Unsupervised clustering using supervised support vector machines

Stephen Winters-Hilt[1,2,*] and Sepehr Merat[1]

[1]Department of Computer Science, University of New Orleans, New Orleans, LA 70148, USA

[2]Research Institute for Children, Children's Hospital, New Orleans, LA 70118, USA

* Corresponding Author

Email addresses:

SWH *: winters@cs.uno.edu; swinters@chnola-research.org

SM: smerat@math.uno.edu

Abstract

Background

The goal of clustering analysis is to partition objects into groups, such that members of each group are more “similar” to each other than the members of other groups. In this paper we describe methods for using (supervised) SVM classifiers to perform unsupervised clustering and partially supervised clustering (projection). The former is accomplished by iteratively changing some of the label information and redoing the SVM training, the latter has no label alteration, but requires a set specified as the positives (a bag model, so partially supervised).

Results

The Results focus on refinements to the SVM-based clustering methods introduced in [1,2]. We demonstrate that it is possible to stabilize the SVM-external clustering method using simulated annealing.

Conclusions

The highest standard of performance for the SVM-based external clustering method would be that it do as well at cluster grouping as its SVM classification counterpart (that trains with the true label information). We occasionally see such excellent levels of performance with SVM-external clustering, and the goal is to be able to single out these cases when selecting clustering solutions via some stabilization method. Here we apply the simulated annealing method and it appears to solve the stability problems encountered in [2], at least on the data sets studied, to reliably yield excellent clustering solutions.

Introduction

In this paper we describe methods for using SVM classifiers to perform unsupervised clustering and partially supervised clustering (projection). The former is accomplished by iteratively changing some of the label information and re-clustering, the latter has no label alteration, but requires a set to specify as the positives (a bag model, so partially supervised). We describe improvements to the SVM-based clustering methods introduced in [1,2], most notable being stabilization of the SVM-external clustering method using simulated annealing. The SVM-based clustering process we use involves no explicit use of an objective function. An objective function may be thought of as being introduced implicitly, perhaps, in the choice of cluster validator, SVM kernel, and SVM cutoff parameters. Even so, such an introduction of objective function is evidently weakly linked in that several validators, kernels, parameters have been used and all perform comparable to each other when optimized with the simulated-annealing learning stabilization. Given the significant strengths seen with many of the solutions of the SVM-external clustering method described in 1,2], and summarized in the Background, a stabilization to always use these best solutions would provide a robust clustering method.

In what follows we provide a brief background on various matters relevant to the design of the experiment (as will be shown in he Methods and Results): (1) the theory of clustering; (2) the theory of classification; (3) the generalization to kernel feature spaces; (4) kernel construction using polarization; (5) supervised learning; and (6) unsupervised learning. In the unsupervised learning background a brief synopsis of prior work with SVM-external clustering is provided, as this is particularly relevant to what we show in the Results.

The theory of clustering

The goal of clustering analysis is to partition objects into groups, such that members of each group are more “similar” to each other than the members of other groups. Similarity, however, is determined subjectively as it does not have a universally agreeable upon definition. In [3] the author suggests a formal perspective on the difficulty in finding such a unification, in the form of an impossibility theorem: for a set of three simple properties described in [3], there is no clustering function satisfying all three. Furthermore, the author demonstrates that relaxations of these properties expose some of the interesting (and unavoidable) trade-offs at work in well-studied clustering techniques such as single-linkage, sum-of-pairs, k-means, and k-median.

Ideally, one would like to solve the clustering problem given all the known and unknown objective functions. This is provably an NP-Hard problem [4]. This brings us to the work presented here, which seeks to provide a new perspective on clustering by introducing an algorithm that does not require an objective function. Hence, it does not inherit the limitations of an embedded objective function. We propose an algorithm that is capable of suggesting solutions that can be later evaluated using a variety of cluster validators.

The theory of classification

The problem of classification is to find a general rule to match a set of objects, or

observations, to their appropriate classes. In its simplest form the classifier's task is to estimate a function $\boldsymbol{f} : \boldsymbol{R}^N \to \{\pm 1\}$, given examples and their $\{\pm 1\}$ classifications:

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n) \in \boldsymbol{R}^N \times Y,\ Y = \{\pm 1\},$$

where $(\boldsymbol{x}, y)$ are assumed to be independent and identically distributed training data drawn from (unknown) probability distribution $\boldsymbol{P}(\boldsymbol{x}, y)$. $\boldsymbol{f}$ perfectly classifies if y = +1 when $\boldsymbol{f}(\boldsymbol{x}) = 0$, with y = -1 otherwise, where this holds for all of the $\boldsymbol{n}$ training instances.

In the loss-function formalism, the optimal $\boldsymbol{f}$ is obtained by minimizing the ***expected risk*** function (expected error) [5]:

$$R[\boldsymbol{f}] = \int \boldsymbol{l}(\boldsymbol{f}(\boldsymbol{x}), y)\mathrm{d}\boldsymbol{P}(\boldsymbol{x}, y)$$

where $\boldsymbol{l}$ is a suitable loss function. For instance, in the case of' "0/1 loss"

$$\boldsymbol{l}(\boldsymbol{f}(\boldsymbol{x}), y) = \Theta(-y\boldsymbol{f}(\boldsymbol{x}))$$

where Θ is the Heaviside function ($\Theta(z) = 0$ for $z < 0$ and $\Theta(z) = 1$ otherwise.) In most realistic cases $\boldsymbol{P}(\boldsymbol{x}, y)$ is unknown and therefore the risk function above cannot be used to find the optimum function $\boldsymbol{f}$. To overcome this fundamental limitation one has to use the information hidden in the limited training examples and the properties of the function class $\boldsymbol{F}$ to approximate this function. Hence, instead of minimizing the expected risk, one minimizes the ***empirical risk***

$$R_{\mathrm{emp}}[\boldsymbol{f}] = 1/\mathrm{n} \sum \boldsymbol{l}(\boldsymbol{f}(\boldsymbol{x}_\mathbf{i}), y), \text{ with sum on } \mathrm{i} \in 1..\mathrm{n}.$$

The learning machine can ensure that for $\mathrm{n} \to \infty$ the empirical risk will asymptotically converge to expected risk, but for a small training set the deviations are often large. This leads to a phenomenon called "over-fitting," where a small generalization error can't be obtained by simply minimizing the training error. One way to avoid the over-fitting dilemma is to restrict the complexity of the function class [6]. The intuition, which will be formalized in the following, is that a "simple" (e.g., linear) function that explains most of the data is preferable to a complex one (i.e., an application of Occam's razor). This is often introduced via a regularization term that limits the complexity of the function class used by the learning machine [7].

A specific way of controlling the complexity of a function class is described by the Vapnik-Chervonenkis (VC) theory and the structural risk minimization (SRM) principle [6,8]. Here the concept of complexity is captured by the VC dimension **h** of the function class $\boldsymbol{F}$ from which the estimate $\boldsymbol{f}$ is chosen. The following set of definitions indicate the role of structural risk minimization (SRM) -- in the SVM construction that follows SRM is implemented via maximum margin clustering.

Definition 1 (Shattering) A Learning Machine $\boldsymbol{f}$ can shatter a set of points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_\mathrm{n}$ if and only if for every possible training set of the form $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\mathrm{n}, y_\mathrm{n})$ there exists some parameter set that gets zero training error.

Definition 2 (VC Dimension) Given a learning machine $\boldsymbol{f}$, the VC-dimension **h** is the maximum number of points that can be arranged so that $\boldsymbol{f}$ shatter them. Roughly speaking, the VC dimension measures how many (training) points can be shattered (i.e.,

separated) for all possible labelings using functions of the class. Constructing a nested family of function classes $\boldsymbol{F}_1 \subset \ldots \subset \boldsymbol{F}_k$ with non-decreasing VC dimension the SRM principle proceeds as follows:

Definition 3 (SRM Principle) Let $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_k$ be the solutions of the empirical risk minimization in the function classes $\boldsymbol{F}_i$. SRM chooses the function class $\boldsymbol{F}_i$ (and the function $\boldsymbol{f}_i$) such that an upper bound on the generalization error is minimized which can be computed making use of theorems such as the following one.

Theorem 4 (Expected Risk Upper bound) Let **h** denote the VC dimension of the function class $\boldsymbol{F}$ and let R_{emp} be defined by using the "0/1 loss." For all delta > 0 and $\boldsymbol{f}$ Î $\boldsymbol{F}$ the inequality bounding the risk

$$R[\boldsymbol{f}] = R_{emp}[\boldsymbol{f}] + [(\mathbf{h}/n)(ln\ (2n/\mathbf{h}) + 1) - (1/n)\ ln\ (\delta/4)]^{1/2}$$

holds with probability of at least 1−δ for n > **h** ([6,8]). Note: this bound is only an example and similar formulations are available for other loss functions [8] and other complexity measures, e.g., entropy numbers [9].

Thus, in the effort to minimize the generalization error $R[\boldsymbol{f}]$ two extremes can arise: i) a very small function class (like $\boldsymbol{F}_1$) yields a vanishing square root term, but a large training error might remain, while ii) a huge function class (like $\boldsymbol{F}_k$) may give a vanishing empirical error but a large square root term. The best class is usually in between, as one would like to obtain a function that explains the data quite well and to have a small risk in obtaining that function. This is very much in analogy to the bias-variance dilemma scenario described for neural networks (see, e.g., [10]).

What these bounds universally indicate is that the minimized generalization error is bounded by a balance between training error and size of function class (i.e., structural risk). The standard SVM formulation (described below) directly implements such an optimization problem by balancing such terms using a Lagrangian formalism. Before proceeding to the SVM derivation, however, a brief description of kernel spaces is introduced as they are critical (and part of the novelty) in the description of the SVM Lagrangian formulations that follow, and in the Results.

Generalization to kernel feature spaces

The so-called curse of dimensionality from statistics says that the difficulty of an estimation problem increases drastically with the dimension N of the space, since in principle as N increases, the number of required patterns to sample grows exponentially. This statement may cast doubts on using higher dimensional feature vectors as input to learning machines. This must be balanced with results from statistical learning theory [6], however, that show that the likelihood of data separability by linear learning machines is proportional to their dimensionality.

Thus, instead of working in the $\boldsymbol{R}^N$, one can design algorithms to work in feature space, $\boldsymbol{F}$, where the data has much higher dimension (but with sufficiently small function class). This can be described via the following mapping

$$\Phi : \boldsymbol{R}^N \to \boldsymbol{F}\ ;\ \boldsymbol{x} \to \Phi(\boldsymbol{x}).$$

Consider the prior training description with data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \boldsymbol{R}^N$ is mapped into a potentially much higher dimensional feature space $\boldsymbol{F}$. For a given learning algorithm one now considers the same algorithm in $\boldsymbol{F}$ instead of $\boldsymbol{R}^N$. Hence, the learning machine works with the following:

$$(\mathrm{F}(\boldsymbol{x}_1), y_1), \ldots, (\mathrm{F}(\boldsymbol{x}_n), y_n) \in \boldsymbol{F} \times Y,\ Y = \{\pm 1\}$$

It is important to note that this mapping is also implicitly done for (one hidden layer) neural networks, radial basis networks [11] and boosting algorithms [12] where the input data is mapped to some representation given by the hidden layer, the radial basis function (RBF) bumps or the hypotheses space, respectively.

As mentioned above, the dimensionality of the data does not detract us from finding a good solution, but it is rather the complexity of the function class $\boldsymbol{F}$ that contributes the most to the complexity of the problem. Similarly, in practice, one need never know the mapping function F. Therefore, the complexity and intractability of computing the actual mapping is also irrelevant to the complexity of the problem of classification. To this end, algorithms are transformed to take advantage of this aspect of the method, which we call the 'Kernel Trick'.

Kernel Trick Definition: Achieve the following two objectives when using kernels with a learning machine:

1) Rewrite the learning algorithm so that instead of using $\mathrm{F}(\boldsymbol{x}_1)$ and $\mathrm{F}(\boldsymbol{x}_2)$ directly, it only uses the dot-product $\mathrm{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{F}(\boldsymbol{x}_1) \cdot \mathrm{F}(\boldsymbol{x}_2)$, and

2) Compute the dot-products $\mathrm{K}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in a manner that avoids computing $\mathrm{F}(\boldsymbol{x}_1)$ and $\mathrm{F}(\boldsymbol{x}_2)$ explicitly.

The Kernel Trick in definition will be used in the SVM derivation that follows. Note: The Kernel Trick is only possible if the key equations in the learning machine involving the training data $\boldsymbol{x}$ are grouped via an inner products, $\boldsymbol{x}_1 \cdot \boldsymbol{x}_2$, such that the occurrences of F are, similarly, always paired in inner product terms. (Then, in turn, inner products on F are replaced with kernel expressions). Even though we don't need to compute F for a given choice of kernel, we still need to know that such a F exists for this to be a standard kernel generalization to the formalism. This is accomplished using Mercer's Condition, which describes the test to determine if there exists a mapping F for the indicated kernel:

Mercer's Condition: For a given function, K, there exists a mapping F and an expansion:

$$\boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_i \mathrm{F}(\boldsymbol{x}_1)_i \mathrm{F}(\boldsymbol{x}_2)_i$$

if and only if, for any function $g(\boldsymbol{x})$ such that $\int g(\boldsymbol{x})^2 d\boldsymbol{x}$ is finite, then

$$\int\int \boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{x}_2) g(\boldsymbol{x}_1) g(\boldsymbol{x}_2) d\boldsymbol{x}_1 d\boldsymbol{x}_2 = 0,$$

i.e., the kernel is positive definite [5].

