_i^m - \alpha_i^m)(\delta_j^m - \alpha_j^m)(K_{ij} + \beta) - \beta \sum_{i,m} \delta_i^m \alpha_i^m$
Constraint: $0 \leq \alpha_i^m$, $\sum_m \alpha_i^m = 1$, i = 1…l; m = 1…k,

where $K_{ij} = x_i.x_j$ is the Kernel generalization [1].

SVM Speedup via differentiating BSVs and SVs

If we track the status of support vectors (SVs) according to whether they are boundary (penalty) support vectors (BSVs) or not, and select accordingly, we can get speedup (even in the binary SVM, where choice of C=100 usually good, but C≥10 typically usually okay too). For the multiclass-internal SVM, on the other hand, the speedup with choice of C can be more significant, where C≥100 typically is needed [1].

In some implementations [23-25], the algorithm does not differentiate between SV and BSV, so a lot of time is spent in trying to adjust the weights of the BSVs, i.e., the weak data. In the alternate algorithms used here [1], as soon as we identify a BSV its weight is no longer adjusted. Hence faster convergence is achieved without sacrificing accuracy.

2.5 Kernel tuning via directed search and via search metaheuristics

It is possible to initiate SVM training with model parameters, such as the kernel or kernel parameter, that are so far out of the operational regime that no convergence is obtained in training. So training must be repeated, minimally, to at least obtain convergence, and is typically explored further with tuning on SVM parameters, to obtain convergence that provides high-confidence class or cluster separation. In many situations the SVM tuning can simply be done manually, or partly automatically, with simple range testing, where only small, separated, subsets of the training data are used in the initial tuning tests, before performing more targetedSVM kernel tuning evaluations on the full dataset. Sometimes more elaborate tuning procedures are necessary. In prior kernel tuning efforts (not shown), with the same datasets used here, massive brute-force search was matched by

genetic algorithm (GA) tuning that ran almost instantly in comparison.
Tuning is a form of optimization, and excellent metaheuristics are known for identifying optimal solutions when a scoring function (a fitness function) can be identified and such is provided by the SVM via sensitivity and specificity scores on training data. Metaheuristic optimizations that have been attempted include genetic algorithms, simulated annealing, and steepest ascent hill-climbing, among others. Applications of the simulated annealing methods are shown here for the results involving SVM-external clustering.
Our ability to assess a score with SVMs, and thereby assign a fitness, allows for a collection of metaheuristics that basically reduce to 'look around and take the best way forward' via a series of tweaks. This isn't possible for some problems; however, because the 'looking around' part isn't that informative, e.g., the fitness landscape has sections that are at a fixed level (with noise variations about that level, for example). This is the typical global-search problem that is addessed via random restart, but if the fitness landscape or configuration space is too large random restart won't offer a solution in a reasonable amount of time (even if it can). This is where more clever metaheuristics must be drawn upon to extend to a more global optimization algorithm.
One of the weaknesses of the brute force random restart approach mentioned is that the parameter 'tweak' involved is with a *bounded* perturbative change, which may *already* exclude the possibility of reaching the solution sought (given the computational resources and a reasonable amount of time). So one generalization is to allow for tweaks that are unbounded, but in some perturbatively stable way, such as via the probability of such a perturbation given via a Boltzmann factor, and in doing this we arrive at the Simulated Annealing approach shown in Fig. 4

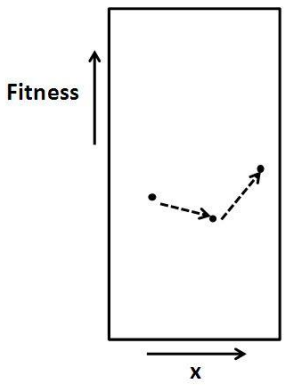


Fig. 4. The Simulated Annealing SearchMetaheuristic. (1) Perform 'tweak', or small perturbation, on the current configuration. (2) The tweak configuration is taken with probability = (1/N)exp[(F(tweak) – F(best))/T], where N is a normalization factor, and T is a 'temperature', e.g., a Boltzmann probability factor for occasionally selecting a lower scoring, possibly non-local, configuration (the case shown for the first transition in the configuration shown to a lower Fitness value). (3) Repeat (1)-(2) until some exit condition is met (no improvements for MAX tries, for example).

3 Methods

The setup for the nanopore experiment, data acquisition, and signal acquisition, is the same as that used in previous nanopore detector analyses [6, 16], and is briefly described in Supplemental Sec. 2. The HMM-based feature extraction methods [6, 16] are described in Sec. S.3, and the Data-rejection heuristics that allow for highly accurate signal calling [6, 16] are described in Sec. S.4. The Methods specifically relevant to the clustering results are given next, including the Bootstrap SVM clustering method in Sec. 3.1, and the

3.1 Bootstrap SVM-clustering

The 'External' SVM Clustering algorithm works by first running a Binary SVM against a data set, with each vector in the set randomly labeled (usually half positives and half negatives), until the SVM converges. Choice of an appropriate kernel and an acceptable sigma value will affect convergence. After the *initial* convergence is achieved, the (sensitivity + specificity) will be low. The algorithm now improves this result by iteratively relabeling the worst misclassified vectors, which have confidence factor values beyond some threshold, followed by rerunning the SVM on the newly relabeled data set. This continues until no more progress can be made. Progress is determined by an increasing value of (sensitivity + specificity). 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 (by iteration on clusters in the binary SVM or direct with merge of multiclass SVM with label flipping algorithms). See Results for a performance comparaison with other clustering methods. The algorithmic variants include modifications to *tolerate* a suitably low number of KKT violators [1, 6].

In practice, the initialization step, that obtains the first SVM convergence, typically takes longer than all subsequent partial re-labeling and SVM rerunning steps. Although convergence is always achieved with the single-convergence SVM-clustering method in the label-flippings, after the initial convergence, convergence to a *global* optimum is not guaranteed. Fig. 5 shows the Purity and Entropy (with the RBF kernel) as a function of Number of Iterations, while Fig. 5 (right) shows the SSE as a function of Number of Iterations [2, 10]. The stopping criteria used for the algorithm is based on the unsupervised (external) SSE measure. Comparison to fuzzy c-means and kernel k-means is shown on the same dataset (the solid blue and black lines in Fig. 5 left and center).

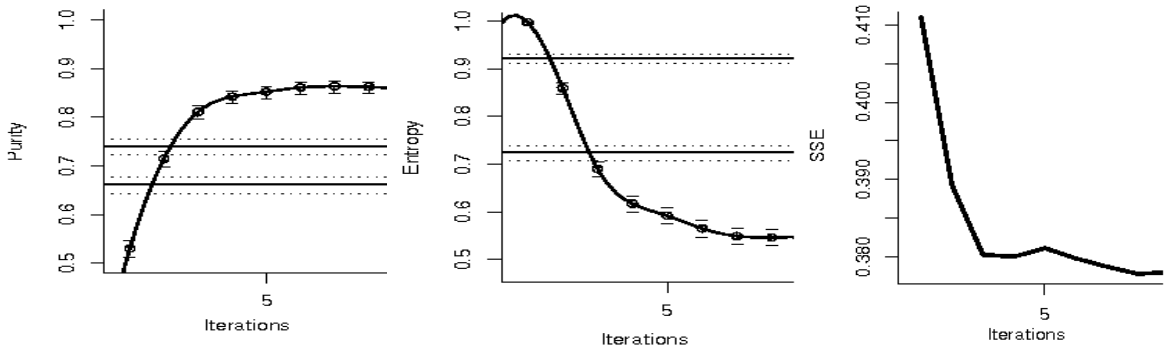


Fig. 5. SVM-external clustering results. (a) and (b) show the boost in Purity and Entropy, respectively, as a function of Number of Iterations of the SVM clustering algorithm. (c) shows that SSE, as an unsupervised measure, provides a good indicator in that improvements in SSE correlate strongly with improvements in purity and entropy. The blue and black lines are the result of running fuzzy c-mean and kernel k-mean (respectively) on the same dataset. In clustering experiments in [2,10], a data set consisting of 8GC and 9GC DNA hairpin data is examined (part of the data sets used in [1,16]). Purity = (TP+TN)/(TP+TN+FP+FN). Entropy = TN/(TN+FN).

Purity = (TP+TN)/(TP+TN+FP+FN) in Fig. 5, which is also known as the Rand Measure [26]. Purity is more typically TP/(TP+FP) (TP are the number of true positive, FP are the number of false positives; this definition is the same as specificity in gene-finding, or positive predictive value in electrical engineering). Entropy = TN/(TN+FN) (the specificity in gene-finding but on negatives).

