ACTOR-CRITIC APPROACH FOR TEMPORAL PREDICTIVE CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Due to the wider availability of modern electronic health records (EHR), patient care data is often being stored in the form of time-series. Clustering such time-series data is crucial for patient phenotyping, anticipating patients' prognoses by identifying "similar" patients, and designing treatment guidelines that are tailored to homogeneous patient subgroups. In this paper, we develop a deep learning approach for clustering time-series data, where each cluster comprises patients who share similar future outcomes of interest (e.g., adverse events, the onset of comorbidities, etc.). The clustering is carried out by using our novel loss functions that encourage each cluster to have homogeneous future outcomes. We adopt actor-critic models to allow "back-propagation" through the sampling process that is required for assigning clusters to time-series inputs. Experiments on two real-world datasets show that our model achieves superior clustering performance over state-of-the-art benchmarks and identifies meaningful clusters that can be translated into actionable information for clinical decision-making.

1 INTRODUCTION

Chronic diseases – such as cystic fibrosis, dementia, and diabetes – are heterogeneous in nature, with widely differing outcomes even in narrow patient subgroups. Disease progression manifests through a broad spectrum of clinical factors, collected as a sequence of measurements over time in electronic health records (EHR), which give a rise to the discovery of complex progression patterns among patients (Samal et al., 2011). For example, cystic fibrosis evolves over a long timespan, allowing for the development of related comorbidities and bacterial infections, and creating distinct behaviors/responses to therapeutic interventions, which in turn make the survival and quality of life substantially different (Ramos et al., 2017). Identifying patient subgroups with similar progression patterns can be advantageous for understanding such heterogeneous underlying diseases. This allows clinicians to anticipate patients' prognoses by comparing "similar" patients for the purpose of designing treatment guidelines that are tailored to homogeneous patient subgroups (Zhang et al., 2019).

Temporal clustering has recently been used as a data-driven framework to partition patients with time-series observations into a set of clusters (i.e., subgroups of patients). Recent research has typically focused on either finding fixed-length and low-dimensional representations (Zhang et al., 2019; Rusanov et al., 2016) or on modifying the similarity measure (Giannoula et al., 2018; Luong and Chandola, 2017) both in an attempt to apply conventional clustering algorithms (e.g., K-means (Lloyd, 1982)) to time-series observations. However, clusters identified from these approaches these approaches are purely unsupervised – they do not account for each patient's observed outcome (e.g., adverse events, the onset of comorbidities, etc.) – which leads to heterogeneous clusters if the clinical presentation of the disease differs even for similar patients. Thus, a common prognosis in each cluster remains unknown which can mystify the understanding of the underlying disease progression (Boudier et al., 2019). For instance, patients who appear to have similar time-series observations may develop different sets of comorbidities in the future which, in turn, require different treatment guidelines to reduce such risks (Wami et al., 2013). To overcome this limitation, we focus on *predictive clustering* (Blockeel et al., 2017) which combines prediction with clustering. Therefore, the cluster assignments are optimized such that patients in a cluster share similar future outcomes to provide a prognostic value.

In this paper, we propose an actor-critic approach for temporal predictive clustering, which we call AC-TPC.[1] Our model consists of three neural networks – an *encoder*, a *selector*, and a *predictor* – and a set of centroid candidates. In particular, the encoder maps an input time-series into a latent encoding; the selector utilizes the encoding and assigns a cluster to which the time-series belongs to via a sampling process; and the predictor estimates the future outcome distribution conditioned on either the encoding or the centroid of the selected cluster. The following three contributions render our model able to identify the predictive clusters. First, to encourage each cluster to have homogeneous future outcomes, we define a clustering objective based on the Kullback-Leibler (KL) divergence between the predictor's output given the input time series, and the predictor's output given estimated cluster assignments. Second, we transform solving the non-trivial combinatorial problem of identifying cluster into iteratively solving two sub-problems: optimization of the cluster assignments and optimization of the cluster centroids. Finally, we allow "back-propagation" through the sampling process by adopting the training of actor-critic models (Konda and Tsitsiklis, 2000).

Throughout the experiments, we show significant performance improvements over the state-of-the-art clustering methods on two real-world medical datasets. Then, to demonstrate the practical significance of our model, we consider a more realistic scenario where the future outcomes of interest are high-dimensional – such as, development of multiple comorbidities in the next year – and interpreting all possible combinations is intractable. Our experiments show that the proposed model can identify meaningful clusters that can be translated into actionable information for clinical decision-making.