Thus, the kernel generalization is broadly applicable to kernels that are positive definite. Note: there is a further subtlety upon implementation -- a kernel may not be positive definite but may have a dense subset of finite training-set kernel matrices that are positive

definite. We have found that the entropic kernel described in what follows is positive definite for all the training sets examined -- where the positive definite property was assumed to hold if the kernel training matrix successfully passed a million g(x) function (Mercer) tests. Whether the entropic kernel is fully analytically positive definite on its restricted domain is unresolved at this time (see [1]).

Kernel construction using polarization

The kernels used in the analysis are based on a family of previously developed kernels [1,13], here referred to as 'Occam's Razor', or 'Razor' kernels. As will be seen, the Gaussian kernel is included in the family of Razor kernels. All of the Razor kernels examined perform strongly on the channel current data analyzed, with some regularly outperforming the Gaussian Kernel itself. 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 feature vector is a discrete probability vector.

Given any metric space ($\mathbf{C}$, d) one can build a positive-definite kernel of the form $e^{-\lambda d^2}$. Conversely, any positive definite kernel with form $e^{-\lambda d^2}$ must have a 'd' that is a metric (this is Mercer's condition in another form). This suggests that the 'simplest' kernel is the Gaussian kernel, since the 'simplest' distance, the Euclidean distance, is used. Functional variations on the Gaussian kernel are described in what follows (see [1] for further details), including variations that are no-longer represented as distances (non-metric), but that operate on a constrained domain (so not clear if Mercer's condition is violated). In what follows a quick synopsis is given of the novel kernels that are used – two of these kernels regularly outperformed the Gaussian kernel (and all other kernels), as will be shown in the following.

Occam's Razor Kernels

The regularized distance kernels are based on the notion of distance. An example of such is the Gaussian kernel (with Euclidian distance). In what follows, we reduce the Gaussian kernel via a log operation and examine the partials of the log kernel – a "polarization" split in the partial derivatives is clearly exhibited:

$$K_{Gaussian}(\vec{x},\vec{y}) = \exp(-\sum_i (x_i - y_i)^2 \Big/ 2s^2)$$

$$\ln K_G(\vec{x},\vec{y}) = -\sum_i (x_i - y_i)^2 \Big/ 2s^2$$

$$\left.\begin{aligned} \frac{\partial \ln K_G(\vec{x},\vec{y})}{\partial x_i} &= -[(x_i - y_i)/s^2] \\ \frac{\partial \ln K_G(\vec{x},\vec{y})}{\partial y_i} &= +[(x_i - y_i)/s^2] \end{aligned}\right\} \quad \text{"polarization" (symmetric)}$$

In what follows, suppose we attempt to modify the form of the kernel at the level of its (symmetric) polarization. The polarization has two attributes, sign and magnitude. This suggests first reducing to the more primitive form of the sign attribute alone (e.g., all have unit magnitude). So, instead of discrimination governed solely by the sign of $(x_i - y_i)$, we use just the "sign" of $(x_i - y_i)$, i.e., the signum function :

Suppose $\left.\begin{aligned} \frac{\partial \ln K(\vec{x},\vec{y})}{\partial x_i} &= -\frac{1}{s^2}\operatorname{sgn}(x_i - y_i) \\ \frac{\partial \ln K(\vec{x},\vec{y})}{\partial y_i} &= +\frac{1}{s^2}\operatorname{sgn}(x_i - y_i) \end{aligned}\right\} \quad \begin{aligned} \frac{\partial |x_i - y_i|}{\partial x_i} &= \operatorname{sgn}(x_i - y_i) \\ \frac{\partial |x_i - y_i|}{\partial y_i} &= \operatorname{sgn}(y_i - x_i) = -\frac{\partial |x_i - y_i|}{\partial x_i} \end{aligned}$

Then recover: $\ln K = -\frac{1}{s^2}\sum_i |x_i - y_i|$, which is another well-known kernel, the Laplace kernel: $K_{Laplace}(\vec{x},\vec{y}) = K_L(\vec{x},\vec{y}) = \exp(-\sum_i |x_i - y_i| \big/ s^2)$. This is the "Occam's razor" kernel for feature discrimination that assumes only a difference-based (geometric) *topology* (nearness but no magnitudes) on the features. The Occam's razor that assumes a difference-based *geometry* on the features (so now not just sign, but magnitude), could have the Gaussian polarization, as a reasonable representative and, thus, arrive back to the Gaussian kernel. Alternatively, another 'simple' formulation would be to seek a polarization term that has an integrable factor that does the opposite of increasing the magnitude of polarization with feature vector difference: boosting the polarization on feature vectors with smaller differences instead. These different cases are described in the following:

$$\frac{\partial \ln K_G(\vec{x},\vec{y})}{\partial x_i} = \underline{(y_i - x_i)\big/ s^2} \;\Rightarrow\; K_G(\vec{x},\vec{y}) = \exp(-\sum_i (x_i - y_i)^2 \big/ 2s^2)$$

$$\Downarrow \text{ sign alone used for polarization of discrimination}$$

$$\operatorname{sgn}(y_i - x_i)\big/ s^2 \;\Rightarrow\; K_L(\vec{x},\vec{y}) = \exp(-\sum_i |x_i - y_i| \big/ s^2)$$

$$\Downarrow \text{ sign with integrable boost factor for polarization of discrimination}$$

$$\frac{\operatorname{sgn}(y_i - x_i)\big/ s^2}{\sqrt{\sum_i |y_i - x_i|}} \;\Rightarrow\; K_I(\vec{x},\vec{y}) = \exp(-2\sqrt{\sum_i |x_i - y_i|} \big/ s^2),$$

where K_I is the 'Indicator' or 'Absdiff' kernel [1,13].

We use a similar basis for construction of an entirely different family of kernel functions (based on an exponentially regularized information divergence). Instead of difference polarization, with (x–y) less than or greater than zero, there is also ratio polarization, with x/y less than or greater than 1, with the unit value subtracted or functionally mapped to zero: $(y_i / x_i - 1)$ does the former and $\ln(y_i / x_i)$ does the latter. Together they are found to be directly integrable:

$$\frac{\partial \ln K_G(\vec{x},\vec{y})}{\partial x_i} = (y_i - x_i)/s^2$$

$\Downarrow$ integrable ratio polarization instead of Gaussian difference

$$[(\frac{y_i}{x_i}) - 1 + \ln(\frac{y_i}{x_i})]\Big/ s^2 \Rightarrow K_E(\vec{x},\vec{y}) = \exp(-\frac{1}{s^2}\sum_i (x_i - y_i)\ln(\frac{x_i}{y_i}))$$

The entropic kernel is so named because of its alternate form $K_{entropic} = \exp(-[D(\vec{y} \parallel \vec{x}) + D(\vec{x} \parallel \vec{y})]/2s^2)$, where D(x||y) is the Kullback-Leibler divergence, otherwise known as relative entropy. Thus, the Occam's razor kernel appropriate to divergence-based differences on an information theoretic space is the entropic kernel (parallels the Gaussian kernel in geometric space). In all of the dataruns with the probability feature vector data considered here, the two best-performing kernels are the entropic and the Indicator 'Adbsdiff' kernels, with the Gaussian trailing noticeably in performance (but still outperforming other methods such as polynomial and dot product).

Supervised Learning

An initial, simple, supervised learning scenario is to assume that the training data is separable by a hyperplane, expressed via a hyperplane separability function, or learning machine:

$$f(\boldsymbol{x}) = (w \bullet \boldsymbol{x}) - b.$$

In [6] it is shown that the for this class of hyperplanes the VC dimension can be bounded in terms of another quantity, the margin. The margin is defined as the minimal distance of a sample to the decision surface (see section 2.5.2). It is rather intuitive that the thicker the margin the better the training would be, and in fact this quantity could be measured by the length of the weight vector w. Since we begin by assuming that the training data is separable we can rescale and normalize w such that the points closest to the hyperplane is a unit away from the hyperplane (i.e., the canonical representation of the hyperplane). This can be done by requiring that $|(w \bullet \boldsymbol{x}) - b)| = 1$. Now consider two samples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from different classes with $(w \bullet \boldsymbol{x}_1) - b = 1$ and $(w \bullet \boldsymbol{x}_2) - b = -1$, respectively. The margin is given by the distance of these two points, measured perpendicular to the hyperplane and therefore, $w/\|w\|^2 \bullet (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 2/\|w\|^2$ (see Fig. 1). This will be used in a Lagrangian optimization in what follows to yield the standard binary SVM.

Binary Support Vector Machines

The binary SVM is the supervised learning method that is co-opted to perform the unsupervised SVM-based clustering we describe in the Results and Methods. In the binary SVM implementation described in what follows we follow the notation and conventions used previously [1]. Feature vectors are denoted by x_{ik}, where index i labels the feature vectors ($1 = i = M$) and index k labels the N feature vector components ($1 = k = N$). For the binary SVM, labeling of training data is done using label variable $y_i = \pm 1$ (with sign according to whether the training instance was from the positive or negative

class). For hyperplane separability, elements of the training set must satisfy the following conditions: $w_ß x_{iß} - b = +1$ for i such that $y_i = +1$, and $w_ß x_{iß} - b = -1$ for $y_i = -1$, for some values of the coefficients $w_1,..., w_N$, and b (**using the convention of implied sum on repeated Greek indices**). This can be written more concisely as: $y_i(w_ß x_{iß} - b) - 1 = 0$. Data points that satisfy the equality in the above are known as "support vectors" (or "active constraints").

Once training is complete, discrimination is based solely on position relative to the discriminating hyperplane: $w_ß x_{iß} - b = 0$. The boundary hyperplanes on the two classes of data are separated by a distance 2/w, known as the "margin," where $w^2 = w_ß w_ß$. By increasing the margin between the separated data as much as possible the optimal separating hyperplane is obtained. In the usual SVM formulation, the goal to maximize w^{-1} is restated as the goal to minimize w^2. The Lagrangian variational formulation then selects an optimum defined at a saddle point of

$$L(\mathrm{w},\mathrm{b};a) = \frac{\mathrm{w}_b \mathrm{w}_b}{2} - a_g y_g (\mathrm{w}_b \mathrm{w}_{gb} - \mathrm{b}) - a_0 \ ,$$

$$\text{where } a_0 = \sum_g a_g \ , \ a_g \geq 0 \quad (1 \leq g \leq M)$$

The saddle point is obtained by minimizing with respect to $\{w_1,...,w_N,b\}$ and maximizing with respect to $\{a_1, ..., a_M\}$. If $y_i(w_ß x_{iß} - b) - 1 = 0$, then maximization on a_i is achieved for $a_i = 0$. If $y_i(w_ß x_{iß} - b) - 1 = 0$, then there is no constraint on a_i. If $y_i(w_ß x_{iß} - b) - 1 < 0$, there is a constraint violation, and a_i ? 8 . If absolute separability is possible, the last case will eventually be eliminated for all a_i, otherwise it is natural to limit the size of a_i by some constant upper bound, i.e., $\max(a_i) = C$, for all i. This is equivalent to another set of inequality constraints with $a_i = C$. Introducing sets of Lagrange multipliers, ?? (see Fig. 1) and $\mu_?$ (1 = ? = M), to achieve this, the Lagrangian becomes:

$$L(\mathrm{w},\mathrm{b};a,x,m) = \frac{\mathrm{w}_b \mathrm{w}_b}{2} - a_g [y_g (\mathrm{w}_b \mathrm{x}_{gb} - \mathrm{b}) + x_g] + a_0 + x_0 C - m_g x_g$$

$$\text{where } x_0 = \sum_g x_g \ , \ \ a_0 = \sum_g a_g \text{ and } a_g \geq 0 \text{ and } x_x \geq 0 \quad (1 \leq g \leq M)$$

At the variational minimum on the $\{w_1,...,w_N,b\}$ variables, $w_ß = a_? y_? x_{?ß}$, and the Lagrangian simplifies to:

$$L(a) = a_0 - \frac{a_d \mathrm{y}_d \mathrm{x}_{db} a_g \mathrm{y}_g \mathrm{x}_{gb}}{2},$$

$$\text{with } 0 \leq a_g \leq C \ \ (1 \leq g \leq M) \text{ and } a_g y_g = 0 ,$$

where only the variations that maximize in terms of the $a_?$ remain (known as the Wolfe Transformation). In this form the computational task can be greatly simplified. By introducing an expression for the discriminating hyperplane: $f_i = w_ß x_{iß} - b = a_? y_? x_{?ß} x_{iß} - b$, the variational solution for L(a) reduces to the following set of relations (known as the Karush-Kuhn-Tucker, or KKT, relations):

(i) $a_i = 0$, $y_i f_i = 1$

(ii) $0 < a_i < C$, $y_i f_i = 1$

(iii) $a_i = C$, $y_i f_i = 1$

When the KKT relations are satisfied for all of the $a_?$ (with $a_? y_? = 0$ maintained) the solution is achieved. The constraint $a_? y_? = 0$ is satisfied for the initial choice of multipliers by setting the a's associated with the positive training instances to 1/N(+) and the a's associated with the negatives to 1/N(-), where N(+) is the number of positives and N(-) is the number of negatives. Once the Wolfe transformation is performed it is apparent that the training data (support vectors in particular, KKT class (ii) above) enter into the Lagrangian solely via the inner product $x_{i\ß} x_{j\ß}$. Likewise, the discriminator f_i, and KKT relations, are also dependent on the data solely via the $x_{i\ß} x_{j\ß}$ inner product. Throughout the rest of this section we consider: $x_{i\beta} x_{j\beta}$ ® $F(x_i)_\beta F(x_j)_\beta = K_{ij}$.

