Ad Hoc Signal Acquisition and Stochastic Carrier-Wave Signal Processing via FSAs, generalized HMMs, and generalized SVMs

Stephen Winters-Hilt[1,2,*], Joshua Morrison[1], and Jorge Chao[1]

[1] Dept. of Computer Science, University of New Orleans, 2000 Lakeshore Drive, New Orleans, LA 70148, USA
[2] Meta Logos Incorporated, 6218 Waldo Dr., New Orleans, LA 70122, USA

*email: winters@cs.uno.edu

Abstract

Signal acquisition and simple, time-domain, feature-extraction, can be done using a Finite State Automaton (FSA) that is based on tuning a variety of threshold parameters, as will be described. When FSA-based methods fail, however, HMM-Viterbi methods are used with possible pMM/SVM enhancements, as described in the stochastic sequential analysis (SSA) Protocol that follows, where generalized HMMs and generalized SVMs are used.

The signal processing "front-end" is the most *ad hoc* signal processing stage by its nature, as the signal can derive from anything imaginable. So initially, just getting a handle on the signal, or simply recognizing that a signal is present, is often the most difficult step in the signal analysis. We describe how signal acquisition can be done according to a stochastic sequential analysis (SSA) Protocol, in channel blockade type signal analysis, and in computational genomic examples, and thereby provide a template for analysis of other signal types. User-friendly web interfaces are being developed to guide users on tuning/selection for their front-end signal processing tasks.

The generalized HMM methods for stochastic sequential analysis, summarized here, provide a synergistic union that is used to arrive at a new form of carrier-based communication, as will be discussed, where the carrier is not periodic but is stochastic, typically with stationary statistics. HMM with binned duration, holographic HMMs, and meta-HMM algorithmic methods, are shown to enable practical stochastic carrier wave encoding/decoding, where stochastic carrier wave signal processing is encountered in a number of settings in science and nanotechnology. Results and applications are shown involving nanopore transduction detector signal analysis (for the nanopore transduction detector 'Nanoscope'), gene structure identification, the SSA Protocol, distributed HMMs and SVMs (to enable speedup of the SSA Protocol), and SVM-based clustering.

1 Introduction

Our objective is to establish a robust method for signal acquisition, and describe a new method for stochastic carrier wave (SCW) signal processing, where both methods are made possible with generalized HMMs that are described in the Methods.

Signal acquisition is typically the most *ad hoc* part of the signal processing problem. *Ad hoc* signal acquisition methods must be broadly applicable, and, thus, typically involve identification of statistical anomalies, including: (1) anomalous threshold detection; (2) anomalous "spike" detection; (3) anomalous Mutual Information detection; (4) anomalous void topologies (open reading frames); (5) emergent grammar detection; (6) emergent phenomeneology detection; and (7) holistic tuning based on observation of phase transitions, where all of these methods will be briefly described in what follows.

The general protocol for stochastic sequence analsysis (SSA) described here, the SSA Protocol [1], has a signal acquisition "front-end" based on a variety of methods involving identification of statistical anaomalies, as mentioned, but in addition has a weakness recovery prototcol that leverages all of the strengths of the generalized methods comprising the SSA Protocol [1]. In particular, if the 'local' statistical anomaly approaches fail, such as can occur in the fintite state automaton (FSA) based methods in what follows, more sophisticated, non-local structure identification statistical models might be needed, e.g., Hidden Markov model (HMM) based methods. In the SSA Protocol, if the FSA front-end method that is designed to pick up on localizable statistical anomolies fails, then the 'fall-through' method is to use a HMM-Viterbi computation to recognize the start-of-signal or full-signal feature. The HMM-Viterbi can be further enhanced via position-dependent Markov model (pMM) enhancements, possibly with SVM boosting (pMM/SVM) with SVM tuning metaheuristics. In the Background section that follows descriptions will be given of the SSA Protocol (with further details in Results), and of the FSAs, HMMs, SVMs, and Metaheuristics comprising its operation (with further details in Methods and Results).

2 Background

2.1 Finite State Automoton based acquisition via anomaly: emission, MI, void topology

Time-domina FSA signal acquisition based on emission thresholding

The time-domain FSA is shown in Fig. 1, Right, where an eight-state automaton is shown with a six base-pair DNA hairpin channel blockade signal (from [2]). The Web interface to the acquisition interface for the automaton involves parameters commonly encounted in channel current blockade experiements, such as the blockade signal's drop in current value at onset of blockade, e.g., the "start drop value". This is an excellent acquisition parameter in the case of molecules that are engineered for channel capture and modulation in the NTD experiments).

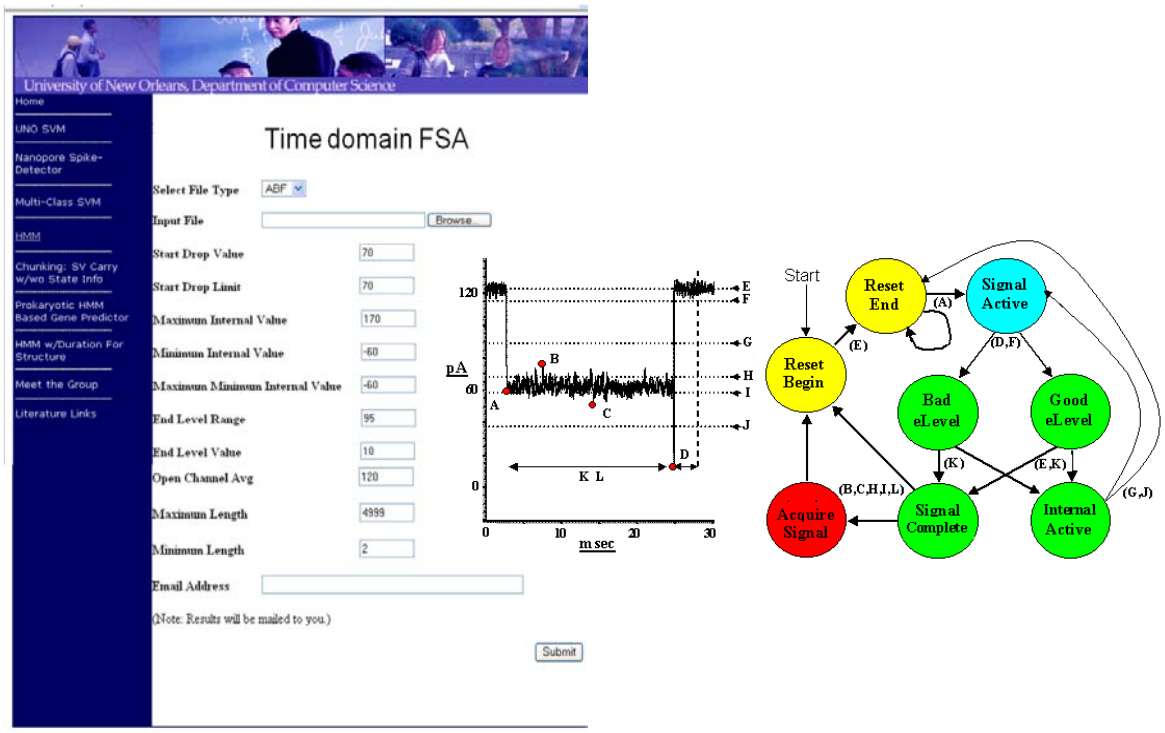


Fig. 1. The Time-domain FSA. The web-interface comes pre-loaded with standard values for those not expert. Descriptions involving prior application of the methods [3], however, make their use by non-experts straightforward.

Acquisition based on MI and Void-topology anomalies are shown in Fig.s 2 Left and 2 Right. Mutual Information, MI(X;Y) is: $\mu=\Sigma_x\Sigma_y p(xy)\log(p(xy)/p(x)p(y))$. If X and Y are independent r.v's then MI=0. If we have a DNA sequence $x_1\ldots.x_ix_{i+1}\ x_{i+2}\ldots\ldots x_n$ (where x_k ={a,c,g, or t}) then we can get counts on pairs x_ix_{i+1} for i=1..n, and assuming stationarity on the data, and large enough n, we can speak of the joint probability p(X,Y). Calculation of MI(X,Y) then gives an indication of the linkage between base probabilities in dinucleotide probabilities. This can be extended to linkages when the two bases aren't sequential (have a base gap between them greater than zero), such as pairs based on x_ix_{i+2} (gap=1), etc. This type of statistical framework can then be iterated to higher order MI calculations in a variety of ways to explore a number of statistical linkages and build towards a motif identifier based on such linkages (gIMM). Such an analysis on the *V. Cholerae* Chr. 1 genome is shown in Fig. 2, and clearly indicates a three-component encoding of data, i.e., the codon structure is revealed.

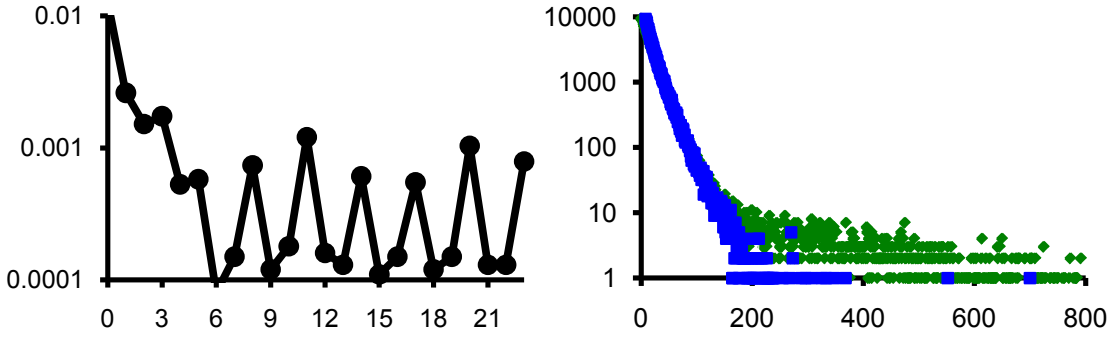


Fig. 2 Left. The y-axis shows the mutual information on base-pairs in he *V. cholerae* gemone, the x-axis is the gap size between bases used to construct the base pairs. A three-component encoding of data, i.e., the codon structure is

revealed. From the strong linkages for gaps 1 through 5, it is also clear that hexamer Markov model statistics will be strong in many regions of the genome.**Right.** An examination of the void sizes encountered for various codons (some more ORFs, or smORFs, see [4]), or groupings of codons, reveals the stop codons very clearly as codons with anomalously lengthy stop-codon void regions. Shown in green are the standard ORFs (voids in the codon subset {(taa),(tag),(tga)}), which are clearly behaving differently from other codon voids (blue) when length is greater than 500 bases. The voids shown in blue are voids in the codon subset {(aaa),(gaa),(gat)}.

2.2 Generic grayscale HMM

An HMM that is designed to generate a particular signal need only have a few states and transitions. In reverse, this HMM 'template' can be used to detect signal with matching statistics. An HMM that is meant to generate a large family of signals, on the other hand, needs to have more states and associated transitions. The 'Generic' HMM or 'grayscale' template HMM is an example of this in the case of the channel current analysis applications in [2] and in many of the examples in this paper. A general HMM [2] can be used to characterize current blockades by identifying a sequence of sub-blockades as a sequence of state emissions. The parameters of the HMM are estimated using Expectation/Maximization to effect de-noising. The HMM Viterbi path features (state histogram and two-component merged transition histogram [2]) can be extracted as a stationary statistics 'fingerprint', bundled as a feature vector and used in SVM classification [2]. The generic HMM is described further in some of the Methods subsections where it is used.

2.3 pMM/SVM application

Markov-based statistical profiles, in a log likelihood discriminator framework, can be used to create a fixed-length feature vector for SVM based classification [5]. Part of the idea of the method is that whenever a log likelihood discriminator can be constructed for classification on stochastic sequential data, an alternative discriminator can be constructed by 'lifting' the log likelihood components into a feature vector description for classification by SVM. Thus, the feature vector uses the individual log likelihood components obtained in the standard log likelihood classification effort, the individual-observation log odds ratios, and 'vectorizes' them rather than sums them. The individual-observation log odds ratios are themselves constructed from positionally defined Markov Models (pMM's), so what results is a pMM/SVM sensor method [5]. This method has utility in a number of areas of stochastic sequential analysis, including splice-site recognition [5] and other types of gene-structure identification, file recovery in computer forensics ('file carving'), and speech recognition, among others.

2.4 SSA Protocol and SCW Signal Analysis

The SSA Protocol and implementations allows an efficient basis for robust stochastic carrier wave signal processing, where the stochastic signal obeys stationary statstics or some other reproducible measure. The SSP Protocol strengthens the critical first step in the signal processing, the signal acquisition step, via use of its weakness recovery protocol. In channel current cheminformatics (CCC) applications, the FSA method for acquisition alone suffices, but in more challenging SCW applications full use of the SSA Protocal may be required for good signal recognition. Stochastic Carrier Wave (SCW) signal processing occurs in both natural and engineered situations [1]. Whenever Nature is observed with a sequence of observations that have stationary statistics (associated with equilibrium and near-equilibrium flow situations, for example), then the basis for SCW signal processing arises. SCW also parallels all electrical

engineering carrier-wave methodologies where periodic wave methods are used in some modulation scheme, thus the number of engineering applications is enormous. AM heterodyning, for example, can be replaced with stochastic carrier wave with pattern recognition informed (PRI) heterodyning . Also have phase modulation equivalence: the standard periodic carrier wave approach has a coherent phase reference, while SCW introduces a stochastic carrier wave with stationary statistics 'phase'. Have similar capabilities as with phased-locked loop (PLL), for example, where the phase tracking is done on SCW encoded information.

The biophysics and 'information flows' associated with the nanopore transduction detector are analyzed using a generalized set of hidden Markov model (HMM) and Support Vector Machine (SVM) based tools, as well as *ad hoc* finite state automata (FSA) based methods, and a collection of distributed (swarm) intelligence and genetic algorithm methods for tuning and selection. Used with a nanopore detector, the channel current cheminformatics use with stationary statics channel blockades provides a method for a highly sensitive nanopore detector for single molecule biophysical analysis, among other things [1,2].

2.5 Support Vector Machines

A classifier is typically a simple rule whereby a class determination can be made, such as a decision boundary (see Fig. 3). 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 usually 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 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.

Support Vector Machine methods are described for classification, clustering, as well as aiding with signal analysis and pattern recognition on stochastic sequential data. Analysis tools for stochastic sequential data, Markovian (or causal) data for example, have broad-ranging application in that almost any device producing a sequence of measurements can be made more sensitive, or "smarter," by efficient learning of measured signal/pattern characteristics via SVM-enhanced SSA Protocol methods.

Using the SSA Protocol, the biophysics and 'information flows' associated with the nanopore transduction detector are analyzed using a generalized set of hidden Markov model (HMM) and Support Vector Machine (SVM) based tools, as well as *ad hoc* finite state automata (FSA) based methods, and a collection of distributed (swarm) intelligence and genetic algorithm methods for tuning and selection. Used with a nanopore detector, the channel blockades with stationary statistics provide a method for a highly sensitive nanopore detector for single molecule biophysical analysis, among other things.

It is conceivable to have a properly coded SVM but to initiate 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, with tuning on SVM parameters, to optimize. For some feature vectors, such as probability vectors, this can partly be

done automatically with choice of kernel. Overall, in many situations the SVM tuning can be done quickly, manually, and to some extent automatically, with simple range testing, where only small, separated, subsets of the training data are used in the tuning tests, before performing SVM training on the full dataset minus the tuning data. Sometimes more elaborate tuning procedures are needed, however, and thus necessary for performance guarantees, and also for the SVM applications in clustering that will be described in the Methods. 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 (such as for the SVM sensitivity and specificity score). Metaheuristics optimization includes genetic algorithms, simulated annealing, swarm intelligence, ACO, steepest ascent hill-climbing, among others. Applications of many of these methods are shown in the results involving SVM-external clustering.

Also shown will be implementation details for distributed SVM training, and other speedup optimizations, for practical deployment of the powerful SVM classification and clustering methods in real-time operational situations (as demonstrated in analysis in nanopore detector experiments).

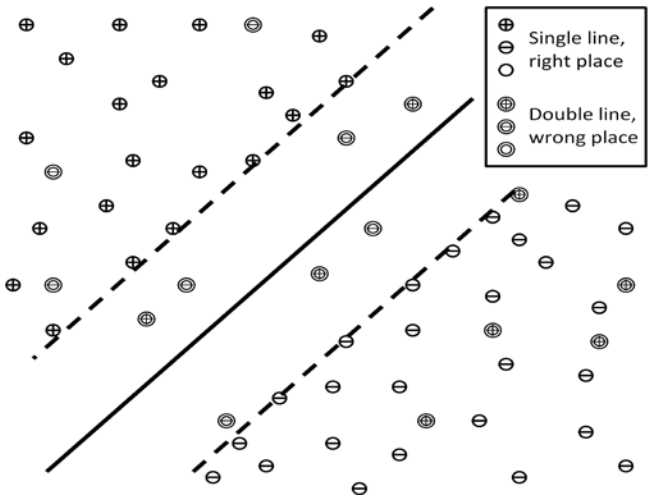


Fig. 3. SVM method for classification. Based on establishing a separating hyperplane with a structural risk minimization contribution to the optimization via incorporation of a maximum margin constraint. Training instances that are mis-labeled (in the "wrong place") are minimized via introduction of a penalty factor in the optimization.

