EasyChair Preprint

№ 7029

Timesmash: Process-Aware Fast Time Series Clustering and Classification

Victor Rotaru, Yi Huang and Ishanu Chattopadhyay

November 10, 2021

Timesmash: Process-aware Fast Time Series Clustering and Classification

Victor Rotaru, Yi Huang, Ishanu Chattopadhyay
Department of Medicine, University of Chicago, Chicago, IL 60637, USA
{virotaru, yhuang10, ishanu}@uchicago.edu

Abstract

We introduce **Timesmash**: a comprehensive suite of clustering and classification algorithms and their implementation as a eponymous python package for stochastic time series analysis. We leverage a subclass of hidden Markov model (HMM), called Probabilistic Finite-State Automaton (PFSA), which are used to first model in an unsupervised setting the underlying generative processes for observed data streams, which then aid in carrying out automatic physics or process aware featurization enabling subsequent clustering and classification. The algorithms in this suite consist of the following tools: a) **LikelihoodDistance** estimating in an unsupervised setting the divergence between ergodic stationary finite valued stochastic processes from the observation of finite and possibly unequal sample paths. b) Featurization algorithms **SymbolicDerivative**, **InferredHMMLikelihood**, and **ClusteredHMMClassifier**, which operate by aiming to recover the underlying hidden generator for the sample paths presented, which then may be used to automatically distill effective features for classification. Our core algorithms require the data streams to take values in a finite alphabet. To extend applicability to continuous-valued time series, a data-driven quantization algorithm, our implementation includes the tool **Quantizer** that discretizes continuous sequences without the assumption of domain knowledge. We evaluate the performance of the **Timesmash** algorithms on problems from the UCR Time Series Classification Archive, and show that we at par or better compared to the state of the art Dynamic Time Warping (DTW) algorithm. In addition, we include brief examples where our unsupervised physical modeling leads to insights not easily obtainable with the current state of the art.

Introduction

Efficiently contrasting and comparing stochastic processes is the key to analyzing time-dependency in complex systems, particularly where randomness cannot be ignored. For such learning to occur, we need to define either a measure of deviation or, more generally, a measure of similarity to compare stochastic time series. Examples of such similarity measures from the literature include the classical l_p distances and l_p distances with dimensionality reduction (Lin et al. 2003), the short time series distance (STS)(Möller-Levet et al. 2003), which takes into account of irregularity in sampling rates, the edit based distances(Navarro 2001) with generalizations to continuous sequences(Chen, Özsu, and Oria 2005), and the dynamic time warping (DTW)(Petitjean, Ketterlin, and Gançarski 2011), which is used extensively in the speech recognition community.

We present a suite of algorithms for time series classification and clustering, implemented as a python package **Timesmash**. The package provides 1) an algorithm **LikelihoodDistance** to calculate the distance matrix given a set of time series; 2) two featurization algorithms, **SymbolicDerivative** and **InferredHMMLikelihood**, that map individual time series to feature vectors appropriate for input to any standard classification algorithm; 3) a domain knowledge-free quantization algorithm **Quantizer** that extends the applicability of the package from categorical to continuous-valued time series. We also discuss a classification algorithm, Cluster InferredHMMLikelihood Classification (**ClusteredHMMClassifier**) implemented using **Timesmash** tools, and compare its performance to Dynamic Time Warping (**DTW**) on the UCR time series classification dataset archive.

The algorithms underlying the tools in **Timesmash** focus on the *generating process* of the data streams. We hypothesize that time series from the same class in the training dataset of a given classification problem *"should"* have similar generating processes. While stochastic processes in general can be arbitrarily complex, here we assume that under appropriate quantization the processes can be well approximated by ergodic stationary finite-valued processes. In particular, we assume that the quantized version of our underlying processes may be generated by Probabilistic Finite State Automata (PFSA). PFSA have the property that we can carry out efficient evaluation of log-likelihood of a sequence as being generated by a specific model, and also PFSAs maybe efficiently inferred from quantized data streams (Chattopadhyay and Lipson 2013) by the algorithm **genESeSS**. **LikelihoodDistance** and **InferredHMMLikelihood** are both based on the PFSA log-likelihood evaluation. **ClusteredHMMClassifier** boosts classification performance by allowing for the possibility that time series from the same prescribed class in training, may actually be generated by different processes. Capturing these within-class variations via an unsupervised learning algorithm, will allow better classification of the test dataset.