The pathology of the single-convergence SVM initialization, to get stuck in local minima, motivates the introduction of perturbations into the methods, as shown in Fig.s 6 & 7 for applications involving the Re-labeler Algorithm with perturbations, and with the hybrid k-means/SVM-clustering method with perturbations (details in Sec. 3.3). It is found that the result of the re-labeler algorithm can be significantly improved by randomly perturbing a weak clustering solution and repeating the SVM-external label-swapping iterations as depicted in Fig. 6. To explore this further, a hybrid SVM-external approach to the above problem is introduced to replace the initial random labeling step with k-means clustering (Fig. 7). The initial SVM-external clustering must then be slightly and randomly perturbed to properly initialize the re-labeling step; otherwise the SVM clustering tends to return to the original k-means clustering solution. A complication is the unknown amount of perturbation of the k-means solution that is needed to initialize the SVM-clustering -- it is generally found that a weak clustering method does best for the initialization (or one weakened by a sufficient amount of perturbation) [6, 27].

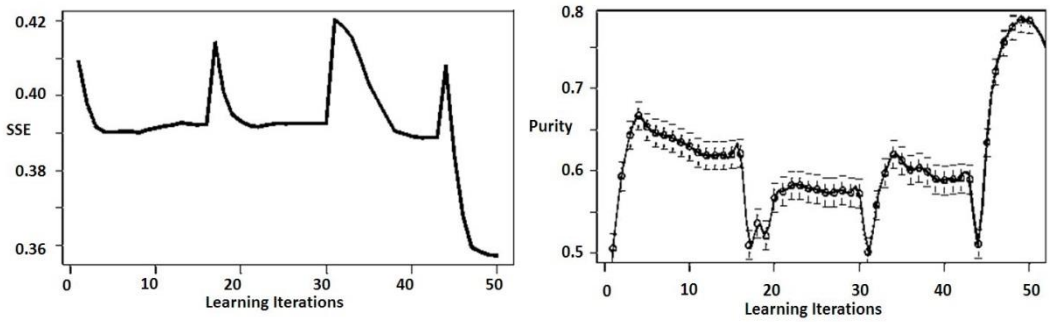


Fig. 6. The Re-labeler Algorithm with Perturbation. The Left plot shows the SSE, the Right the Purity, as the learning process proceeds (vs. learning iteration).The spikes are drops followed by recovery in the validity of the clusters as a result of random perturbation. Note that after 4 runs of perturbation best solution is typically recovered.

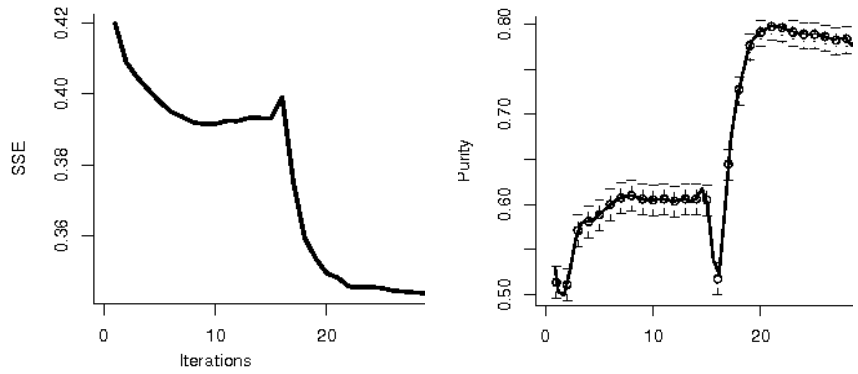


Fig. 7. Left and Right represent the SSE and Purity evaluation of hybrid Re-labeler with Perturbation on the same dataset. Data is initially clustered using k-means to initialize the Re-labeler algorithm. The first segment of the plot (right before the spike at 16) is the result of Re-labeler after 10% perturbation, while the second segment is the result after 30% perturbation.

3.2 Clustering using K-means variants

In work mainly presented elsewhere [27] the Kernel K-means step was used in hybrid clustering efforts to first roughly identify each data vector's cluster before the data vectors were run through an SVM clustering method. A bootstrap Kernel K-means, with intial retraining on data with weak cluster scores dropped (distant from centroid), was also attempted. Minimal details are given on these methods, however, as they never matched the purely SVM-based clustering performance, seemingly inheriting a local minima performance (and limitations) of the K-means approaches.

The data is first mapped into kernel space. The absdiff kernel is used [1, 6, 16], and the K-Means algorithm is performed on the data. The K-Means implementation used in the results that follow use the constraint that every k-means clustering has two clusters (with one cluster 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. This form of initialization is the same as that used in the SVM clustering methods, allowing a more direct comparison of the methods.

3.3 Hybrid K-means/SVM-drop Clustering

The External-Drop SVM clustering method improves the accuracy on a data set that has already been split into two clusters. 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.

Results on the Hybrid K-means/SVM-drop clustering methods are not the focus of this paper, so are described elsewhere [27].

3.4 The SVR Method

Support Vector Reduction (SVR) is a process that is run right after the SVM learning step is complete [6, 15]. Instead of going on to testing data against the training results to get accuracy, we further reduce the support vector set. One way to do this is to coerce some alphas to zero which means they would now fall into the polarization set. Converting the smaller alphas to zeros makes the most sense since a larger alpha indicates that the data point is stronger towards its grouping (polarized sign). This is done using a user-defined alpha cut off value. All alpha values that are under the cut off are pushed to zero. It is not entirely trivial since certain mathematical constraints must be met. The constraint that must be met for this method is the linear equality constraint:

$$\sum_{i=1}^{N} y_i \alpha_i = 0$$

Therefore, the alpha values not meeting the cutoff cannot just be forced to zero unless the value is retained somewhere else in the set. This is done by first sorting the alpha values of the support vectors. Then for each alpha that does not meet the cut off value, the small left over value is added to the largest alpha of the same

polarity (further biases towards SVR). Since the list is sorted it can loop through and evenly distribute the left over values through the larger alphas starting with the largest. The reduction process can cut the number of support vectors significantly, while not significantly diminishing the accuracy. Other observations have shown that the easier the dataset to classify, the larger the reduction via this process.

3.5 Distributed SVM Learning (Chunking Protocols)

Distributed learning on SVMs can be accomplished by breaking the training set into smaller chunks, running separate SVM processes on each of those chunks, and pooling the information that is 'learned', e.g., the support vectors identified as well as nearby (in terms of confidence value) training data vectors and outliers. The reduced pool of data is randomly repartitioned into another round of chunk processing. This is a general approach can be repeated until only a single chunk remains, whose solution is then the overall solution sought or close to it (other minor refinements could be sought).

The training set chunking used in distributed learning is a need that arises even if not interested in a distributed processing speedup since there is a fundamental memory limit encountered with larger SVM training sets (where we need to use a sequential chunk-handling process if on a single machine). For this circumstance, for sequential processing of chunks, we take the SV's identified from the prior round of chunk training and merge it with the next chunk to be trained, and iterate. In this way we never have a pure SV training set [6, 15]. If multiple CPU's are available, however, we can distribute the processing on chunks amongst the CPUs or machines, and pool their SV's (i.e., pass their SV's to the training pool for the next round). The resulting 'pure SV' training sets, however, are often found to not converge, thus requiring more care to avoid convergence pathologies [15].

Chunking the SVM learning process becomes a necessity when training (and classifying) large datasets. The number and size of the chunks depends on the size of the dataset to be trained and the capacity of the computational resource used. When training on an individual chunk is complete, the resulting trained feature vectors split into distinct sets (support vectors, polarization set, penalty set, and KKT violator). If the SVM learning is done well, the largest set consists of the support and polarization feature vectors. The polarization set consists of the feature vectors that have been properly classified. These feature vectors pass the KKT relations and have an alpha coefficient equal to zero. The penalty set consists of the feature vectors which pass the KKT relations, and have alpha coefficients equal to C (the max value). The KKT violators make up another set consisting of feature vectors that violate one of the KKT relations. (The KKT violator set is usually zero at the end of the training process, unless some minimal number of violators, the aforementioned 'tolerance' parameter, is allowed upon learning completion.) These sets give the user different categories of feature vectors

that they can pass to the next chunk(s). To keep the SVM converging to a better solution on the next chunk run, however, support vectors (and sometimes some of the polarization set) are passed to the next chunk(s). The optimal pass-percentages of each feature vector set depend on which kernel is used and the dataset, thus compound what is already a complex SVM tuning optimization problem.

There are different methods of extracting a specified percentage of the feature vectors from the different sets. The specified percentages of feature vectors are randomly chosen from each of the sets except for one. *The support feature vectors extraction method differs since it extracts the support vectors that are nearest to the decision hyperplane.* We choose feature vectors that are closer to the hyperplane (i.e. with smallest magnitude confidence values) in order to pass a tighter hyperplane on to the next chunk(s), and manage accumulation of outliers.