3 Methods

3.1 HMM Feature Extraction and EVA

3.1.1 Viterbi Path

In the Viterbi algorithm, a recursive variable is defined: $v_{kn} = v_k(n) = v_k(b_n) =$ *"the most probable path ending in state $\lambda_n=k$ with observation b_n"*. The recursive definition of $v_k(n)$ is then: $v_l(n+1) = e_l(b_{n+1}) \max_k [v_k(n) a_{kl}]$. From which the optimal path information is recovered according to the (recursive) trace-back:

$\Lambda^* = \text{argmax}_{\Lambda} P(B, \Lambda) = (\lambda^*_0, \ldots, \lambda^*_{L-1})$; $\lambda^*_n|_{\lambda^*_{n+1}=l} = \text{argmax}_k [v_k(n) a_{kl}]$, and where $\lambda^*_{L-1} = \text{argmax}_k [v_k(L-1)]$, for length L sequence.

The recursive algorithm for the most likely state path given an observed sequence (the Viterbi algorithm) is expressed in terms of v_{ki} (the probability of the most probable path that ends with observation $b_n = i$, and state $\lambda_n=k$). The recursive relation is lifted directly from the underlying probability definition: $v_{ki} = \max_n\{e_{ki}a_{nk}v_{n(i-1)}\}$, where the $\max_n\{\ldots\}$ operation returns the maximum value of the argument over different values of index n, and the boundary condition on the recursion is $v_{k0} = e_{k0}p_k$. The emission probabilities are the main place where the data is brought into the HMM-EM algorithm. An inversion on the emission probability is possible when the states and emissions share the same alphabet of states/quantized-emissions (details in Sec. 2.3.6). The Viterbi path labelings are, thus, recursively defined by $p(\lambda_i|\lambda_{(i+1)}=n) = \text{argmax}_k\{v_{ki}a_{kn}\}$. The evaluation of sequence probability (and its Viterbi labeling) take the emission and transition probabilities as a given. Estimates on those emission and transition probabilities themselves can be obtained by an Expectation/Maximization (EM) algorithm that is known as the Baum-Welch algorithm in this context. A 50-state generic HMM is used extensively in [2], and will be described further in the EVA and other methods that follow.

3.1.2 pMM/SVM

For start-of-coding recognition, for example, one can create a profile Markov model (pMM) based log-likelihood ratio (LLR) classifier given by $\log[P_{start}/P_{non\text{-}start}] = \Sigma_i \log[P_{start}(x_i=b_i)/P_{non\text{-}start}(x_i=b_i)]$. Rather than a classification built on the sum of the independent log odds ratios, however, the sum of components could be replaced with a vectorization of components:

$\Sigma_i \log[P_{start}(x_i=b_i)/P_{non\text{-}start}(x_i=b_i)]$ --> {…., $\log[P_{start}(x_i=b_i)/P_{non\text{-}start}(x_i=b_i)]$, ….}

These can be viewed as feature vectors (f.v.'s), and can be classified by use of an SVM. The SVM partially recovers linkages lost with whatever order of Markov model dependency that is imposed. For the 0th order MM in the example, the positional probabilities are approximated as entirely independent -- which is far from accurate. The SVM approach can recover statistical linkages between components in the f.v.'s in the SVM training process.

There are generalizations for the MM sensor and its SVM f.v. implementation, and all are compatible with the SVM f.v. classification profiling. Markov Profiling with component-sum to component feature-vector mapping for SVM/MM profiling, thus, encompasses use of MMs, IMMs, gIMMs, hIMMs, and ghIMMs [4,5], and SVM usage via "vectorization" to SVM/MM, SVM/IMM, SVM/gIMM classification profiling (with use of the SVM's confidence parameter).

3.1.3 Feature Extraction via EVA projection (for use with power signal analysis)

Emission variance amplification (EVA) projection is used in the SSA Protocol to go from a power signal (or anything sampled from a continuum domain of possibilities) to a sparser, projected 'EVA state', representation of the data. Quantization on the sparser representation can then provide a discrete representation. Once all states are discrete, higher order structure (or encoding) can be extracted by use of the meta-HMM generalization described in Sec. 3.2, and other methods.

In the CCC analysis in [2], we have an HMM with emissions probabilities parameterized by Gaussian distributions: emission_probabilities[i][k] = exp(-(k-i)*(k-i)/(2*variance)), where "i" and "k" are each a state where 0 <= i,k <= 49 in a 50 state system. To perform EVA in this setting, the variance is simply multiplied by a factor that essentially widens the gaussian distribution parameterized to best fit the emissions, and the equation simply becomes exp(-(k-i)*(k-i)/(2*variance*eva_factor)). For a sizable range of this parameter, HMM with EVA will remove the noise from the power signal while *strictly* maintaining the timing of the state transitions.

After EVA-projection, a simple FSA can easily extract level duration information. Each level is identified by a simple threshold of blockade readings, typically one or two percent of baseline. When EVA boosts the variance of the distribution, for states near a dominant level in the blockade signal, the transitions are highly favored to points nearer that dominant level. This is a simple statistical effect having to do with the fact that far more points of departure are seen in the direction of the nearby dominant level than in the opposite direction. When in the local gaussian tail of sample distribution around the dominant level, the effect of transitions towards the dominant level over those away from the dominant level can be very strong. In short, a given point is much more likely to transition towards the dominant level than away from it, thereby arriving at a "focusing" on the levels, while preserving level transitions. In this respect, other distributional parameterizations could be used than Gaussian, but Gaussian is a good starting point (a mixture of a sufficient number of distributions will arrive at a Gaussian distribution overall, an expression of the law of large numbers).

3.1.4 Feature Extraction via Data Absorption (a.k.a. Emission Inversion)

A new form of "inverted" data injection is possible during HMM training when the states and quantized emission values share the same alphabet. This is typically the case in the CCC power signal analysis examples given here. Results from channel current signal classification consistently show approximately 5% improvement in accuracy (sensitivity + specificity) with the aforementioned data inversion upon SVM classification (and this holds true over wide ranges of SVM kernel parameters and collections of feature sets). Transition & "absorption" statistical profiles are thought to work better than standard transition & emission profiles, in generalized classification performance, due to regularization with an effective SRM (structural risk minimization [6]) constraint, via optimization with an added term that depends on the relative entropy between state prior probabilities and emission posterior probabilities.

By swapping $e_b(k)$ for $e_k(b)$ we introduce a multiplicative factor, the ratio of the priors on states to the frequencies on emissions: $e_k(b) = e_b(k)\ [P(b)/P(k)]$. This factor weights the computations in a manner that seems to track, and minimize, on the Kullback-Leibler divergence between the state prior distribution and the emission frequency distribution. This approximate notion follows from the evaluation of the extra terms that will occur on the maximum log-prob calculation for the Viterbi path. On the Viterbi solution, using the swapped emission probabilities, the sum (on log probabilities) at the end will differ by a sum of log ratios: $\log [P(k_i)/P(b_i)] = - \log[P(b_i) / P(k_i)]$ Normalized by length 'L' over different k and b, this term is approximated by Diff Term = $– D(P(Z)||P(S))$, maximizing on this term is, thus, minimizing on the divergence, $D(P(Z)||P(S))$, between the priors and the emissions.

3.1.5 Modified AdaBoost for Feature selection and Data fusion

AdaBoost [7] can learn a sequence of weak classifiers and then boosts them by a linear combination into a single strong classifier. As a classification method, one of the main disadvantages of AdaBoost is that it is prone to overtraining. However, AdaBoost is a natural fit for feature selection. Here, overtraining is not a problem, as AdaBoost is only used to finds diagnostic features and those features are then passed on to a classifier that does not suffer from overtraining (such as an SVM). HMM features, and other features (from neural net, wavelet, or spike profiling, etc.), can be fused and selected via use of the Modified Adaboost selection algorithm [3].

In Modified AdaBoost [3] weights are given to the weak learners as well as the training data. The key modifications here are to give each column of features in a training set a weak learner and to update each weak learner every iteration, not just update the weights on the data. In an example where there is a set of 150-component feature vectors, 150 weak learners would be created. As previously mentioned, each weak learner corresponds to a single component and classifies a given feature vector based solely on that one component. Then, weights for these weak learners are introduced. In each iteration of this modified AdaBoost process, weights for both the input data and the weak learners are updated. The weights for the input data are updated as in the standard AdaBoost implementation, while weights on the individual weak learners are updated as if each were a complete hypothesis in the standard AdaBoost implementation. At the end of the iterative process, the weak learners with the highest weights, that is, the weak learners that represent the most diagnostic features, are selected and those features are passed to an SVM for classification (see [3] for more details). Thus, the benefits of both AdaBoost and SVMs are obtained.

3.2 Channel Current Cheminformatics

The SSA Protocol has been developed for the discovery, characterization, and classification of localizable, approximately-stationary, statistical signal structures in stochastic sequential data, such as in channel current cheminformatics (CCC), as shown in Fig. 4.

The general components for a stochastic signal analysis protocol and a stochastic carrier wave communications protocol are described in the next section. NTD, with the channel current cheminformatics implementation of the SSA protocol, provides proof-of-concept examples of the SSA methods utilization, and can be used as an example of finite state communication. From the CCC/NTD starting point, it is easier to convey the unique signal boosting capabilities when working with real-time capable HMMBD signal processing [8] and other SSA methods. In the larger sense, recognition of stationary statistics transitions allows one to generalize to full-scale encoding/decoding in terms of stationary statistics ‘phases’, i.e., stochastic phase modulation, a form of stochastic carrier-wave communications. Many of the Proof-of-concept experiments described in what follows involve SSA applications in a CCC implementation or a context for the NTD platform. The SSA Protocol, however, is a general signal processing paradigm for characterizing stochastic sequential data, as will be detailed next.

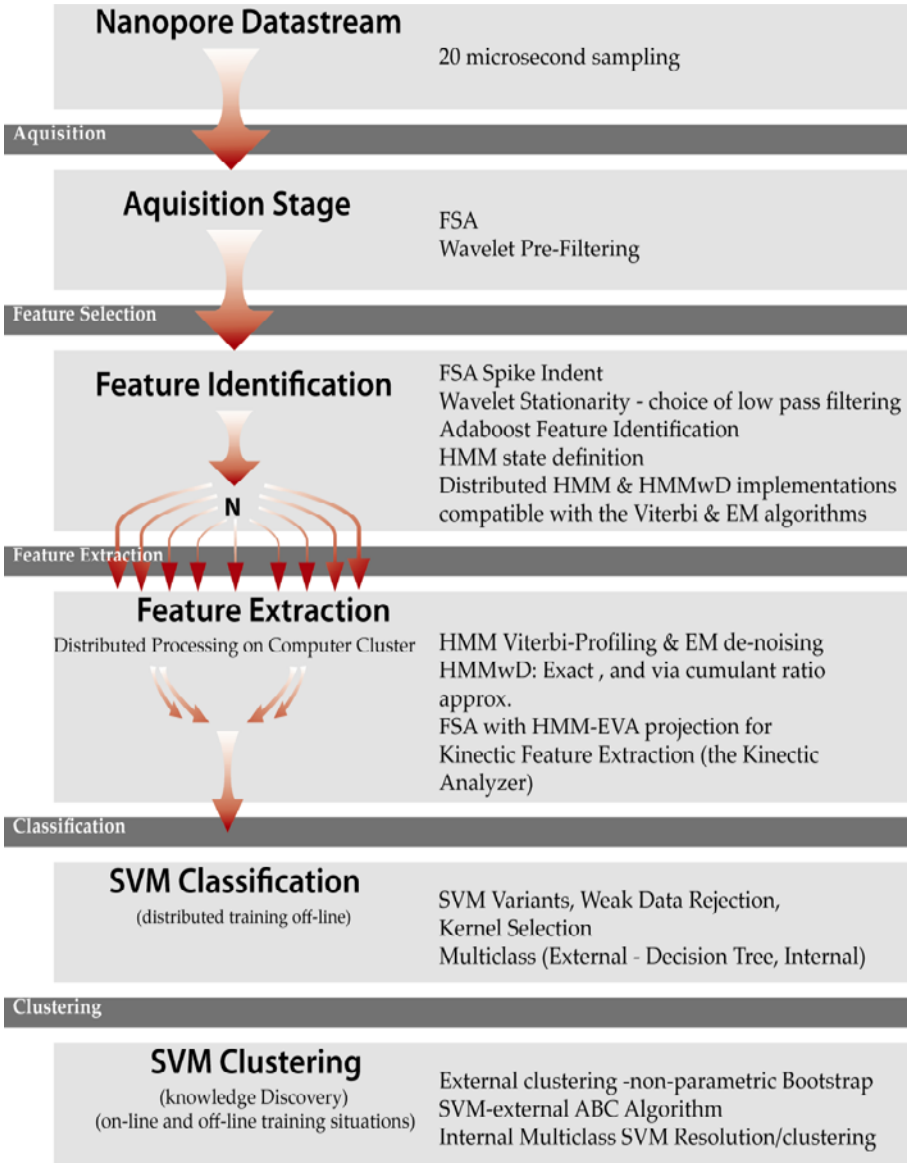


Figure 4. Nanopore cheminformatics & data-flow control architecture. Aside from the modular design with the different machine learning methods shown (HMMs, SVMs, etc.), recent augmentations to this architecture for real-time processing include use of a networked server to link to the patch-clamp amplifier, and the 'real-time' pattern recognition informed signal processing architecture (the latter shown in Fig. 7).

3.3 Generalized HMMs (Gene-structure identification)

Hidden Markov models are an amazing tool at the nexus where Bayesian probability and Markov models meet dynamic programming. To properly define/choose the HMM model in a machine learning context, however, further generalization is usually required. This is because the 'bare-bones' HMM description has critical weaknesses in most applications, which are summarized below. Fortunately, these weaknesses can be addressed, and in computationally efficient ways, see Fig. 5, with further details in the Methods.

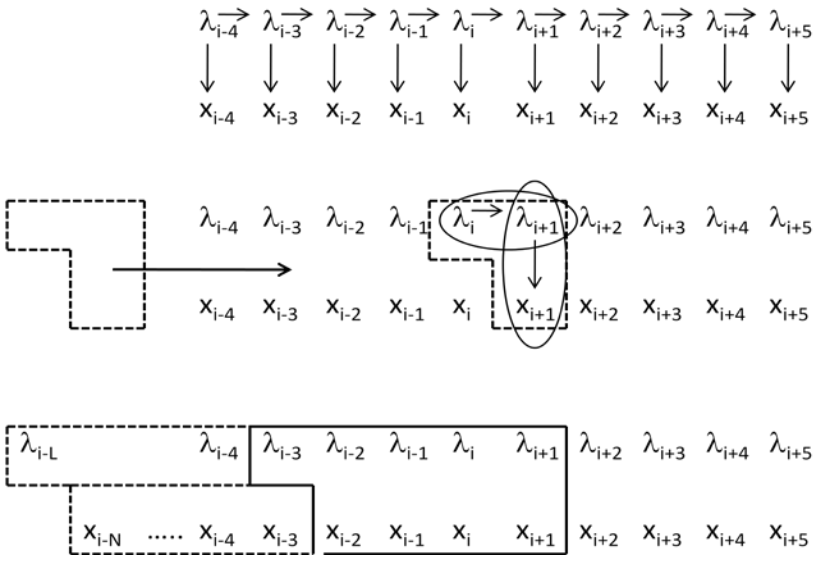


Fig. 5. Comparison of standard HMM and the clique-generalized HMM. The upper graphical model is for the standard HMM and shows the 'emission' observation sequence x_i, and the associated hidden label sequence λ_i, and the arrows denote the conditional probability approximations used in the model (for the transition and emission probabilities). Focusing at the level of the core joint-probability construct at instant 'i' in the middle graph, the standard HMM is a subset of the joint probability construct $P(\lambda_i, \lambda_{i+1}, x_{i+1})$. The generalized-clique HMM is shown in the graphical model at the bottom for one particular clique generalization. The model can be exact on emission positionally, then extend via zone dependence and use of gIMM interpolation. The model can be exact to higher order in state (referred to as footprint states', see [9]), and also extends modeling to have HMM with duration modeling. When doing the latter, zone-dependent and position dependent modeling can be incorporated via reference to the duration in the model, and can be directly incorporated into a generalized Viterbi algorithm (and other generalized HMM algorithms), as well as any other side-information of interest [10].

A brief list of the typical weaknesses encountered with the standard HMM:

(1) Standard HMMs are at low Markov order in transitions (first) and in emissions (zeroth), and transitions are decoupled from emissions, which can miss critical structure in the model (e.g., state transition probabilities that are strongly sequence dependent). This weakness is eliminated if we generalize to the largest state-emission clique possible, fully interpolated on the data set, with use of a minimal state-length constraint to obtain an efficient implementation (see Fig. 5).