Table 1: Performance Comparison on UCR Time Series Classification Datasets

Dataset	Baseline	SD	CH	Dataset	Baseline	SD	CH
ChlorineConcentration	**0.3500**	**0.2753**	0.3518	MiddlePhalanxTW	0.4870	**0.4286**	**0.4675**
Computers	**0.3000**	0.3360	**0.2760**	MixedShapesRegularTrain	0.0911	**0.0833**	**0.0680**
Crop	**0.2883**	**0.4169**	0.4426	PhalangesOutlinesCorrect	**0.2389**	0.2914	**0.2541**
Distal...Group	**0.2302**	0.2878	**0.1942**	PowerCons	**0.0667**	0.1667	**0.1278**
Distal...Correct	0.2754	**0.2681**	**0.2391**	Proximal...Group	0.1951	**0.1268**	**0.1415**
DistalPhalanxTW	**0.3669**	0.3741	**0.3094**	Proximal...Correct	0.1924	0.1924	**0.1890**
Earthquakes	0.2734	**0.2518**	**0.2590**	ProximalPhalanxTW	**0.2439**	0.2683	**0.2293**
ECG5000	**0.0749**	0.0802	**0.0798**	RefrigerationDevices	0.5360	**0.4027**	**0.5013**
ElectricDevices	**0.3806**	1.0000	**0.4889**	ScreenType	0.5893	**0.5467**	**0.5760**
EthanolLevel	0.7180	**0.6860**	**0.7140**	SemgHandGenderCh2	**0.1550**	0.2417	**0.1317**
FordA	0.3091	**0.1644**	**0.1515**	SemgHandMovementCh2	**0.3622**	0.3911	**0.2778**
FordB	0.3802	**0.3432**	**0.2827**	SemgHandSubjectCh2	**0.2000**	0.3889	**0.1956**
FreezerRegularTrain	0.0930	**0.0596**	**0.0196**	SmallKitchenAppliances	0.3280	**0.2133**	**0.2293**
GunPointAgeSpan	**0.0348**	0.0443	**0.0348**	StarLightCurves	0.0934	**0.0244**	**0.0211**
Gun...Female	**0.0032**	0.1108	**0.0285**	Strawberry	**0.0541**	**0.0757**	**0.0757**
Gun...Young	**0.0349**	0.0540	**0.0063**	UWaveGestureLibraryAll	**0.0343**	0.3398	**0.2990**
Ham	**0.4000**	0.4190	**0.3524**	UWaveGestureLibraryX	**0.2267**	**0.2993**	0.3099
HandOutlines	**0.1189**	0.2595	**0.2432**	UWaveGestureLibraryY	**0.3009**	**0.3903**	0.4013
LargeKitchenAppliances	**0.2053**	**0.3173**	0.3520	UWaveGestureLibraryZ	**0.3222**	0.4534	**0.3473**
MelbournePedestrian	**0.1518**	**0.2743**	0.2922	WormsTwoClass	0.3766	**0.2208**	**0.2857**
Middle...Group	0.4286	**0.3831**	**0.3571**	Wafer	0.0045	**0.0008**	**0.0015**
Middle...Correct	**0.2337**	**0.2337**	0.2680	Yoga	**0.1560**	0.2540	**0.2027**

1. Smallest error in red. Baseline error rate is the minimum error listed in the UCR Time Series Classification Archive.
2. SD for **SymbolicDerivative**, CH for **ClusteredHMMClassifier**.

Background: Models of Discrete Processes

Definition 1 (probabilistic Finite-State Automaton (PFSA))**.** A **probabilistic finite-state automaton**, or PFSA for short, G is specified by a quadruple $(Q, \Sigma, \delta, \widetilde{\pi})$, where Q is a finite set of states, Σ is a finite alphabet, δ is a partial map from $Q \times \Sigma$ to Q, called transition map, and $\widetilde{\pi}$, called observation probability, is a map from Q to $\mathbf{P}_\Sigma$, where $\mathbf{P}_\Sigma$ is the space of probability distributions over Σ. The entry indexed by σ of $\widetilde{\pi}(q)$ is written as $\widetilde{\pi}(q, \sigma)$.

We call the directed graph (not necessarily simple with possible loops and multi-edges) with vertex set Q and edges specified by δ the **graph of the PFSA** and assume it to be **strongly connected** (Bondy and Murty 2008), which means for any pair $q, q' \in Q$, there is a sequence $\sigma_1\sigma_2\cdots\sigma_k$ such that $\delta(q_{i-1}, \sigma_i) = q_i$ for $i = 1, 2, \ldots, k$ with $q_0 = q$ and $q_k = q'$.