The SVM discriminators are trained by solving their KKT relations using the SMO procedure of [14]. The method described here follows the description of [14] and begins by selecting a pair of Lagrange multipliers, $\{a_1, a_2\}$, where at least one of the multipliers has a violation of its associated KKT relations. For simplicity it is assumed in what follows that the multipliers selected are those associated with the first and second feature vectors: $\{x_1, x_2\}$. The SMO procedure then "freezes" variations in all but the two selected Lagrange multipliers, permitting much of the computation to be circumvented by use of analytical reductions:

$$L(a_1, a_2; a_{b\,'\geq 3}) = a_1 + a_2 - \frac{(a_1^2 K_{11} + a_2^2 K_{22} + 2a_1 a_2 y_1 y_2 K_{12})}{2} - a_1 y_1 v_1 - a_2 y_2 v_2 + a_{b'} U_{b'} - \frac{a_{b'} a_{y'} y_{b'} K_{b'y'}}{2} ,$$

with ß',?' = 3, and where $K_{ij} = K(x_i, x_j)$, and $v_i = a_{ß'} y_{ß'} K_{iß'}$ with ß' = 3. Due to the constraint $a_ß y_ß = 0$, we have the relation: $a_1 + s a_2 = -?$, where $? = y_1 a_{ß'} y_{ß'}$ with ß' = 3 and $s = y_1 y_2$. Substituting the constraint to eliminate references to a_1, and performing the variation on a_2: $?L(a_2; a_{ß'} = 3)/?a_2 = (1 - s) + ?a_2 + s?(K_{11} - K_{22}) + s y_1 v_1 - y_2 v_2$, where $? = (2K_{12} - K_{11} - K_{22})$. Since v_i can be rewritten as $v_i = w_ß x_{iß} - a_1 y_1 K_{i1} - a_2 y_2 K_{i2}$, the variational maximum $?L(a_2; a_{ß'} = 3)/?a2 = 0$ leads to the following update rule:

$$a_2^{new} = a_2^{old} - \frac{y_2((w_b x_{1b} - y_1) - (w_b x_{2b} - y_2))}{h}$$

Once a_2^{new} is obtained, the constraint $a_2^{new} = C$ must be re-verified in conjunction with the $a_ß y_ß = 0$ constraint. If the $L(a_2; a_{ß'} = 3)$ maximization leads to a a_2 new that grows too large, the new a_2 must be "clipped" to the maximum value satisfying the constraints. For example, if $y_1 ? y_2$, then increases in a_2 are matched by increases in a_1. So, depending on whether a_2 or a_1 is nearer its maximum of C, we have $\max(a_2) = \text{argmin}\{a_2 + (C - a_2) ; a_2 + (C - a_1)\}$. Similar arguments provide the following boundary conditions:
(i) if s = -1, $\max(a_2) = \text{argmin}\{a_2 ; C + a_2 - a_1\}$, and $\min(a_2) = \text{argmax}\{0 ; a_2 - a_1\}$, and
(ii) if s = +1, $\max(a_2) = \text{argmin}\{C ; a_2 + a_1\}$, and $\min(a_2) = \text{argmax}\{0 ; a_2 + a_1 - C\}$.
In terms of the new $a_2^{new, clipped}$, clipped as indicated above if necessary, the new a_1 becomes:

$$a_1^{new} = a_1^{old} + s(a_2^{old} - a_2^{new,clipped}) ,$$

where $s = y_1y_2$ as before. After the new a_1 and a_2 values are obtained there still remains the task of obtaining the new b value. If the new a_1 is not "clipped" then the update must satisfy the non-boundary KKT relation: $y_1f(x_1) = 1$, i.e., $f^{new}(x_1) - y_1 = 0$. By relating f^{new} to f^{old} the following update on b is obtained:

$$b_1^{new} = b - (f^{new}(x_1) - y_1) - y_1(a_1^{new} - a_1^{old})K_{11} - y_2(a_2^{new,clipped} - a_2^{old})K_{12}$$

If a_1 is clipped but a_2 is not, the above argument holds for the a_2 multiplier and the new b is:

$$b_2^{new} = b - (f^{new}(x_2) - y_2) - y_2(a_2^{new} - a_2^{old})K_{22} - y_1(a_1^{new,clipped} - a_1^{old})K_{12}$$

If both a_1 and a_2 values are clipped then we don't have a unique solution for b. The Platt convention was to take:

$$b^{new} = \frac{b_1^{new} + b_2^{new}}{2}$$

and this works well much of the time. Alternatively, Keerthi [15] has devised an alternate formulation without this weakness, as have Crammer and Singer [16]. Perhaps just as good as any exact solution for 'b' in the double-clipped scenario is to manage this special case by rejecting the update and picking a new pair of alphas to update (in this way only unique 'b' updates are made). Alpha-selection variants are briefly discussed in the Section after next.

Chunking

Chunking is described in [17] and [18], where use is made of sparsity and efficient testing of the KKT conditions. Depending on the problem, many of the a's will either be zero or C. If we could easily identify $a = 0$, the corresponding calculation could be avoided without changing the value of the quadratic form. Then, at every chunking step we could solve the problem by reducing to the $a \neq 0$ plus the KKT violating a's. The size of this problem varies but is finally equal to the number of non-zero a's [19]. While this technique is suitable for fairly large problems it is still limited by the maximal number of support vectors that it can handle. This is because for every chunk a quadratic optimizer is still necessary. In [20] we describe a new method for distributed SVM learning that eliminates many of these limitations on learning set size.

Binary classification scoring conventions

In this paper we adopt the following conventions:

SN = TP/(TP + FN)
SP = TP/(TP + FP)
nSN = TN/(TN + FP)
nSP = TN/(TN + FN)

Bioinformatics researchers, in gene prediction for example [21], take as primary pair: {SN, SP}; ROC people, or people using a Confusion Matrix diagram, take as primary pair {SN, nSN}; Purity/Entropy researchers use {SP, nSP}; no one uses the pair {nSN, nSP} since it is a trivial label flip from being {SN, SP}. Label flipping leaves the sensitivity {SN, nSN} pair the same, similarly for the specificity pair {SP, nSP}. In other

work we use the conventions that are commonly employed in gene prediction, and other instances where the signal identification is biased towards identification of positives. Specifically, they have two sets they focus on, the actual positives (genes) and the predicted positives (gene predictions). For gene prediction there is either a gene, or there is no-gene (i.e., junk, or background noise). In situations where your objective is to make the sets of actual positives (AP) and predicted positives (PP) maximally overlap, SP=specificity and SN=sensitivity are natural parameters and can be nicely described with a prediction accuracy Venn diagram (similar to a confusion matrix). For the problem here, we also describe the purity/entropy measures, in some examples, to be in-line with other efforts in the clustering research community. Thus, we work with the pair {SP,nSP} as well as {SP,SN}.

Unsupervised Learning (Clustering)

K-means is a simple algorithm for clustering a set of un-labeled feature vectors $\boldsymbol{X}$: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ that are drawn independently from the mixture density $p(\boldsymbol{X}|q)$ with a parameter set q. At the heart of K-means algorithm is optimization of the sum-of-squared-error criterion function (SSE), $\mathbf{J_i}$ defined in definition SSE below.

Definition SSE (Sum-of-squared-error): Given a cluster c_i, the sum-of-squared, $\mathbf{J_i}$ is defined by

$$\mathbf{J_i} = \sum_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{m_i}\|^2 , \boldsymbol{x} \hat{I} c_i ,$$

and where $\boldsymbol{m_i}$ is the mean of the samples belonging to c_i. The geometric interpretation of this criterion function is that for a given cluster c_i the mean vector $\boldsymbol{m_i}$ is the centroid of the cluster by minimizing the length of the vector $\boldsymbol{x} - \boldsymbol{m_i}$. This can be shown by taking the variation of $\mathbf{J_i}$ with respect to the "centroid" $\boldsymbol{m_i}$ and setting it zero,

$$\partial/\partial \boldsymbol{m_i} \mathbf{J_i} = \partial/\partial \boldsymbol{m_i} \sum_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{m_i})\mathrm{T} (\boldsymbol{x} - \boldsymbol{m_i}) = 2\sum_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{m_i}) = 0$$

with minimum when equal to zero, where $\boldsymbol{x} \hat{I} c_i$ in the sums, and solving for $\boldsymbol{m_i}$:

$$\boldsymbol{m_i} = 1/n_i \sum_{\boldsymbol{x}} \boldsymbol{x}$$

where $n_i = |c_i|$ is the number of feature vectors belonging to c_i. The total SSE for all of the clusters, $\mathbf{J_e}$ is the sum of SSE for individual clusters. The value of $\mathbf{J_e}$ depends on the cluster membership of the data, (i.e. the shape of the clusters), and the number of clusters. The optimal clustering is the one that minimizes $\mathbf{J_e}$ for a given number of clusters, k, and K-means tries to do just that.

Kernel K-means [22] is a natural extension of K-means. Denote $\mathbf{M_{in}}$ to be the cluster assignment variables such that $\mathbf{M_{in}} = 1$ if and only if $\boldsymbol{x_i}$ belongs to cluster n and 0 otherwise. As for K-means, the goal is to minimize the $\mathbf{J}_n$ for all clusters, n in feature space, by trying to find k means $F(\boldsymbol{m}_n)$ such that each observation in the data set when mapped using F is close to at least one of the means. Since the means lie in the span of $F(\boldsymbol{x_1}), \ldots, F(\boldsymbol{x_n})$, we can write them as:

$$\mu_v \equiv F(\boldsymbol{m}_n) = \sum_{\mathbf{j}} g_{vj} F(\boldsymbol{x_j})$$

We can then substitute this in the $\mathbf{J_i}$ of definition (10) to obtain,

$$\mathbf{J}_n = \sum_{\chi} \|F(\boldsymbol{x}) - \boldsymbol{\mu}_n\|^2 = \sum_{\chi} \|F(\boldsymbol{x}) - \sum_{\mathbf{j}} g_{vj} F(\boldsymbol{x}_{\mathbf{j}})\|^2$$
$$= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) - 2 \sum_{\mathbf{j}} g_{vj} \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}_{\mathbf{j}}) + \sum_{\mathbf{ij}} g_{vi} g_{vj} \boldsymbol{K}(\boldsymbol{x}_{\mathbf{i}}, \boldsymbol{x}_{\mathbf{j}}),$$

where $\boldsymbol{x} \hat{I} c_{\mathbf{i}}$ in $\sum_{\chi}$, $j \hat{I} 1..n$ in $\sum_{\mathbf{j}}$, and $i,j \hat{I} 1..n$ in $\sum_{\mathbf{ij}}$. We initially assign random feature vectors to means. Then Kernel K-means proceeds iteratively as follows: each new remaining feature vectors, $\boldsymbol{x}_{t+1}$, is assigned to the closest mean μ_a:

$$M_{t+1,a} = 1 \text{ if for all } n \neq a, \|F(\boldsymbol{x}_{t+1}) - \mu_a\|^2 < \|F(\boldsymbol{x}_{t+1}) - \mu_n\|^2,$$

$M_{t+1,a} = 0$ otherwise. Or, in terms of the kernel function,

$$M_{t+1,a} = 1 \text{ if for all } n \neq a, \sum_{\mathbf{ij}} g_{ai} g_{aj} \boldsymbol{K}_{\mathbf{ij}} - \sum_{\mathbf{j}} g_{aj} \boldsymbol{K}_{\mathbf{t+1,j}} < \sum_{\mathbf{ij}} g_{vi} g_{vj} \boldsymbol{K}_{\mathbf{ij}} - \sum_{\mathbf{j}} g_{vj} \boldsymbol{K}_{\mathbf{t+1,j}},$$

$M_{t+1,a} = 0$ otherwise, where $\boldsymbol{K}_{\mathbf{i,j}} \equiv \boldsymbol{K}_{\mathbf{ij}} \equiv \boldsymbol{K}(\boldsymbol{x}_{\mathbf{i}}, \boldsymbol{x}_{\mathbf{j}})$. The update rule for the mean vector is then given by,

$$\mu_{t+1,a} = \mu_{t,a} + D(F(\boldsymbol{x}_{t+1}) - \mu_{t,a}),$$

where,

$$D \equiv M_{t+1,a} / \sum_{1..t+1} M_{ia}$$

SVM-Internal Clustering

The SVM-Internal approach to clustering was originally defined by [1]. Data points are mapped by means of a kernel to a high dimensional feature space where we search for the minimal enclosing sphere. In what follows, Keerthi's method [15] is used to solve the dual (see implementation).

The minimal enclosing sphere, when mapped back into the data space, can separate into several components; each enclosing a separate cluster of points. The width of the kernel (say Gaussian) controls the scale at which the data is probed while the soft margin constant helps to handle outliers and over-lapping clusters. The structure of a data set is explored by varying these two parameters, maintaining a minimal number of support vectors to assure smooth cluster boundaries.

The SVM-internal Lagrangian formulation is presented in what follows, where we adopt the notation used by [23]. The SVM-internal results are shown in the comparative study described in the Results.