The chunk learning topology used in our distributed approach is slightly different from the Binary Tree splitting described in the Cascade SVM [28]. As discussed above, the large dataset is broken into smaller chunks and the SVM is run on each separate chunk. Instead of bringing the results of paired chunks together, all chunk results are brought together and re-chunked as occurred in the first layer. This process occurs until the final chunk is calculated which gives the trained result. At each training stage, the user has the option to tune the percentage of support vectors and non support vectors to pass to the next set of chunks. Additionally, passed support vectors can be chosen to satisfy some max value (approx. C/10 in cases examined) to produce a tighter hyperplane to better distinguish the polarization sets and eliminate outliers. We also incorporate SVR post-processing in some of the dataruns, where SVR runs as part of the core SVM learning task on each chunk. It uses a user-defined alpha cutoff value for further tuning and can significantly reduce the number of support vectors passed to the next set of chunks (with bias towards elimination of outliers and the large non-boundary alphas). These additional steps reduce the size of the chunks, thus making the algorithm run faster without loss of accuracy. The SVR post-processing also appears to offer similar immunity to the convergence pathology (noted in cases involving 100% SV passing on distributed learning topologies).

There are a variety of ways to avoid the pure SV training-set pathology [6, 15, 29]. Since we are interested in training set reduction overall, we consider the possibility of simply reducing the SV set. This appears to work in preliminary tests on well-studied datasets of interest (see Table 1), where the SV's nearest to the decision hyperplane (most supporting the hyperplane) are retained. For the channel current data examined in, with 150-component feature vectors, we find that 30% SV passing is optimal on distributed learning topologies. The low SV-passing percentage that is found to work in *distributed* chunking might fundamentally be an issue of outlier control during distributed learning. Further reduction of SV passed is possible with dropping SV's with confidence values at

the other extreme, near zero (i.e., those nearest and most strongly supporting the hyperplane). This entails a additional Support Vector reduction (SVR) process that is run right after the SVM learning step is complete, where we further reduce the support vector set according to some confidence cut-off (actually imposed via cut-off on associated Lagrange multiplier in the SVM/SMO implementation). By reducing the number of support vectors propagated into the next round, we further accelerate the chunked processing. In this way, a strongly performing distributed chunk-training process is possible, with speedup by ~10 for the multicore PC used in the example shown in the table shown in Table 1 (with no significant loss in accuracy). It appears possible to automate the tuning & selection procedures. To achieve this, it is necessary to examine the stability of the algorithmic parameters such as the pass percentages on the different types of learned data (e.g., see Table 1 for pass percentages indicated). Further results on distributed SVM learning is given in [6, 15, 29].

SVM Method	**Sensitivity**	**Specificity**	**(SN+SP)/2**	**Time (ms)**
SMO (non-chunked)	0.87	0.84	0.86	47708
Sequential Chunking	0.84	0.86	0.85	27515
Multi-threaded Chunking	0.88	0.78	0.83	7855
SMO (non-chunked) with SV Reduction	0.91	0.81	0.86	43662
Sequential Chunking with SV Reduction	0.90	0.82	0.86	18479
Multi-threaded Chunking with SV Reduction	0.85	0.83	0.84	5232
Multi-threaded Dist. Chunking with SVR	0.85	0.83	0.84	5973

Table 1. Performance comparison table for the different SVM methods. The distributed chunking used three identical networked machines. Dataset = 9GC9CG_9AT9TA (1600 feature vectors, 400 each from class {9GC,9CG,9AT,9TA}, where {9GC,9CG} are labeled positives, and {9AT,9TA} are labeled as negatives). SVM Parameters: Absdiff kernel (with sigma=.5, C = 10, Epsilon = .001, Tolerance = .001). For chunking methods: Pass 90% of support vectors, Starting chunk size = 400, maxChunks = 2. For SVR methods: Alpha cut off value = 0.15.

To further enhance processing speed, one can also boost thread-processing speed on a given computer via use of GPU processing. This has already been undertaken,where distributed chunks of SVM training data were processed using a CPU/GPU that, at marginal added cost (a graphics card), provided as much as a 32-fold speedup on the channel current blockade classification [17]. Similar GPU speed enhancements to the other machine learning algorithms are possible as well.

3.6 Bootstrap Tuning Methods

Tuning is needed to optimize the choice of kernel & kernel-parameter used by the SVM. This is often handled simply by ranging over a collection of roughly 10

kernel types and each with roughly 10 attempted kernel parameter settings (assuming each kernel is single-parameter kernel), and to do this only on smaller test sets in the training data, where the time complexity of the SVM training is directly tied to the training-kernel computation, which is quadratic in the number of training instances. Although caching can modify the assumptions on time-complexity, there is generally an approximately quadratic time-complexity in the size of the training instances regardless. Chunking must be used to break past this, possibly with use of GPU capabilities (where there is still the need to do chunking).

Once the small test set is done on the initial kernel screening indicated above, a sub-set of kernel will emerge as best, and these are considered again with larger training sets, eventually allowing selection of a good choice of kernel and kernel parameter that is trained and tested on the full dataset. The more directed tuning paradigms typically involve simulated annealing in this kernel optimization setting (and in the clustering process described in the Results to follow). Algorithmic and implementation parameters can also be considered in the tuning, which means we now have a collage of different parameter types in a coupled optimization task. For this type of generalization, genetic algorithms have been applied with success (but not shown in what follows). These more sophisticated tuning methods may not always be necessary in the SVM classification applications, however, but do allow for successful classifications in some situations where simple methods do not. In the SVM-based clustering methods to be described in what follows, these tuning methods generally play an even more important role.

4 Results

4.1 Clustering Method Comparison

The single-convergence initialized SVM-based clustering algorithm [1, 2, 6] clusters data with no *a priori* knowledge of input classes. The algorithm initializes by first running a binary SVM classifier against a data set with each vector in the set randomly labeled, this is repeated until an *initial convergence* occurs. The convergence may have to be attempted several times (with different randomized initializations) before a SVM solution is obtained -- but convergence is usually seen on the first or second try. Once an SVM solution is obtained, however, the strengths of the SVM classifier can start to be levereged to full advantage. SVMs are ideal in this effort as they not only classify, but offer a confidence parameter with their classification, and can do so in a generalized kernel space. Thus, once a convergent solution is obtained, label-flipping can be done for high confidence mislabels (where the confidence score has large magnitude but is not in agreement with the current label sign). At each label-flipping iteration we can potentially have unequal numbers of positives and negatives changing their labels, thus, asymmetrically sized clusters can be realized from a half-positive/half-negative initialization. This iterative process continues until there is no longer a high-confi-

dence mislabel by the SVM, or until an external cluster validation, such as the sum-of-squared error (SSE) on each cluster, remains relatively unchanged. There are numerous tuning parameters in the SVM-classification process itself, as well as in the SVM-clustering halting specification, and even tuning choices in the SVM chunk-training (that may be necessary for larger data sets). As shown in Fig. 8, SVM-based clustering [1] often outperforms other methods. Further details for the implemenations for each of the methods is given in [1, 6, 30].

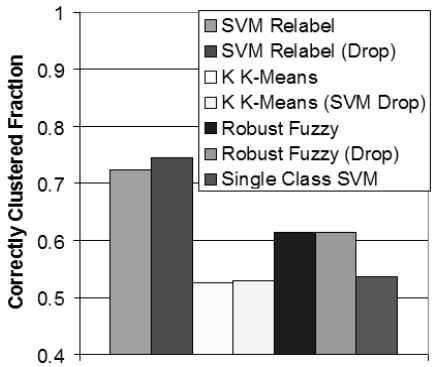


Fig. 8. Clustering performance comparisons: SVM-external clustering compared with explicit objective function clustering methods. Nanopore detector blockade signal clustering resolution from a study of blockades due to individual molecular capture-events with 9AT and 9CG DNA hairpin molecules [6, 30]. The SVM-external clustering method consistently out-performs the other methods. The optimal drop percentage on weakly classified data differed for the different methods for the scores shown: Our SVM relabel clustering with drop: 14.8%; Kernel K-means with drop: 19.8%; Robust fuzzy with drop: 0% (no benefit); Vapnik's Single-class SVM (internal) clustering: 36.1%.