2 Problem Formulation

Let $\mathbf{X} \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be random variables for an input feature and an output label (i.e., one or a combination of future outcome(s) of interest) with a joint distribution p_{XY} (and marginal distributions are p_X and p_Y, respectively) where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the label space. Here, we focus our description on C-class classification tasks, i.e., $\mathcal{Y} = \{1, \cdots, C\}$.[2] We are given a time-series dataset $\mathcal{D} = \{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^N$ comprising sequences of realizations (i.e., observations) of the pair $(\mathbf{X}, Y)$ for N patients. Here, $(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}$ is a sequence of T^n observation pairs that correspond to patient n and $t \in \mathcal{T}^n \triangleq \{1, \cdots, T^n\}$ denotes the time stamp at which the observations are made.From this point forward, we omit the dependency on n when it is clear in the context and denote $\mathbf{x}_{1:t} = (\mathbf{x}_1, \cdots, \mathbf{x}_t)$ for ease of notation.

Our aim is to identify a set of K *predictive clusters*, $\mathcal{C} = \{\mathcal{C}(1), \cdots, \mathcal{C}(K)\}$, for time-series data. Each cluster consists of homogeneous data samples, that can be represented by its centroid, based on a certain similarity measure. There are two main distinctions from the conventional notion of clustering. First, we treat subsequences of each times-series as data samples and focus on partitioning $\{\{\mathbf{x}_{1:t}^n\}_{t=1}^{T^n}\}_{n=1}^N$ into $\mathcal{C}$. Hence, we define a cluster as $\mathcal{C}(k) = \{\mathbf{x}_{1:t}^n | t \in \mathcal{T}^n,\ s_t^n = k\}$ for $k \in \mathcal{K} \triangleq \{1, \cdots, K\}$ where $s_t^n \in \mathcal{K}$ is the cluster assignment for a given $\mathbf{x}_{1:t}^n$. This is to flexibly update a patient's cluster assignment (in real-time) to which a patient belongs as new observations are being accrued over time. Second, we define the similarity measure with respect to the label distribution and associate it with clusters to provide a prognostic value. More specifically, we want the distribution of output labels for subsequences in each cluster to be homogeneous so that it can be well-represented by the centroid of that cluster. Let S be a random variable for the cluster assignment – that depends on a given subsequence $\mathbf{x}_{1:t}$ – and $Y|S = k$ be a random variable for the output given cluster k. Then, such property of predictive clustering can be achieved by minimizing the following Kullback-Leibler (KL) divergence:

$$KL\big(Y_t|\mathbf{X}_{1:t} = \mathbf{x}_{1:t} \big\| Y_t|S_t = k\big) \quad \text{for } \mathbf{x}_{1:t} \in \mathcal{C}(k) \tag{1}$$

where $KL\big(Y_t|\mathbf{X}_{1:t} = \mathbf{x}_{1:t} \big\| Y_t|S_t = k\big) = \int_{y \in Y} p(y|\mathbf{x}_{1:t}) \big(\log p(y|\mathbf{x}_{1:t}) - \log p(y|s_t)\big) dy$. Here, $p(y|\mathbf{x}_{1:t})$ and $p(y|s_t)$ are the label distributions conditioned on a subsequence $\mathbf{x}_{1:t}$ and a cluster assignment s_t, respectively. Note that (1) achieves its minimum when the two distributions are equivalent.

[1] Source code available at `https://github.com/ICLR2020-ACTPC/ACTPC_submission.git`

[2] In this paper, we focus our description on C-class classification task, i.e., $\mathcal{Y} = \{1, \cdots, C\}$; in Appendix A, we discuss simple modifications of our model for regression and M-dimensional binary classification tasks, i.e., $\mathcal{Y} = \mathbb{R}$ and $\mathcal{Y} = \{0, 1\}^M$, respectively.

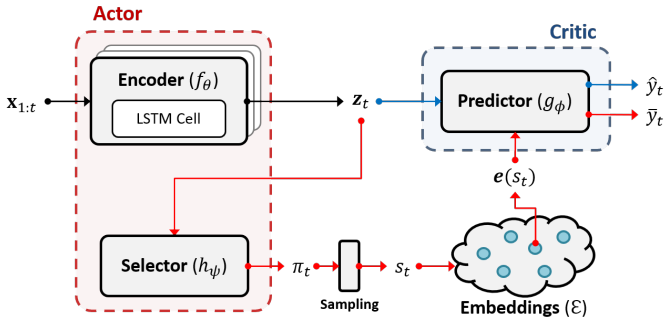


Figure 1: A block diagram of AC-TPC. The red line denotes the procedure of estimating $p(y|S_t = s_t)$ which includes a sampling process and the blue line denotes that of estimating $p(y|\mathbf{X}_{1:t} = \mathbf{x}_{1:t})$.