SVM-Internal Lagrangian Formulation

Let $\{x_i\}$ be a data set of N points in R^d (*Input Space*.) Similar to the nonlinear SVM formulation, using a non-linear transformation f, we transform x to a high-dimensional space – *Kernel space* – and look for the smallest enclosing sphere of radius R. Hence we have:

$$\|f(x_j) - a\|^2 = R^2 \text{ for all } j = 1,...,N$$

where a is the center of the sphere. Soft constraints are incorporated by adding slack variables $?_j$:

$$\|f(x_j) - a\|^2 = R^2 + ?_j \text{ for all } j = 1,...,N$$

Subject to: $?_j = 0$

We formulate the Lagrangian as:

$$L = R^2 - ? _j ß_j (R^2 + ?_j - \|f(x_j) - a\|^2) - ? _j ?_j \mu_j + C? _j ?_j$$

subject to: $ß_j = 0, \mu_j = 0,$

where C is the cost for outliers and therefore $C? _j ?_j$ is the penalty term. Taking the derivative of L w.r.t. R, a and $?$ and setting them to zero we have:

$$? _j ß_j = 1,$$

$$a = ? _j ß_j f(x_j), \text{ and}$$

$$ß_j = C - \mu_j .$$

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

$$W = 1 - ? _{i,j} ß_i ß_j K_{ij} \text{ where } 0 = ß_i = C; K_{ij} = exp(-\|x_i - x_j\|^2 / 2s^2)$$

subject to: $? _i ß_i = 1$

By KKT relations we have:

$$?_j \mu_j = 0 \text{ and } ß_j (R^2 + ?_j - \|f(x_j) - a\|^2) = 0.$$

In the *feature space*, $ß_j = C$ only if $?_j > 0$; hence it lies outside of the sphere i.e. $R^2 < \|f(x_j) - a\|^2$. This point becomes a bounded support vector or BSV. Similarly if $?_j = 0$, and $0 < ß_j < C$, then it lies on the surface of the sphere i.e. $R^2 = \|f(x_j) - a\|^2$. This point becomes a support vector or SV. If $?_j = 0$, and $ß_j = 0$, then $R^2 > \|f(x_j) - a\|^2$ and hence this point is enclosed with-in the sphere.

SVM-External clustering

Although the internal approach to SVM clustering is only weakly biased towards the shape of the clusters in *feature* space (the bias is for spherical clusters in the *kernel* space), it still lacks robustness. In the case of most real-world problems and strongly overlapping clusters, the SVM-Internal Clustering algorithm above can only delineate the relatively small cluster cores. Additionally, the implementation of the formulation is tightly coupled with the initial choice of kernel; hence the static nature of the formulation and implementation does not accommodate numerous kernel tests. One way around this excessive geometric constraint, and others like it, is to use an external-SVM clustering algorithm (introduced in [1]) that clusters data vectors with no a priori knowledge of each vector's class.

The algorithm works by first running a Binary SVM against a data set, with each vector in the set randomly labeled, until the SVM converges. Choice of an appropriate kernel and an acceptable sigma value will affect this convergence. After the initial convergence is achieved, the (sensitivity + specificity) will be low, likely near 1. The algorithm now improves this result by iteratively re-labeling *only* 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), hopefully

nearly reaching 2. This method provides a way to cluster data sets without prior knowledge of the data's clustering characteristics, or the number of clusters. In practice, the initialization step, that arrives at the first SVM convergence, typically takes longer than all subsequent partial re-labeling and SVM rerunning steps.

The SVM-External clustering approach is not biased towards the shape of the clusters, and, unlike the internal approach, the formulation is not fixed to a single kernel class. Nevertheless, there are robustness and consistency issues that arise in the SVM-External clustering approach. To rectify these issues, stabilization methods are described for SVM-external clustering that take into account the robustness required in realistic applications.

Unsupervised Scoring: Cluster Entropy and Purity

Let p_{ij} be the probability that an object in cluster i belongs to class j. Then entropy for the cluster i, e_i, can be written as:

$$e_i = -\sum_j p_{ij} \log p_{ij} \quad \text{, where } j \in 1..J,$$

and J is the number of classes. Similarly, the purity p_i for the cluster i can be expressed as,

$$p_i = \max_j \; p_{ij}$$

Note that the probability that an object in cluster i belongs to class j can be written as the number of objects of class j in cluster i, n_{ij} , divided by the total number of objects in cluster i, n_i, (*i.e.*, $p_{ij} = n_{ij} / n_i$) Using this notation the overall validity, f_V, of a cluster i using the measure f_i (for either entropy or purity) is the weighted sum of that measure over all clusters. Hence,

$$f_V = 1/N \sum_i n_i f_i \quad \text{, where } i \in 1..K,$$

and K is number of clusters and N is the total number of patterns. For our 2-class clustering problem:

$$p_1 = max(SP, 1 - SP), \;\; p_2 = max(nSP, 1 - nSP)$$

and:

$$p_V = max(\; (TP + TN)/(TP + FP + TN + FN); \; 1–(TP + TN)/(TP + FP + TN + FN) \;)$$

Entropy is a more non-localized (holistic) measure than *purity*. Rather than considering either the frequency of patterns that are within a class or the frequency of patterns that are outside of a class, *entropy*, takes into account the entire distribution. Further details on the entropic measure, however, are not discussed here.

Early SVM-external Results

The SVM-external approachl is first used on some simple synthetic data using some simple kernels [1,2]. In Fig. 2 we see correct clustering very quickly with the polynomial kernel (compared with no convergence when using the linear kernel).

In clustering experiments in [2], a challenging data set consisting of 8GC and 9GC DNA hairpin data is examined (part of the data sets used in [13]). The hairpin dataset consists of 400 elements. Half of the elements belong to each class. Although convergence is always achieved, convergence to a *global* optimum is not guaranteed. Figures 3a and 3b show the Purity and Entropy (with the RBF kernel) as a function of Number of Iterations, while Fig 3c shows the SSE as a function of Number of Iterations. Note: the stopping criteria for the algorithm is based on the **unsupervised** measure of SSE (see Methods). Comparison to fuzzy c-means and kernel k-means is shown on the same dataset (the solid blue and black lines in Fig. 3a and 3b). In this early effort it was noted 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.

Thus, the SVM-external clustering method appears to offer very strong solutions about half the time – which allows for simple optimization by repeated clustering attempts (external optimization), or optimization via internal optimization, such as simulated annealing. Comparative results with other clustering methods are shown in Fig. 4 (using one of the most challenging pairs of DNA signals to resolve from analysis done in [1]). In the Results presented in what follows, a simulated annealing formulation of the internal optimization is shown.

Results

The SVM-Relabeler clustering algorithm is capable of clustering a wide range of data sets from simple multi-dimensional data sets (e.g., the Iris data set) to the complex 150-dimensional Nanopore DNA hairpin data. These applications will now be described, along with Results upon introducing simulated annealing stabilization to the clustering process.

Iris Data Set

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Sir Ronald Aylmer Fisher in [24] as an example of discriminant analysis. The data set consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample, they are the length and the width of sepal and petal (see Fig. 5). Based on the combination of the four features, Fisher developed a linear discriminant model to predict classes to which each sample belongs. We partly analyze the Iris data by performing one binary clustering evaluation. (This can be iterated on the binary clusters identified until no further clusters can be resolved to arrive at a multiple cluster resolution method without specification of cluster number, but that will not be discussed further here.) Fig. 6 shows the binary cluster splitting and it precisely identifies one cluster and groups the other two, demonstrating excellent performance.

DNA Hairpin Data Set

In [1, 25-27] it is shown that the alpha-hemolysin nanometer-scale channel can be used to associate ionic current measurements with single-molecule channel blockades. In

particular, the nanopore dimensions are such that they allow for lengthy dsDNA terminus measurements This is because the alpha-Hemolysin entry aperture is 2.6 nm in diameter which is large enough to capture approximately 10 base-pairs of dsDNA (the limiting aperture prevents further passage).

The data set is chosen to be a symmetric sample of 200 8GC blockade signals (i.e., blockade signals due to 8 base-pair DNA hairpins ending in 5'-GC-3') and 200 9GC blockade signals. 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 [13]). Although convergence was easily achieved with the SVM Relabeler algorithm (see Methods), convergence to a global optimum was not guaranteed. Figure 7a & 7b illustrates the characteristic behavior of different possible solutions with the data sets indicated. At the end of a successful run of this algorithm it is hypothesized that the generalization error (testing error) will be very small. In Figure 8 a small value of Kernel-SSE (herein referred to as SSE) is shown to provide us with a reliable cluster validation measure.

The 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 (see Fig. 7). Unsuccessful solutions and local optima solutions are undesirable and the objective is to find a method to eliminate their usage by simply re-clustering for objectively improved clustering (via SSE scoring, for example). Since, the solutions in the unsuccessful set are expected to be easily identified in any experiment that calculates the SSE of a randomly labeled data set, they can be simply eliminated by post-processing. For instance, in a similar 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.

To substantially reduce the local optimum solutions, however, thresholding does not scale well. One solution is a 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 (so far) if it has a lower SSE than the previously recorded value. This can be a very slow learning process, and is a familiar scenario in statistical learning, and one of the popular solution in those situations presents itself here as well – simulated annealing.

Refinement Methods Using 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 could be found. (Note that $p_{pert} = 0.50$ has the effect of random reinitialization and $p_{pert} = 1$ flips the entire

labels.) The hope is that perturbation with $p_{pert} \leq 0.50$ results in a faster convergence. Reliability can be achieved by searching through the solution space. To do this efficiently, Monte Carlo Methods could be used by taking advantage of perturbation to evaluate the neighboring configuration. The procedure described next uses a modified version of Simulated Annealing to achieve this desired reliability.

As shown in Figure 10a, 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 (Figure 10b, 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.

Projection Clustering – clustering in Decision space

SVM methods for clustering are described that are based on the SVM's ability to not only classify, but also to give a confidence parameter on its classifications. Even without modifying the label information (passive clustering), there is often strong clustering information in an SVM training solution. One such instance occurs when one set, the positives, are a known signal species (or collection of species). If you have mixture data with known and unknown signal species, and wish to identify (i.e., cluster) the unknown species, then an SVM training attempt with the mixture taken as negatives leads to a cluster identification method via an SVM "projection-score" histogram. (i.e., cluster partitioning in Decision Space). Real channel blockade data has been examined in this way, biotinylated DNA hairpin blockades comprised the positives, and scored as a sharp peak at around 1.0 (see Fig. 11). The mixture signals seen after introduction of streptavidin cluster with scores around 0.5, corresponding to (unbound) biotinylated DNA hairpin signals, and signals that score < –1.0, corresponding to the streptavidin-bound biotinylated DNA hairpins. SVM projection clustering can be a very powerful clustering tool in and of itself as can be seen in this cheminformatics application.

Discussion

The highest standard of performance for the SVM-based external clustering method would be that it do as well at cluster grouping as its SVM classification counterpart (that trains with the true label information). We occasionally see such excellent levels of performance with SVM-external clustering, and the goal is to be able to single out these cases when selecting clustering solutions via some stabilization method. Here we apply the simulated annealing method and it appears to solve the stability problems encountered in [2], at least on the data sets studied, to reliably yield excellent clustering solutions. If in the general data clustering context and simulated annealing becomes unmanageable, genetic algorithm tuning could also be used. Early tests with SVM tuning using genetic algorithms show strong performance (results not shown).

Cluster Validators

The fitness of the cluster identified by the SVM-Relabeler algorithm is modeled using a supervised measure, purity and an unsupervised measure, Kernel SSE. In other words, we assume that the fitness of a cluster can be tracked using the compactness of that cluster as the algorithm progresses. It is necessary to note that i) the notion of compactness as a

way to evaluate clusters favours spherical clusters over clusters spread over a linear region, and ii) the known classes may not properly correspond to the geometric structure of the clusters. However, this limitation does not normally affect our methodology, since this limitation is imposed over the feature space, and with a choice of proper kernels and parameters, feature space can be more likely to have higher variation in the lifted dimensions such that spherical-cluster representations are accurate.

Fisher Validator

Scaling and transformation requires knowledge about the geometry of the problem at hand. This information is often unavailable to data analysts. SVM-Relabeler can be improved to take advantage of the discriminant function provided by the SVM solution. In Fig. 8 we have shown the histogram of the decision space and demonstrated that the method is effectively separating the clusters. Let m_1, m_2, s_1 and s_2 be the means and standard deviation of the decision space projection shown in Fig. 8. The function,

$$J(\omega) = |m_1 - m_2| / [(s_1)^2 + (s_2)^2]$$

for ω is the derived weight factor, can be used to validate the clusters. Generally, larger $J(\omega)$ corresponds to better separation, while taking into account the compactness of the clusters. Note: $J(\omega)$ is used as the objective function of Fisher's discriminants [22].

Tuning SVM-Relabeler

Kernel selection is highly data set dependant and so far accurate methods for choosing kernels are not automated [28-31] (although initial efforts using genetic algorithm tuning, with the channel current data, appear to be successful). The SVM-Relabeler results produced in this work assume the value of 1.5 for the SVM regularization constant, C, and make use of the Absdiff, Sentropic, and Gaussian kernels. When working with similar data in a standard single-pass SVM classification, the typical settings used are C = 10.0, with a significantly shifted optimum in kernel parameter tuning. The choices of the SVM regularization constant, C, has profoundly different utility when used in the context of SVM-Relabeler. The choice of C in the supervised SVM controls the trade off between the training error and the generalization error, and therefore, it effectively controls the soft margin. Since this margin is not known to us when clustering, the value of C has to be chosen empirically. While we generally find C >= 10 to be good for classification, for SVM *clustering*, tuning to larger values of C tends to reduce performance much sooner than with SVM *classification*. This is because a larger C value reduces the number of misclassifications (boundary support vectors) accessible to the SVM-Relabeler algorithm (see Methods), where a specified (tuned) percentage of boundary support vectors have their labels flipped. It is observed that the percentage of label flipping is stable, however, given a good choice of kernel and parameters, i.e., when everything else is tuned properly, small changes in the relabeling percentage are not seen to alter clustering performance significantly.

Conclusion

In this work, new methods for clustering are explored. The method uses the Support Vector Machine's discriminant learning and confidence evaluation, and does not explicitly depend on an objective function. The solution yielded by this algorithm is evaluated using a criterion function, but it remains external to the algorithm. Simulated is

successfully used to stabilize this process on the datasets studied. The hope is that this work provides a new perspective on unsupervised learning processes.