The generalized clique HMM (Fig. 5) begins by enlarging the primitive hidden states associated with the individual base labels (as exon, intron, or junk) to substrings of primitive hidden states or footprint states. There is a key constraint, however, to keep the scaling of footprint states linear with footprint size: the footprint states are constrained to have self-transitions with a minimal length such that a footprint, and the mostly overlapping 'next' footprint, together can only have one transition between states of different type. The emissions are likewise expanded to higher order in the fundamental joint probability that is the basis of the generalized-clique, or 'meta-State', HMM. Further details on the meta-state HMM [9] are in the Methods that follow

(2) Need Method for directly incorporating side-information into the dynamic programming table based optimizations (used in the Viterbi and Baum-Welch algorithms, etc.). This is solved in [10], where an HMM is 'bootstrapped' into a HMM-with-duration, see Methods that follow for brief description.

(3) Standard HMMs don't properly model self-transition durations, imposing a 'best-fit' geometric distribution on self-transition duration distributions instead. This weakness is eliminated if we generalize to a HMM-with-duration (HMMD) formalism, where direct modeling on self-transition duration distributions is incorporated (see Fig. 5). Standard HMMD methods are computationally expensive, however, when compared to Standard HMM. This weakeness can be addressed, without loss of generality, via use of HMM with binned duration (HMMBD) representations. Further details on HMMBD [8] are in the Methods that follow.

(4) Standard HMM and HMMD have latency bottleneck if full table computation is used on a lengthy data sequence, so need method for distributed processing'chunking' with overlaps sufficient for recovery. This is demonstrated in the Results.

(5) Need Method for HMM Feature Extraction Selection, Compression, and Fusion. A modified form of Adaboost is used for this purpose, see methods in preceding sections.

(6) Need Multitrack (Holographic) Generalization. In particular, need to show that hidden constraints can significantly limit model complexity, as seen in the clique generalization with application in gene-finding in [9], allowing significant scaling in multiple hidden-track (holographic) model complexity. In the Results we show the preliminary statistical support to justify the Two-track HMM alternative-splice gene-finder model.

(7) Need Method for Standardized HMM application to power signal data, and this is described in the Results.

(8) Need Method for Standardized HMM usage: the SSA Protocol is described in the Results, and outlined in what follows next.

3.4 Overview of the SSA Protocol

The SSA protocol is shown in Fig. 6 in a common signal-processing flow topology. The SSA Protocol is for the discovery, characterization, and classification of localizable, approximately-stationary, statistical signal structures in channel current data, or genomic data, or sequential data in general, and changes between such structures. The core signal processing stage in Fig. 6 is usually the feature extraction stage, where central to the signal processing protocol is the Hidden Markov model. The SSA Protocol has a built-in weakness recovery protocol, outlined next, where the strengths of the HMM are further leveraged to full advantage.

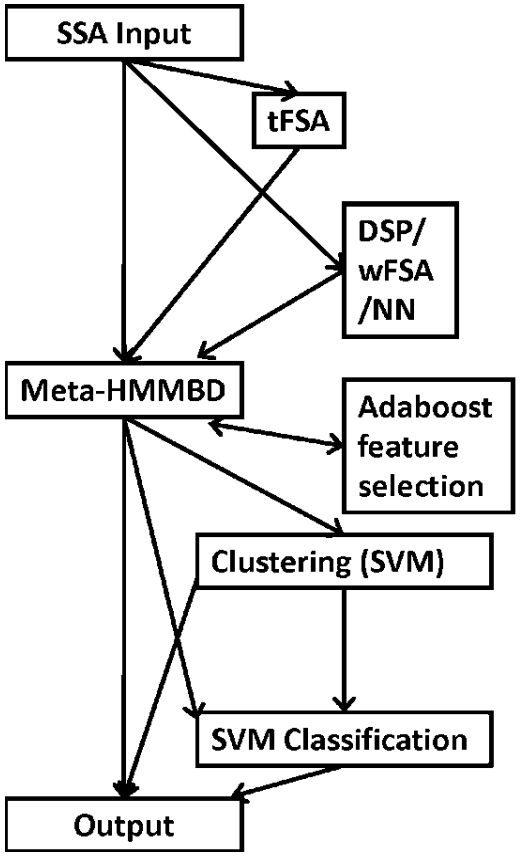


Fig. 6. The most common stochastic sequential analysis flow topology. The main signal processing flow is typically Input → tFSA→ Meta-HMMBD → SVM → Output. Notable differences occur in channel current cheminformatics where there is use of EVA-projection, or similar method, to achieve a quantization on states, then have Input → tFSA → HMM/EVA → meta-HMMBD-side → SVM → Output. While, in gene-finding just have: Input → meta-HMMBD-side → Output. In gene-finding, however, the HMM internal 'sensors' are sometimes replaced, locally, with profile-HMMs [2] or SVM-based profiling [5], so topology can differ not only in the connections between the boxes shown, but in their ability to embed in other boxes as part of an internal refinement.

3.4.1 Weakness Recovery Protocol (Acquisition strengths boosted by the entire SSA Protocol)

The sequence of algorithmic methods used in the SSA Protocol comprise a weakness recovery protocol: (i) the weakness in the Fintie State Automoton (FSA) methods will be shown to be its difficulty in strong structure identification (especially non-local structure), for which HMM methods (and tuning metaheuristics) are the solution. (ii) for the HMM, in turn, the main weakness is in local sensing 'classification', and as a classification problem, the problem can be solved via incorporation of generalized SVM methods. If facing a classification task independent of a prior signal processing need, the SVM will also be the method of choice in what follows. (iii) The weakness of the SVM, whether used for classification or clustering, but especially for the latter, is the need to optimize over algorithmic, model (kernel), chunking, and other process parameters during learning. This is solved via use of metaheuristics for optimization such as simulated annealing, genetic algorithm optimization, particle swarm optimization, etc. (iv) The main weaknesses in the metaheuristic effort is partly resolved via use of the "front-end" methods, like the FSA, and partly resolved by a knowledge discovery process (that can be done using the SVM clustering methods). The SSA Protocol weakness recovery method, thus, comes full circle, thereby establishing a robust signal processing platform. To have a specific example, consider the situation where the 'front-end' FSA acquisition methods fail, the weakeness recovery is to fall through to a HMM-Viterbi based feature extraxction, with feature set classification via HMM 'templates' or via a generic HMM feature-extractor with SVM-based classifier (the latter is done in the CCC implementation and is recommended for robustness).

The HMM methods are the central methodology/stage in the channel current cheminformatics (CCC) protocol in that the other stages can be dropped or merged with the HMM stage in many

incarnations. For example, in some data analysis situations the time-domain Finite State Automoton (tFSA) methods could be totally eliminated in favor of the more accurate HMM-based approach to the problem, with signal states defined/explored in much the same setting, but with the optimized Viterbi path solution taken as the basis for the signal acquisition.

The HMM features, and other features (from neural net, wavelet, or spike profiling, etc.) can be fused and selected via use of various data fusion methods, such as a modified Adaboost selection (from [3,11]). The HMM-based feature extraction provides a well-focused set of 'eyes' on the data, no matter what its nature, according to the underpinnings of its Bayesian statistical representation. The key is that the HMM not be too limiting in its state definition, while there is the typical engineering trade-off on the choice of number of states, N, which impacts the order of computation via a quadratic factor of N in the various dynamic programming calculations used (comprising the Viterbi and Baum-Welch algorithms among others).

The HMM 'sensor' capabilities can be significantly improved via switching from profile-MM sensors to pMM/SVM-based sensors, as indicated in [5], where the superior performance and generalization capability of this approach was demonstrated. A martingale feature vector is described in this context in [1].

3.4.2 Stochastic Carrier-Wave Signal Processing

We establish a new type of communication process where the carrier wave is a stochastic observation sequence that obeys stationary statistics. In standard periodic carrier wave signal processing convolving with the carrier frequency allows the signal modulations of that carrier to be obtained. Here we have something analogous, but we have a carrier with stationary statistics, not fixed frequency, and can recognize different phases of stationary statistics via HMM methods for class-independent feature extraction, with Support Vector Machines (SVMs) for sparse data classification, or via HMM methods for class-dependent HMM generative projection.

In standard bandlimited signal analysis with periodic waveforms, sampling is done at the Nyquist rate for full reproducible capability. If the sample information is needed elsewhere, it is then compressed (possibly lossy) and transmitted (a 'smart encoder'). The received data is then decompressed and reconstructed (by simply summing wave components, e.g., a 'simple' decoder). If the signal is sparse or compressible, then compressive sensing [12] can be used, where sampling and compression are combined into one efficient step to obtain compressive measurements (refered to as 'dumb' encoding in [12] since a set of random projections are employed), which are then transmitted. On the receiving end, the decompression and reconstruction steps are, likewise, combined using an asymmetric 'smart' decoding step. This progression towards asymmetric compressive signal processing can be taken a step further if we consider signal sequences to be equivalent if they have the same stationary statistics. What is obtained is a method similar to compressive sensing, but involving stationary-statistics generative-projection sensing, where the signal processing is non-lossy at the level of stationary statistics equivalence. In the SCW signal analysis the signal source is generative in that it is describable via use of a hidden Markov model, and the HMM's Viterbi-derived generative projections are used to describe the sparse components contributing to the signal source. In SCW encoding the modulation of stationary statistics can be man-made or natural, with the latter in many experimental situations that involve flow phenomologies that have stationary statistics. If the signal is man-made, usually the underlying stochastic process is still a natural source, where

it is the changes in the stationary statistics that is under the control of the man-made encoding scheme. Transmission and reception are then followed by generative projection via Viterbi-HMM template matching or via Viterbi-HMM feature extraction followed by separate classification (using SVM). So in the SCW approach the encoding is even 'dumber' in that it can be any noise source with stationary statistics (the case for many experimental observations), with stationary statistics phase modulation for encoding. The decoding must be even 'smarter', on the other hand, in that generalized Viterbi algorithms are used to perform a generative projection (and possibly other machine learning methods as well, SVMs in particular). An example of the stationary statistics sensing with a machine learning based decoder is described in application to channel current cheminformatics studies in what follows.

3.5 Generalized HMM Algorithms

3.5.1 The Meta-HMM – a clique-generalized HMM

The traditional HMM assumes that a 1st order Markov property holds among the states and that each observable depends only on the corresponding state and not any other observable. The meta-HMM entails a maximally-interpolated departure from that convention (limited according to the size of the training dataset) in an attempt to leverage anomalous statistical information in the neighborhood of non-self state transitions. The regions of anomalous statistics are often highly structured, having consensus sequences that strongly depart from the strong independence assumptions of the 1st order HMM. The existence of such consensus sequences suggests that we adopt an observation model that has a higher order Markov property with respect to the observations. Furthermore, since the consensus sequences vary by the type of transition, this observational Markov order should be allowed to vary depending on the state.

The gap and hash interpolating Markov Models (gIMM and hIMM) [4] can be directly incorporated into meta-HMMBD gene-finding models as a further enhancement to the underlying Markov models, since they are already known to extract additional information that may prove useful, particularly in the zone-dependent emission regions (denoted 'zde's as in [4]) where promoters and other gapped motifs might exist. Promoters and transcription factor binding sites often have lengthy overall gapped motif structure, and with the hash-interpolated Markov models it is also possible to capture the conserved higher order sequence information in the zde sample space. The hIMM and gIMM methods, thus, will not only strengthen the gene structure recognition, but can also provide the initial indications of anomalous motif structure in the regions identified by a gene-finder (in a post-genomic phase of the analysis) [4].

By viewing state transitions, such as e_0e_1 or e_0i_0, as transition "dimer states", or as two-element "footprint" states, we begin to shift to a meta-HMM footing where we can model emissions more accurately. For the footprint states introduced in what follows, a critical assumption is made – ***at most, one non-self transition is allowed per footprint transition***. This assumption is a equivalent to a minimum length constraint on regions of self-transitions to be footprint size or greater. For genomic applications this is not a problematic constraint, and when a concern, different 'gene-scans' can always be performed with different footprint sizes.

When encountered sequentially in the Viterbi algorithm, the sequence of (single) non-self state transition 'dominated' *footprint* states would conceivably score highly when computed for the footprint-width number of footprint-states that overlap the non-self transition. In other words, we can expect a *natural boosting* effect for the correct prediction at such non-self transitions

(compared to the standard HMM). To describe bases in the irreducible joint probability we have: $w_n = b_{n-L+1}, \ldots, b_n, \ldots, b_{n+R}$, and $\widetilde{w}_n = b_{n-L+1}, \ldots, b_n, \ldots, b_{n+R-1}$ describes the base observations, while $s_n = \lambda_n\lambda_{n+1}$ (dimer states, length in λ's =2), and $f_n = s_{n-\ell+1}, \ldots, s_{n+r} \cong \lambda_{n-\ell+1}, \ldots, \lambda_n, \ldots, \lambda_{n+r+1}$ (footprint state, length in s's= ℓ+r), describes the associated labels. Given the above, the clique-factorized HMM is as follows:

$$P(B, \Lambda) = P(w_{-R}, f_{-R}) \{ \mathbf{\Pi}_{n=-R+1}^{N+L-2} [P(w_n, f_{n-1}, f_n) / P(\widetilde{w}_n, f_{n-1})] \},$$

with appropriate boundary terms (see [9]). A generalization to the Viterbi algorithm can be directly implemented, using the above clique-factorized HMM form [9], to establish an efficient dynamic programming table construction. Generalized expressions for the Baum-Welch algorithm are also possible. Some of the generalizations are straightforward extensions of the algorithms from 1st order theory with its minimal clique. Sequence-dependent transition properties in the generalized-clique formalism, however, have no counterpart in the standard 1st Order HMM formalism.

The core term in the clique-factorization can also be written by introducing a Bayesian parameter, one that happens to provide a matching joint probability construct (to the extent possible) with the term in the numerator:

$$\rho = \frac{P(w_n, f_{n-1}, f_n)}{P(\widetilde{w}_n, f_{n-1})} = \frac{P(w_n, f_{n-1}, f_n)}{\sum_{f'_{n(allowed)}} P(\widetilde{w}_n, f_{n-1}, f'_n)} = \frac{P(w_n|f_{n-1}, f_n)\, P(f_n|f_{n-1})\, P(f_{n-1})}{\sum_{f'_n} P(\widetilde{w}_n|f_{n-1}, f'_n)\, P(f'_n|f_{n-1})\, P(f_{n-1})}$$

In the above expression we clearly have sequence dependent transitions. For $f_{n-1} = ii,$ $and\ f_n = ie$ for example, we have:

$$\rho|_{GCHMM} = \frac{P(w_n, f_{n-1}, f_n)}{P(\widetilde{w}_n, f_{n-1})}\Big|_{\substack{f_{n-1}=ii,\\ f_n=ie}} = \frac{P(w_n|ie)\, P(ie|ii)}{P(\widetilde{w}_n|ie)P(ie|ii)+P(\widetilde{w}_n|ii)P(ii|ii)} = \frac{P(b_{n+R}|\widetilde{w}_n, ie)P(ie|ii)}{P(ie|ii)+P(ii|ii)\left(\frac{P(\widetilde{w}_n|ii)}{P(\widetilde{w}_n|ie)}\right)}$$

Use of the meta-HMM formalism resolves complications due to heavy-tail duration distributions and weak contrast. This is a new HMM modeling capability. The form of the clique factorization in [9] also has LLR terms such as $P(\widetilde{w}_n|ie)/P(\widetilde{w}_n|ii)$ that allow for a simple switch from internal scalar-based state discriminant to a vector-based feature, allowing for a similar substitution of a discriminant based on an SVM as demonstrated for splice sites in [5] and described in the pMM/SVM sub-section. These alternate representations do not introduce any significant increase in computational time complexity.

3.5.2 Hidden Semi-Markov model and HMM-with-duration

In the standard HMM, when a state i is entered, that state is occupied for a period of time, via self-transitions, until transiting to another state j. If the state interval is given as d, the standard HMM description of the probability distribution on state intervals is implicitly given by:

$$p_i(d) = a_{ii}^{d-1}(1 - a_{ii})$$

where a_{ii} is self-transition probability of state i. As mentioned previously, this geometric distribution is inappropriate in many cases. The standard HMMD replaces the equation above with a $\boldsymbol{p_i(d)}$ that models the real duration distribution of state i. In this way explicit knowledge about the duration of states is incorporated into the HMM. When entered, state i will have a duration of d according to its duration density $\boldsymbol{p_i(d)}$; it then transits to another state j according to

the state transition probability a_{ij} (self-transitions, a_{ii}, are not permitted in this formalism). It is easy to see that the HMMD will turn into a HMM if $p_i(d)$ is set to the geometric distribution shown above. The first HMMD formulation was studied by Ferguson [13]. A detailed HMMD description was later given by [14]. There have been many efforts to improve the computational efficiency of the HMMD formulation given its fundamental utility in many endeavors in science and engineering. Notable amongst these are the variable transition HMM methods for implementing the Viterbi algorithm introduced in [15], and the hidden semi-Markov model (HSMM) implementations of the forward-backward algorithm [16].