Finally, we establish our goal as identifying a set of predictive clusters $\mathcal{C}$ that optimizes the following objective:

$$\underset{\mathcal{C}}{\text{minimize}} \sum_{k \in \mathcal{K}} \sum_{\mathbf{x}_{1:t} \in \mathcal{C}(k)} KL\big(Y_t|\mathbf{X}_{1:t} = \mathbf{x}_{1:t} \big\| Y_t|S_t = k\big). \tag{2}$$

Unfortunately, the optimization problem in (2) is highly non-trivial. We need to estimate the objective function in (2) while solving a non-convex combinatorial problem of finding the optimal cluster assignments and cluster centroids.

3 Actor-Critic Approach for Temporal Predictive Clustering

To effectively estimate the objective function in (2), we introduce three networks – an *encoder*, a *selector*, and a *predictor* – and an *embedding dictionary* as illustrated in Figure 1. These components together provide the cluster assignment and the corresponding centroid based on a given sequence of observations and enable us to estimate the probability density $p(y|s_t)$. More specifically, we define each component as follows:

- The *encoder*, $f_\theta : \prod_{i=1}^{t} \mathcal{X} \rightarrow \mathcal{Z}$, is a RNN (parameterized by θ) that maps a (sub)sequence of a time-series $\mathbf{x}_{1:t}$ to a latent representation (i.e., encoding) $\mathbf{z}_t \in \mathcal{Z}$ where $\mathcal{Z}$ is the latent space.
- The *selector*, $h_\psi : \mathcal{Z} \rightarrow \Delta^{K-1}$, is a fully-connected network (parameterized by ψ) that provides a probabilistic mapping to a categorical distribution from which the cluster assignment $s_t \in \mathcal{K}$ is being sampled.
- The *predictor*, $g_\phi : \mathcal{Z} \rightarrow \Delta^{C-1}$, is a fully-connected network (parameterized by ϕ) that estimates the label distribution given the encoding of a time-series or the centroid of a cluster.
- The *embedding dictionary*, $\mathcal{E} = \{\mathbf{e}(1), \cdots, \mathbf{e}(K)\}$ where $\mathbf{e}(k) \in \mathcal{Z}$ for $k \in \mathcal{K}$, is a set of cluster centroids lying in the latent space which represents the corresponding cluster.

Here, $\Delta^{D-1} = \{\mathbf{q} \in [0,1]^K : q_1 + \cdots + q_D = 1\}$ is a $(D-1)$-simplex that denotes the probability distribution for a D-dimensional categorical (class) variable.

At each time stamp t, the *encoder* maps an input (sub)sequence $\mathbf{x}_{1:t}$ into a latent encoding $\mathbf{z}_t \triangleq f_\theta(\mathbf{x}_{1:t})$. Then, based on the encoding $\mathbf{z}_t$, the cluster assignment s_t is drawn from a categorical distribution that is defined by the *selector* output, i.e., $s_t \sim Cat(\pi_t)$ where $\pi_t = [\pi_t(1), \cdots, \pi_t(K)] \triangleq h_\psi(\mathbf{z}_t)$. Once the assignment s_t is chosen, we allocate the latent encoding $\mathbf{z}_t$ to an embedding $\mathbf{e}(s_t)$ in the *embedding dictionary* $\mathcal{E}$. Since the allocated embedding $\mathbf{e}(s_t)$ corresponds to the centroid of the cluster to which $\mathbf{x}_{1:t}$ belongs, we can, finally, estimate the density $p(y|s_t)$ in (2) as the output of the *predictor* given the embedding $\mathbf{e}(s_t)$, i.e., $\bar{y}_t \triangleq g_\phi(\mathbf{e}(s_t))$.

3.1 Loss Functions

In this subsection, we define loss functions to achieve our objective in (2); the details of how we train our model will be discussed in the following subsection.