Definition 2 (Observation and Transition Matrices)**.** Given a PFSA $(\Sigma, Q, \delta, \widetilde{\pi})$, the **observation matrix** $\widetilde{\Pi}$ is the $|Q| \times |\Sigma|$ matrix with the (q, σ)-entry given by $\widetilde{\pi}(q, \sigma)$, and the **transition matrix** Π is the $|Q| \times |Q|$ matrix with the (q, q')-entry, written as $\pi(q, q')$, given by

$$\pi(q, q') = \sum_{\{\sigma : \delta(q,\sigma) = q'\}} \widetilde{\pi}(q, \sigma).$$

It is straightforward to verify that both Π and $\widetilde{\Pi}$ are stochastic, *i.e.* non-negative with rows of sum 1. Since the graph of a PFSA is strongly connected, we have there is a unique probability vector $\mathbf{p}_G$ that satisfies $\mathbf{p}_G^T\Pi = \mathbf{p}_G^T$ (Vidyasagar 2014). We call $\mathbf{p}_G$, or simply $\mathbf{P}$ if G is understood, the **stationary distribution** of G.

Definition 3 (Stochastic process Generated by a PFSA)**.** Let $G = (Q, \Sigma, \delta, \widetilde{\pi})$ be a PFSA and $\mathbf{p}_G$ be the stationary distribution on Q. G generates sequences in the following fashion. To start, a state q_0 is chosen following $\mathbf{p}_0 = \mathbf{p}_G$, and then a symbol σ_1 is generated following $\widetilde{\pi}(q_0)$ and the system moves to $q_1 = \delta(q_0, \sigma_1)$. Then, a symbol σ_2 is generated following $\widetilde{\pi}(q_1)$ and the system moves to $p_2 = \delta(q_1, \sigma_2)$, so on and so forth.

Entropy Rate, KL Divergence, and Log-likelihood

Definition 4 (Entropy rate and KL divergence)**.** The entropy rate of a PFSA G is the entropy rate of the stochastic process G generates (Cover and Thomas 2012). Similarly, the KL divergence of a PFSA G' from the PFSA G is the KL divergence of the process generated by the G' from that of G. More precisely, we have the

$$\mathcal{H}(G) = -\lim_{d\to\infty} \frac{1}{d} \sum_{x\in\Sigma^d} p_G(x) \log p_G(x),$$

and the KL divergence

$$\mathcal{D}_{\mathrm{KL}}(G \,\|\, G') = \lim_{d\to\infty} \frac{1}{d} \sum_{x\in\Sigma^d} p_G(x) \log \frac{p_G(x)}{p_{G'}(x)},$$

if the limits exist, and where Σ^d is teh d times Cartesian product of Σ. Importantly, PFSA has closed-form formula for both entropy rate and KL divergence (Chattopadhyay, Huang, and Evans 2020).

Definition 5 (Log-likelihood)**.** The log-likelihood (Cover and Thomas 2012) of a PFSA G generating $x \in \Sigma^d$ is:

$$L(x, G) = -\frac{1}{d} \log p_G(x).$$

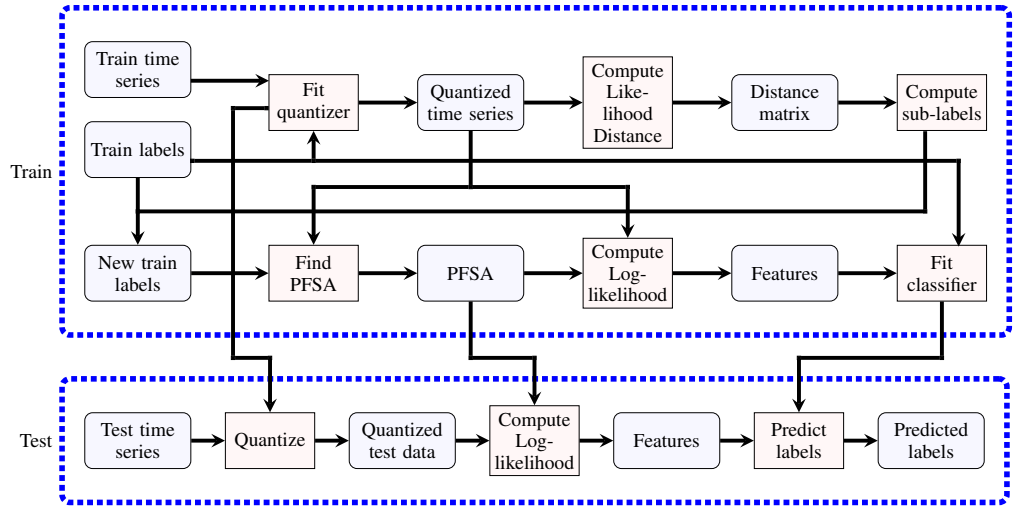

Figure 1: Flow chart of the **ClusteredHMMClassifier** on a continuous-valued dataset.