4.2 Stabilized, single-convergence intitialized, SVM-External Clustering

The External ('bootstrap') SVM Clustering data set is chosen to initialize with an equal number of positives and negatives. In the dataset studeud there are 200 8GC blockade signals and 200 9GC blockade signals (see [16] for details about these molecules). Each feature vector is 150 dimensional and normalized to satisfy the L_1 (norm = 1) constraint. Features from the 8 and 9 base-pair blockade signals were extracted using Hidden Markov Models (for details, see [16]). Although convergence was easily achieved with the External SVM Clustering algorithm (see the Methods), convergence to a global optimum was not guaranteed.

In [10, 27], we see that a small value of Kernel-SSE (herein referred to as SSE) is shown to provide us with a reliable cluster validation measure. The External SVM Clustering (SVM-Relabeler) algorithm does not use an objective function, and the hope is that by running the algorithm in its purest form, the resulting clusters are reliable solutions. However, running this algorithm in this basic fashion does not consistently provide us with a satisfying clustering solution. In fact, the solution

space can be divided into three sets: successful, local-optimum, and unsuccessful. Unsuccessful solutions and local optima solutions are undesirable and the objective is to find a method to eliminate their selection by simply re-clustering for objectively improved clustering (via SSE scoring, for example). Since, the solutions in the unsuccessful set are easily identified by an SSE comparable to that of a randomly labeled data set, they can be easily eliminated by post-processing. In a control experiment we have randomly labeled the dataset 5000 times and calculated the SSE distribution for the experiment. The resulting distribution has a good fit to Johnson's SB distribution and is illustrated in the histogram of Fig. 9. Using a fitted distribution one can calculate the p-value of a given SSE. For a SSE threshold of 170.5 (accidentally very unlikely) we can directly eliminate the unsuccessful set.

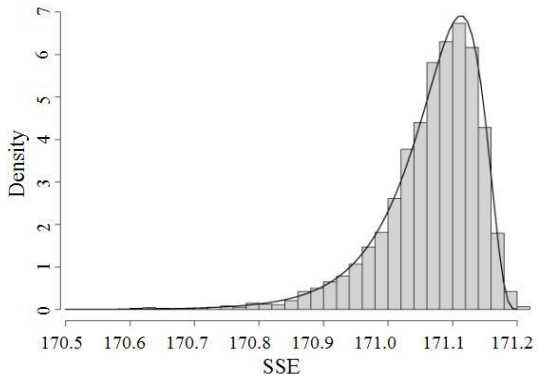


Fig. 9. Nanopore feature vector data (in standard 150 component, L1-norm, format) is randomly labeled 5000 times followed by evaluation of SSE values and production of a histogram of those values as shown. The resulting distribution has a good fit to Johnson's SB distribution with gamma= −5.5405, delta = 1.8197, lambda = 2.7483, epsilon = 168.46.

To substantially reduce the local optimum solutions, however, SSE thresholding does not scale well. One solution is to use a simple hill climbing algorithm which is to run the algorithm for a sufficiently long number of iterations to find the solution with the lowest SSE value. To do this, the clustering algorithm is run repeatedly and randomly initialized every time. A solution is accepted as the best solution if it has a lower SSE than the previously recorded value. This can be a very slow learning process, and suggest and alternative tuning method from statistical learning – simulated annealing.

It is observed that random perturbation by flipping each label at some probability, p_{pert}, is often sufficient to switch to another subspace where a better solution can be found. (Note that $p_{pert} = 0.50$ has the effect of random reinitialization and $p_{pert} = 1$ flips all of the labels.) The hope is that perturbation with $p_{pert} \leq 0.50$ results in a faster convergence. Reliability can then be achieved by searching through the solution

space. The procedure described next uses a modified version of Simulated Annealing to achieve this desired reliability.

As shown in Fig. 10 left, top panel, constant perturbation with $p_{pert} = 0.10$ results in a local-optimum solution that could be otherwise avoided by using a perturbation function depending on the number of iterations of unchanged SSE (Fig. 10 right, top panel). These results were produced using an exponential cooling function, $T_{k+1} = \beta^k T_k$, with $\beta = 0.96$ and $T_0 = 10$. The initial temperature, T_0 should be large enough to be comparable with the change of SSE, ΔSSE, and therefore increase the randomness by making the Boltzman factor $e^{-\Delta SSE / T} \approx e^0$, while β (< 1) should be large enough to speed up the cooling effect. See [6, 10, 27] for further details.

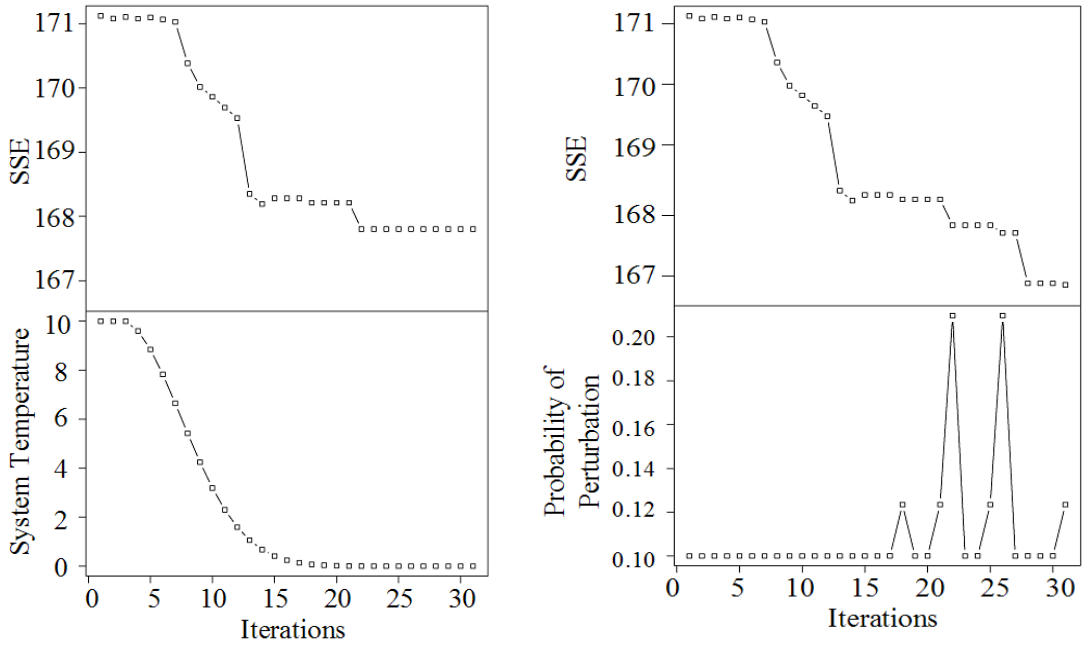


Fig. 10. (left) Simulated annealing with constant perturbation, (right) Simulated annealing with variable perturbation. As shown in left, top panel, simulated annealing with a 10% initial label-flipping results in a local-optimum solution. In the right panel this is avoided by boosting the perturbation function depending on the number of iterations of unchanged SSE (right, top panel). These results were produced using an exponential cooling function, $T_{k+1} = \beta^k T_k$, with $\beta = 0.96$ and $T_0 = 10$. Using Absdiff kernel with gamma = 1.8.

In the effort shown in Fig. 10 it was found that random perturbation and hybridized methods (with more traditional clustering methods) could help stabilize the clustering method, but often at significant cost to its performance edge over other clustering methods (apparently due to getting stuck in local minima traps to which the other parametric clustering methods are susceptible).

The ‘pure’ SVM-external clustering method appears to offer very strong solutions about half the time – which allows for optimization simply by repeated clustering attempts and looking for the most tightly clustered (smallest SSE) solution. Details on the more informed (multiple-conversion initialized) version of the process given next.

4.2 Stabilized, multiple-convergence, SVM-External Clustering

In Fig. 11 the SSE scoring is tracked with learning iteration on three representative SVM clustering processes for the successful, unsuccessful, and local minima solutions, where we use the same datasetas in Sec. 4.1, and same kernel settings (Absdiff kernel with gamma = 1.8), and computing SSE, but now iterating with multiple convergences instead of perturbations. See [6, 10, 27] for further details.