Predictive Clustering Loss: Since finding the cluster assignment of a given sequence is a probabilistic problem due to the sampling process, the objective function in (2) must be defined as an expectation over the cluster assignment. Thus, we can estimate solving the objective problem in (2) as minimizing the following loss function:

$$\mathcal{L}_1(\theta, \psi, \phi, \mathcal{E}) = \mathbb{E}_{\mathbf{x}, y \sim p_{XY}} \left[\sum_{t=1}^{T} \mathbb{E}_{s_t \sim Cat(\pi_t)} \left[\ell_1(y_t, \bar{y}_t) \right] \right] \tag{3}$$

where $\ell_1(y_t, \bar{y}_t) = -\sum_{c=1}^{C} y_t^c \log \bar{y}_t^c$. Here, we slightly abuse the notation and denote $y = [y^1 \cdots y^C]$ as the one-hot encoding of y, and y^c and $\bar{y}^c$ indicates the c-th component of y and $\bar{y}$, respectively. It is worth highlighting that minimizing ℓ_1 is equivalent to minimizing the KL divergence in (2) since the former term of the KL divergence is independent of our optimization procedure.

One critical question that may arise is how to avoid trivial solutions in this unsupervised setting of identifying the cluster assignments and the centroids (Yang et al., 2017). For example, all the embeddings in $\mathcal{E}$ may collapse into a single point or the selector simply assigns equal probability to all the clusters regardless of the input sequence. In both cases, our model will fail to correctly estimate $p(y|s_t)$ and, thus, end up finding a trivial solution. To address this issue, we introduce two auxiliary loss functions that are tailored to address this concern. It is worth highlighting that these loss functions are not subject to the sampling process and their gradients can be simply back-propagated.

Sample-Wise Entropy of Cluster Assignment: To motivate sparse cluster assignment such that the selector ultimately selects one dominant cluster for each sequence, we introduce sample-wise entropy of cluster assignment which is given as

$$\mathcal{L}_2(\theta, \psi) = \mathbb{E}_{\mathbf{x} \sim p_X} \left[-\sum_{t=1}^{T} \sum_{k \in \mathcal{K}} \pi_t(k) \log \pi_t(k) \right] \tag{4}$$

where $\pi_t = [\pi_t(1) \cdots \pi_t(K)] = h_\psi(f_\theta(\mathbf{x}_{1:t}))$. The sample-wise entropy achieves its minimum when π_t becomes an one-hot vector.

Embedding Separation Loss: To prevent the embeddings in $\mathcal{E}$ from collapsing into a single point, we define a loss function that encourages the embeddings to represent different label distributions, i.e., $g_\phi(\mathbf{e}(k))$ for $k \in \mathcal{K}$, from each other:

$$\mathcal{L}_3(\mathcal{E}) = -\sum_{k \neq k'} \ell_1(g_\phi(\mathbf{e}(k)), g_\phi(\mathbf{e}(k'))) \tag{5}$$

where ℓ_1 is reused to quantify the distance between label distributions conditioned on each cluster. We minimize (5) when updating the embedding vectors $\mathbf{e}(1), \cdots, \mathbf{e}(K)$.

3.2 Optimization

The optimization problem in (2) is a non-convex combinatorial problem because it comprises not only minimizing the KL divergence but also finding the optimal cluster assignments and centroids. Hence, we propose an optimization procedure that iteratively solves two subproblems: i) optimizing the three networks – the encoder, selector, and predictor – while fixing the embedding dictionary and ii) optimizing the embedding dictionary while fixing the three networks. Pseudo-code of AC-TPC can be found in Appendix F.

3.2.1 Optimizing the Three Networks – f_θ, h_ψ, and g_ϕ

As discussed in the previous subsection, finding the predictive clusters incorporates the sampling process which is non-differentiable. Thus, to allow for "back-propagation", we use the training procedure of actor-critic models (Konda and Tsitsiklis, 2000). More specifically, we view the combination of the encoder (f_θ) and the selector (h_ψ) as the "actor" parameterized by $\omega_A = [\theta, \psi]$, and the predictor ($g_\phi$) as the "critic". These two networks are trained iteratively. The critic takes as input the output of the actor and determines the corresponding loss. This, in turn, enables the actor to change its output distribution to minimize such loss. Thus, it is important for the critic to

perform well on the updated output of the actor while it is important for the actor to perform well on the updated loss estimation. As such, the parameters for the actor and the critic need to be updated iteratively.