Algorithm 1: PFSA Log-likelihood $L(x, G)$

Data: A PFSA $G = (Q, \Sigma, \delta, \widetilde{\pi})$ and a sequence x of length n.
Result: Log-likelihood of G generating x

Get the stationary distribution $\mathbf{p}_G$ as the left eigenvector of Π_G of eigenvalue 1;
Let $\mathbf{p}$ be the current distribution on states, and initialize it with $\mathbf{p}_G$;
Let L be the log-likelihood of G generating x and initialize it with 0;
for *each symbol* σ *in* x **do**
  Get the current distribution on symbols $\phi = \mathbf{p}_G^T \widetilde{\Pi}_G$;
  Update $L = L - \log \phi(\sigma)$;
  Let $\mathbf{p}_{\text{new}}$ be the new distribution on states, and initialize all its entries with 0;
  for *each state* $q \in Q$ **do**
    Let the next the state $q_{\text{new}} = \delta(q, \sigma)$;
    Let $\mathbf{p}_{\text{new}}(q_{\text{new}}) = \mathbf{p}_{\text{new}}(q_{\text{new}}) + \mathbf{p}(q)\widetilde{\pi}(q, \sigma)$;
  end
  Update $\mathbf{p}$ with $\mathbf{p}_{\text{new}} / \|\mathbf{p}_{\text{new}}\|_1$;
end
Let $L = L/n$;
return L;

Theorem 1 (Convergence of Log-likelihood)**.** *Let G and H be two irreducible PFSA, and let $x \in \Sigma^d$ be a sequence generated by G. Then we have $L(x, H) \to \mathcal{H}(G) + \mathcal{D}_{KL}(G \,\|\, H)$, in probability as $d \to \infty$.*

LikelihoodDistance: Log-likelihood Distance

A (pseudo-)distance between two sequences is calculated by choosing a set of basis PFSA $\mathcal{G} = \{G_1, \ldots, G_k\}$ and mapping the sequence x to the vector $\mathbf{v}_x = (L(x, G_1), \ldots, L(x, G_k))$, where $L(x, G)$ is the log-likelihood of G generating x as defined in Defn. 5. The distance between a pair of sequences can be any valid distance between their coordinates (See Alg. 1 and Alg. 3). For unsupervised learning, $\mathcal{G}$ can be a pre-determined set of PFSA that is well-separated in a certain sense, for example, KL divergence. For supervised problems, $\mathcal{G}$ consists of one or more PFSA inferred from each class of the training dataset using algorithm **genESeSS** (Chattopadhyay and Lipson 2013).

Algorithm 2: Compute_likelihoods

Data:
- A dataset $X = \{x_1, \ldots, x_{|X|}\}$ over alphabet Σ;
- A set of PFSA $\mathcal{G} = \{G_1, \ldots, G_{|\mathcal{G}|}\}$, each PFSA over alphabet Σ.

Result: Feature matrix of X.

return the $|X| \times |\mathcal{G}|$ matrix with the (i, j)-th entry being $L(x_i, G_j)$;

Algorithm 3: LikelihoodDistance

Data:
- A collection X of sequences over alphabet Σ;
- A set $\mathcal{G}$ of basis PFSA over the same alphabet.

Result: A distance matrix of dimension $|X| \times |X|$

Let $F =$ **Compute_likelihoods**$(X, \mathcal{G})$;
Let D be a matrix of dimension $|X| \times |X|$;
for $i = 1, \ldots, |X|$ **do**
  Let $\mathbf{v}_i$ be the i-th row of F;
  for $j = 1, \ldots, |X|$ **do**
    Let $\mathbf{v}_j$ be the j-th row of F;
    Let the (i, j)-entry of D be $d(\mathbf{v}_i, \mathbf{v}_j)$;
  end
end
return D;

SymbolicDerivative

Definition 6 (Empirical Symbolic Derivative)**.** For $x \in \Sigma^\star$ (*i.e.* x is a finite but unbounded sequence over Σ), the **empirical symbolic derivative** $\hat{\phi}_y^x$ of a subsequence y of x is a probability vector:

$$\forall \sigma \in \Sigma,\ \hat{\phi}_y^x(\sigma) = \frac{\text{number of subsequence } y\sigma \text{ in } x}{\text{number of subsequence } y \text{ in } x}$$

where $y\sigma$ is teh concatenation of y and σ.