Method

SVM-Relabeler: An External Method of SVM Clustering

Although the internal approach to SVM (see svm-internal) clustering is only weakly biased towards the shape of the clusters in the input space (the bias is for spherical clusters in the feature space), it still lacks robustness. In the case of most real-world problems and strongly overlapping clusters, the SVM- Internal Clustering algorithm above can only delineate the relatively small cluster cores. Additionally, the implementation of the formulation is tightly coupled with the initial choice of kernel; hence the formulation does not accommodate numerous kernel tests. We avoid this constraint by making use of an external-SVM clustering algorithm (called SVM-Relabeler in [2]). SVM-Relabeler clusters data vectors with no a priori knowledge of each vector's class. SVM-Relabeler algorithm works by first running a Binary SVM against a data set, with each vector in the set randomly labeled, until the SVM converges. The SVM-Relabeler algorithm now improves convergent SVM label result by iteratively re-labeling only 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 further progress can be made.

Unsupervised Cluster Validator

Unlike purity, unsupervised evaluation techniques do not have external class information. These measures are often optimization functions in many clustering algorithms. Sum-of-Squared-Error, SSE, measures the compactness of a single cluster, and other measures evaluate the isolation of a cluster from other clusters. Standard SSE is typically calculated in the input space, which does not necessarily correspond with a good cluster measure in the kernel feature space. Below, we show the unsupervised cluster validator that works in the kernel's feature space instead.

Sum-of-Squared-Error, SSE, in input space, can be written as:

$$J_e = \tfrac{1}{2} \sum_i n_i S_i, \text{ where } i \in 1..K,$$

and where for any similarity function $s(x, x')$

$$S_i = 1/(n_i)^2 \sum_x \sum_{x'} s(x, x'), \text{ where } x, x' \in D_i$$

We use Euclidean distance as the measure of similarity. Hence,

$$s(x, x') = \|x - x'\|^2$$

As before, let $F : D_i \rightarrow F_i$ and $\boldsymbol{K}(x,y) = \{F(x), F(y)\}$, then J_e can be rewritten (this time feature space) as:

$$\boldsymbol{J_e} = \sum_i \boldsymbol{J_i}, \text{ where } i \in 1..K,$$

and

$$\boldsymbol{J_i} = \sum_{x} \boldsymbol{K}(x,x) - 1/n_i \sum_{x} \sum_{x'} k(x,x'), \text{ where } x, x' \in D_i$$

Note that SSE, like any other unsupervised criterion, may not reveal the true underlying clusters, since the Euclidean distance simplification favors spherically shaped clusters. However, this geometry is often imposed after the data is mapped to the higher dimensional feature space, so this measure is often acceptable.

Competing interests

The authors declare that there are no competing interests.

Author's Contribution

SWH outlined the SVM approach. SM implemented the simulated annealing stabilization for SVM clustering. The authors co-wrote the paper.

Acknowledgments

Funding was derived from grants from NIH and the LA Board of Regents. Federal funding was provided by NIH K-22 (SWH PI, 5K22LM008794). State funding was provided from a LaBOR Enhancement (SWH PI). The authors thank Anil Yelundur for data analysis used in comparative analysis plot shown in Fig. 4.

References

1. Winters-Hilt S, Yelundur A, McChesney C, Landry M: Support Vector Machine Implementations for Classification & Clustering. BMC Bioinf. 7 Suppl 2: S4, 2006.
2. Winters-Hilt S and Merat S. SVM Clustering. BMC Bioinf. 8 Suppl 7, S18, 2007.
3. Kleinberg J. An impossibility theorem for clustering. Proc. of the 16th conference on Neural Information Processing Systems, (12), 2002.
4. Eppstein D and Bern M. Approximation algorithms for geometric problems, pages 296–345. Approximation algorithms for NP-hard problems. PWS Publishing Co, 1996.
5. Burgess CJC. A tutorial on support vector machines for pattern recognition. Knowledge Discovery and Data Mining, 2(2):121–167, 1998.
6. V.N.Vapnik. The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1995.
7. Girosi F and Poggio T. Regularization algorithms for learning that there are equivalent to multilayer networks. Science, 247:978–982, 1990.
8. V.N.Vapnik. Statistical Learning Theory. New York: Wiley, 1998.
9. Scholkopf B, Williamson RC, Smola AJ. Regularization algorithms for learning that there are equivalent to multilayer networks. Science, 247:978–982, 1990.
10. Doursat R, Geman S, and Bienenstock E. Neural network and bias/variance dilemma. Neural Computation, 4(2):1–58, 1992.
11. Bishop CM. Neural Networks for Pattern Recognition. Oxford University Press, 1995.
12. Schapire RE and Freund Y. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci., 55(1):119–139, 1997.
13. Winters-Hilt S, Vercoutere W, DeGuzman VS, Deamer DW, Akeson M, and Haussler D. Highly Accurate Classification of Watson-Crick Basepairs on Termini of Single DNA Molecules. Biophys. J. 84:967-976. 2003.
14. Platt JC. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In *Advances in Kernel Methods -- Support Vector Learning.* Edited by Scholkopf B, Burges CJC, and Smola AJ. MIT Press, Cambridge, USA;. 1998: Ch. 12.
15. Keerthi SS, Shevade SK, Bhattacharyya C and Murthy KRK. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation,* Vol. 13, 637-649. 2001.
16. Crammer K and Singer Y. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. Journal of Machine Learning Research 2 pp. 265-292. 2001.
17. V.N.Vapik. Estimation of Dependences Based on Empirical Data. Berlin: Springer-Verlag, 1982.
18. Osuna E, Freund R, and Girosi. F. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing VII*. Edited by Principe J, Gile L, Morgan N, and Wilson E. editors. IEEE, New York. 276-85. 1997.

19. Weston J, Bottou L, Scholkopf B, Smola A, Saunders C, and Stitson MO. Support vector machine reference manual. Technical Report CSD-TR-98-03, Royal Holloway Univ. Tech Rep., 1998.
20. Winters-Hilt S and Armond Jr. K. Distributed SVM Learning and Support Vector Reduction. Submitted to BMC Bioinformatics – preprint available at http://www.cs.uno.edu/~winters.
21. Guigo R and Burset M. Evaluation of gene structure prediction programs. Genomics, 3(34):353–367, 1996.
22. Duda RO, Hart PE and Stork DG, *Pattern classification*, Second Edition, John Wiley and Sons, New York, 2001.
23. Ben-Hur A, Horn D, Siegelmann HT, and Vapnik VN. Support vector clustering. Journal of Machine Learning Research, 2:125–137, 2001.
24. Fisher RA. The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7:179–188, 1936.
25. Vercoutere W, Winters-Hilt S, DeGuzman VS, Deamer D, Ridino S, Rogers JT, Olsen HE, Marziali A, and Akeson M, "Discrimination Among Individual Watson-Crick Base-Pairs at the Termini of Single DNA Hairpin Molecules," *Nucl. Acids Res. Vol.31, 1311-1318,* 2003.
26. Vercoutere W, Winters-Hilt S, Olsen HE, Deamer D, Haussler D, and Akeson M, "Rapid Discrimination Among Individual DNA Molecules at Single Nucleotide Resolution Using an Ion Channel," *Nature Biotechnology*, Vol. 19, pg 248, 2001.
27. Winters-Hilt, S. and M. Akeson, "Nanopore cheminformatics," *DNA and Cell Biology, Vol. 23 (10), Oct. 2004*.
28. Micchelli CA, Pontil M, Argyriou A, Hauser R. A dc-programming algorithm for kernel selection. ACM International Conference Proceeding Series, 148:41–48, 2006.
29. Furesjo F, Smola A, Caputo B, and Sim K. Appearance-based object recognition using svms: which kernel should i use? Whistler: Proc of NIPS workshop on Statitsical methods for computational experiments in visual processing and computer vision, 2002.
30. Takahashi H and Debnath R. Kernel selection for the support vector machine. IEICE Transactions on Information and Systems, E87-D(12):2903–2904, 2004.
31. Boyd S, Kim SJ, Magnani A. Optimal kernel selection in kernel fisher discriminant analysis. ACM International Conference Proceeding Series, 148:465–472, 2006.

Figures

Figure 1. Hyperplane Separability. A general hyperplane is shown with its decision-function feature-space splitting role implicit in the indicated margin width and offset from the origin, also noted is misclassified case for the general formalism.

Figure 2. A test of the SVM-external clustering algorithm, using a simple kernel. The randomized label data set clearly falls into two clear clusters, one central, one at fixed radius, both radially symmetric about the origin. The linear kernel (i.e., the dot-product non-kernel case) fails to converge to a solution. The polynomial kernel does converge, and very quickly, within 3 iterations, as can be seen.

Figure 3. SVM-external clustering results from [2]. (a) and (b) show the boost in Purity and Entropy as a function of Number of Iterations of the Re-labeler 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 on the same dataset.

Figure 4. Clustering performance comparisons: SVM-external clustering compared with explicit objective function clustering methods. In those cases where drop percentages are indicated, tuning was used to drop 'weaker' data. The figure shows the nanopore detector blockade signal clustering resolution where the blockades are due to individual molecular capture-events with 9AT and 9CG DNA hairpin molecules. This was a very difficult data set to resolve with the SVM classifier – even when 'bag-labeled' training data is used (i.e., bag-labeled from experiments run only with one or the other molecule introduced). The label 9AT is used for the control molecule that is nine base-pairs in length and ends with 5'-AT-3' and has a 4dT loop, the same for 9CG, except it has terminal base-pair 5'-CG-3'. The SVM-external clustering method consistently out-performs the other methods (see [1] for further details on this effort).

Figure 5. Fisher's Iris data set. The Iris data set consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample, they are the length and the width of sepal and petal.

Figure 6. Recovery of main binary splitting in three-species Iris data set. The figure shows the labeling recovered after doing one binary clustering operation (on the full dataset). The Gaussian kernel was used with = 2.0, C = 1.5, and $\sigma = 0.5$.

Figure 7. (a) Kernel SSE validation during SVM-external clustering with 8GC and 9GC DNA hairpin blockade data. There are 200 samples of each, where each sample results in a 150-component feature vector that is normalized to 1.0. Plots show Kernel SSE values (Absdiff kernel with $\sigma = 1.8$) as the SVM-external clustering algorithm iterates (post initial convergence).

Figure 7. (b) Purity scoring during operation of Kernel SSE validator. Plots show

Purity values as the SVM-external clustering algorithm iterates (post initial convergence).

Figure 8. (a) The histogram of the projection of feature vectors on a unit vector orthogonal to the decision hyperplane. Features are classified according to the sign of this projection. As can be seen, by iteration 17 the data set has been clearly separated. For these experiments: Absdiff kernel (with $\sigma = 2.0$), C = 1.5, and relabeling factor = 0.15.

Figure 8. (b) The learning plots. The plots show iteration by iteration the view of the experiment shown in Fig. 8a. as can be seen, there is correlated decrease in Kernel SSE value, test error, and training error, and simultaneous increase in the purity.

Figure 9. The distribution of Kernel SSE values over random binary-labeled data. The histogram shows what to expect from random labeling on the training problem shown in Fig. 7, and provides a validator criterion on clustering solutions when they fall well outside the support of the random label SSE values. Again, use is made of the Absdiff kernel with $\sigma = 1.8$. The best-fitted Distribution we could identify was Johnson's SB distribution, and that is the curve that is shown.

Figure 10. (a) Simulated annealing with constant perturbation. The behavior of the simulated annealing algorithm using a constant perturbation functions is shown (the variable case is in the net figure). The top panel shows the learning behavior when constant 10% perturbation is applied at every iteration. The bottom panel shows the annealing process.

Figure 10. (b) Simulated annealing with variable perturbation. The behavior of the simulated annealing algorithm using a variable perturbation functions is shown. The top panel shows the learning behavior when constant 10% perturbation is applied at every iteration along with boosts in the perturbation probability based on the number of iterations during which SSE remained constant (Bottom panel). (The timing of the annealing process is shown in 10a Bottom panel.)

Figure 11. Projection clustering with channel current blockade data. A nanopore detector experiment is performed in which two reactants are introduced: a biotinylated DNA hairpin and streptavidin. Only the biotinylated DNA is likely to be observed at the channel, or the bound streptavidin – biotinylated DNA, due to the negatively charged DNA molecules. The biotinylated DNA hairpin signal by itself is well studied so makes an excellent example of the positives to use in the projection clustering. The negatives consist of the mixture file observations made upon introduction of streptavidin, in excess, to a solution in the analyte chamber that contains only biotinylated DNA hairpins. Bound complexes are hypothesized to form upon introduction of the streptavidin, with a new signal class emerging, as can be seen, along with some of the old signal class. The biotinylated DNA hairpin blockades comprised the positives, and scored as a sharp peak at around 1.0. The mixture signals seen after introduction of streptavidin cluster with scores around 0.5, corresponding to (unbound) biotinylated DNA hairpin signals, and signals that score < -1.0, are hypothesized to correspond to the streptavidin-bound biotinylated DNA hairpins.

Figures

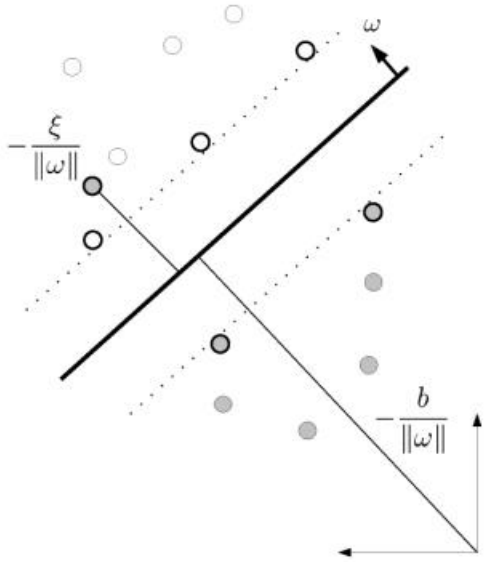


Figure 1.