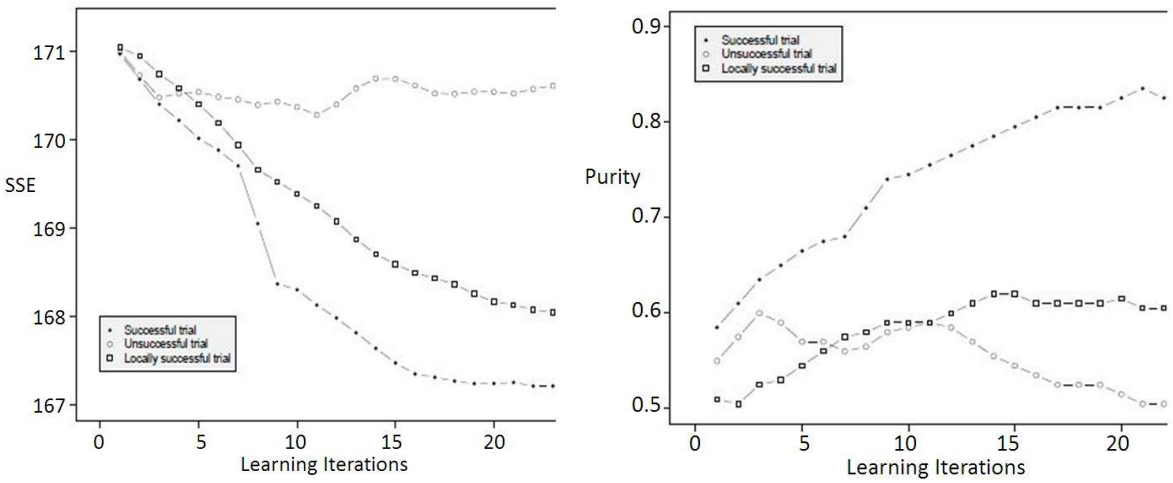


Fig. 11. Multiple-convergence, SVM-External Clustering. Three multiple clustering convergences (different trials) of SVM-Relabeler algorithm demonstrating the range of the possible solution space as measured by SSE and purity (a ‘successful’, an unsuccessful, and a partly-successful trial). Choosing the good SSE external measure (Left Panel) typically provides a generalized clustering that has high purity (Right Panel). The improvments in Purity and SSE with learning iterations are shown. Once Learning slows (SSE unimproved), a restart for a new convergence clustering is done. Usually in the first two or three attempts a strongly performing converegence is seen as with the example shown here.

5 Discussion

Single-convergence SVM-clustering, that uses simulated annealing to tune over the clustering relabel rate, is shown to be very effective at obtaining good clustering

solutions. This is shown to be enhanced further with introduction of perturbations. Multiple-convergence SVM-clustering is also shown to be very effective, and provides an overall non-parametric means to clustering. In preliminary work [1,6], it is found that the relabel-based SVM-based clustering method also offers prospects for inheriting the very strong performance of standard SVMs from the supervised classification setting. This offers a remarkable prospect for knowledge discovery via recognition of patterns and clusters without the limitations imposed by requiring a parametric model (e.g., no cooling rate for the simulated annealing, etc.). Resolution of the clusters can be at an accuracy comparable to the supervised setting (i.e., where cluster identities are already specified).
The completely non-parametric clustering approach is to first obtain *multiple* SVM convergences from separate initializations. An objective SSE evaluation can be done on the solutions, and the best could simply be taken, as shown in the Results. This clustering algorithm already provides very good results but could do even better with further use of the multiple convergences and the confidence information they obtain (to then be used in repeated multiple convergences). This is similar to steepest ascent search where numerous configurations are generated and the one with the best fitness is chosen at the end of each iteration.

6 Conclusion

SVMs are fast, easily trained, discriminators, for which strong discrimination is possible without over-fitting complications. SVMs are firmly grounded as a variational-calculus based optimization method that is constrained to have structural risk minimization (SRM), unlike neural net classifiers, such that an SVM offers noise tolerant solutions for pattern recognition in a variety of settings. An SVM determines a hyperplane that optimally separates one class from another, while the structural risk minimization (SRM) criterion manifests as the hyperplane having a thickness, or "margin," that is made as large as possible in the process of seeking a separating hyperplane. The SVM approach thereby encapsulates model fitting and discriminatory information in the choice of kernel in the SVM, and a number of novel kernels are used. SVMs are good at both classifying data and evaluating a confidence in the classifications given, which leaves an opening for use of metaheuristics to bootstrap into a clustering capability, as explored in a number of algorithmic variations in this paper. SVM use in clustering appears to offer a very robust clustering platform when enhanced with simulated annealing tuning on clustering parameters.

Acknowledgements. The author would like to thank lab technicians Amanda Alba and Eric Morales for help performing the nanopore experiments and University of New Orleans Master's students Anil Yelundur, Ken Armond, Charlie McChesney, and Sepehr Merat for help with the JAVA RMI implementation and dataruns based on the authors C/Perl implementation.The author would like to thank NIH, NSF, NASA, and the Louisiana Board of Regents for research support.

References

[1] S. Winters-Hilt, A. Yelundur, C. McChesney, M. Landry, Support Vector Machine Implementations for Classification & Clustering, *BMC Bioinformatics*, **7** (2006), Suppl. 2, S4. https://doi.org/10.1186/1471-2105-7-s2-s4

[2] S. Winters-Hilt and S. Merat, SVM Clustering, *BMC Bioinformatics*, **8** (2007), Suppl. 7, S18. https://doi.org/10.1186/1471-2105-8-s7-s18

[3] W.B. Croft, D. Metzler, T. Strohman, *Search Engines: Information Retrieval in Practice*, Pearson Education Inc., 2015.

[4] S. Guha, R. Rastogi and K. Shim, ROCK: A Robust Clustering Algorithm for Categorical Attributes, *Proceedings 15th International Conference on Data Engineering*, (1999), 512-521. https://doi.org/10.1109/icde.1999.754967

[5] D. Wang and Z. Zhou, Application of Data Field Clustering in Computer Forensics, *Third International Conference on Information and Computing*, **1** (2010), 147-150. https://doi.org/10.1109/icic.2010.43

[6] S. Winters-Hilt, *Machine-Learning Based Sequence Analysis, Bioinformatics & Nanopore Transduction Detection*, Lulu, 2011.

[7] SCOTUS 378 U.S. at 197 (Stewart, J., concurring) 1964.

[8] D.L. Davies and Donald W. Bouldin, A Cluster Separation Measure, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-1** (1979), no. 2, 224-227. https://doi.org/10.1109/tpami.1979.4766909

[9] A. Ben-Hur, D. Horn, H.T. Siegelmann, V. Vapnik, Support Vector Clustering, *Journal of Machine Learning Research*, **2** (2001), 125-137.

[10] S. Winters-Hilt and S. Merat, Classification and Clustering using Support Vector Machines, 2007 preprint.

[11] D.E. Krane and M.L. Raymer, *Fundamental Concepts of Bioinformatics*, 1st Ed., Benjamin/Cummings, 2002.

[12] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995. https://doi.org/10.1007/978-1-4757-2440-0

[13] V.N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[14] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[15] S. Winters-Hilt, Distributed SVM Learning and Support Vector Reduction, Submitted April 2017.

[16] S. Winters-Hilt, W. Vercoutere, V.S. DeGuzman, D.W. Deamer, M. Akeson and D. Haussler, Highly Accurate Classification of Watson-Crick Basepairs on Termini of Single DNA Molecules, *Biophys. Journal*, **84** (2003), 967-976. https://doi.org/10.1016/s0006-3495(03)74913-3

[17] Z. Hang, *Distributed Support Vector Machines with Graphics Processing Units*, MS Thesis, University of New Orleans, Stephen Winters-Hilt Advisor, 2009.

[18] B. Roux and S. Winters-Hilt, Hybrid MM/SVM Structural Sensors for Stochastic Sequential Data, *BMC Bioinformatics*, **9** (2008), Suppl. 9, S12. https://doi.org/10.1186/1471-2105-9-s9-s12

[19] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, USA, 1981. https://doi.org/10.1007/978-1-4757-0450-1

[20] W. Du, K. Inoue, K. Urahama, Robust Kernel Fuzzy Clustering, Chapter in *Fuzzy Systems and Knowledge Discovery*, Lecture Notes in Computer Science, Vol. 3613, Springer-Verlag Berlin Heidelberg, 2005, 454-461. https://doi.org/10.1007/11539506_58

[21] M. Girolami, Mercer kernel-based clustering in feature space, *IEEE Trans. on Neural Netw.*, **13** (2002), 780-784. https://doi.org/10.1109/tnn.2002.1000150

[22] D.W. Kim, K.Y. Lee, D. Lee, K.H. Lee, Evaluation of the performance of clustering algorithms in kernel-induced feature space, *Patt. Recog*., **38** (2005), 607-611. https://doi.org/10.1016/j.patcog.2004.09.006

[23] K. Crammer and Y. Singer, On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines, *Journal of Machine Learning Research*, **2** (2001), 265-292.