SymbolicDerivative maps each sequence in x over alphabet Σ to a feature vector composed of empirical symbolic derivatives. More specifically, let $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$, the feature vector of the subsequence y is defined to be the k-dimensional vector $\phi_y^x = \left(\phi_y^x(\sigma_1), \ldots, \phi_y^x(\sigma_{k-1})\right)$. Let l be a fixed length, the feature vector $\mathbf{v}_x$ of a sequence x is

Algorithm 4: InferredHMMLikelihood featurization

Data:
- Dataset $X = \left(X^{\text{train}}, X^{\text{test}}\right)$ over alphabet Σ;
- Labels $L^{\text{train}} = \left(l_1, \ldots, l_{|X^{\text{train}}|}\right)$;

Result: Feature matrices of X^{train} and X^{test}

Let $\mathcal{L}$ be the set of unique labels;
Let $\mathcal{G} = \emptyset$ be the set of class PFSA;
for *each* l *in* $\mathcal{L}$ **do**
    Let $X_l = \{x_i : y_i = l\}$;
    Add PFSA $G_l =$ **genESeSS**(X_l) to $\mathcal{G}$;
end
Let $F^{\text{train}} =$ **Compute_likelihoods** $\left(X^{\text{train}}, \mathcal{G}\right)$;
Let $F^{\text{test}} =$ **Compute_likelihoods** $\left(X^{\text{test}}, \mathcal{G}\right)$;
return F^{train} and F^{test};

given by the concatenation of ϕ_y^x for all y with length less than or equal to l.

InferredHMMLikelihood

InferredHMMLikelihood infers a PFSA G_i from each class i of a k-class classification dataset X and returns **LikelihoodDistance** matrix produced with X and $\mathcal{G} = \{G_1, \ldots, G_k\}$ (See Alg. 9).

Quantizer

The simplest approach to turn a continuous sequence to a symbolic one with alphabet size k is by choosing $k-1$ cut-off points $p_1 < p_2 < \cdots < p_{k-1}$. With $p_0 = -\infty$ and $p_k = +\infty$, we can replace a data point p in the continuous sequence with symbol i if $p \in [p_i, p_{i+1})$. The set of cut-off points is called a *partition*. A commonly used principle for choosing a partition is entropy maximization, in which the p_is are chosen so that there are as equal as possible numbers of data points falling in each $[p_i, p_{i+1})$. In the current implementation of **Quantizer**, users can specify the following quantization parameters: 1) Alphabet size (default is 2 and 3); 2) Whether to take derivative first (default is to try both); 3) Whether to standardize (default is to try both); 4) Maximum number of quantization schemes returned (default is to return all). For a fixed alphabet size k, by default, **Quantizer** produces $2 \cdot 2 \cdot (2k+1)$ quantization schemes, where $2k+1$ is because of the k quantizations at 90% maximum entropy, k at 95%, and 1 with maximum entropy. Hence, by default, **Quantizer** generates totally 48 quantization schemes.

With quantization schemes $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_m\}$, we get m discrete datasets $X_1, \ldots, X_m$ from a continuous dataset X. Let F_i, $i = 1, \ldots, m$, be the feature matrix produced by either **SymbolicDerivative** or **InferredHMMLikelihood** from X_i, the feature matrix of X is given by $F = (F_1, \ldots, F_m)$. We define the **LikelihoodDistance** matrix D of a continuous dataset to be $(1/m)\sum_{i=1}^{m} D_i$, where D_i is the **LikelihoodDistance** matrix of X_i. For supervised learning, **Quantizer** uses **separation ratio** to measure the quality of quantization schemes. Let l_i be the class label of the i-th sequence in X, the mean inter-class distance s and mean intra-class distance d are defined by

$$s = \frac{\sum_{i,j} \delta_{l_i l_j} D_{i,j}}{\sum_{i,j} \delta_{l_i l_j}}, \quad d = \frac{\sum_{i,j} \left(1 - \delta_{l_i l_j}\right) D_{i,j}}{\sum_{i,j} \left(1 - \delta_{l_i l_j}\right)},$$

respectively, where $\delta_{ab} = 1$ if $a = b$ and 0 if otherwise. The bigger the separation ratio $r_{\mathcal{Q}} = d/s$, the better separation $\mathcal{Q}$ produces between different classes of the dataset.