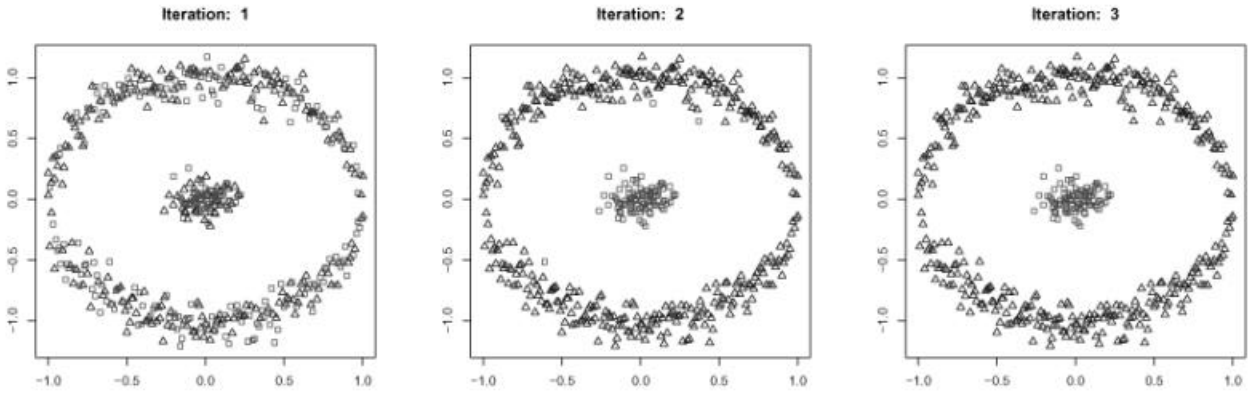
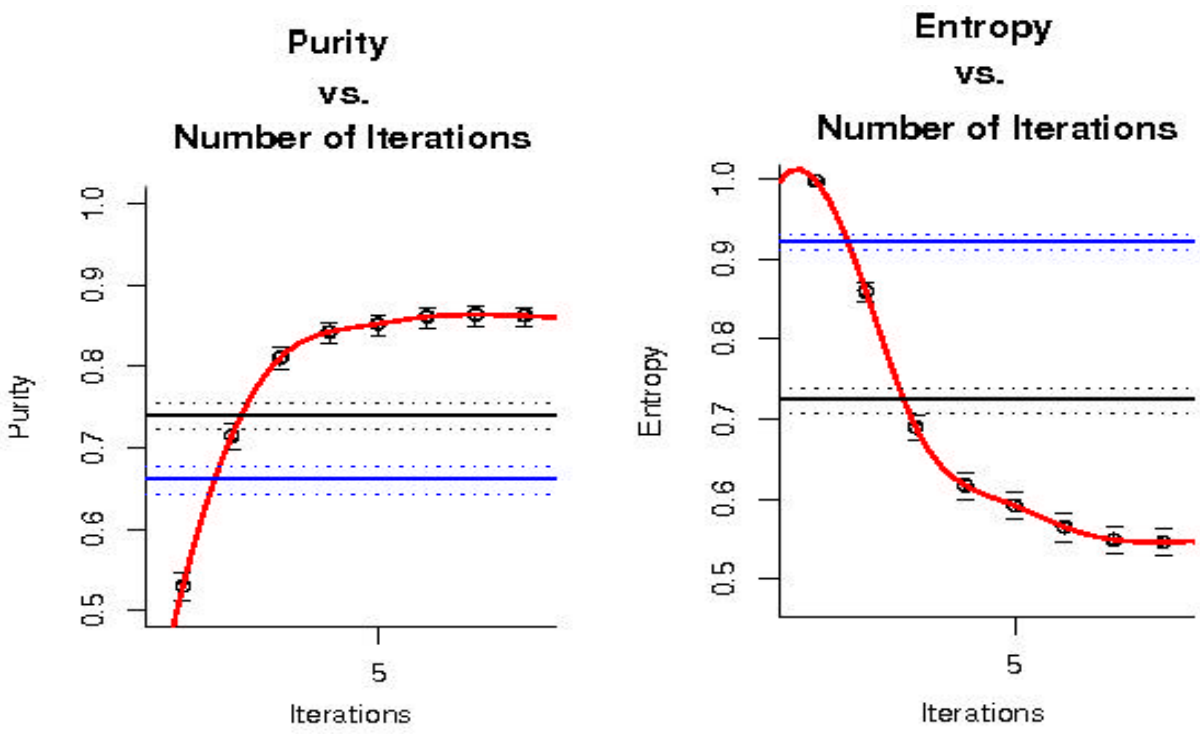
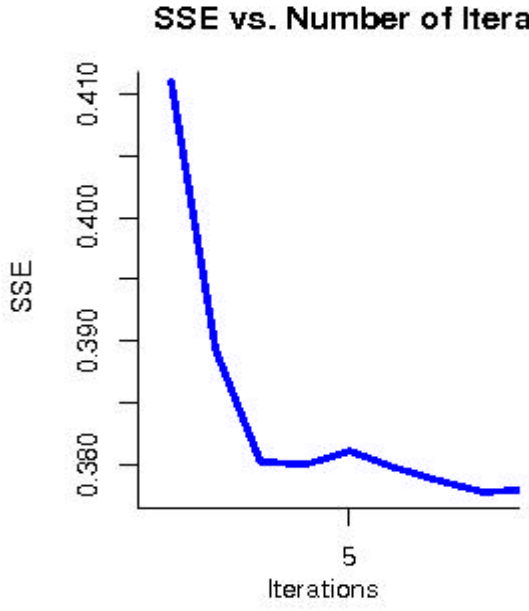


Figure 2.


Purity
vs.
Number of Iterations
Purity
0.5
0.6
0.7
0.8
0.9
1.0
5
Iterations
Entropy
vs.
Number of Iterations
Entropy
0.5
0.6
0.7
0.8
0.9
1.0
5
Iterations

(a) **(b)**


SSE vs. Number of Iterations
SSE
0.380
0.390
0.400
0.410
5
Iterations

(c)

Figure 3.

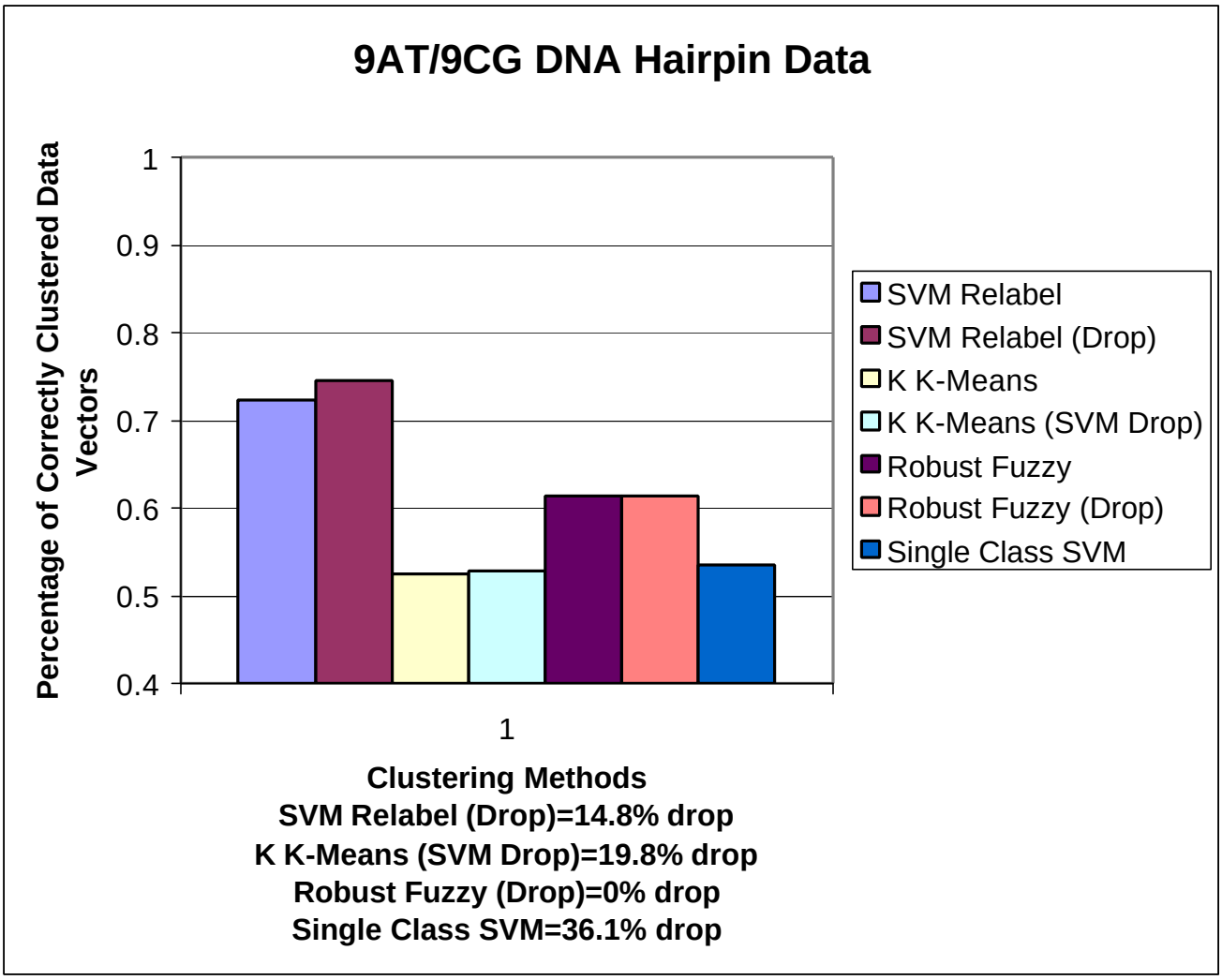

9AT/9CG DNA Hairpin Data
Percentage of Correctly Clustered Data Vectors
1
0.9
0.8
0.7
0.6
0.5
0.4
1
SVM Relabel
SVM Relabel (Drop)
K K-Means
K K-Means (SVM Drop)
Robust Fuzzy
Robust Fuzzy (Drop)
Single Class SVM
Clustering Methods
SVM Relabel (Drop)=14.8% drop
K K-Means (SVM Drop)=19.8% drop
Robust Fuzzy (Drop)=0% drop
Single Class SVM=36.1% drop

Figure 4.

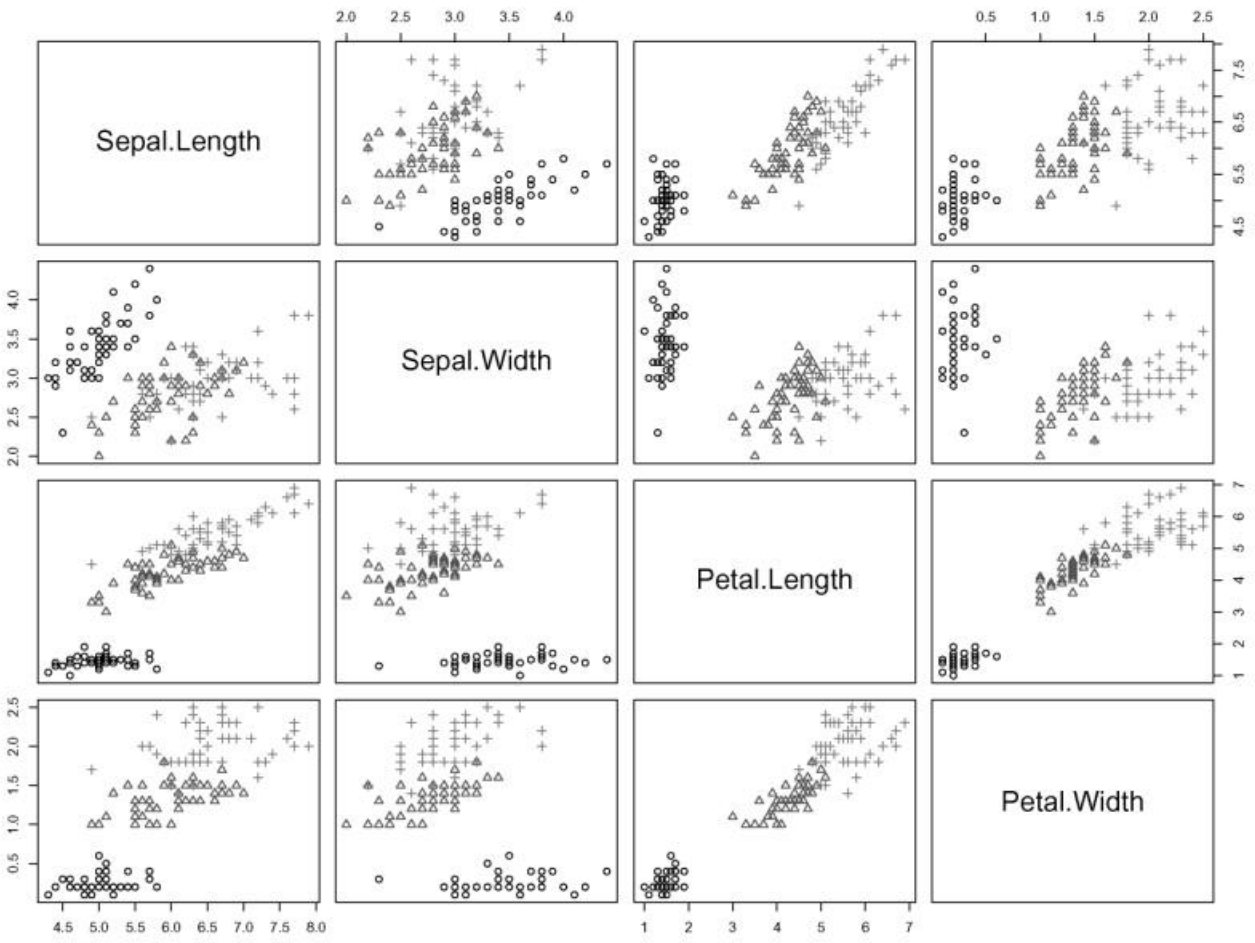


Figure 5.