[24] C.W. Hsu and C.J. Lin, A Comparison of Methods for Multi-class Support Vector Machines, *IEEE Transactions on Neural Networks*, **13** (2002), 415-425. https://doi.org/10.1109/72.991427

[25] Y. Lee, Y. Lin and G. Wahba, *Multicategory Support Vector Machines*, Technical Report 1043, Department of Statistics, University of Wisconsin,

Madison, WI, 2001. http://citeseer.ist.psu.edu/lee01multicategory.html

[26] W. M. Rand, Objective criteria for the evaluation of clustering methods, *J. of the Am. Stat. Assoc*., **66** (1971), no. 336, 846-850. https://doi.org/10.2307/2284239

[27] S. Merat, *Clustering Via Supervised Support Vector Machines*, MS Thesis, University of New Orleans, Stephen Winters-Hilt Advisor, 2008.

[28] H.P. Graf, E. Cosatto, L. Bottou, I. Durdanovic and V.N. Vapnik, Parallel Support Vector Machines: The Cascade SVM, *Advances in Neural Information Processing Systems*, Vol. 17, 2004.

[29] K.C. Armond, Jr., *Distributed Support Vector Machine Learning*, MS Thesis, University of New Orleans, Stephen Winters-Hilt Advisor, 2008.

[30] C. McChesney, *External Support Vector Machine Clustering*, MS Thesis, University of New Orleans, Stephen Winters-Hilt Advisor, 2006.

[31] G. Ratsch, K. Tsuda, K. Muller, S. Mika and B. Scholkopf, An introduction to kernel-based learning algorithms, *IEEE Trans. Neural Netw*., **12** (2001), no. 2, 181-201. https://doi.org/10.1109/72.914517

[32] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, **2** (1998), no. 2, 121-167. https://doi.org/10.1023/a:1009715923555

[33] J. Platt, *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, Microsoft Research Tech. Rep. MSR-TR-98-14, 1998.

[34] S. Kullback, *Information Theory and Statistics*, Dover, 1968.

Supplement

S.1 The Binary SVM

S.1.1 SVM Lagrangian formulation [6]

The SVM approach encapsulates a key Structured Risk Minimization (SRM) criterion when it seeks to obtain the separable solution for which margin thickness for the separating hyperplane, "d", is the greatest. This is the solution for which the separating (decision) hyperplane is the furthest distance possible from the positive & negative support vectors (the nearest data points if separable), which permits structural risk minimization.

In order to formulate the SVM Lagrangian, we first need multipliers for the collection of separability constraints on solutions: $y_i(\omega \bullet \boldsymbol{x} - b) - 1 \geq 0 \ \forall i$. For SRM, we then need to maximize $d = 2/\|\omega\|$, or minimize $\|\omega\|^2$, which is chosen due to simplifications in the formalism that follows (i.e., if we max $2/\|\omega\|$ by min on $\|\omega\|$, it could just as well be done with min on $\|\omega\|^2$). The Lagrangian formulation then should have one multiplier constraint for each training instance, where $y_i(\omega \bullet \boldsymbol{x} - b) - 1 \geq 0$, and minimize on $\|\omega\|^2$ overall, so:

$$L(\vec{\omega}, b, \vec{\alpha}) = \frac{1}{2}\|\omega\|^2 - \sum_i \alpha_i [y_i(\omega x_i - b) - 1], \alpha_i \geq 0$$

We seek to minimize L on $\{\vec{\omega}, b\}$ and to extremize (maximize in this case) L on $\{\vec{\alpha}\}$, i.e., we seek a minimization -- maximization saddle-point optimization for the solution.

The Wolfe Dual Calculations, with or without slack variable, have the form:

$$\tilde{L}(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j, \ \alpha_i \geq 0$$

where we want to find the α 's that maximize $L(\alpha)$. Similarly, for the L_σ Dual:

$$\tilde{L}_\sigma(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j, \ C \geq \alpha_i \geq 0$$

So, the duals are the same, with or without slack variable, aside from the $C \geq \alpha_i$ ($\max(\alpha) \leq C$) constraint.

S.1.2 SVM Kernels and Algorithm Variants

Notice how in the Dual reduction the dependence on the training data only appears in the inner product term. We can generalize from the simple inner product term in a number of ways, and in doing so arrive at the SVM Kernel generalization. The choice of kernel eliminates the need for refining a choice on feature vector mappings beyond a certain point, such as requiring some consistent normalization on feature vectors, for example, which is made consistent with choice

of kernel (e.g., one could take a discrete probability distribution as feature vector, with its L_1-norm, and its pairing with the entropic kernel). The SVM kernels used in the analysis are referred to as 'Occam's Razor', or 'Stability' kernels [1,6]. All of the stability kernels examined perform strongly on channel current data, often outperforming the Gaussian Kernel. The kernels fall into two classes: regularized distance (squared) kernels; and regularized information divergence kernels. The first set of kernels strongly models data with classic, geometric, attributes or interpretation. The second set of kernels is constrained to operate on $(\boldsymbol{R}^+)^{\mathbf{N}}$, the feature space of positive, non-zero, real-valued feature vector components. The space of the latter kernels is often also restricted to feature vectors obeying an L_1-norm = 1 constraint, i.e., the space of discrete probability vectors. In all of the data-runs with the probability feature vector channel current data considered in [1,6], the two best-performing kernels are the entropic and the indicator 'Adbsdiff' (or 'Variational') kernels, with the Gaussian trailing in performance in general (but still outperforming other methods such as polynomial and dot product). The L_1-norm channel current feature vector components appear to encapsulate a key constraint of a discrete probability vector via its domain selection and its associated optimal kernel sets.

If the distance term in the Gaussian is denoted $d_G = |\mathbf{x}_i - \mathbf{x}_j| = \sqrt{(\sum_k (x_j^k - x_i^k)^2)}$, the Gaussian Kernel can be written as $K_G(\mathbf{x}_i,\mathbf{x}_j) = \exp(-(d_G)^2/2\sigma^2)$. In general, exponential regularization of a metric on the feature vectors, as in the Gaussian, will provide a Kernel satisfying Mercer's conditions [6,31]. Since the "kernels" considered in what follows are an extension from those justified by the geometric heuristic to those justified by an information-theoretic heuristic (the final arbiter of performance being empirical results), ***the key property from the above, in obtaining alternate kernels, will be the exponential regularization***.

It is found that ***the other key property is a stability property*** that ties together the best performing kernels from the various cases. For the Gaussian kernel, the stability property is exhibited when the log Kernel variation on feature vector components is calculated:
$\partial \ln (K_G(\mathbf{x}_i , \mathbf{x}_j)) / \partial x_i^k = (x_j^k - x_i^k)/\sigma^2$,
where "x_i^k" is the k^{th} component of the i^{th} feature vector and "stability" is indicated by the sign of the difference term $(x_j^k - x_i^k)$, e.g., for

$$K_G(\vec{y},\vec{z}) = \exp(-\| \vec{y} - \vec{z} \|^2 / 2\sigma^2) : \frac{\partial \ln K_G(\vec{y},\vec{z})}{\partial y_K} = (y_K - z_K)/\sigma^2$$

Clearly, the sign is important, as is a notion of difference. Suppose we generalize on this basis to decouple the sign (stability in orientation) convention from the "notion of differance", here providing a new kernel expression, for the "variational kernel" by way of an integration factor:

$$\frac{\partial \ln K_G(\vec{y},\vec{z})}{\partial y_K} = (\frac{-1}{2\sigma^2})\,(\frac{\text{sign}(y_k - z_k)}{\sqrt{\sum_k |\, y_k - z_k\,|}})$$

$$K_v(\vec{y},\vec{z}) = \exp(-\sqrt{\sum_k |\, y_k - z_k\,|}\,\Big/ 2\sigma^2)$$

The subscript "V" in K_V is meant to denote "variational" kernel (sometimes referred to as "indicator" kernel or "Absdiff" kernel). For suitable choice of tuning parameter σ, the variational kernel offers the best performance on the data sets considered. The regularized distance in K_V is the square root of the "Variational" distance: $V(\mathbf{x}_i|\,|\,\mathbf{x}_j)=\sum_k |\, x_j^k - x_i^k|$. It is found that the variational kernel is usually the best performing kernel on the L_1 normed data considered in the channel current analysis (L_1 norm: $|x|_1=\sum_k |\, x_k\,|$, a discrete prob. dist if $x_k > 0$ also). The argument of the exponential in the variational kernel is a distance squared, with $K_v = \exp(-d_v^2/2\sigma^2)$, thus the variational kernel automatically satisfies Mercer's conditions.