ClusteredHMMClassifier

ClusteredHMMClassifier proceeds by first clustering sequences from the training dataset belonging to the same class using **LikelihoodDistance** and a standard specified clustering algorithm, and then infers a PFSA from each *subclass*. A feature matrix is generated from the inferred PFSA, for the final classifier training (See Alg. 19).

Algorithm 5: ClusteredHMMClassifier: Cluster InferredHMMLikelihood Classification

Data:
- Dataset $\left(X^{\text{train}}, X^{\text{test}}\right)$;
- Labels $L^{\text{train}} = (l_1, \ldots, l_n)$;
- Optional quantization parameters;
- A set of basis PFSA $\mathcal{G}$ for **LikelihoodDistance**;
- A clustering algorithm `clu`;
- A classification algorithm `clf`.

Result: Predicted labels for X^{test}.

Let quantization schemes $\mathcal{Q}_1, \ldots, \mathcal{Q}_m =$ **Quantizer** $\left(X^{\text{train}}, L^{\text{train}}\right)$;
Let $\mathcal{L}$ be the set of unique labels;
for $i = 1, \ldots, m$ **do**
    Let $X_i^{\text{train}}, X_i^{\text{test}} = \mathcal{Q}_i\left(X^{\text{train}}, X^{\text{test}}\right)$;
    for *each* $l \in \mathcal{L}$ **do**
        Let $X_{i,l}^{\text{train}}$ be the subset of X_i^{train} with label l;
        Let $D =$ **LikelihoodDistance** $\left(X_{i,l}^{\text{train}}, \mathcal{G}\right)$;
        Let $\left\{X_{i,l,c}^{\text{train}} : c = 1, \ldots, C\right\} =$ `clu`(D);
        for $c = 1, \ldots, C$ **do**
            Assign a **new label** l_c to sequences in $X_{i,l,c}^{\text{train}}$;
        end
    end
    Let L_i^{new} be the new labels;
    Let $F_i^{\text{train}}, F_i^{\text{test}} =$ **InferredHMMLikelihood** $\left(X_i^{\text{train}}, X_i^{\text{test}}, L_i^{\text{new}}\right)$;
end
Let $F^{\text{train}} = \left(F_1^{\text{train}}, \ldots, F_m^{\text{train}}\right)$;
Let $F^{\text{test}} = \left(F_1^{\text{test}}, \ldots, F_m^{\text{test}}\right)$;
Train `clf` with F^{train};
return prediction `clf` $\left(F^{\text{test}}\right)$;

We show the flow chart for **ClusteredHMMClassifier** in Fig. 1, and compare the performance of **ClusteredHMMClassifier** with **SymbolicDerivative** and **DTW** in Tab. 1.

Performance Comparison

We first compare runtimes of dynamic time warping (**DTW**) (Berndt and Clifford 1994), **LikelihoodDistance** on a synthetic symbolic dataset. We implement both algorithms in C++ and use the implementation documented in (Rakthanmanon et al. 2012) for **DTW**. The synthetic dataset contains 200 randomly generated bi-class classification sub-datasets with 25 sequences of length 500 in each class. Sequences in each class are sample paths from a randomly

generated hidden Markov model with binary output. For comparing performances, we use the separation ratio defined in discussion of **Quantizer**. For **DTW**, we try window sizes $5, 10, 20, 30, 40, 50$, and 100. The average run time of **LikelihoodDistance** is $.042$ second. **LikelihoodDistance** achieves an average separation ratio that is comparable to DTW of window size 30 but with run time 2 magnitude smaller. The run time of **DTW** with window size 30 and **LikelihoodDistance** on a synthetic dataset constructed as before but with sequence lengths ranging from 200 to 2000 with 200 increment.

Performance Comparison on UCR Datasets

In Tab. 1, we compared the error rates of **DTW**, **SymbolicDerivative**, and **ClusteredHMMClassifier** on datasets from the UCR Time series classification archive. We did the comparison on all datasets that contain at least 50 time series per class for the comparison since the inference algorithm **genESeSS** does need a moderate sample size to work optimally. Compared to the **DTW** baseline, **SymbolicDerivative** and **ClusteredHMMClassifier** perform at or better on 29 out of the 44 of datasets. **SymbolicDerivative** is a featurization algorithm and requires a standard classification algorithm to be specified. **ClusteredHMMClassifier** also needs user-specified clustering and classification algorithm. The classification and clustering algorithms we used were selected from the algorithms available in the **scikit-learn** (Pedregosa et al. 2011) package: **RandomForestClassifier**,**AdaBoostClassifier**,**GradientBoostingClassifier**,and **SVC** And for clustering algorithms, we considered **KMeans** and **AffinityPropagation**.