Given the embedding dictionary $\mathcal{E}$ fixed (we omit the dependency on $\mathcal{E}$), we train the actor, i.e., the encoder and the selector, by minimizing a combination of the predictive clustering loss $\mathcal{L}_1$ and the entropy of cluster assignments $\mathcal{L}_2$, which is given by

$$\mathcal{L}_A(\theta, \psi, \phi) = \mathcal{L}_1(\theta, \psi, \phi) + \alpha \mathcal{L}_2(\theta, \psi) \tag{6}$$

where $\alpha \geq 0$ is a coefficient chosen to balance between the two losses. To derive the gradient of this loss with respect $\omega_A = [\theta, \psi]$, we utilize the ideas from actor-critic models (Konda and Tsitsiklis, 2000) as follows; please refer to Appendix B for the detailed derivation:

$$\begin{aligned} \nabla_{\omega_A} \mathcal{L}_A(\theta, \psi, \phi) &= \mathbb{E}_{\mathbf{x}, y \sim p_{XY}} \left[\nabla_{\omega_A} \left(\sum_{t=1}^{T} \mathbb{E}_{s_t \sim Cat(\pi_t)} \big[\ell_1(y_t, \bar{y}_t) \big] \right) \right] + \alpha \nabla_{\omega_A} \mathcal{L}_2(\theta, \psi) \\ &= \mathbb{E}_{\mathbf{x}, y \sim p_{XY}} \left[\sum_{t=1}^{T} \mathbb{E}_{s_t \sim Cat(\pi_t)} \big[\ell_1(y_t, \bar{y}_t) \nabla_{\omega_A} \log \pi_t(s_t) \big] \right] + \alpha \nabla_{\omega_A} \mathcal{L}_2(\theta, \psi). \end{aligned} \tag{7}$$

Note that since no sampling process is considered in $\mathcal{L}_2(\theta, \psi)$, we can simply derive $\nabla_{\omega_A} \mathcal{L}_2(\theta, \psi)$.

Iteratively with training the actor, we train the critic, i.e., the predictor, by minimizing the predictive clustering loss $\mathcal{L}_1$ as the following:

$$\mathcal{L}_C(\phi) = \mathcal{L}_1(\theta, \psi, \phi) \tag{8}$$

whose gradient with respect to ϕ can be givens as $\nabla_\phi \mathcal{L}_C(\phi) = \nabla_\phi \mathcal{L}_1(\theta, \psi, \phi)$. Note that since the critic is independent of the sampling process, the gradient can be simply back-propagated.

3.2.2 Optimizing the Cluster Centroids

Now, once the parameters for the three networks (θ, ψ, ϕ) are fixed (we omit the dependency on θ, ψ, and ϕ), we updated the embeddings in $\mathcal{E}$ by minimizing a combination of the predictive clustering loss $\mathcal{L}_1$ and the embedding separation loss $\mathcal{L}_3$, which is given by

$$\mathcal{L}_E(\mathcal{E}) = \mathcal{L}_1(\mathcal{E}) + \beta \mathcal{L}_3(\mathcal{E}) \tag{9}$$

where $\beta \geq 0$ is a coefficient chosen to balance between the two losses.

3.2.3 Initializing AC-TPC via Pre-Training

Since we transform the non-trivial combinatorial optimization problem in (2) into iteratively solving two sub-problems, initialization is crucial to achieve better optimization as has been shown in (Yang et al., 2017), which addressed a similar concern.

First, we pre-train the encoder and the predictor by minimizing the following loss function based on the predicted label distribution given the latent encodings of input sequences, i.e., $\hat{y}_t \triangleq g_\phi(\mathbf{z}_t) = g_\phi(f_\theta(\mathbf{x}_{1:t}))$:

$$\mathcal{L}_I(\theta, \phi) = \mathbb{E}_{\mathbf{x}, y \sim p_{XY}} \left[- \sum_{t=1}^{T} \ell_1(y_t, \hat{y}_t) \right]. \tag{10}$$

Minimizing (10) encourages the latent encoding to be enriched with information for accurately predicting the label distribution. Then, we perform K-means (other clustering method can be also applied) based on the learned representations to initialize the embeddings $\mathcal{E}$ and the cluster assignments $\{\{s_t^n\}_{t=1}^{T^n}\}_{n=1}^{N}$. Finally, we pre-train the selector h_ψ by minimizing the cross entropy treating the initialized cluster assignments as the true clusters.

4 Related Work

Temporal clustering, also known as time-series clustering, is a process of unsupervised partitioning of the time-series data into clusters in such a way that homogeneous time-series are grouped together

based on a certain similarity measure. Temporal clustering is challenging because i) the data is often high-dimensional – it consists of sequences not only with high-dimensional features but also with many time points – and ii) defining a proper similarity measure for time-series is not straightforward since it is often highly sensitive to distortions (Ratanamahatana et al., 2005). To address these challenges, there have been various attempts to find a good representation with reduced dimensionality or to define a proper similarity measure for times-series (Aghabozorgi et al., 2015).