Consider now the case where the notion of difference is not arithmetic but multiplicative, i.e., based on $(1-z_k/y_k)$ rather than (y_k-z_k) (for the Gaussian). In doing so, we must restrict to $y_k \neq 0$ of course. As before, the sign of (y_k-z_k) is information preserved in $(1-z_k/y_k)$, but the latter is not integrable. However, $\ln(y_k/z_k)$ also provides sign info -- positive when $y_k > z_k$, etc., as before, and also includes a ratio. Which to go with? A combination seems best as this is integrable:

$$\frac{\partial \ln K_G(\vec{y},\vec{z})}{\partial y_K} = (\frac{-1}{2\sigma^2})\,[(1-\frac{z_k}{y_k}) + \ln(\frac{y_k}{z_k})]$$

$$K_\sigma(\vec{y},\vec{z}) = \exp(-[D(y\,\|\,z) + D(z\,\|\,y)]/2\sigma^2)$$

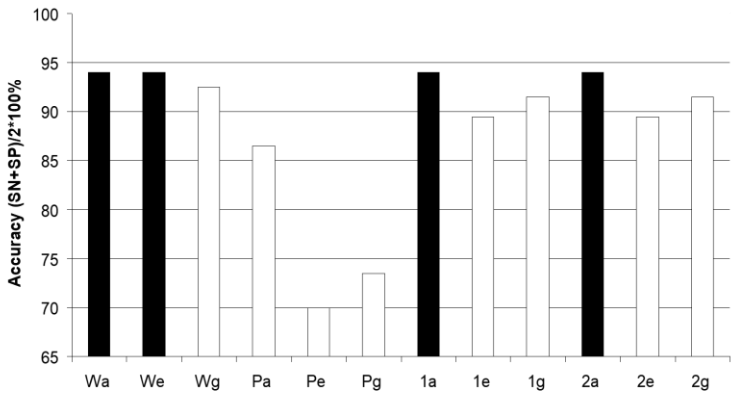
This is usually a close 2nd to the K_v kernel, in performance, sometimes outperforming for the datasets examined. This kernel relates feature vectors via relative entropy terms:

$$D(y\,\|\,z) = \sum_k y_k \ln(\frac{y_k}{z_k})$$

The doubly novel aspect of the entropic kernel is that it would be the very first guess if one wanted to generalize from kernels based on exponentially regularized, square distances, to exponentially regularized, symmetrized, divergences (beginning with the most fundamental, symmetrized "relative entropy" also known as the Kullback-Leibler information divergence).

A comparison of some of the SVM Kernels of interest is shown in Suppl. Fig. 1, with "regularized" distances or divergences, where they are regularized if in the form of an exponential with argument the negative of some distance-measure squared ($d^2(x,y)$) or symmetrized divergence measure ($D(x,y)$), the former if using a geometric heuristic for comparison of feature vectors, the latter if using a information divergence heuristic. Results in Suppl. Fig. 1 are shown for the Gaussian Kernel: $d^2(x,y)=\Sigma_k(x_k-y_k)^2$; for the Absdiff or Variational Kernel $d^2(x,y)=(\Sigma_k|x_k-y_k|)^{1/2}$; and for the Symmetrized Relative Entropy Kernel $D(x,y)= D(x||y)+D(y||x)$, where $D(x||y)$ is the standard relative entropy.

In the standard Platt SMO algorithm, $\eta=2*K_{12}-K_{11}-K_{22}$, for which speedup variations are described to avoid calculation of this value entirely [6,32,33]. A middle ground is obtained with the following definition $\eta =2*K_{12}-2$; If ($\eta >=0$) $\{\eta \geq -1;\}$ (labeled WH SMO in Suppl. Fig. 1, with underflow handling and other details that differ slightly in the WH-SMO implementation as well).



Suppl. Fig. 1. Comparative results are shown on SVM performance with different kernels and algorithmic variants. The classification is between two DNA hairpins. The classification is done on blockade signals produced by molecules when occluding ion flow through a nanometer-scale channel (for 9TA vs (GC types of molecular signals, with feature vector extraction as described in the Methods). Implementations: WH SMO (W); Platt SMO (P); Keerthi1 (1); and Keerthi2 (2). Kernels: Absdiff (a); Entropic (e); and Gaussian (g).

The best algorithm/kernel in Suppl. Fig. 1, and in other channel blockade data studied, has consistently been the WH SMO variant and the Absdiff and Entropic Kernels. Another benefit of the WH SMO variant is its significant speedup over the other methods (about half the time of Platt SMO and one fourth the time of Keerthi 1 or 2). The alpha handling and other modifications in WH SMO [6] relate to boundary support vector (BSV) handling (associated with handling on outliers), which is also critical to enhancements to a multiclass SVM solution described in the Background.

Given any metric space (χ, d) one can build a positive-definite kernel of the form $e^{-\lambda d^2}$. Conversely, any positive definite kernel with such form must have a 'd' that is a metric (this is Mercer's condition in another form). This suggests that the 'simplest' distance-based kernel is the Gaussian kernel, since the 'simplest' distance, the Euclidean distance, is used. Likewise, this suggests that the simplest divergence-based kernel would be the aforementioned entropic kernel.

The use of probability vectors, and L_1-norm feature vectors in general (often in conjunction with the entropic kernel), turns out to provide a very general formulation, wherein feature extraction makes use of signal decomposition into a complete set of separable states that can be interpreted or represented as a probability vector (or normalized collection of such, or concatenation, then normalization, etc.). A probability vector formulation also provides a straightforward hand-off to the SVM classifiers since all feature vectors have the same length with such an approach. What this means for the SVM, however, is that geometric notions of distance are no longer the best measure for comparing feature vectors. For probability vectors (i.e., discrete distributions), the best measures of similarity are the various information-theoretic divergences: Kullback–Leibler, Renyi, etc. By symmetrizing over the arguments of those divergences, the entropic kernels are obtained, where the (symmetrized) Kullbach-Leibler Diveregence [34] is used in the entropic kernel in [1, 6] and in the Results.

S.2 Experimental Data and FSA Acquisition

S.2.1 Nanopore Detector Experiments [6,16]

Each experiment is conducted using one alpha-hemolysin channel inserted into a diphytanoyl-phosphatidylcholine/hexadecane bilayer across a, typically, 20-micron-diameter horizontal Teflon aperture. The alpha-hemolysin pore has a 2.0 nm width vestibule opening allowing a dsDNA molecule to be captured (while a ssDNA molecule translocates). The effective diameter of the bilayer ranges mainly between 1-25 μm. This value has some fluctuation depending on the condition of the aperture, which nanopore station is used, and the bilayer applied on a day to day basis. Seventy microliter chambers on either side of the bilayer contain 1.0 M KCl buffered at pH 8.0 (10 mM HEPES/KOH) except in the case of buffer experiments where the salt concentration, pH, or identity may be varied. Voltage is applied across the bilayer between Ag-AgCl electrodes. DNA control probes are typically added to the *cis* chamber at 10-20 nM final concentration. All experiments are maintained at room temperature (23 ± 0.1 °C), using a Peltier device.

S.2.2 NTD control probes

The five DNA hairpins studied in [1, 6, 16] have been carefully characterized, so are used as highly sensitive controls (obtained from IDT DNA with PAGE purification). The nine base-pair hairpin molecules share an eight base-pair hairpin core sequence, with addition of one of the four permutations of Watson-Crick base-pairs that may exist at the blunt end terminus, i.e., 5'-G|C-3', 5'-C|G-3',

5'-T|A-3', and 5'-A|T-3'. Denoted 9GC, 9CG, 9TA, and 9AT, respectively. The full sequence for the 9GC hairpin is 5'-GTTCGAACGTT TTCGTTCGAAC-3'. The eight base-pair DNA hairpin (8GC) is identical to the core eight base-pair part of the 9GC sequence, except the terminal base-pair is changed to be 5'-G|C-3' (e.g., 5'-GTCGAACGTT TTCGTTCGAC-3'). Each hairpin was designed to adopt one base-paired structure.

S.2.3 Data acquisition and FSA-based Signal acquisition [16]

Data is acquired and processed in two ways depending on the experimental objectives: (i) using commercial software from Axon Instruments (Redwood City, CA) to acquire data, where current was typically filtered at 50 kHz bandwidth using an analog low pass Bessel filter and recorded at 20 μs intervals using an Axopatch 200B amplifier (Axon Instruments, Foster City, CA) coupled to an Axon Digidata 1200 digitizer. Applied potential was 120 mV (*trans* side positive) unless otherwise noted. In some experiments, semi-automated analysis of transition level blockades, current, and duration were performed using Clampex (Axon Instruments, Foster City, CA). (ii) using LabView based experimental automation. In this case, ionic current was also acquired using an Axopatch 200B patch clamp amplifier (Axon Instruments, Foster City, CA), but it was then recorded using a NI-MIO-16E-4 National Instruments data acquisition card (National Instruments, Austin TX). In the LabView format, data was low-pass filtered by the amplifier unit at 50 kHz, and recorded at 20 μs intervals. Signal acquisition from the 20 μs sample stream was done using a Finite State Automaton (FSA) [6, 16].