Clustering based on infectious disease outbreaks

To illustrate a particularly relevant applicability of to the modeling of bio-physical systems, we calculate a county-wise disease risk factor from historical outbreaks of infectious diseases. Using a comprehensive database of insurance claims records (Truven Marketscan database of ~ 150 million patients in the US tracked over approximately a decade (Hansen 2017)), we obtain weekly county-wise time series of diagnosed cases for specific infections over a period of 471 weeks spanning from 2003 to 2011.Considering two infections, influenza (flu) and Styphlococcus Aureus (staph) here, we use **LikelihoodDistance** to get distance matrix D_{flu} and D_{staph}, with which we cluster the counties using standard spectral clustering. Selecting the cluster with the highest average case counts, we run **genESeSS** on the time series from each cluster, and generate PFSA models G_{flu} and G_{staph}. By Thm. 1, we have

$$\mathcal{D}_{\text{KL}}\left(G \,\|\, H\right) \leftarrow L(x, H) - \mathcal{H}(G) \tag{1}$$

Hence, for a particular disease d, a county from which PFSA G_d has a smaller divergence has a higher risk. Approximating the entropy rate in Eq. (1) by the empirical entropy of time series x_c of a county c, we can evalute the risk of county c with respect to disease d by $R_d(c) = -\left(L(x_c, G_d) - \hat{E}(x_c)\right)$, where $L(x_c, G_d)$ can be evaluated by **Compute_likelihoods** and $\hat{E}(x_c)$ is the binary entropy of the frequency of 0 in x_c. The risk factor obtained in this manner from flu was used to the development a COVID-19 cases forecast model that achieves the smallest mean absolute error in one-week-ahead forecasts among the top performing teams from the COVID-19 Forecast Community (team UCHICAGOCHATTOPADHYAY-UnIT, https://covid19forecasthub.org/community/) In Fig. 3a and b, we show results for flu and staph, respectively. The chloropleth in i. of each panel is the averaged normalized case count over the weeks. In ii., we show G_{flu} and G_{staph}. The chloropleth of the computed risk R is shown in iii.

Acknowledgement

This work is funded in part by the Defense Advanced Research Projects Agency (HR00111890043/P00004). The claims made in this study do not necessarily reflect the position or the policy of the US Government.

References

Berndt, D. J.; and Clifford, J. 1994. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, 359–370. Seattle, WA.

Bondy, J.; and Murty, U. 2008. Graph theory (2008). *Grad. Texts in Math* .

Chattopadhyay, I.; Huang, Y.; and Evans, J. 2020. Deep Learning Without Neural Networks: Fractal-nets for Rare Event Modeling. doi:10.21203/rs.3.rs-86045/v1. URL https://doi.org/10.21203/rs.3.rs-86045/v1.

Chattopadhyay, I.; and Lipson, H. 2013. Abductive learning of quantized stochastic processes with probabilistic finite automata. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371(1984): 20110543.

Chen, L.; Özsu, M. T.; and Oria, V. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 491–502. ACM.

Cover, T. M.; and Thomas, J. A. 2012. *Elements of information theory*. John Wiley & Sons.

Hansen, L. 2017. The Truven health MarketScan databases for life sciences researchers. *Truven Health Ananlytics IBM Watson Health* .

Lin, J.; Keogh, E.; Lonardi, S.; and Chiu, B. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2–11. ACM.

Möller-Levet, C. S.; Klawonn, F.; Cho, K.-H.; and Wolkenhauer, O. 2003. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *International Symposium on Intelligent Data Analysis*, 330–340. Springer.

Navarro, G. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)* 33(1): 31–88.