In [10] it is shown how to 'lift' side information that is associated with a region, or transition between regions, by 'piggybacking' that side information along with the duration side information. We use, as example, HMM incorporation of duration itself as the guide in what follows. In doing so, we arrive at a hidden semi-Markov model formalism for a HMMD. An equivalent formulation of the HSMM was introduced in [15] for the Viterbi algorithm and in [16] for Baum-Welch. In these derivations, however, the maximum-interval constraint is still present (comparisons of these methods were subsequently detailed in [17]). Other HMM generalizations include Factorial HMMs [18] and hierarchical HMMs [19]. For the latter, inference computations scaled as $O(T^3)$ in the original description, and havesince been improved to O(T) by [20].

The HSMM formalism introduced here, however, is directly amenable to incorporation of side-information and to adaptive speedup (as described in [8,10] and in Sec. 3.3). For the state duration density $p_i(x = d)$, $1 \le x \le D$, we have:

$$p_i(x = d) = p_i(x \ge 1) \cdot \frac{p_i(x \ge 2)}{p_i(x \ge 1)} \cdot \frac{p_i(x \ge 3)}{p_i(x \ge 2)} \cdots \frac{p_i(x \ge d)}{p_i(x \ge d-1)} \cdot \frac{p_i(x = d)}{p_i(x \ge d)}$$

where $p_i(x = d)$ is abbreviated as $p_i(d)$ if no ambiguity. Define "self-transition" variable $s_i(d)$ = probability that next state is still $\lambda_t = i$, given that i has consecutively occurred d times up to now.

$$p_i(x = d) = \left[\prod_{j=1}^{d-1} s_i(j)\right]\left(1 - s_i(d)\right), where\ s_i(d) = \begin{cases} \frac{p_i(x \ge d+1)}{p_i(x \ge d)}\ if\ 1 \le d \le D-1 \\ 0\ if\ d = D \end{cases}$$

We see with comparison of the equation for $p_i(d)$ above and $p_i(d) = (a_{ii})^{d-1}(1 - a_{ii})$, that we now have similar form, there are 'd-1' factors of 's' instead of 'a', with a 'cap term' '(1-s)' instead of '(1-a)', where the 's' terms are not constant, but only depend on the state's duration probability distribution. In this way, each 's' can mesh with the HMM's dynamic programming table construction for the Viterbi algorithm at the column-level in the same manner that 'a' does. Side-information about the local strength of EST matches or homology matches, etc., that can be put in similar form, can now be 'lifted' into the HMM model on a proper, locally optimized Viterbi-path. The derivations of the Baum-Welch and Viterbi HSMM algorithm is in [10].

The memory complexity of this method is O(TN). No forward table needs to be saved. The computation complexity is $O(TN^2+TND)$. In an actual implementation, a scaling procedure may be needed to keep the forward-backward variables within a manageable numerical interval. One common method is to rescale the forward-backward variables at every time index t using the scaling factor $c_t = \Sigma_i f_t(i)$. Here we use a dynamic scaling approach. For this we need two

versions of θ(k, i, d). Then at every time index, we test if the numerical values is too small, if so, we use the scaled version to push the numerical values up; if not, we keep using the unscaled version. In this way, no additional computation complexity is introduced by scaling.

As with Baum-Welch, the Viterbi algorithm for the HMMD is $O(TN^2+TND)$. Because logarithm scaling can be performed for Viterbi in advance, however, the Viterbi procedure consists only of additions to yield a very fast computation. For both the Baum-Welch and Viterbi algorithms, use of the HMMBD algorithm [8] can be employed (as in this work) to further reduce computational time complexity to $O(TN^2)$, thus obtaining the speed benefits of a simple HMM, with the improved modeling capabilities of the HMMD.

3.5.3 HMMD with binned duration

The intuition guiding the HMMBD approach is that the standard HMM already does the desired duration modeling when the distribution modeled is geometric, suggesting that, with sufficient effort, a self-tuning explicit HMMD might be possible to achieve HMMD modeling capabilities at HMM computational complexity in an adaptive context.

The duration distribution of state i consists of rapidly changing probability regions (with small change in duration) and slowly changing probability regions. In the standard HMMD all regions share an equal computation resource (represented as D substates of a given state) -- this can be very inefficient in practice. In this section, we describe a way to recover computational resources, during the training process, from the slowly changing probability regions. As a result, the computation complexity can be reduced to $O(TN^2+TND^*)$, where D* is the number of "bins" used to represent the final, coarse-grained, probability distribution. A "bin" of a state is a group of substates with consecutive duration. For example, ***f(i, d)***, ***f(i, d+1)***, ..***f (i, d+δd)*** can be grouped into one bin. The bin size is a measure of the granularity of the evolving length distribution approximation. A fine-granularity is retained in the active regions, perhaps with only one length state per bin, while a coarse-granularity is adopted in weakly changing regions, with possibly hundreds of length states per bin. An important generalization to the exact, standard, length-truncated, HMMD is suggested for handling long duration state intervals – a "tail bin". Such a bin is strongly indicated for good modeling on certain important distributions, such as the long-tailed distributions often found in nature, the exon and intron interval distributions found in gene-structure modeling in particular. In practice, the idea is to run the exact HMMD on a small portion, δT, of the training data, at O(δTNN + δTND) cost, to get an initial estimate of the state interval distributions. Some preliminary course-graining is then performed, where strongly indicated, and the number of bins representing the length distribution is reduced from D to D′. The exact HMMD is then performed on the D′ substate model for another small portion of the training data, at computational expense O(δTNN + δTND′). This is repeated until the number of bin states, D*, reduces no further, and the bulk of the training then commences with the D* bin-states length distribution model at expense $O(TN^2+TND^*)$. The key to this process is the retention of training information during the 'freezing out' of length distribution states, and such that the D* bin state training process can be done at expense $O(TN^2+TND^*) \approx O(TN^2)$, which is the same complexity class as the standard HMM itself. Starting from the above binning idea, for substates in the same bin, a reasonable approximation is applied:

$$\sum_{d'=d}^{d+\delta_d} f_t(i,d')\theta(b_t,i,d') \;=\; \theta\left(b_t,i,\bar{d}\right)\sum_{d'=d}^{d+\delta_d} f_t(i,d'),$$

where $\bar{d}$ is the duration representative for all substates in this bin.

3.5.4 Adaptive null-state binning for O(TN) computation

During the HMM Viterbi table construction for each of T sequence data values, there is a column entry, and for each of N states there is a row. At each column the HMM Viterbi algorithm must look to the past column entries as it populates the table from left to right, thus leading to an $O(TN^2)$ computation. If we establish an adaptive binning capability, reminiscent of what was done with the HMMBD method, then we can keep track of lists with respect to each state that correspond to prior column transitions to that state. If we, in particular, track those Viterbi most-probable-paths that arrive at our state cell with probability below some cutoff (with respect to the other probabilities arriving at that cell), we can ignore transitions from such cells in later column computations. What results is an initial $O(tN^2)$ ($t<<T$) computation to learn the state lists for above cut-off transitions (suppose K on average), followed by the main body of the O(TNK) computation (with $K<<N$).

A method is also possible comprising use of a "fastViterbi" process where $O(TN^2) \rightarrow O(TmN)$ via learned, local, max-path ordering in a given column of the Viterbi computation for the highest 'm' values. Subsequent columns first only examine the top 'm' max-paths and if their ordering is retained, and their total probability advanced sufficiently, then the other states remain 'frozen-out' with a large grouping (binning) on the probabilities on those states used to maintain their probability information (and correct normalization summing) when going forward column-by column, with reset to full column evaluation on the individual state level when the m values fall out of their initially identified ordering.

A method is possible comprising use of a fastViterbi with null-binning process where $O(TN^2) \rightarrow O(Tmn) \rightarrow O(T)$ via learned global and local aspects of the data as indicated above. This approach offers significant utility as a purely HMM-based alignment algorithm that may outperform BLAST (Basic local alignment search tool. Altschul SF et al. 1990) with comparable time complexity.

3.6 NTD 'Binary' event communication is a form of stochastic 'phase' modulation (SPM)

In the nanopore transduction detector (NTD) experiments [1,21], the molecular dynamics of a (single) captured transducer molecule provide a unique stochastic reference signal with stable statistics on the observed, single-molecule blockaded, channel current, somewhat analogous to a carrier signal in standard electrical engineering signal analysis. Discernible changes in blockade statistics, coupled to SSA signal processing protocols, enable the means for a highly detailed characterization of the interactions of the transducer molecule with binding targets (cognates) in the surrounding (extra-channel) environment.

The transducer molecule is specifically engineered to generate distinct signals depending on its interaction with the target molecule. Statistical models are trained for each binding mode, bound and unbound, for example, by exposing the transducer molecule to zero or high (excess) concentrations of the target molecule. The transducer molecule is engineered so that these different binding states generate distinct signals with high resolution. Once the signals are characterized, the information can be used in a real-time setting to determine if trace amounts of the target are present in a sample through a serial, high-frequency sampling, and pattern recognition, process.

Thus, in NTD applications of the SSA Protocol, due to the molecular dynamics of the captured transducer molecule, a unique reference signal with stationary (or approximately stationary)

statistics is engineered to be generated during transducer blockade, analogous to a carrier signal in standard electrical engineering signal analysis. The adaptive SSA machine learning algorithms for real-time analysis of the stochastic signal generated by the transducer molecule offer a "lock and key" level of signal discrimination. The heart of the signal processing algorithm is an adaptive Hidden Markov Model (AHMM) based feature extraction method, implemented on a distributed processing platform for real-time operation. For real-time processing, the AHMM is used for feature extraction on channel blockade current data, while classification and clustering analysis are implemented using a Support Vector Machine. In addition, the design of the machine learning based algorithms allow for scaling to large datasets, real-time distributed processing, and are adaptable to analysis on any channel-based dataset, including resolving signals for different nanopore substrates (e.g. solid state configurations) or for systems based on translocation technology. The machine learning software has also been integrated into the nanopore detector for "real-time" pattern-recognition informed (PRI) feedback [22,23] (see Fig. 7). The methods used to implement the PRI feedback include *distributed* HMM and SVM implementations, which enable the processing speedup that is needed.

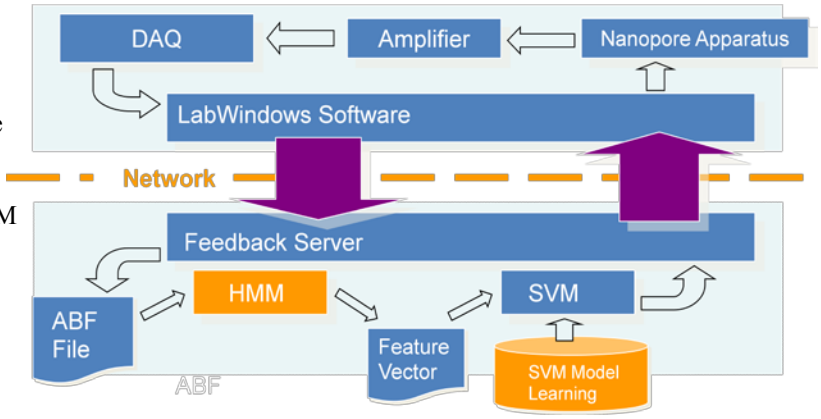


Figure 7. PRI Sampling Control (see [23] for specific details). Labwindows Feedback Server Architecture with Distributed CCC processing. The HMM learning (on-line) and SVM learning (off-line), denoted in orange, are network distributed for N-fold speed-up, where N is the number of computational threads in the cluster network.

A mixture of two DNA hairpin species (denoted {9TA, 9GC} in [2]) is examined in an experimental test of the PRI system [23]. In separate experiments, data is gathered for the 9TA and 9GC blockades in order to have known examples to train the SVM pattern recognition software. A nanopore experiment is then run with a 1:70 mix of 9GC:9TA, with the goal to eject 9TA signals as soon as they are identified, while keeping the 9GC's for a full 5 seconds (when possible, sometimes a channel-dissociation or melting event can occur in less than that time). The results showing the successful operation of the PRI system is shown in Fig. 8 as a 4D plot, where the radius of the event 'points' corresponds to the duration of the signal blockade (the 4th dimension). The result in Fig. 8 demonstrates an approximately 50-fold speedup on data acquisition of the desired minority species.

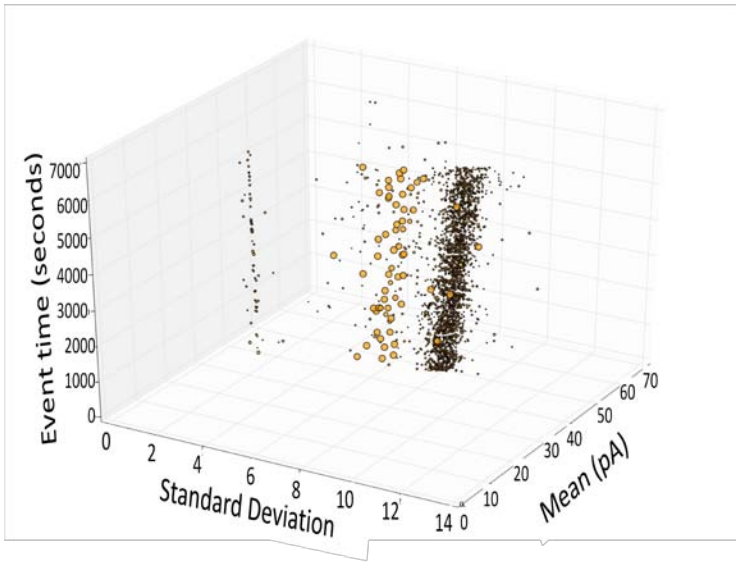


Fig. 8. PRI Mixture Clustering Test with 4D plot [23]. The vertical axis is the event observation time, and the plotted points correspond to the standard deviation and mean values for the event observed at the indicated event time. The radius of the points correspond to the duration of the corresponding signal blockade (the 4th dimension). Three blockade clusters appear as the three vertical trajectories. The abundant 9TA events appear as the thick band of small-diameter (short duration, ~100ms) blockade events. The 1:70 rarer 9GC events appear as the band of large-diameter (long duration, ~ 5s) blockade events. The third, very small, blockade class corresponds to blockades that partially thread and almost entirely blockade the channel.

3.7 Generalized SVM with novel kernels

The SVM implementations described involve SVM algorithmic variants, kernel variants, and chunking variants; SVM classification tuning metaheuristics; and SVM clustering metaheuristics. The SVM training metaheuristics enable use of the SVM's confidence parameter to bootstrap from a strong classification engine to a strong clustering engine via use of label changes, and repeated SVM training processes with the new label information obtained.

SVM Methods and Systems are given for classification, clustering, stochastic sequential analysis and nanopore transduction detection, with a broad range of applications: sequential-structure identification, pattern recognition, knowledge discovery, Bioinformatics, Nanopore Detector Cheminformatics, the nanopore transduction detection Nanoscope, and computational engineering with information flows using stochastic sequential analysis tools.

In the Results we show a number of SVM classification algorithms and new SVM/metaheuristics bootstrap algorithms for clustering. SVMs are fast, easily trained, discriminators, for which strong discrimination is possible without over-fitting complications. SVMs are firmly grounded as variational-calculus based optimization methods that are constrained to have structural risk minimization (SRM), unlike neural net classifiers, such that they provide noise tolerant solutions for pattern recognition. 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 shown in Fig. 9. 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 be a very robust platform and, from initial results shown here, promises to be one of the best clustering approaches.

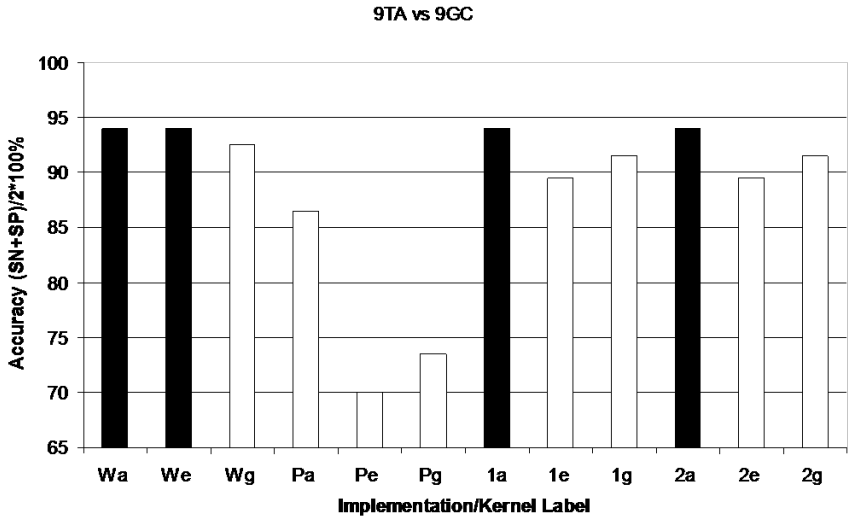


Fig. 9. Comparative results are shown on performance of Kernels and algorithmic variants. The classification is between two DNA hairpins (in terms of features from the blockade signals they produce when occluding ion flow through a nanometer-scale channel). Implementations: WH SMO (W); Platt SMO (P); Keerthi1 (1); and Keerthi2 (2). Kernels: Absdiff (a); Entropic (e); and Gaussian (g). The best algorithm/kernel on this and 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).