Recently, Baytas et al. (2017) and Madiraju et al. (2018) proposed temporal clustering methods that utilize low-dimensional representations learned by RNNs. These works are motivated by the success of applying deep neural networks to find "clustering friendly" latent representations for clustering static data (Xie et al., 2017; Yang et al., 2017). In particular, Baytas et al. (2017) utilized a modified LSTM auto-encoder to find the latent representations that are effective to summarize the input time-series and conducted K-means on top of the learned representations as an ad-hoc process. Similarly, Madiraju et al. (2018) proposed a bidirectional-LSTM auto-encoder that jointly optimizes the reconstruction loss for dimensionality reduction and the clustering objective. However, these methods do not associate a target property with clusters and, thus, provide little prognostic value in understanding the underlying disease progression.

Our work is most closely related to SOM-VAE (Fortuin et al., 2019). This method jointly optimizes a static variational auto-encoder (VAE), that finds latent representations of input features, and a self-organizing map (SOM), that allows for mapping the latent representations into a more interpretable discrete representations, i.e., the embeddings. However, there are three key differences between our work and SOM-VAE. First, SOM-VAE aims at minimizing the reconstruction loss that is specified as the mean squared error between the original input and the reconstructed input based on the corresponding embedding. Thus, similar to the aforementioned methods, SOM-VAE neither associates future outcomes of interest with clusters. In contrast, we focus on minimizing the KL divergence between the outcome distribution given the original input sequence and that given the corresponding embedding to build association between future outcomes of interest and clusters. Second, to overcome non-differentiability caused by the sampling process (that is, mapping the latent representation to the embeddings), Fortuin et al. (2019) applies the gradient copying technique proposed by (van den Oord et al., 2017), while we utilize the training of actor-critic model (Konda and Tsitsiklis, 2000). Finally, while we flexibly model time-series using LSTM, SOM-VAE handles time-series by integrating a Markov model in the latent representations. This can be a strict assumption especially in clinical settings where a patient's medical history is informative for predicting his/her future clinical outcomes (Ranganath et al., 2016).

5 EXPERIMENTS

In this section, we provide a set of experiments using two real-world time-series datasets. We iteratively update the three networks – the encoder, selector, and predictor – and the embedding dictionary as described in Section 3.2. For the network architecture, we constructed the encoder utilizing a single-layer LSTM (Hochreiter and Schmidhuber, 1997) with 50 nodes and constructed the selector and predictor utilizing two-layer fully-connected network with 50 nodes in each layer, respectively. The parameters (θ, ψ, ϕ) are initialized by Xavier initialization (Glorot and Bengio, 2010) and optimized via Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.001 and keep probability 0.7. We chose the balancing coefficients $\alpha, \beta \in \{0.1, 1.0, 3.0\}$ utilizing grid search among the possible values; the effect of different loss functions are further investigated in the experiments. Here, all the results are reported using 5 random 64/16/20 train/validation/test splits.

5.1 REAL-WORLD DATASETS

We conducted experiments to investigate the performance of AC-TPC on two real-world medical datasets; detailed statistics of each dataset can be found in Appendix C:

- **UK Cystic Fibrosis registry (UKCF)**[3]**:** This dataset records annual follow-ups for 5,171 adult patients (aged 18 years or older) enrolled in the UK CF registry over the period from 2008 and 2015, with a total of 25,012 hospital visits. Each patient is associated with 89 variables (i.e., 11 static and 78 time-varying features), including information on demographics and genetic

[3] https://www.cysticfibrosis.org.uk/the-work-we-do/uk-cf-registry

mutations, bacterial infections, lung function scores, therapeutic managements, and diagnosis on comorbidities. We set the development of different comorbidities in the next year as the label of interest at each time stamp.

- **Alzheimer's Disease Neuroimaging Initiative (ADNI)**[4]**:** This dataset consists of 1,346 patients in the Alzheimer's disease study with a total of 11,651 hospital visits, which tracks the disease progression via follow-up observations at 6 months interval. Each patient is associated with 21 variables (i.e., 5 static and 16 time-varying features), including information on demographics, biomarkers on brain functions, and cognitive test results. We set predictions on the three diagnostic groups – normal brain functioning, mild cognitive impairment, and Alzheimer's disease – as the label of interest at each time stamp.