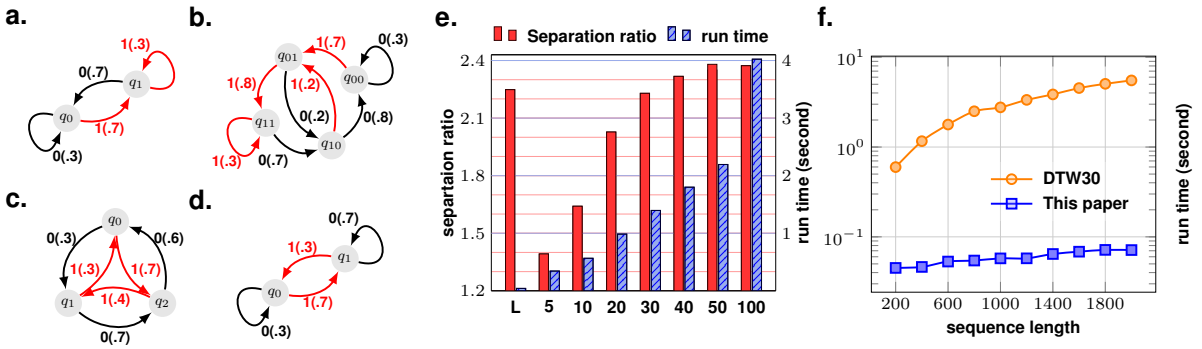

Figure 2: **Panel a-d.** Four pre-specified PFSA models chosen to act as the basis to estimate similarity between stochastic sample paths. An edge connecting state q to q' is labeled as $\sigma\left(\widetilde{\pi}(q,\sigma)\right)$ if $\delta(q,\sigma) = q'$ (See Defn. 1). **Panel e.** Performance and run time comparisons of **LikelihoodDistance** and DTW on a synthetic dataset. We denote the **LikelihoodDistance** by L and DTW by their window size in Panel e. The average run time of of **LikelihoodDistance** is .042 second. **Panel f.** Run time v.s. sequence length comparison between DTW30 and the **LikelihoodDistance** distance.

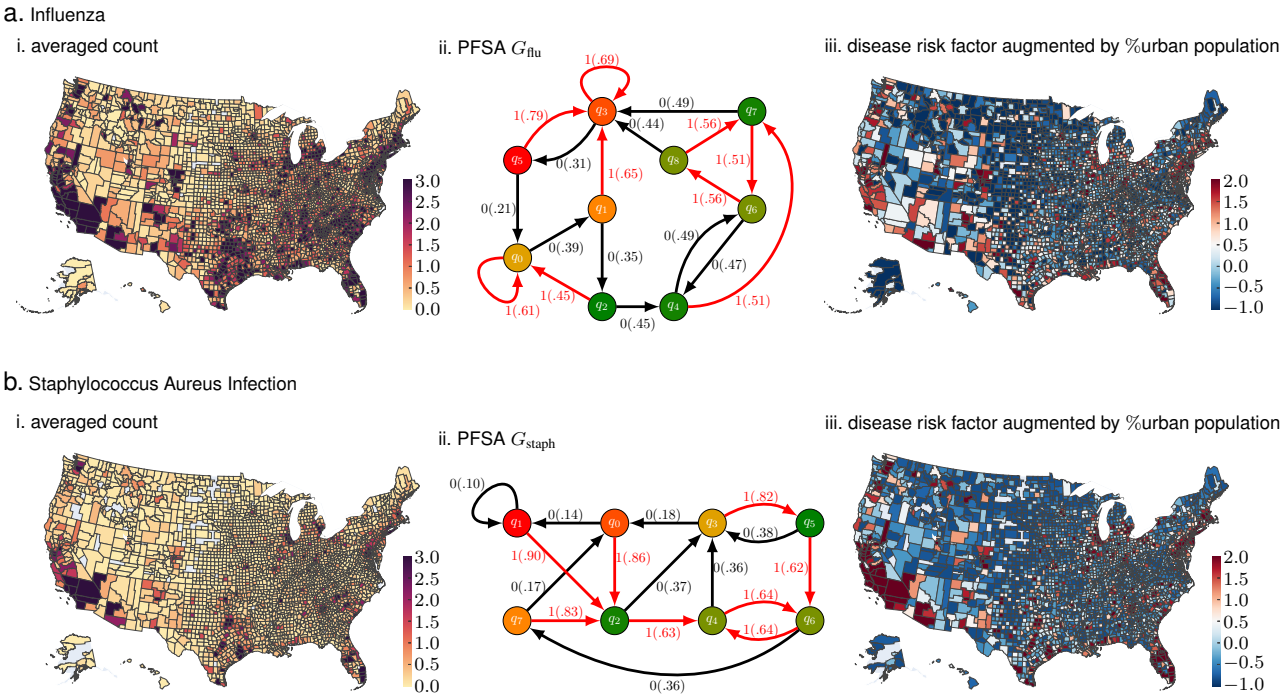

Figure 3: Use of **LikelihoodDistance** to quantify risk phenotype of US counties to specific infections

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Petitjean, F.; Ketterlin, A.; and Gançarski, P. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition* 44(3): 678–693.

Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; and Keogh, E. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 262–270.

Vidyasagar, M. 2014. *Hidden markov processes: Theory and applications to biology*, volume 44. Princeton University Press.