3.8 SVM Tuning

Our objective is to establish an automated tuning solution for SVM classification over a variety of novel kernel and algorithmic parameters. In Proof-of-Concept work, this has been done by implementing a genetic algorithm tuning procedure, where SVM performance on training data is used to define a fitness function. In initial efforts with genetic algorithm tuning (in analysis of channel current data), the genetic algorithm tuning results were as good as or better than those obtained by an expert manually. Alternative, easily distributed, tuning approaches will probably work well also, and are considered as needed in ongoing efforts, and include ACO, and other multi-agent *distributed* intelligence approaches (see descriptions in the next section).

3.8.1 Simple heuristics for SVM Tuning

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 at roughly 10 kernel parameter setting (where each 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, or more extensive use of GPU capabilities (still need to eventually do chunking). Chunking algorithms will be shown to be effective, but susceptible to training-failure pathologies if certain safeguards aren't observed, as will be discussed.

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. More directed tuning paradigms typically involve simulated annealing in this setting (to be shown later). 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 amazing success (but not shown in what follows). These more sophisticated tuning methods may not always be necessary in the SVM classification applications, but will 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 important role.

3.8.2 Metaheuristics – Parameter Search and Tuning

Simulated Annealing

Our ability to assess a score, or 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 larger problem of the simple globalization algorithm, via random restart: if the fitness landscape or configuration space is too large random restart won't offer a solution, even if it can, in a reasonable amount of time. This is where more clever metaheuristics are involved, to extend to a global optimization algorithm.

Global Optimization

One of the weaknesses of the brute force random restart approach mentioned so far is that the 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 with a Boltzmann factor for regularization, and in doing so we arrive at the Simulated Annealing approach:

Population-based metaheuristics

In seeking global optimization metaheuristics we are starting to see more sophisticated configuration selection at the component level, on the one hand, and at the population level, on the other hand, especially as we work with the probabilistic simulated annealing approach and the history-based taboo approach, especially with the component-based versions of the latter. This is because the population and history aspects point to a general metaheuristics that operates on populations of configurations (or populations of 'agents' that interact as intermediaries to determining a configuration selection). The notion of 'history' also must address the conveyance of this information or 'artefact'. In the case of ACO, described in what follows, this will be via stygmergy.

Population with evolution
A fixed-size (or size otherwise constrained) population of configurations can have a birth/death cycle or be static. If it has a birth/death cycle, one popular method is the evolutionary computation approach (Darwinian evolution; asexual reproduction):

Metaheuristic: Evolutionary Optimization (Darwinian Evolution; asexual reproduction)
Starting population of parent configurations ("parents") undergoes initial selection according to cut-off (truncation selection) that is chosen. Those surviving produce offspring, typically via a simple configuration tweak mutation, and those child configurations ("children") are then added to the pool of the current population. Repeat.

Depending on algorithm, the parents may be selected against generationally also (such as for salmon). The reproduction step can be done by the population all at once (generationally), as described here, or individually, out-of-phase, as commonly done with the GA's given next.

Metaheuristic: Genetic Algorithm (Darwinian Evolution; sexual reproduction)
Starting population of parent configurations ("parents") undergoes initial selection according to cut-off (truncation selection) that is chosen.
Those surviving produce offspring, typically via both simple configuration tweak mutation and non-local configuration component-level swapping, and those child configurations ("children") are then added to the pool of the current population.
Repeat.

Both of the evolutionary algorithms have tuning parameters in their manner of iteration that shift between less reliance on what has been learned (random search) and strong use of existing information (highly directed gradient ascent). These parameters can be summarized as follows:

Populations with interactions (and sub-populations, e.g., speciation)
Once we consider that the configurations in a population may interact with one another, we have a situation where different sub-populations may be given (and now not trivially de-couple), i.e., speciation is possible in the evolutionary population. From there whole ecologies of evolutionary complexity can be developed.

Population with swarm intelligence
With population-based interactions have possible direct coordination between agents (sexual reproduction providing cross-over mutation in GA's, as mentioned above, and swarm activity that provides global information to all agents with action defined accordingly to desired local and global swarm behavior, as defined in what follows:
Metaheuristic: Particle swarm optimization (PSO) (Lamarckian Evolution)
Particle swarm optimization (PSO) also takes its cue from Biology, but not from evolutionary model, but from a swarm model. Here the population is static (the other case than the birth/death cycle case), and there is no selection of any kind. Now the configurations in the population are themselves directly tweaked in response to new information obtained. This is a form of directed mutation and is part of a Lamarckian evolutionary paradigm. The configurations are often viewed as describing particles in a space and the configurations undergo directed mutation, 'motion', in the configuration space, with motion towards the best known configuration, where

three levels of knowledge are weighed in the balance: (i) the fittest configuration ascertained by a particular during its history; (ii) the fittest configuration ascertained by the informants of a particular particle (often just a randomly chosen set of particles); and (iii) the fittest configuration discovered by any particle.

3.9 SVM Clustering

The SVM-based clustering method (Fig. 10) makes use of the SVM-classifier convergence process. Single-convergence initialized clustering methods, involving label-flipping between SVM convergence training runs, have been studied previously and will be described in the Background Section. The single-convergence methods outperform other methods on the test sets considered, but 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 [24]. A different approach is to initialize with information from *multiple* SVM convergences, with selection on training data based on algorithmic methods that leverage the clustering groupings indicated. This can be done to more effectively cluster, or cluster with less sophistication, and initial efforts along these lines have been very promising.

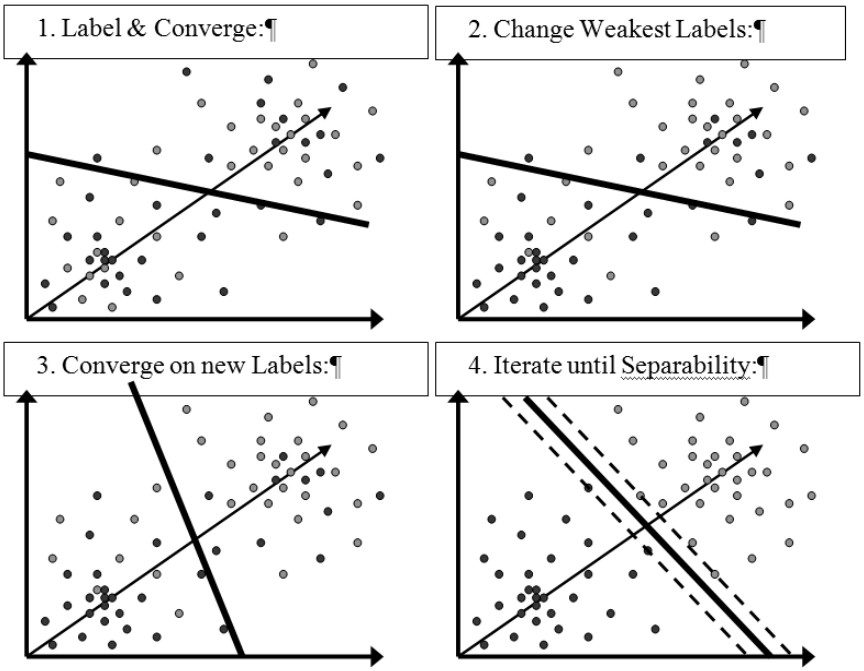


Fig. 10. SVM-

4 Results

4.1 Finite State Automaton Results

4.1.1 Holistc Tuning

The FSA in Fig. 1 enables acquisition of localizable channel current signals using 'holistic' tuning and 'emergent grammar' tuning. This is an example of how a "holistic engine" of multiply connected variables/states/interactions can be used to acquire localizable signals. Here, the implementation of the engine entails a finite-state automaton (FSA), where running-time scales as O(L), where L is the length of the sequence data (i.e., the same time complexity as simply making a copy of the data). For acquisition we seek minimal feature identification comprising identification of signal beginnings and ends (and thus durations as well). Holistic tuning is done by testing global features, such as the signal acquisition numbers, as different parameters are considered over a reasonable range of values (see Fig. 11). Part of the trick with holistic tuning is the design of the initial tuning state, e.g., multiple parameters must be within their 'lock range' on tuning parameters analogous to the PLL lock-range constraint.

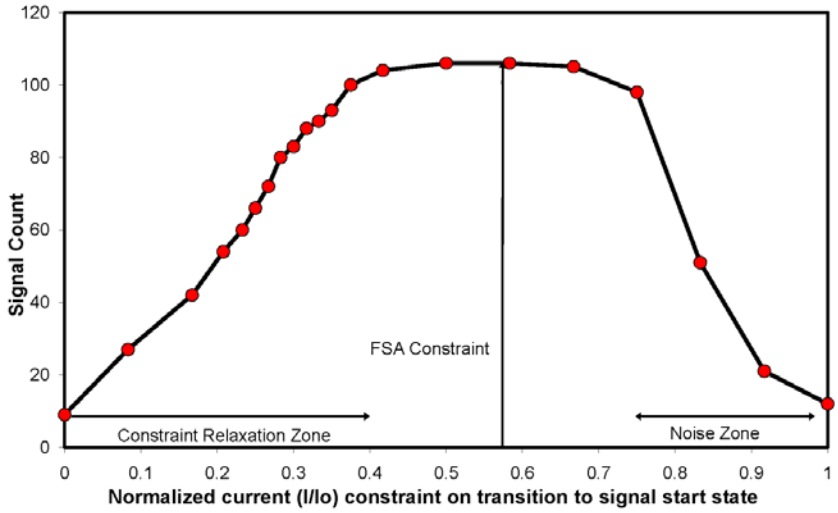


Fig.11. Tuning on 'start_drop_value for a collection of DNA hairpins with 6 base-pair stem length.

4.1.2 Other tuning methods

Emergent grammar tuning is described in [2,22], where it is used to establish the lowpass filtering cut-off. As the name suggests, a collection of states are defines and a sequence of those states is observed. As the state defintisions are broadened, such as with bin-merging on binned emission values, then a grammar may emerge from from the sequence of observations. Once a lock-range is established on an emergent grammar structure, other tuning can be performed.

Emergent phenomenology tuning is akin to emergent grammar tuning, but now the tuning over state definitions, or groupings, is used to encompass a range of phenomenological models and SCW information flows (rather than grammar models). In some respects this is the underlying approach that has provided the 'laws' of physics, where the scientific method drives the tuning process. In the context of signal analysis, however, this is a very weakly informed situation, a method of last resort unless some specific phenomenomly is known to be involved and the tuning can be constrained appropriately. Due to the latter, practical usage limitation, the emergent phenomenology tuning approach is also referred to as the DarkStar Approach -- named after 1974 film Dark Star that ends with crew trying to teach an AI superbomb phenomenology in order to defuse it, the AI eventually learns a phenomenology, but not the one desired, and

explodes. Same problem here, the approach may identify a structure, but obtaining one considered relevant or desirable is another problem.

4.1.3 Spike detection

The spike detector software is designed to count "anomalous" spikes, i.e., spike noise not attributable to the gaussian fluctuations about the mean of the dominant blockade-level. Spike count plots are generated to show increasing counts as cut-off thresholds are relaxed (to where eventually any downward deflection will be counted as a spike). The plots are automatically generated and automatically fit with extrapolations of their linear phases (exponential phases occur when cut-offs begin to probe the noise band of a blockade state – typically gaussian noise "tails"). The extrapolations provide an estimate of "true" anomalous spike counts.

In Fig. 12 are shown the automatically generated plot of spike characteristics for blockade data when DNA hairpins were examined, one radiated and one not. The plots are automatically generated and automatically fit with extrapolations of their linear phases (exponential phases occur when cut-offs begin to probe the noise band of a blockade state – typically gaussian noise "tails"). The extrapolations provide a stable, "robust", estimate of anomalous spike counts. By this method, the non-radiated DNA exhibited a full-blockade "spike" from its lower-level blockade with a frequency of 5 spikes per second (indicating a fraying of the blunt ended terminus of the molecule at that rate). For the radiated molecule the frequency of spikes was 15 spikes per second, indicating a much greater fraying rate (dissociation of the terminal base-pair), consistent with that molecule being weakened by radiation such that its terminal base-pair frays more frequently.

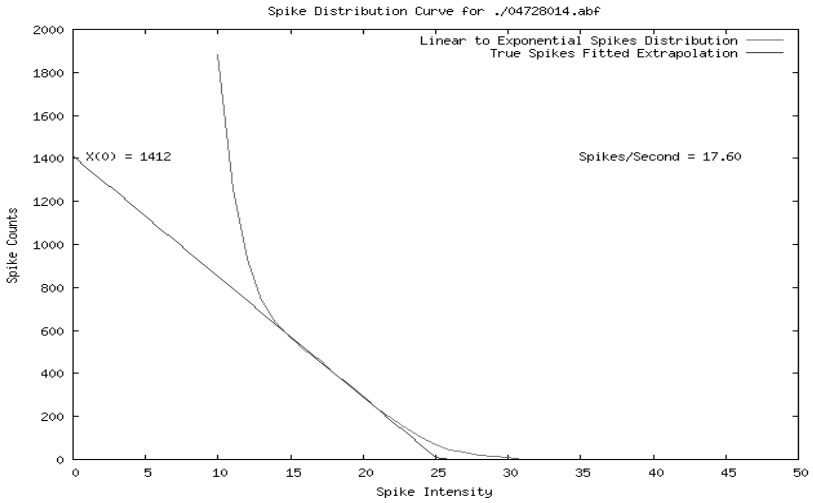


Fig. 12. The time-domain FSA shown in Fig. 1 is used to extract fast time-domain features, such as "spike" blockade events. Automatically generated "spike" profiles are created in this process. One such plot is shown here for a radiated 9 base-pair hairpin, with a fraying rate indicated by the spike events per second (from the lower level sub-blockade). Results: the radiated molecule has more "spikes" which are associated with more frequent "fraying" of the hairpin terminus--the radiated molecules were observed with 17.6 spike events per second resident in the lower sub-level blockade.

4.2 Generalized HMM Results

4.2.1 The SSA Protocol

The sequence of algorithmic methods used in the SSA Protocol comprise a weakness recovery protocol: (i) the weakness in the FSA methods will be shown to be its difficulty in strong structure identification (especially non-local structure), for which HMM methods (and tuning metaheuristics) are the solution. (ii) for the HMM, in turn, the main weakness is in local sensing 'classification', and as a classification problem, the problem can be solved via incorporation of generalized SVM methods. If facing a classification task independent of a prior signal processing need, the SVM will also be the method of choice in what follows. (iii) The weakness of the SVM, whether used for classification or clustering, but especially for the latter, is the need to optimize over algorithmic, model (kernel), chunking, and other process parameters during learning. This is solved via use of metaheuristics for optimization such as simulated annealing, genetic algorithm optimization, particle swarm optimization, etc. (iv) The main weaknesses in the metaheuristic effort is partly resolved via use of the "front-end" methods, like the FSA, and partly resolved by a knowledge discovery process (that can be done using the SVM clustering methods). The weakness recovery protocol, thus, comes full circle, thereby establishing a robust signal processing platform.

The SSA Protocol tries to associate acquisition, feature extraction, classification, and clustering tasks with their most appropriate machine learning method, given the data, the noise properties, the operational time-constraints, and other constraints involved. Since data processing is often encountered in stages, the decomposition described in what follows is in terms of stages for acquisition, feature extraction, classification, and clustering, but the methods can have more complex seqneces of operation or embedded operation in another method, etc., as is described, to some extent, in what follows as well.

(Stage 1) primitive feature identification: this stage is typically finite-state automaton based, with feature identification comprising identification of signal regions (critically, their beginnings and ends), and, as-needed, identification of sharply localizable 'spike' behavior in any parameter of the 'complete' (non-lossy, reversibly transformable) classic EE signal representation domains: raw time-domain, Fourier transform domain, wavelet domain, etc. (The methodology for spike detection is shown applied to the time-domain in [3].) Primitive feature extraction can be operated in two modes: off-line, typically for batch learning and tuning on signal features and acquisition; and on-line, typically for the overall signal acquisition (with acquisition parameters set – e.g., no tuning), and, if needed, 'spike' feature acquisition(s).

The FSA method that is primarily used in the channel current cheminformatics signal discovery and acquisition is to identify signal-regions in terms of their having a valid 'start' and a valid 'end', with internal information to the hypothesized signal region consisting, minimally, of the duration of that signal (e.g., the duration between the hypothesized valid 'end and hypothesized valid 'start'). One approach along these lines is a signal 'fishing' protocol " …constraints on valid 'starts' that are weak (with prominent use of 'OR' conjugation) and constraints on valid 'ends' that are strong (with prominent use of 'AND' conjugation)." We underpin our approach to signal analysis in a fundamentally different way, however, although the signal fishing method indicated above is still used as needed. The FSA signal analysis methodology used here, for example, involves identifying anomalously long-duration regions. Identification of anomalously-

long duration regions in the more sophisticated Hidden Markov model representation would require use of a HMM-with-duration to not lose the information on the anomalous durations, which is one of the application areas for the HMMBD method (described in the Methods).