S.3 HMM-based Feature Extraction

With completion of FSA preprocessing, an HMM is used to remove noise from the acquired signals, and to extract features from them. The HMM in one configuration (for control probe validation) is implemented with fifty states, corresponding to current blockades in 1% increments ranging from 20% residual current to 69% residual current [6]. The HMM states, numbered 0 to 49, corresponded to the 50 different current blockade levels in the sequences that are processed. The state emission parameters of the HMM are initially set so that the state j, 0 <= j <= 49 corresponding to level L = j+20, can emit all possible levels, with the probability distribution over emitted levels set to a discretized Gaussian with mean L and unit variance. All transitions between states are possible, and initially are equally likely. Each blockade signature is de-noised by 5 rounds of Expectation- Maximization (EM) training on the parameters of the HMM. After the EM iterations, 150 parameters are extracted from the HMM. The 150 feature vectors obtained from the 50- state HMM-EM/Viterbi implementation are: the 50 dwell percentage in the different blockade levels (from the Viterbi trace-back states), the 50 variances of the emission probability distributions associated with the different states, and the 50 merged transition probabilities from the primary

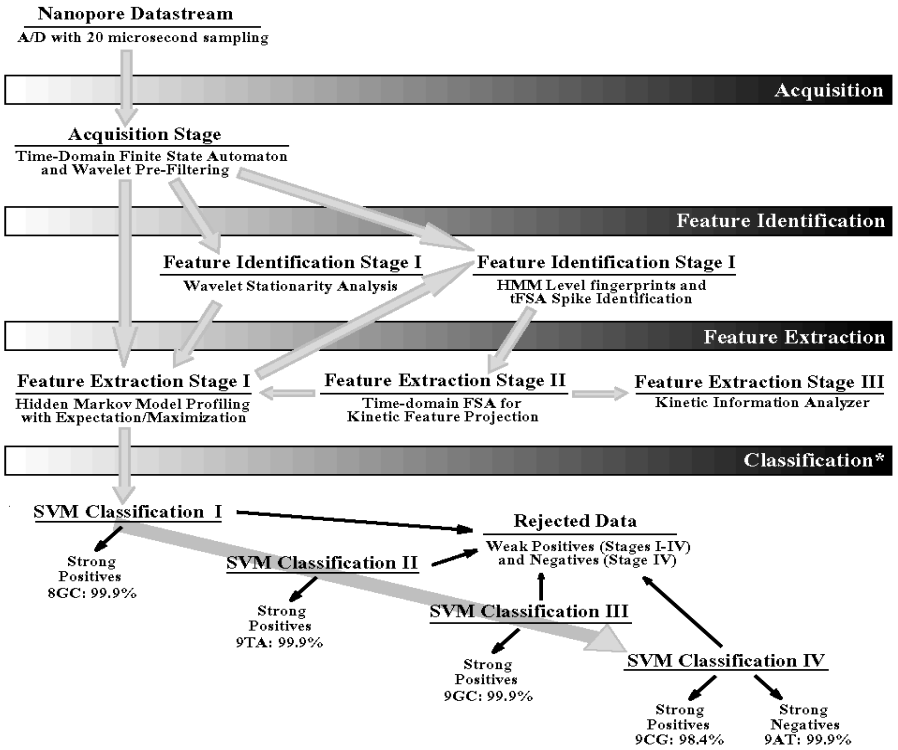
and secondary blockade occupation levels (fits to two-state dominant modulatory blockade signals). Variations on the HMM 50 state implementation are made as necessary to encompass the signal classes under study.

S.4 Data-rejection heuristics

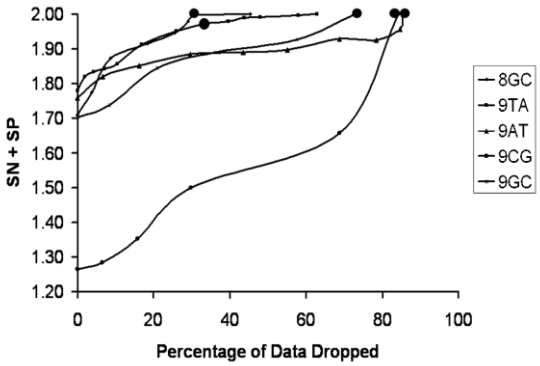
The SVM Decision Tree shown in Suppl. Fig. 2 obtained nearly perfect sensitivity and specificity, with a high data rejection rate, and a highly non-uniform class signal-calling throughput. In Suppl. Fig. 3, the Percentage Data Rejection vs SN+SP curves are shown for test data classification runs with a binary classifier with one molecule (the positive, given by label) versus the rest (the negative). Since the signal calling wasn't passed through a Decision Tree, the way these curves were generated, they don't accurately reflect total throughput, and they don't benefit from the "shielding" shown in the Decision Tree in Suppl. Fig. 2 prototype. In the SVM Decision Tree implementation described in Suppl. Fig. 2 [16], this is managed more comprehensively, to arrive at a five-way signal-calling throughput at the furthest node of 16% (in Suppl. Fig. 2, 9CG and 9AT have to pass to the furthest node to be classified), while the best throughput, for signal calling on the 8GC molecules, is 75%.

The SVM Decision Tree classifier's high, non-uniform, rejection can be managed by generalizing to a collection of Decision Trees (with different species at the furthest node). The problem is that tuning and optimizing a single decision tree is already a large task, even for five species (as in [16]). With a collection of trees, this problem is seemingly compounded, but can actually be lessened in some ways in that now each individual tree need not be so well-tuned/optimized. Although more complicated to implement than an SVM-External method, the SVM-Internal multiclass methods are not similarly fraught with tuning/optimization complications.

Suppl. Fig. 4 shows the Percentage Data Rejection vs SN+SP curves on the same train/test data splits as used for Suppl. Fig. 3, except now the drop curves are to be understood as *simultaneous* curves (not sequential application of such curves as in Suppl. Fig. 3). Thus, comparable, or better, performance is obtained with the multiclass-internal approach and with far less effort, since there is no managing and tuning of Decision Trees. Another surprising, and even stronger argument for the SVM-Internal approach to the problem, for many situations, is that a natural drop zone is indicated by the margin.



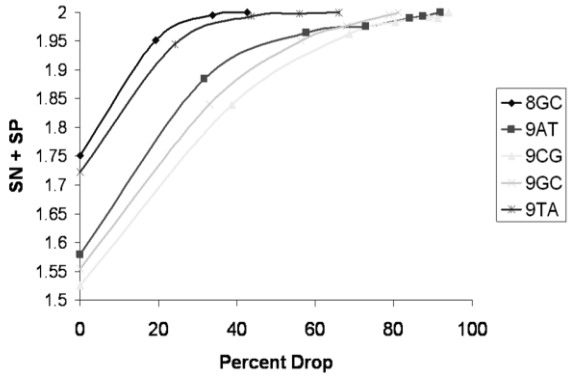
Suppl. Fig. 2. Nanopore Detector signal analysis architecture, with use of an SVM Decision Tree for classification.



Suppl. Fig. 3. The Percentage Data Rejection vs SN+SP curves for test data classification runs with a binary classifier with one molecule (the positive, given by label) versus the rest (the negative). Since the signal calling wasn't passed through a Decision Tree, it doesn't accurately reflect total throughput, and they don't benefit from the "shielding" shown in the Decision Tree in Suppl. Fig. 2 prototype. The Relative Entropy Kernel is shown because it provided the best results (over Gaussian and Absdiff).

Suppose we define the criteria for dropping weak data as the margin: For any data point x_i; let $\max_m\{f_m(x_i)\} = f_{yi}$, and Let $f_m = \max_m\{f_m(x_i)\}$ for all $m \neq yi$, then we

define the margin as: (f_{yi} - f_m), hence data point x_i is dropped if (f_{yi} - f_m) ≤ Confidence Parameter. (For this data set using Gaussian, AbsDiff & Sentropic kernel, a confidence parameter of at least (0.00001)*C was required to achieve 100% accuracy.) Using the margin drop approach, there is even less tuning, and there is improved throughput (approximately 75% for *all* species) [16].



Suppl. Fig. 4. The Percentage Data Rejection vs SN+SP curves are shown for test data classification runs with a *multiclass* discriminator. The following criterion is used for dropping weak data: for any data point x_i; if $\max_m\{f_m(x_i)\}$ ≤ Confidence Parameter, then the data point x_i is dropped. For this data set using AbsDiff kernel (σ^2 = 0.2) performed best, and a confidence parameter of 0.8 achieve 100% accuracy.

Received: May 21, 2017; Published: July 18, 2017