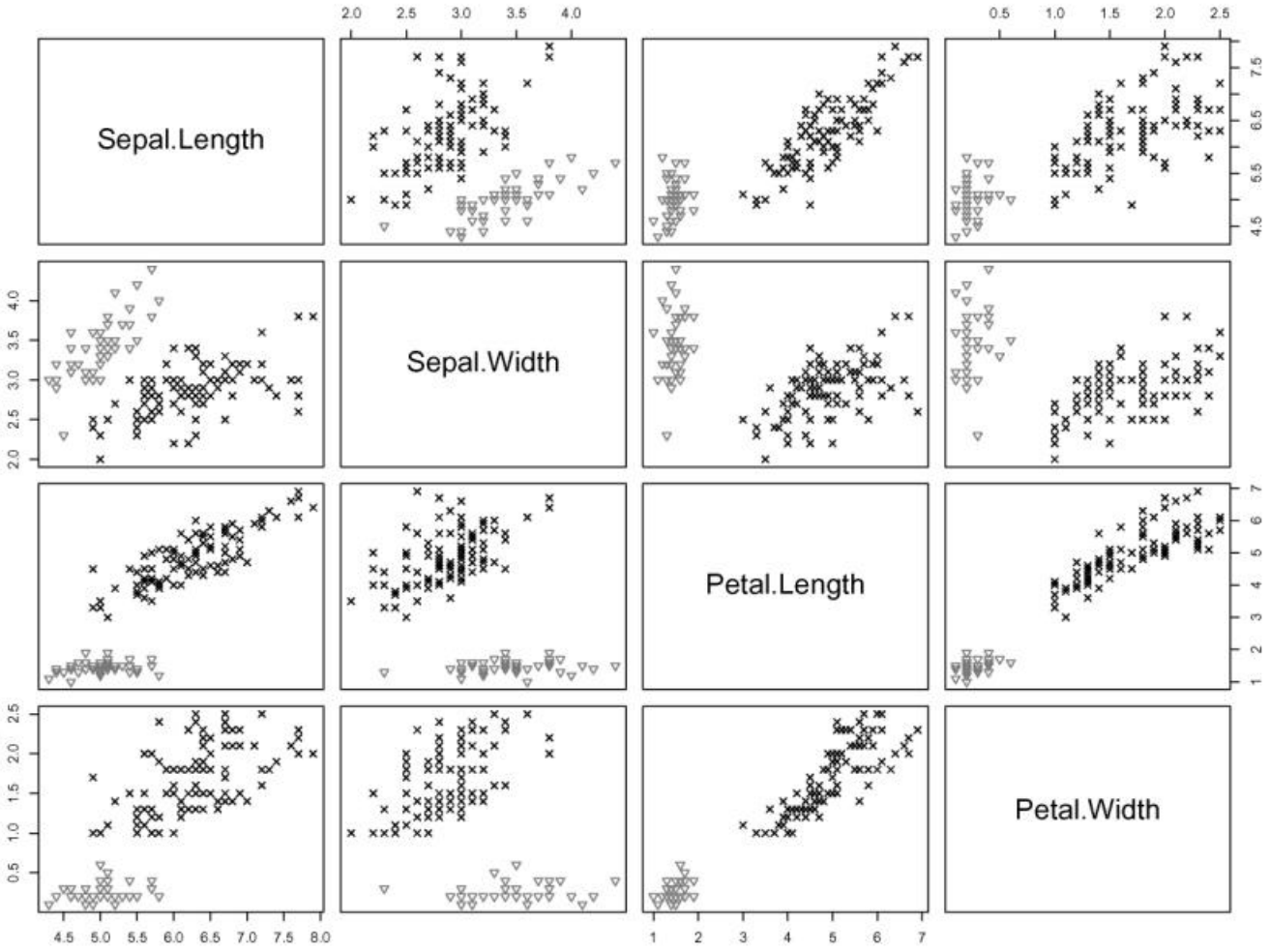


Figure 6.

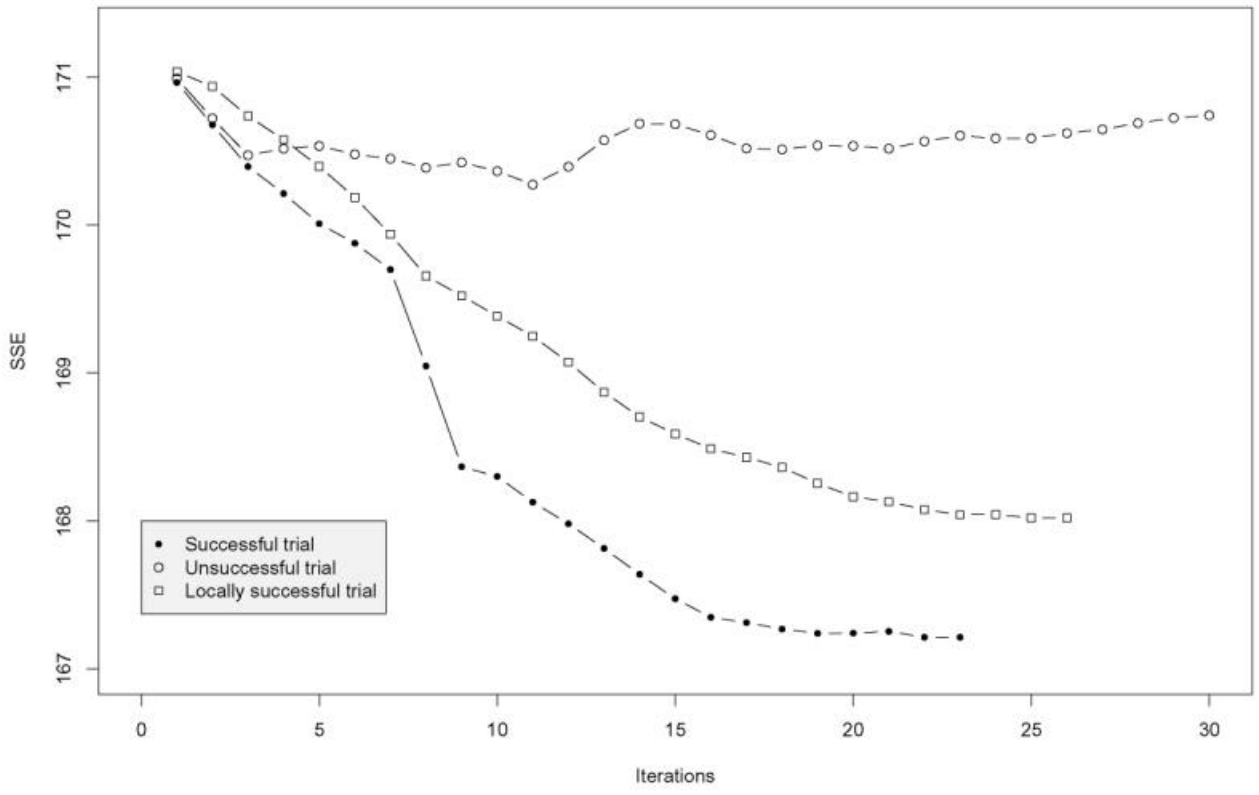

SSE
171
170
169
168
167
0
5
10
15
20
25
30
Iterations
Successful trial
Unsuccessful trial
Locally successful trial

Figure 7a.

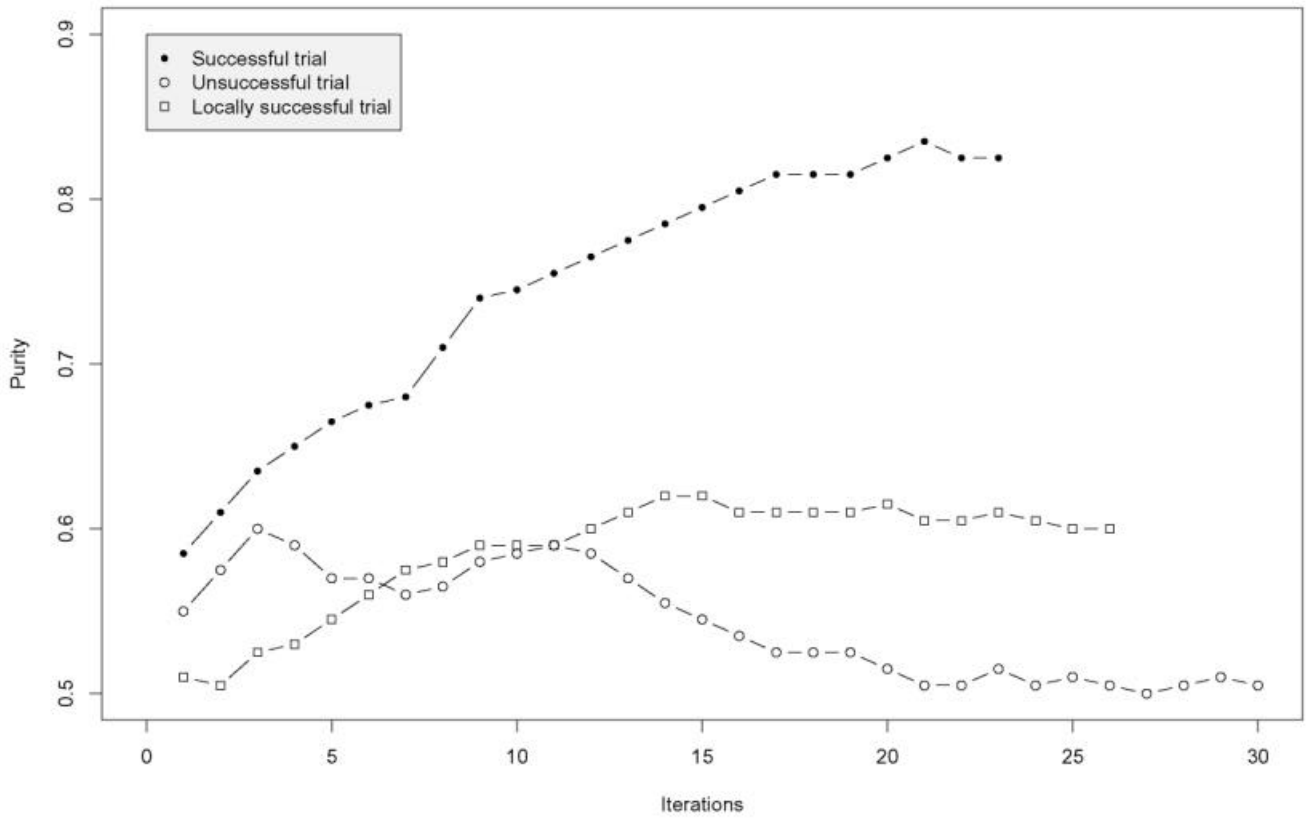

Successful trial
Unsuccessful trial
Locally successful trial
Purity
0.5
0.6
0.7
0.8
0.9
0
5
10
15
20
25
30
Iterations

Figure 7b.

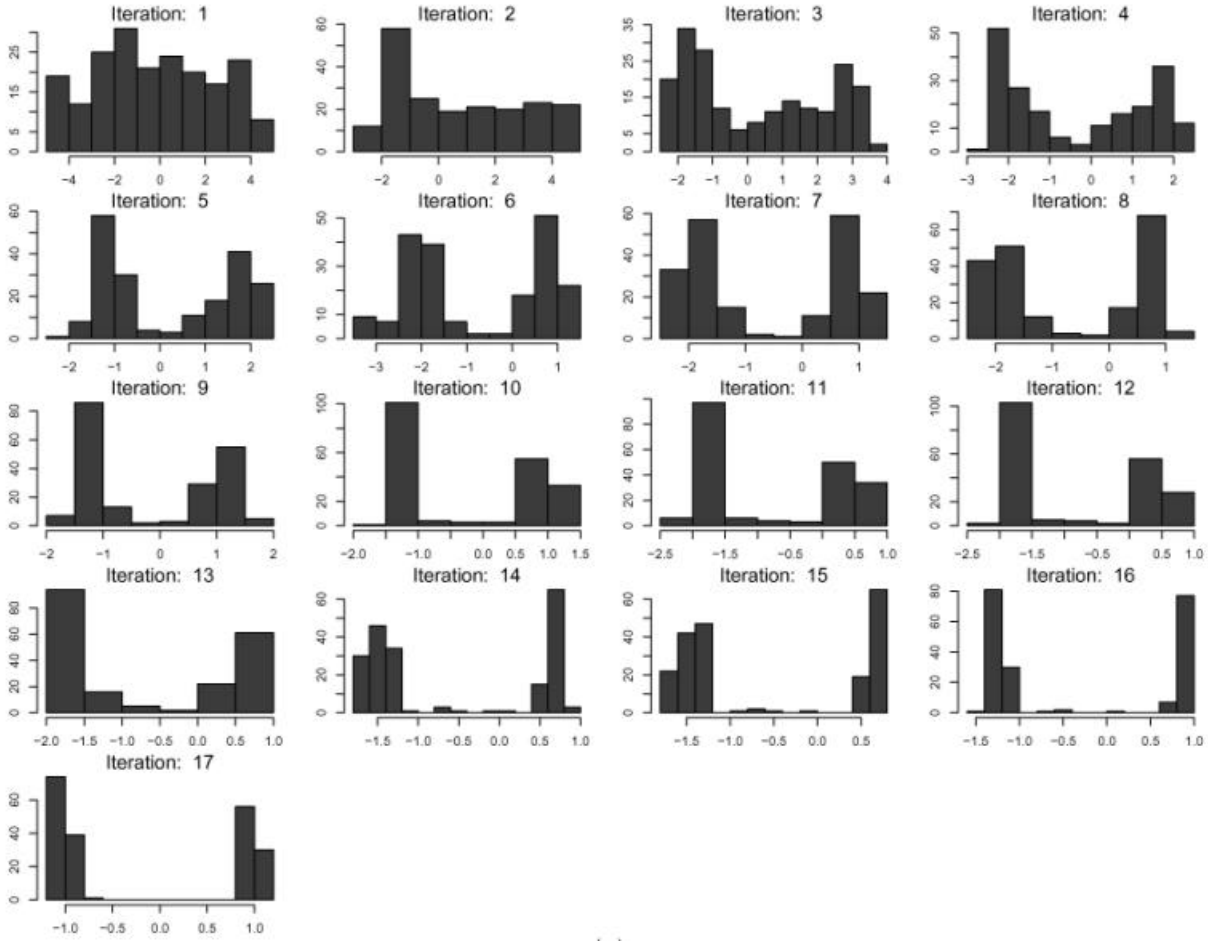

Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
Iteration: 6
Iteration: 7
Iteration: 8
Iteration: 9
Iteration: 10
Iteration: 11
Iteration: 12
Iteration: 13
Iteration: 14
Iteration: 15
Iteration: 16
Iteration: 17

Figure 8a.