5.2 BENCHMARKS

We compare AC-TPC with clustering methods ranging from conventional approaches based on K-means to the state-of-the-art approaches based on deep neural networks. All the benchmarks compared in the experiments are tailored to incorporate time-series data as described below; please refer to Appendix D for more details:

- **Dynamic time warping followed by K-means**[5]: Dynamic time warping (DTW) is utilized to quantify pairwise distance between two variable-length sequences and, then, K-means is applied (denoted as **KM-DTW**).
- **K-means with deep neural networks**: To handle variable-length time-series data, we utilize our encoder and predictor that are trained based on (10) for dimensionality reduction; this is to provide fixed-length and low-dimensional representations for time-series. Then, we apply K-means on the latent encodings $\mathbf{z}$ (denoted as **KM-E2P ($\mathcal{Z}$)**) and on the predicted label distributions $\hat{y}$ (denoted as **KM-E2P ($\mathcal{Y}$)**), respectively.
- **Extensions of DCN**[6] (Yang et al., 2017): Since the DCN is designed for static data, we replace their static auto-encoder with a sequence-to-sequence network to incorporate time-series data (denoted as **DCN-S2S**).[7] In addition, to associated with the label distribution, we compare a DCN whose static auto-encoder is replaced with our encoder and predictor (denoted as **DCN-E2P**) to focus on dimensionality reduction while still preserving information for predicting the label.
- **SOM-VAE**[8] (Fortuin et al., 2019): Although this method is oriented towards visualizing input data via SOM, we compare SOM-VAE since it naturally clusters time-series data (denoted as **SOM-VAE**). In addition, we compare with a variation of SOM-VAE by replacing the decoder with our predictor in order to find embeddings that capture information for predicting the label (denoted as **SOM-VAE-P**). For both cases, we set the dimension of SOM to K.

Note that the label information is not provided for training KM-DTW, DCN-S2S, and SOM-VAE. We summarized major components of these benchmarks – in comparison with our model – in Table 6 in Appendix D.

5.3 PERFORMANCE METRICS

We applied the following three standard metrics for evaluating clustering performances: *purity score*, *normalized mutual information* (NMI) (Vinh et al., 2010), and *adjusted Rand index* (ARI) (Hubert and Arabie, 1985). More specifically, the purity score assesses how homogeneous each cluster is (ranges from 0 to 1 with 1 being a cluster consists of a single class), the NMI is an information theoretic measure of how much information is shared between the clusters and the labels that is adjusted for the number of clusters (ranges from 0 to 1 with 1 being a perfect clustering), and ARI

[4] https://adni.loni.usc.edu

[5] https://github.com/rtavenar/tslearn

[6] https://github.com/boyangumn/DCN

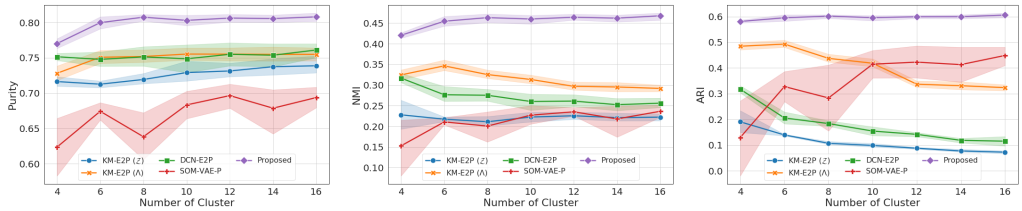
[7] This extension is a representative of recently proposed deep learning approaches for clustering of both static data (Xie et al., 2017; Yang et al., 2017) and time-series data (Baytas et al., 2017; Madiraju et al., 2018) since these methods are built upon the same concept – that is, applying deep networks for dimensionality reduction to conduct conventional clustering methods, e.g., K-means.

[8] https://github.com/ratschlab/SOM-VAE

Table 1: Performance Comparison on the UKCF and ADNI datasets.

Methods	UKCF			ADNI		
	Purity	NMI	ARI	Purity	NMI	ARI
KM-DTW	0.573±0.01*	0.010±0.01*	0.014±0.01*	0.566±0.02*	0.019±0.02*	0.006±0.02*
KM-E2P ($\mathcal{Z}$)	0.719±0.01*	0.211±0.01*	0.107±0.01*	0.736±0.03†	0.249±0.02	0.230±0.03†
KM-E2P ($\mathcal{Y}$)	0.751±0.01*	0.325±0.01*	0.440±0.02*	0.776±0.05	0.264±0.07	0.317±0.11
DCN-S2S	0.607±0.06*	0.059±0.08*	0.063±0.09*	0.567±0.02*	0.005±0.00*	0.000±0.01*
DCN-E2P	0.751±0.02*	0.275±0.02*	0.184±0.01*	0.749±0.06	0.261±0.05	0.215±0.06†
SOM-VAE	0.573±0.01*	0.006±0.00*	0.006±0.01*	0.566±0.02*	0.040±0.06*	0.011±0.02*
SOM-VAE-P	0.638±0.04*	0.201±0.05*	0.283±0.17†	0.586±0.06*	0.085±0.08*	0.038±0.06*
Proposed	**0.807±0.01**	**0.463±0.01**	**0.602±0.01**	**0.786±0.03**	**0.285±0.04**	**0.330±0.06**