Once identification rules, often threshold-based, are established for the signal starts and signal ends, then those definitions can be explored/used in signal acquisition. As those definitions are tuned over, by exploring the different signal acquisition results obtained with different parameter settings, the signal acquisition counts can undergo radical phase transitions, providing the most rudimentary of the holistic tuning methods on the primitive feature acquisition FSA. By examining those phase transitions, and the stable regimes in the signal counts (and other attributes in more involved holistic tuning), the recognition of good parameter regimes for accurate acquisition of signal can be obtained. As more internal signal structure is modeled by the FSA, the holistic tuning can involve more sophisticated tuning recognition of emergent grammars on the signal sub-states. The end-result of the tuning is a signal acquisition FSA that can operate in an on-line setting, and very efficiently (computation on the same order as simply reading the sequence) in performing acquisition on the class of signals it has been 'trained' to recognize. On-line learning is possible via periodic updates on the batch learning state/tuning process. For typical SSA (and CCC) applications, the tFSA is used to recognize and acquire 'blockade' events (which have clearly defined start and stop transitions).

A computationally 'expensive' HMM signal acquisition at Stage 1 may be necessary for very weak signals, for example, if the typical Stage 1 methods fail. In this situation the HMM will probably have a very weak signal differential on the different signal classes if it were to attempt direct classification (and eliminate the need for a separate Stage 3). In this setting, the HMM would probably be run in the finest grayscale generic-state mode, with a number of passes with different window sample sizes to 'step through' the sequence to be analyzed. Then, there are two ways to proceed: (1) with a supervised learning 'bias', where windows on one side of a 'cut' are one class, and those on the other side the other class, can the SVM classify at high accuracy on train/test with the labeled data so indicated. If so, a transition is identified. In (2) the idea is to use an unsupervised learning SVM-based clustering method where we look for a strong knife-edge split on clustered populations along the sequence of window samples. When this occurs, there is a strong identification of a transition. Since regions are identified (delineated) by their transition boundaries, we arrive at a minimally-informed means for state and state-transition discovery in stochastic sequential data involving HMM/SVM based channel current signal processing.

(Stage 2a) feature identification and feature selection: this stage in the signal processing protocol is typically Hidden Markov model based, where identified signal regions are examined using a fixed state HMM feature extractor or a template-HMM (states not fixed during template learning process where they learn to 'fit' to arrive at the best recognition on their train-data, the states then become fixed when the HMM-template is used on test data). The Stage 2 HMM methods are the central methodology/stage in the CCC protocol in that the other stages can be dropped or merged with the Stage 2 HMM in many incarnations. For example, in some data analysis situations, the Stage 1 methods could be totally eliminated in favor of the more accurate HMM-based approach to the problem, with signal states defined/explored in much the same setting, but with the optimized Viterbi path solution taken as the basis for the signal acquisition

structure identification. The reason this is not typically done is that the FSA methods sought in Stage 1 are usually only O(T) computational expense, where 'T' is the length of the stochastic sequential data that is to be examined. The typical HMM Viterbi algorithm, on the other hand, is $O(TN^2)$, where 'N' is the number of states in the HMM. Stage 1 provides a faster, and often more flexible, means to acquire signal, but it is more hands-on. If the core HMM/Viterbi method can be approximated such that it can run at O(TN) or even O(T) in certain data regimes, for example, then the non-HMM methods in stage 1 could be phased out. Such HMM approximation methods are described in the Methods, and present a data-dependent branching in the most efficient implementation of the protocol. If the data is sufficiently regular, direct tuning and regional approximation with HMM's may allow Stage 1 FSA methods to be avoided entirely. For general data, however, some tuning and signal acquisition according to Stage 1 will be needed (possibly off-line) if only to then bootstrap (accelerate) the learning task of the HMM approximation methods.

The HMM emission probabilities, transition probabilities, and Viterbi path sampled features, among other things, provide a rich set of data to draw from for feature extraction (to create 'feature vectors'). The choice of features is optimized according to the classification or clustering method that will make use of that feature information. In typical operation of the protocol, the feature vector information is classified using a Support Vector Machine. This is described in Stage 3 to follow. Once again, however, the Stage 3 classification could be totally eliminated in favor of the HMM's log likelihood ratio classification capability at Stage 2, for example, when a number of template HMMs are employed (one for each signal class). This classification approach is inherently weaker and slower than the (off-line trained) SVM methodology in many respects, but, depending on the data, there are circumstances where it may provide the best performing implementation of the protocol.

(Stage 2b) Stochastic carrier wave encoding/decoding
Using HMMBD, we have an efficient means to establish a new form of carrier-based communications where the carrier is not periodic but is stochastic, with stationary statistics. The HMMBD algorithmic methodology, [8], enables practical stochastic carrier wave encoding/decoding with this method.

Stochastic carrier wave signal processing is also encountered at the forefront of a number of efforts in nanotechnology, where it can result from establishing or injecting signal modulations so as to boost device sensitivity. The notion of modulations for effectively larger bandwidth and increased sensitivity is also described in [21]. Here we choose modulations that specifically evoke a signal type that can be modeled well with a HMMD but not with a HMM. This is a generally applicable approach where conventional, periodic, signal analysis methods will often fail. Nature at the single-molecule scale may not provide a periodic signal source, or allow for such, but may allow for a signal modulation that is stochastic with stationary statistics, as in the case of the nanopore transduction detector.

(Stage 3) classification: this stage is typically SVM based. SVMs are a robust classification method. If there are more classes to discern than two, the SVM can either be applied in a Decision Tree construction with binary-SVM classifiers at each node, or the SVM can internally represent the multiple classes, both are done in proof-of-concept experiments that are described.

Depending on the noise attributes of the data, one or the other approach may be optimal (or even achievable). Both methods are typically explored in tuning, for example, where a variety of kernels and kernel parameters are also chosen, as well as tuning on internal Karush-Kuhn-Tucker (KKT) handling protocols. Simulated annealing and genetic algorithms have been found to be useful in doing the tuning in an orderly, efficient, manner. If the feature vectors produced correspond to complete data information/profiling in some manner, which is explicitly the case in a probability feature vector representation on a complete set of signal event frequencies (where all the feature 'components' are positive and sum to 1), then kernels can be chosen that conform to evaluating a measure of distance between feature vectors in accordance with that notion of completeness (or internal constraint, such as with the probability vectors). Use of divergence kernels with probability feature vectors in proof-of-concept experiments have been found to work well with channel blockade analysis and is thought to convey the benefit of having a better pairing of kernel and feature vector, here the kernels have probability distribution measures (divergences), for example, and the feature vectors are (discrete) probability distributions.

(Stage 4) clustering: this stage is often not performed in the 'real-time' operational signal processing task, as it is more for knowledge discovery, structure identification, etc., although there are notable exceptions, one such being the jack-knife transition detection, via clustering consistency with a causal boundary. This stage can involve any standard clustering method, in a number of applications, but the best performing in the channel current analysis setting is often found to be an SVM-based external clustering approach (see [24]), which is doubly convenient when the learning phase ends because the SVM-based clustering solution can then be fixed as the supervised learning set for a SVM-based classifier (that is then used at the operational level).

4.2.2 Stochastic Carrier Wave Communications

The original description of an explicit HMMD required computation of order $O(TN^2+TND^2)$ [13] (where T is the sequence length to be examined, N is the number of states in the HMM/HMMD model, and D is the maximum duration length allowed in the HMMD model). The 'D^2' term made the original approach prohibitively computationally expensive in practical, real-time, operations, and introduced a severe maximum-duration constraint on the duration-distribution model. Improvements via hidden semi-Markov models to computations of order $O(TN^2+TND)$ are described in [15,16], where the maximum-interval constraint is still employed, and comparisons of these methods were subsequently detailed in [17]. In [8] we show that $O(TN^2+TND^*)$ is possible with the HMMBD algorithm, where D^* is the number of binned length states. The HMMBD implementation brings the HMMD modeling within the range of computational viability for many applications. In the HMMBD approach we also eliminate the maximum-duration constraint. We can often reduce to a bin representation with $D^*<10$, such that $D^*<<N$ in many situations, in which case that the HMMBD requires computations of order $O(TN^2)$, the same as for the HMM alone.

One important application of the HMM-with-duration method used in the CCC context [3,22] includes kinetic feature extraction from EVA projected channel current data (the HMM-with-Duration is shown to offer a critical stabilizing capability). The EVA-projected/HMMD processing offers a hands-off (minimal tuning) method for extracting the mean dwell times for

various blockade states (the core kinetic information on the blockading molecule's channel interactions). We have synthetically generated data where we employ the HMM's generative capability to generate signal profiles with the stationary statistics indicated in the model (to synthetically generate signals like that seen in current NTD experiments, e.g., with the same stationary statistics). The synthetic data allows NTD situations to be simulated where multiple channels, or other noise sources, might be present. In [3], experiments with synthetic data with two blockade levels were considered, with lifetime in each level determined by a governing distribution (Poisson and Gaussian distributions with a range of mean values were considered). The results clearly demonstrate the superior performance of the HMMD over the simpler standard HMM formulation on data with non-geometrically distributed same-state interval durations. In the stochastic carrier wave context this describes a means to discern carrier with HMMD (while with HMM alone we are much weaker in this regard and cannot robustly discern carrier). With use of the EVA-projection method, this also affords a robust means to obtain kinetic-type feature extraction. The HMM with duration is critical for accurate kinetic feature extraction when using EVA, and the results in [3] suggest that this problem can be elegantly solved with a pairing of the HMM-with-Duration stabilization with EVA-projection.

In [3] we describe a state-decoding on synthetic data that is representative of a biological-channel, two-state ion-current decoding problem, or an encode/decode software radio signal. For this problem 120 data sequences were generated that have two states with channel blockade levels set at 30 and 40 pA (a typical scenario in practice). Every data sequence has 10,000 samples. Each state has emitted values in a range from 0 to 49 pA. The maximum duration of states is set at 500. The mean duration of the 40 pA state is given as 200 samples (typically have one sample every 20 microseconds in actual experiments), while the 30 pA level has mean duration set at 300 samples. The task is to train using 100 of the generated data sequences and attempt state-decoding on the remaining 20 data sequences. The performance difference is stark: the exact and adaptive HMMD decodings are 97.1% correct, while the HMM decoding is only correct 61% of the time (where random guessing would accomplish 50%, on average, in a two-state system).

4.2.3 Holographic HMMs -- multi-track HMMs with generalized clique, maximally interpolated, with minimum-size meta-state constraints

The label and other counts described in what follows (Tables 1-5), show statistical support for a two-track HMM with generalized clique, with the meta-states indicated for the *C. elegans* Genome. Consider exon forward read labels made according to (012) frame: 0, 1, 2. Likewise, consider exon reverse read labels according to (CBA) frame: A, B, C. For intron in forward gene: use 'i' for the label. For intron in reverse gene: use 'I' for the label. For non-coding, non-intron, 'junk: use 'j' for the label. There are, thus, 9 labels: (0,1,2,A,B,C,i,I,j) in the "single-track" label scheme. The first chromosome of *C. elegans* has 14,025,570 bases, and with annotation according to the above label scheme, the counts on different labels are shown in Table 1:

0 571,187	A 518,431	I 1,634,653
1 571,187	B 518,431	i 1,779,392
2 571,187	C 518,431	j 7,336,733

Table 1. Counts on Labels.

For the 9-label alphabet used in the annotation we find 25 transitions between labels (see Table 2), or transition "states", with the following counts showing consistency with the 25 'allowed transition' out of 9x9=81 possible (i.e., only 25 transitions with nonzero counts):

01	569,483		BA	516,874		II	1,628,572	
12	569,490		CB	516,868		ii	1,772,795	
20	566,732		AC	514,309		jj	7,334,177	
0i	1,704 → i1	1,704*	IA	1,557 → BI	1,557	j0	1,257 → 2j	1,257
1i	1,696 → i2	1,696	IB	1,563 → CI	1,563	Aj	1,161 → jC	1,161
2i	3,197 → i0	3,197	IC	2,961 → AI	2,961			

Table 2. Counts on Transitions.

The label convention on introns is such that a sequence of transitions between to the next exon might look like the following: …20 0i ii ii ii ii ---- ii i1 12 20 01 …., thus it is expected that the number of 0i transitions will equal the number of i1 transitions, etc., as is verified above.

Now suppose that there were multiple annotations regarding the labeling of a base (i.e., alternative splicing). As the genome is traversed in the forward direction, gene annotations not in conflict with annotations already seen are used to determine labels on the first label track. If a gene annotation is in conflict (an alternative splicing), then its label information is recorded on a second, adjacent, label track. The above tables are actually the label counts on track one, in Table 3 below are the label counts on track two (where the default base label is taken to be 'j'):

0 21,599	A 64,475	I 325,471
1 21,599	B 64,471	i 81,289
2 21,599	C 64,467	j 13,354,661

Table 3. Counts on Track '2' Labels.

Since the j count on track two is 13,354,661, this indicates that about 5% of the coding regions in the first chromosome of *C. elegans* have alternate splicing at the coding *base* level. The counts on Track 2 transitions are shown in Table 4.

01	21,554		BA	64,296		II	324,751	
12	21,548		CB	64,275		ii	81,073	
20	21,441		AC	63,986		jj	13,354,350	
0i	45 → i1	45	IA	175 → BI	175	j0	38 → 2j	38
1i	51 → i2	51	IB	192 → CI	192	Aj	136 → jC	136
2i	120 → i0	120	IC	353 → AI	353			

Table 4. Counts on Track '2' Transitions.

Consider the preliminary results on the alternative splice data in a new way. Now let us consider a label scheme that is truly multi-track insofar as integrating with the HMM hidden label formalism. For this, consider the 'V-label' -- the two-element "vertical" label comprising the track 1 and 2 values. So if a base has label '0' on track 1 and label 'A' on track 2, its V-label is 'V0A'. 72 V-labels are found to have nonzero counts (out of 9*9=81 possible). Most of the V-labels describe an overlap of noncoding on one track with coding on the other track. These are the counts on V-labels describing *coding region* overlaps, shown in Table 5:

V00, V11, V22 17,839	VA0, VB2, VC1 0	V0A, V1C, V2B 741	VAA, VBB, VCC 16,169
V01, V12, V20 3	VA1, VB0, VC2 0	V0B, V1A, V2C 957	VAB, VBC, VCA 0
V02, V10, V21 58	VA2, VB1, VC0 829	V0C, V1B, V2A 5164	VAC, VBA, VCB 54

Table 5. Counts 'Vertical" labels (V-labels), consisting of the track 1 and track 2 labels at a particular emission instance being grouped as (track 1 label)(track 2 label), of which only 72 out of 81 possible have non-zero counts and the nontrivial coding-coding overlaps are shown here. Notice how the V-labels tend NOT to favor simple frame-shifts in a given read direction (i.e., the V01 count is very low compared to V00, etc.).

Some of the 263 'V-transitions' only have one count, and an analysis indicates that other, similar type transitions may be allowable as well. The initial model that we will adopt has the following two-track transition overlap rules: (1) if coding overlaps and has the same read direction it must have same framing; (2) transitions between coding and non-coding ('eij-transitions') can only overlap other transitions between coding and non-coding if half in agreement (the signature of a common splice variant); (3) transitions between coding and non-coding can overlap coding to coding transitions (coding 'xx-transitions') regardless of frame mismatch; (4) non-coding to non-coding transitions (coding 'xx-transitions') can overlap any other transition. This leads to 389 V-transitions, with 320 eij-transitions and 69 xx-transitions. In the clique-generalized version of the 389-element 'base model' the states grow as N=69+320(F-1), where 'F' is the size of the clique 'footprint' state, and the order of computation for the footprint-size F meta-HMM scales as 320*F (not N^2). The 389 V-transitions present a notable reduction from the 25x25 possible overlaps (625 total). This is a tractable number of states to manage in the HMM analysis, suggesting a simple and direct approach to alternative splice HMM analysis. The 389 V-transitions do not require 389 independent tables -- 311 independent tables are needed, and this is the same even for the N-element footprint state versions, where many of the xx-transitions and eij-transitions have merged counts (and shared look-up tables), just as with the 25-transition model having 11 independent tables in the original clique-generalization analysis [9]. Although sufficient support for statistical modeling is already at hand, improvement to the V-transitions statistical model can also be pursued by incorporating information from related genomes (such as *C. Briggsae*).

4.2.4 Distributed HMM processing via 'Viterbi-overlap-chunking' with GPU speedup

Distributed processing has been done by use of simple chunking with overlaps 'sufficient' for recovery, where the details of the latter are described in what follows. A central Viterbi-like feature of the chunking methods employed is first described: the table chunking methods for the dynamic programming algorithms that have been developed involve only a single-pass computation analogous to the Viterbi algorithm (ignoring O(L) traceback) [1]. The Viterbi algorithm efficiently calculates the most probable state path. The Baum-Welch algorithm calculates the probability of having a state at a particular index, summing over all path probabilities that arrive at that state-instance, and is usually implemented as two passes, for the forward and backward parameters. In the Linear Memory HMM introduced in [25], however (see Methods),the Baum-Welch implementation has a distinctive trait other than a linear memory implementation, it's also a 'single-pass' implementation for the algorithm, which is needed for the Viterbi single-pass referenced, overlap-stitched, reconstituted signal in a distributed

processing setting (see Methods for details). This can used be for brute force, and massively scalable, computational speed-up on all the HMM-based algorithms used in the SSA Protocol.