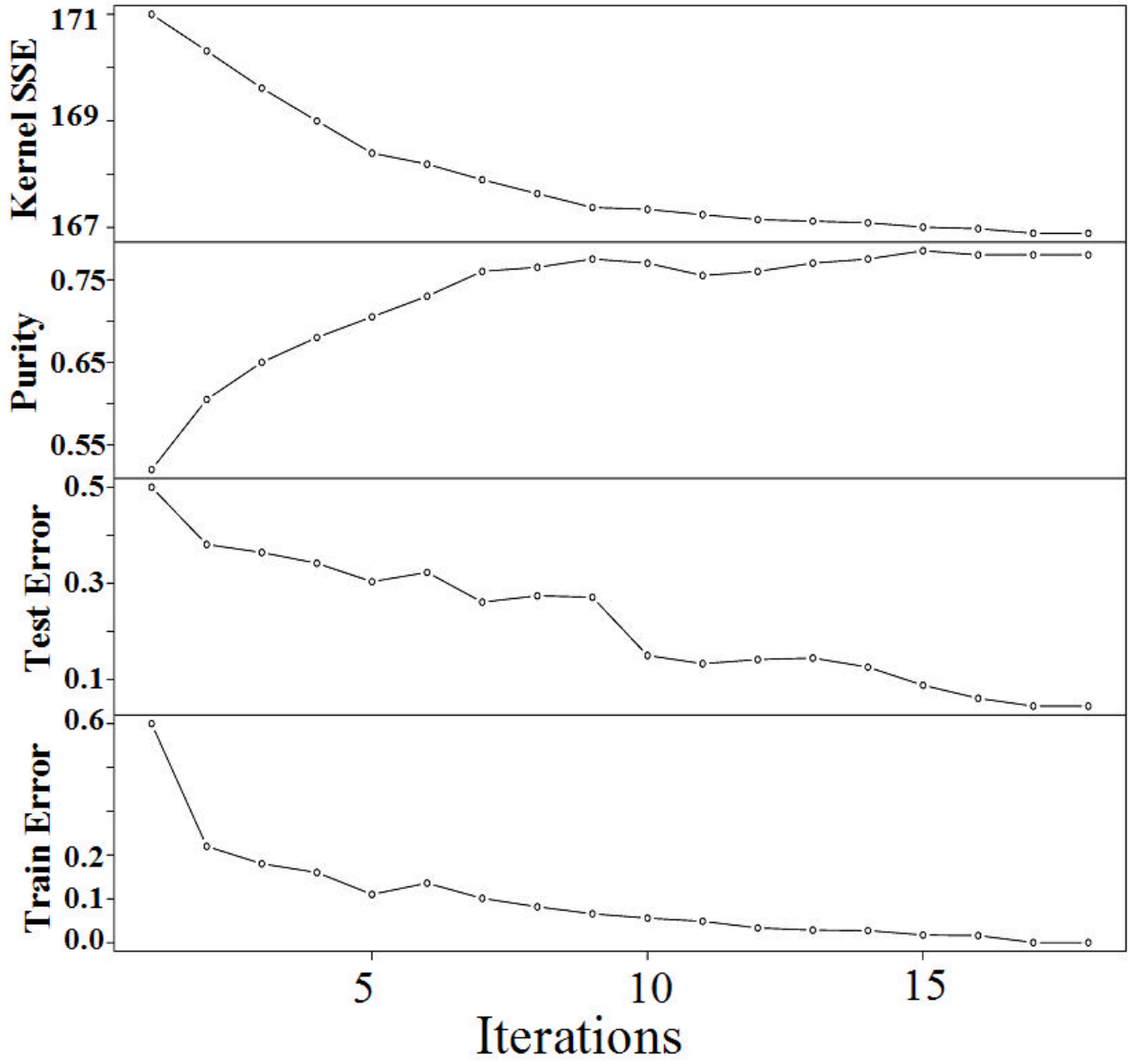

Kernel SSE
171
169
167
Purity
0.75
0.65
0.55
Test Error
0.5
0.3
0.1
Train Error
0.6
0.2
0.1
0.0
5
10
15
Iterations

Figure 8b.

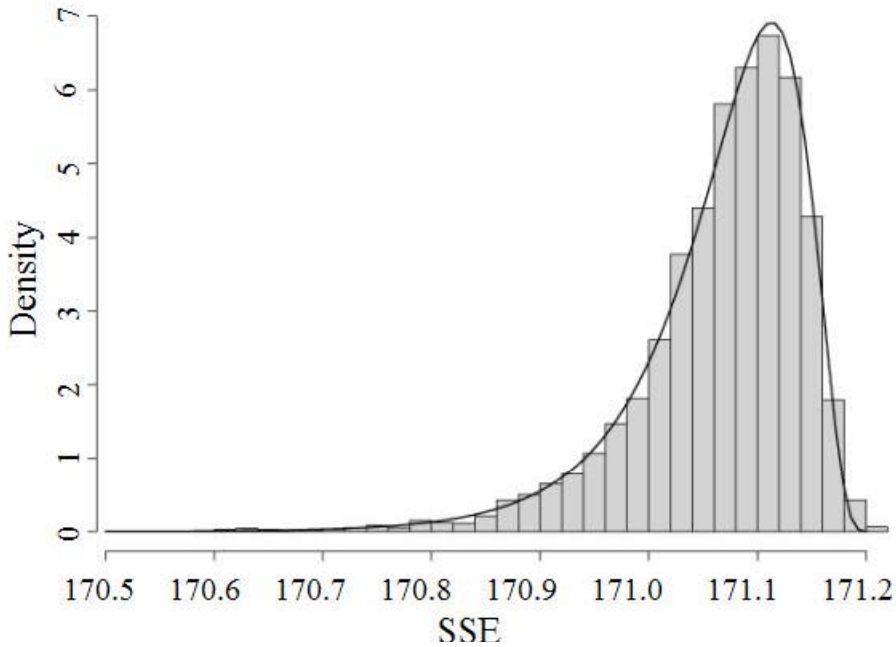

Density
0
1
2
3
4
5
6
7
170.5
170.6
170.7
170.8
170.9
171.0
171.1
171.2
SSE

Figure 9.

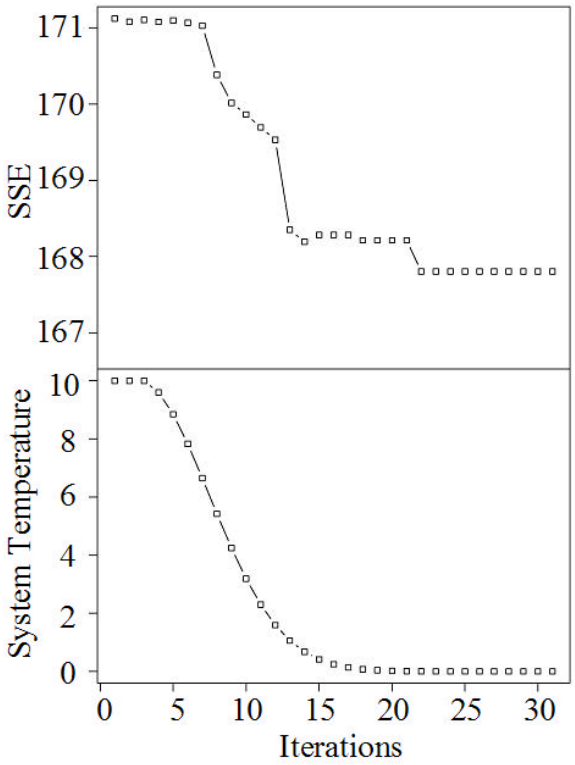

SSE
171
170
169
168
167
System Temperature
10
8
6
4
2
0
0
5
10
15
20
25
30
Iterations

Figure 10a.

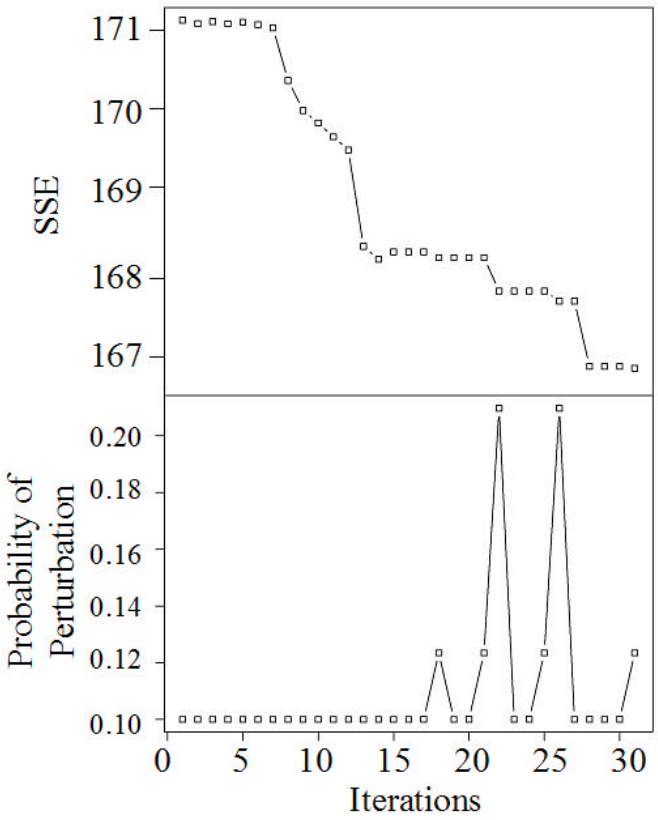

SSE
171
170
169
168
167
Probability of Perturbation
0.20
0.18
0.16
0.14
0.12
0.10
0
5
10
15
20
25
30
Iterations

Figure 10b.

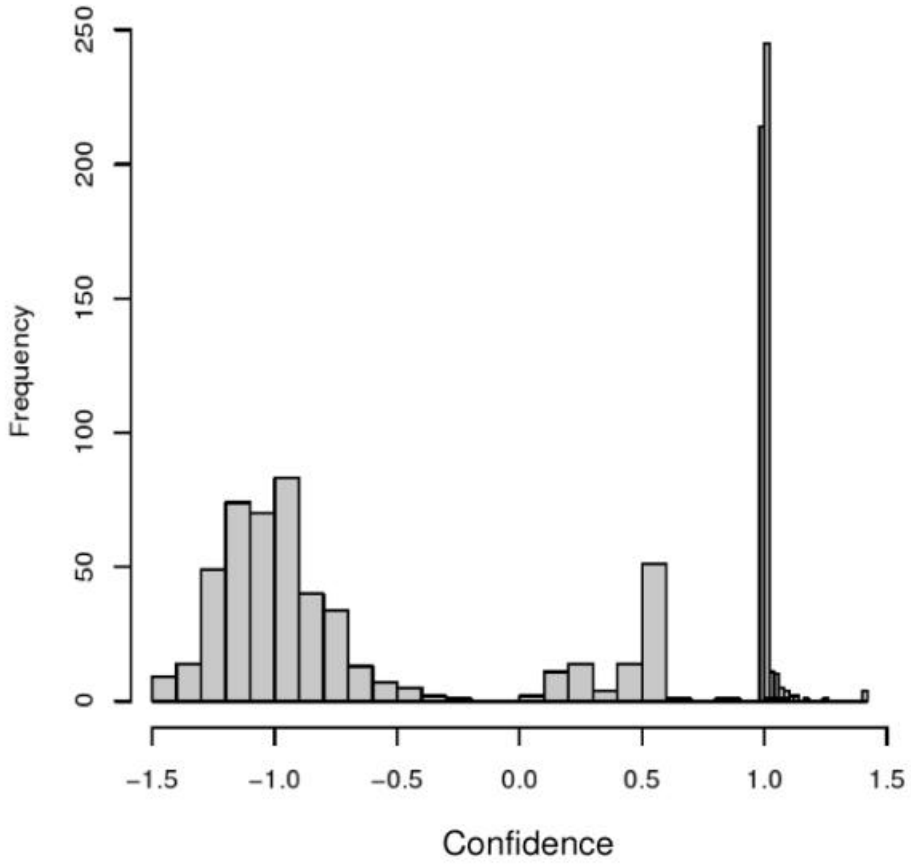


Figure 11.