Single-Pass Table Algorithm

The table chunking methods for the dynamic programming algorithms that are described in what follows make use of a single-pass computation analogous to the Viterbi algorithm (ignoring O(L) traceback). In the Linear Memory HMM introduced in [25], and described below, the Baum-Welch algorithm implementation has a distinctive trait other than a linear memory implementation; itis also a 'single-pass' implementation.

Following the notation used in [25], $t_{i,j}(t,m)$ is the weighted sum of probabilities of all possible state paths that emit subsequence $b_1,\ldots,b_t$ and finish in state $\lambda_t = m$, taking an λ_t=**i** → λ_{t+1}=**j** (**i→j**) transition at least once (for some t) where the weight of each state path is the number of i→j transitions that it takes. Processing of the entire $t_{i,j}(t,m)$ recurrence takes memory proportional to $O(NQ)$ and processor time $O(TNQQ_{max})$.

Initially, since no transitions have been made, $t_{i,j}(1,m)=0$. After initialization we have the following recurrence steps

$$t_{i,j}(t,m) = f_{i(t-1)}\, a_{im}\, e_m(b_t)\,\delta(m=j) + \sum_{n=1}^{N} t_{i,j}(t\text{-}1,n)\, a_{nm}\, e_m(b_t)$$

The computation is in-step with the forward variable as a single-pass computation, where the delta function is defined as: $\delta(m=j)=\begin{cases} 1, & if\ m=j \\ 0, & otherwise \end{cases}$. At a certain time moment t we need to score the evidence supporting transition between nodes i and j, which is the sum of probabilities of all possible state paths that emit subsequence $b_1,\ldots,b_{t-1}$, and finish in state i (forward probability $f_{i(t-1)}$), multiplied by transition a_{ij} and emission $e_j(b_t)$ probabilities upon arrival to b_t. We extend the weighted paths containing evidence of i→j transitions made at previous time moments $1,\ldots,t\text{-}1$ further down the trellis in the second part of the equation above. Finally, by the end of the recurrence, we marginalize the final state m out of probability $t_{i,j}(T,m)$ to get a weighted sum of state paths taking transition i→j at various time moments. Thus, we estimate transition utilization using

$$a_{ij} = \frac{\sum_{m=1}^{N} t_{i,j}(T,m)}{\sum_{j\in out(state\ i)} \sum_{m=1}^{N} t_{i,j}(T,m)},$$

where *out(state i)* of nodes connected by edges from state *i*.

The following algorithm updates the 'emission' parameters for the set of discrete symbol probability distributions E={e_1(b),…, e_N(b)} in O(NED) memory and O(TNEDQ$_{max}$) time. According to [25], $e_i(b,t,m)$ is the weighted sum of probabilities of all possible state paths that emit subsequence $b_1,\ldots,b_t$ and finish in state m, for which state i emits observation b at least once where the weight of each state path is the number of b emissions that it makes from state i. Initialization step: $e_i(b,1,m)= f_{m1}\delta(i=m)\delta(b=b_1)$. After initialization we make the recurrence steps, where we correct emission recurrence presented in [26]:

$$e_i(b,t,m)= f_{mt}\delta(i=m)\delta(b=b_t) + \sum_{n=1}^{N} e_i(b,t-1,n)\, a_{nm}\, e_m(b_t)$$

Finally, by the end of the recurrence, we marginalize the final state m out of $e_i(b,T,m)$ and estimate the emission parameters through normalization

$$e_j(b) = \frac{\sum_{m=1}^{N} e_i(b,T,m)}{\sum_{\gamma=1}^{D} \sum_{m=1}^{N} e_i(b,T,m)},$$

The forward sweep takes $O(TNQ_{max})$ time, where only the values of $f_{i(t-1)}$ for $1 \le i \le N$ are needed to evaluate f_{it}, thus rendering memory requirement to $O(N)$ for the forward algorithm. Computing $e_i(b,t,m)$ takes $O(NED)$ previous probabilities of $e_i(b,t-1,m)$ for $1 \le m \le N$, $1 \le i \le E$, $1 \le b \le D$. Recurrent updating of each $e_i(b,t,m)$ probability elements takes $O(Q_{max})$ summations, totaling $O(TNEDQ_{max})$.

Chunking with overlap resolution

In HMM signal processing latency becomes very prohibitive when attempting to increase device bandwidth or when input datasets are large. Described in what follows are results from performing HMM algorithms in a distributed manner by breaking into overlapping chunks and leveraging the Markovian assumption underlying the HMM to help arrive at a chunk-data reconstruction. The pathological instances where the distributed merges can fail to exactly reproduce the non-distributed HMM calculation can be made as least likely as desired with sufficiently strict, but not computationally expensive, segment join conditions. In this way, the distributed HMM provides a feature extraction that is equivalent to that of the sequentially run, general definition HMM, and with a speedup factor approximately equal to the number of independent CPUs operating on the data. The Viterbi most probable path calculation and the Expectation/Maximization (EM) calculation can both be performed in this distributed processing context.

The linear memory implementation described above (and in [26]) was optimized according to the observation that Viterbi traceback paths in the Viterbi procedure typically converge to the most likely state path and travel all together to the beginning of the decoding table – the picture being much like a river with minor tributaries backtracking onto that river, and maybe those 'tributaries' themselves have more minor state paths converging into them, etc. But the trait that is most notable in the convergence-durations to the 'main-tributary', or what is to be the most likely (Viterbi) path, is that it is usually a modest number of columns for many data types. This backwards Markovian memory loss on a tributary with respect to its origin (said to occur when backtracked and mixed with the main, Viterbi, convergence path of the tributaries) is hypothesized to be an indicator of the span of sequence needed to have Viterbi path probabilities in a given column that have settled into their properly ordered relative probabilities in that column. Further column processing refinement to bring the relative values of the Viterbi-path probabilities into better estimation is then possible. In distributed processing efforts, this "Viterbi relaxation time" is a key parameter that can be used to design an optimally overlapping chunking of the data sequence in a distributed speed-up on the sequence analysis.

A distributed signal processing test of some basic chunk reconstruction heuristics was performed on 5 computers with 300 signals. Each signal had 5000 samples. The resulting viterbi paths

matched between the distributed HMM and standard HMM on a 10-column segment. For the standard HMM, EM training (5 loops) the Viterbi algorithm took 272 seconds. For distributed HMM with 5 CPU's, the computational time was reduced to 69 seconds. So using 5 computers, we had a speedup of 3.94. A perfect de-segmentation was performed with an N=10 match window as indicated, initally, but it was found that a perfect re-stitching of segments was also possible simply with N=1 (see Fig. 13), due to the implicit stringency of the simultaneity condition (the overlap match, at the one position corresponding to N=1, must globally index to the same observation data index for both segments). The multi-chunk re-stitching makes use of the Viterbi path and the entire set of Viterbi traceback pointers in a given overlap set of columns [1].

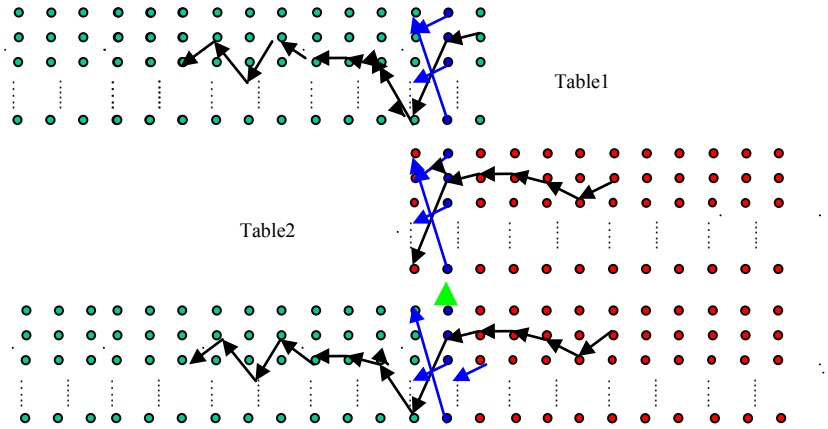


Figure 13. Viterbi column-pointer match de-segmentation rule. Table1 and Table2 are overlapped. And their blue columns have the same pointers. Then the index of this blue column becomes the joint. The black pointers form the final viterbi path.

4.3 Support Vector Machine Results

4.3.1 SVM Feature-Set Robustness

In a parallel datarun to that indicated in Fig. 14, with 150 component feature vectors, feature vectors with the full set of 2600 components were extracted (i.e., no compression was employed on the transition probabilities). SVM performance on the same train/test data splits, but with 2600 component feature vectors instead of 150 component feature vectors, offered similar performance after drop optimization. This demonstrates a significant robustness to what the SVM can "learn" in the presence of noise if a small weak-data drop is allowed (where some of the 2600 component have richer information, but even more are noise contributors).

Classification improvement with Adaboost taking the best 50 from the Inverted-emission 150 feature set is shown in Fig. 14 as 'First 50' (above cutoff). An accuracy of 95% is possible for discriminating 9GC from 9TA hairpins with no data dropped with use of Adaboost. This demonstrates a significant robustness to what the SVM can "learn" in the presence of noise (some of the 2600 components have richer information, but even more are noise contributors). This also validates the effectiveness with which the 150 parameter compression was able to describe the two-state dominant blockade data found for the nine base-pair hairpin and other types of "toggler" blockades, as well as the utility of the inverted features.

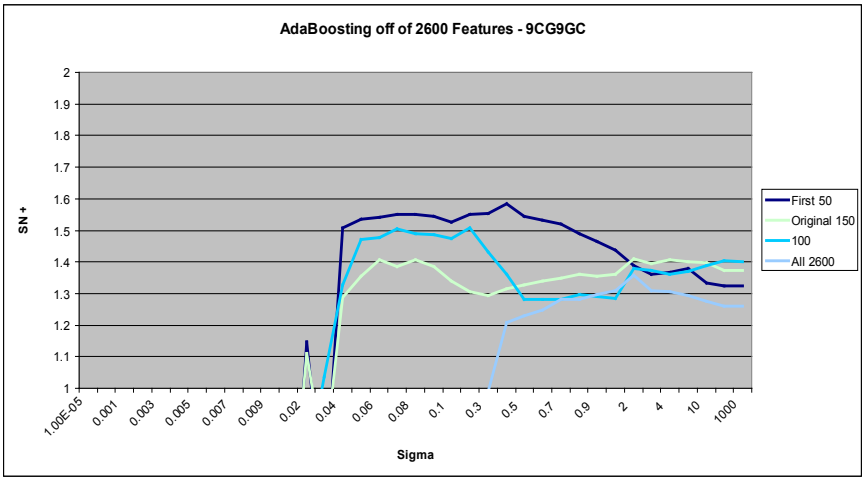


Fig. 14. AdaBoosting to select 100 of the full set of 2600 features improves classification over just passing all 2600 components to the SVM. The best performance is still obtained when working with the Adaboosting from the manual set. (A principal component analysis (PCA) is done on the HMM projection data. 90% of the PCA information is contained in the first 50 principal components. The first 50 principal components are also listed as a feature set.)

4.3.2 SVM Clustering

Single-convergence initialized SVM-clustering: cluster refinement using simulated annealing

The data set is chosen to be an equal positives vs negatives sample of 200 8GC blockade signals and 200 9GC blockade signals (see [27] 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 [2]). Although convergence was easily achieved with the SVM Relabeler algorithm (see [27]), convergence to a global optimum was not guaranteed. Fig. 15 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 will be very small.

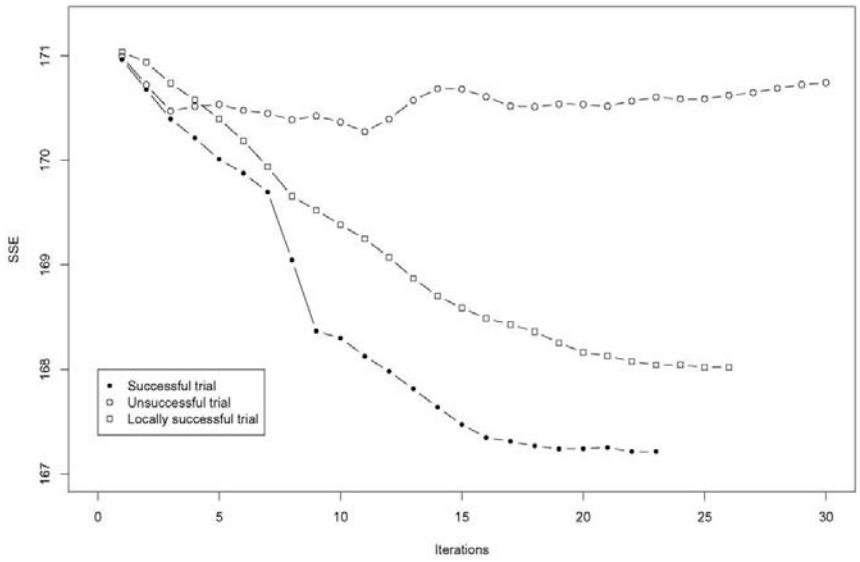


Fig. 15a. Training history on SSE for three types of clustering result: successful, unsuccessful; and locally successful.

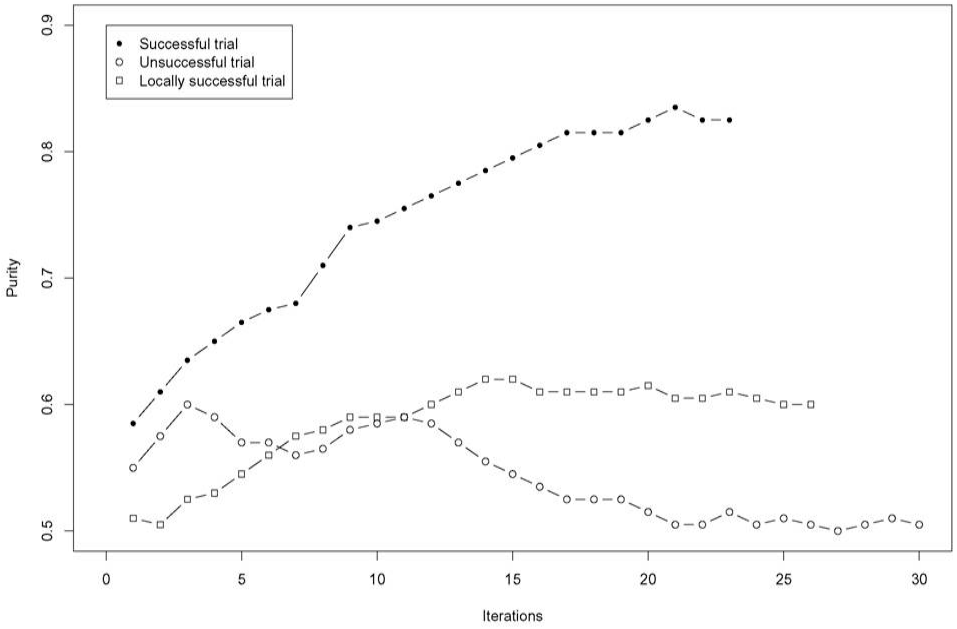
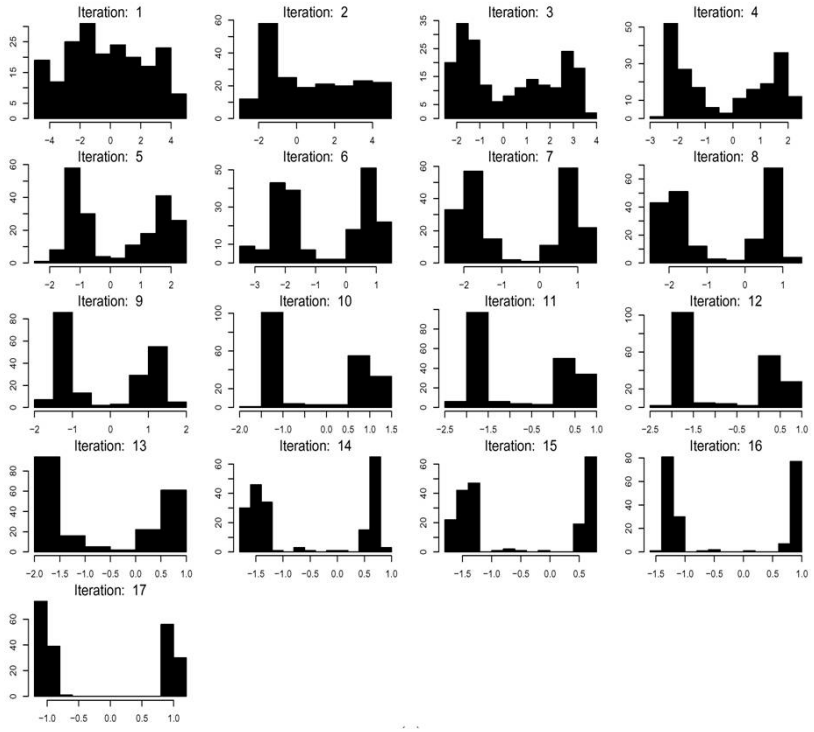


Fig. 15b. Training history on cluster purity for three types of clustering result: successful, unsuccessful; and locally successful.

In Fig. 16 a small value of Kernel-SSE (herein referred to as SSE) is shown to provide us with a reliable cluster validation measure.

(TOP)



(Bottom)

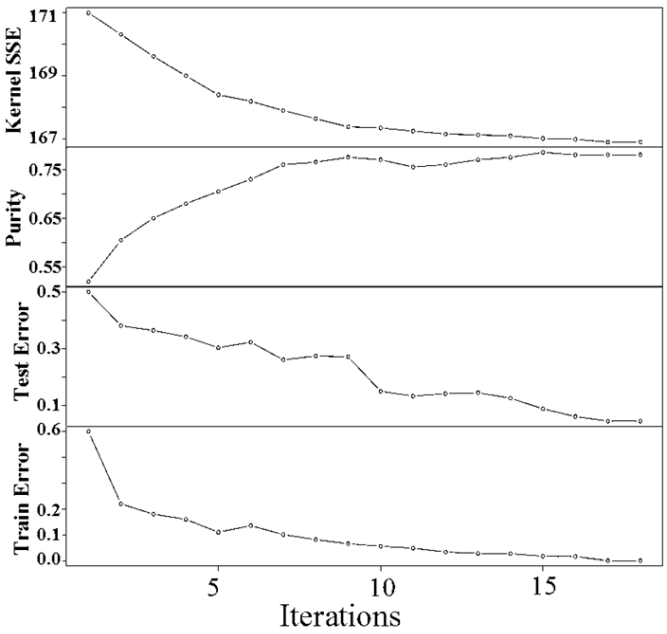


Fig. 16 TOP. SVM clustering training run. BOTTOM. SVM clustering training parameters during the training process shown in TOP.

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. 15). 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. 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. 17. 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 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 solutions in those situations works well here as well – simulated annealing.

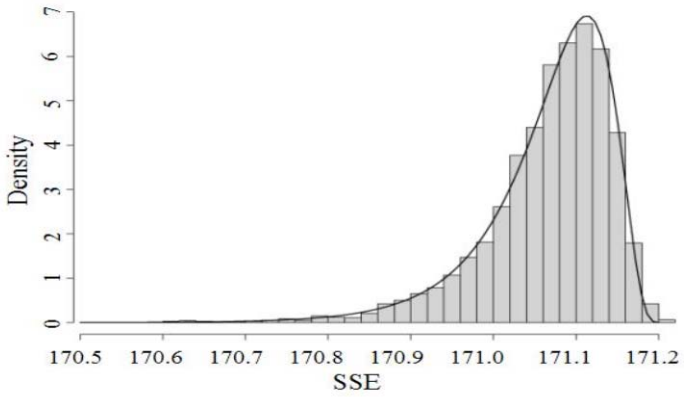


Fig. 17. Plot of randomly generated SSE profiles for the type of feature vectors considered in the DNA molecule analysis.

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 Fig. 18 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. 18 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.

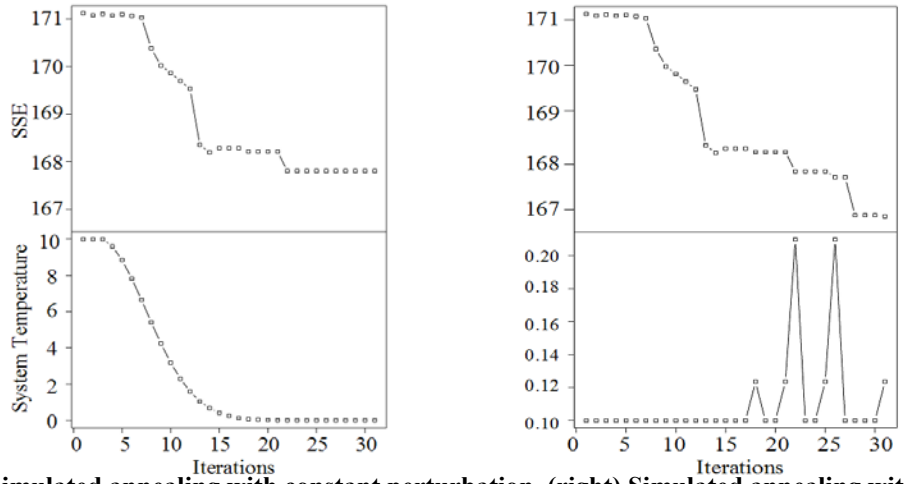


Fig. 18. (left) Simulated annealing with constant perturbation, (right) Simulated annealing with variable perturbation. As shown in left, top panel, simulated annealing with a 10% initial label-flipping results in a local-

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 $T_0 = 10$.

In the effort shown in [27] 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, which suggested a simulated annealing approach for greater computational efficiency, as shown in Fig. 18 (more recent Proof of Concept work with Genetic Algorithms not shown, but were found to exhibiting even stronger stability). Results of this effort (Fig. 18) significantly improve and stabilize the SVM clustering process.

Given the wide variety of dissimilar tuning parameters in the SVM classification process alone, tests on SVM classification with genetic algorithm (GA) based tuning seems optimal. The very robust and rapid auto-tuning with the GA approach on SVM classification in initial tests strongly suggests that this, or any swarm intelligence search/tuning paradigms, offer important refinement to the SVM-classification efforts and critical refinement to the single-convergence initialization SVM-clustering efforts. As mentioned previously, however, we also have a more informed version of the process, as discussed in the next section.

5 Discussion

5.1 SCW in nanotechnology

Recently developed HMM implementations, summarized in the Methods, allow a new form of carrier-based communications, where the carrier is not periodic but is stochastic with stationary statistics. The "stochastic carrier wave" approach is not only a means to understand the messages Nature provides (in near-equilibrium flow phenomenologies with stationary statistics), but also provides a hidden carrier method, enabling secure communications, making signal jamming much more difficult, and making signal location much more difficult. An algorithmic methodology that allows for 100-fold, or faster, implementation of a Hidden Markov model with duration (the HMMBD algorithm [8]), is critical to this encoding/decoding method.SCW communications are found at the forefront of a number of efforts in nanotechnology. This is because nature at the single-molecule scale has a signal modulation that is stochastic, sometimes with stationary statistics. Such is the case with the signal analysis in a nanopore transduction detector.

If states have self-transitions with a notably non-geometric distribution on their self-transition 'durations', then a fit to a geometric distribution in this capacity, as will be forced by the standard HMM, will be weak, and HMMD modeling will serve better. In engineered communications protocols, or in engineered, modulated, nanopore transduction detector signals, highly non-geometric distributions can be sought. One encoding scheme that is strongly non-geometric in same-state duration distribution is Nature's familiar *long* open-reading-frame

(ORF) encoding found in genomic data. This suggests a similar ORF-like encoding scheme to establish a carrier duration peak in the self-transition distribution's tail region, e.g., a second peak in the duration distribution (perhaps one even more skewed from the geometric distribution than the heavy-tail distributions found for ORFs).

The NTD signal analysis demonstrates the simplest stochastic carrier wave utilization in a biophysics experimental setting – a stochastic phase modulation (with just two phases of stationary statistics). A minor elaboration on the signal analysis, to go from a simple two-state (bound/unbound) signal recognition to a two-phase SCW telegraph signal, then yields the rudimentary implementation for stochastic carrier communications purposes.

5.2 NTD with multiple channels (or high noise)

The nanopore transduction detection platform involves functionalizing a standard nanopore detector platform in a new way that is cognizant of signal processing and machine learning capabilities and advantages, such that a highly sensitive biosensing capability is achieved. In the NTD functionalization of the standard nanopore detector, we design a molecule that can be drawn into the channel (by an applied potential) but be too big to translocate, instead becoming stuck in a bistable 'capture' such that it modulates the ion-flow in the *single* nanopore channel established in a distinctive way. An approximately two-state 'telegraph signal' is engineered for this purpose. If the channel modulator is bifunctional, in that one end is meant to be captured and modulate while the other end is linked to an aptamer or antibody for specific binding, then we have the basis for a remarkably sensitive and specific biosensing capability. The biosensing task is reduced to the channel-based recognition of bound or unbound NTD modulators. Preliminary results demonstrate successful application of this method in a streptavidin (toxin) detection scenario using a biotinylated DNA hairpin. In typical NTD biosensing there is only one (nanometer-scale) channel established in the detector apparatus, however, where other channels bridging the same membrane (bilayer) would do so in parallel with the first (single) channel. In a naïve setting, additional channel noise sources degrade sensitivity and offset gains from having multiple channel 'receptors'. ***In the stochastic carrier wave encoding/decoding with HMMD, it may be possible to have multiple channels but avoid signal degradation such that the full benefits of a multiple receptor gain can be realized.***

In the NTD platform, sensitivity increases with observation time in contrast to translocation technologies where the observation window is fixed to the time it takes for a molecule to move through the channel. The key to the sensitivity and versatility of the NTD platform is the unique ability to couple real-time adaptive signal processing algorithms to the complex blockade current signals generated by the captured transducer molecule. The NTD approach can provide exquisite sensitivity and can be deployed in many applications where trace level detection is required.

Consider the case where 100 parallel channels are in operation, a scenario that has the potential to increase the sensitivity of the NTD 100-fold, but the signal analysis typically becomes more challenging, and sensitivity gains limited, since there are 100 parallel noise sources. The HMMD recognition of a transducer signal's stationary statistics, however, is analogous to 'time integration' heterodyning a radio signal with a periodic carrier in classic electrical engineering, in that there is improved carrier-signal recognition with longer observation time. In order to introduce a 'time integration' benefit in the recognition of a transducer signal, periodic (or

stochastic) modulations may be introduced to the transducer environment. In a high noise background, modulations may allow some of the transducer states to have heavy-tailed, or multimodal, self-transition duration distributions. With these modifications to the signal processing software, a single transducer molecule signal is recognizable in the presence of 100s of channels. Increasing the number of channels by 100 and retaining the capability of recognizing a single transducer blockading one of those channels provides a direct gain in sensitivity according to the number of channels (e.g., 100 channels would provide a sensitivity boost of 100). It is important to note that the increase in sensitivity is mostly implemented computationally and does not add complexity or cost to the NTD device itself.

5.3 pMM/SVM sensor boosting in HMMs

Markov-based statistical profiles, in a log likelihood discriminator framework, can also be used to create a fixed-length feature vector for SVM based classification [5]. Part of the idea of the method is that whenever a log likelihood discriminator can be constructed for classification on stochastic sequential data, an alternative discriminator can be constructed by 'lifting' the log likelihood components into a feature vector description for classification by SVM. Thus, the feature vector uses the individual log likelihood components obtained in the standard log likelihood classification effort, the individual-observation log odds ratios, and 'vectorizes' them rather than sums them. The individual-observation log odds ratios are themselves constructed from positionally defined Markov Models (pMM's), so what results is a pMM/SVM sensor method. This method may have utility in a number of areas of stochastic sequential analysis, including splice-site recognition and other types of gene-structure identification [1], file recovery in computer forensics ('file carving') , and speech recognition.

6 Conclusions

Hidden Markov models are a pervasive and fundamental tool in sequential data analysis and signal communications. Critical HMM tools have recently been improved via a number of computationally efficient generalizations. Generalized HMMDs that are clique-generalized and with side-information provide an efficient means to establish a new form of carrier-based communications, where the carrier is not periodic but is stochastic, with stationary statistics. The generalized HMMD algorithmic methodology enables practical stochastic carrier wave encoding/decoding. SCW type signal processing is encountered at the forefront of a number of efforts in science and nanotechnology, since Nature offers up stationary statistics in many near-equilibrium situations and flow situations, and since engineered stationary statistics systems are common (such as with the nanopore transduction detection examples).

The recently improved signal processing and clique-scaling functionality with HMMs enables robust stochastic carrier wave signal processing, especially when implemented within the weakness recovery protocol outlined for the SSA Protocol described in the Results. The new stochastic carrier wave functionality described here offers a significant and new dimension to signal processing, with numerous applications.

7 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 students Anil Yelundur, Ken Armond, and Sepehr Merat for help with the algorithmic implementations and the channel current cheminformatics analysis. The author would like to thank the University of New Orleans, NIH, NSF, NASA, and the Louisiana Board of Regents for research support. The author would also like to thank META LOGOS Inc., for research support and a research license. (META LOGOS was co-founded by the author in 2009, when it obtained exclusive license to the NTD and machine-learning based signal processing intellectual property.) The author would also like to thank Robert Adelman (CEO META LOGOS, Inc.), Andrew Peck (CEO PxBioSciences), and Mike Lewis (Professor, University of Missouri-Columbia), for insights into the potential impact of the NTD approach.

8 References

[1] Winters-Hilt, S. and R. Adelman. Method and System for Characterizing or Identifying Molecules and Molecular Mixtures. USPTO Filing. Meta Logos Inc. 2010.

[2] Winters-Hilt, S., W. Vercoutere, V. S. DeGuzman, D. Deamer, M. Akeson, and D. Haussler, "Highly Accurate Classification of Watson-Crick Base-Pairs on Termini of Single DNA Molecules," *Biophys. J.* Vol. 84, pg 967, 2003.

[3] Landry M, Winters-Hilt S. Analysis of nanopore detector measurements using machine learning methods, with application to single-molecule kinetic analysis. BMC Bioinf. 8 S7, S12 (2007).

[4] Winters-Hilt S: Hidden Markov Model Variants and their Application. BMC Bioinf. 2006, 7 S2: S14.

[5] Roux B and Winters-Hilt S. Hybrid SVM/MM Structural Sensors for Stochastic Sequential Data. BMC Bioinf. 9 S9, S12 (2008).

[6] Vapnik, V. N. 1999. The Nature of Statistical Learning Theory (2nd Ed.). Springer-Verlag, New York.

[7] 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.

[8] Winters-Hilt S and Jiang Z. A hidden Markov model with binned duration algorithm. IEEE Trans. on Sig. Proc., Vol. 58 (2), Feb. 2010.

[9] Winters-Hilt, S. and C. Baribault. A Meta-state HMM with application to gene structure identification in eukaryotes. EURASIP Journal of Advances in Signal Processing, Special Issue, Genomic Signal Processing, 2010.

[10] Winters-Hilt, S., Jiang, Z., and C. Baribault. Hidden Markov model with duration side-information for novel HMMD derivation, with application to eukaryotic gene finding. EURASIP Journal of Advances in Signal Processing, Special Issue on Genomic Signal Processing, 2010.

[11] Iqbal R, Landry M, Winters-Hilt S: DNA Molecule Classification Using Feature Primitives. BMC Bioinformatics 2006, 7 S2: S15.

[12] Donoho D. Compressed sensing. IEEE Trans. On Information Theory, 52(4), pp. 1289 - 1306, April 2006.

[13] Ferguson, J.D. Variable duration models for speech. Proceedings of Symposium on the Application of Hidden Markov models to Text and Speech, pages 143-179, 1980.

[14] Rabiner, L.R. A tutorial on hidden markov models and selected application in speech recognition. Proceedings of the IEEE, 77:257-286, 1989.

[15] Ramesh, P., and J.G. Wilpon. Modeling state durations in hidden markov models for automatic speech recognition. Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 1:381-384, 1992.

[16] Yu, SZ. and H. Kobayashi. An efficient forward-backward algorithm for an explicit-duration hidden markov model. IEEE Signal Processing Letters, 10:11-14, 2003.

[17] Johnson, M.T. Capacity and complexity of hmm duration modeling techniques. IEEE Signal Processing Letters, 12:407-410, 2005.

[18] Ghahramani, Z. and M. Jordan. Factorial hidden markov models. Machine Learning, 29:245-273, 1997.

[19] Singer, Y., S. Fine and N. Tishby. The hierarchical hidden markov model: Analysis and applications. Machine Learning, 32:41, 1998.

[20] Murphy, K. and M. Paskin. Linear time inference in hierarchical hmms. Proceedings of Neural Information Processing Systems, 2001.

[21] Winters-Hilt, S., and Pincus, S. Nanopore-based biosensing. PATENT, UNO filing, 2004.

[22] Baribault, C and Winters-Hilt S. A novel, fast, HMM-with-Duration implementation -- for application with a new, pattern recognition informed, nanopore detector. BMC Bioinformatics 2007, 8 S7: S19.

[23] Eren AM, Amin I, Alba A, Morales E, Stoyanov A, and Winters-Hilt S. Pattern Recognition Informed Feedback for Nanopore Detector Cheminformatics. Accepted paper in book “Advances in Computational Biology”, Springer: Advances in Experimental Medicine and Biology, June 2010

[24] Winters-Hilt S, Yelundur A, McChesney C, Landry M: Support Vector Machine Implementations for Classification & Clustering. BMC Bioinformatics 2006, 7 S2: S4.

[25] Churbanov, Alexander and S. Winters-Hilt. Implementing EM and Viterbi algorithms for Hidden Markov Model in linear memory. BMC Bioinformatics 2008, 9:228.

[26] Miklos I, Meyer I: A linear memory algorithm for Baum-Welch training. *BMC Bioinformatics* 2005, 6(231).

[27] Winters-Hilt S and Merat S. SVM Clustering. BMC Bioinf. 8 S7, S18 (2007).