$*$ indicates p-value < 0.01, † indicates p-value < 0.05



(a) The averaged purity score. (b) The averaged NMI. (c) The averaged ARI.

Figure 2: The purity score, NMI, and ARI (mean and 95% confidence interval) for the UKCF dataset ($C = 8$) with various K.

is a corrected-for-chance version of the Rand index which is a measure of the percentage of correct cluster assignments (ranges from -1 to 1 with 1 being a perfect clustering and 0 being a random clustering). In a nutshell, all the three performance metrics are commonly used but all have its pros and cons; for instance, the purity score easily converges to 1 when there are as many clusters as data samples and does not work for imbalanced data. Thus, using them together suffices to demonstrate the effectiveness of the clustering methods.

5.4 Clustering Performance

To evaluate the clustering performance, we start with a simple scenario where the true class is available and the number of classes is tractable. In particular, we set $C = 2^3 = 8$ based on the development of three common comorbidities of cystic fibrosis – diabetes, ABPA, and intestinal obstruction – in the next year for the UKCF dataet and $C = 3$ based on the mutually exclusive three diagnostic groups for the ADNI dataset. We compare AC-TPC against the aforementioned benchmarks with respect to the purity score, NMI, and ARI in Table 1.

As shown in Table 1, AC-TPC achieved performance gain over all the tested benchmarks – where most of the improvements were statistically significant with p-value < 0.01 or p-value < 0.05 – for both datasets. Importantly, clustering methods – i.e., KM-DTW, DCN-S2S, and SOM-VAE – that do not associate with the future outcomes of interest performed poorly and are thus proven to provide little prognostic value. Here, the ARI near 0 indicates that the clustering has no difference with random assignment. This implies that sequences that are similar in their latent representations tailored for reconstruction or that are similar with respect to the shape-based measurement using DTW can have very different future outcomes of interest.

In Figure 2, we further investigate the purity score, NMI, and ARI by varying the number of clusters K from 4 to 16 on the UKCF dataset in the same setting with that stated above (i.e., $C = 8$). Here, the three methods – i.e., KM-DTW, DCN-S2S, and SOM-VAE – are excluded for better visualization. As we can see in Figure 2, our model rarely encounters performance loss on both NMI and ARI while the benchmarks (except for SOM-VAE-P) showed significant decrease in the performance as K increased (higher than C). This is because the number of clusters identified by AC-TPC (i.e., the number of activated clusters where we define cluster k is activated if $|\mathcal{C}(k)| > 0$) was the same with C most of the times, while the DCN-based methods identified exactly K clusters

(due to the K-means). Since the NMI and ARI are adjusted for the number of clusters, a smaller number of identified clusters yields, if everything else is equal, a higher performance. In contrast, while our model achieved the same purity score for $K \geq 8$, the benchmarks showed improved performance as K increased since the purity score does not penalize having many clusters. This is an important property of AC-TPC that we do not need to know a priori what the number of clusters is which is a common practical challenge of applying the conventional clustering methods (e.g., K-means).

In addition, when compared with SOM-VAE-P, the improvements of our model over SOM-VAE-P come from two possible sources: i) SOM-VAE-P mainly focuses on visualizing the input with SOM which makes both the encoder and embeddings less flexible – this is why it performed better with higher K – and ii) the Markov property can be a too strict assumption for time-series data especially in clinical settings where a patient's medical history is informative for predicting his/her future clinical outcomes (Ranganath et al., 2016). The same conclusions are reached for SOM-VAE.

5.5 Contributions of the Auxiliary Loss Functions

As described in Section 3.1, we introduced two auxiliary loss functions – the sample-wise entropy of cluster assignment (4) and the embedding separation loss (5) – to avoid trivial solutions that may arise in identifying the predictive clusters. To analyze the contribution of each auxiliary loss function, we report the average number of activated clusters, clustering performance, and discriminative performance on the UKCF dataset with 3 comorbidities as described in Section 5.4. Here, we use both area under receiver operator characteristic curve (AUROC) and area under precision-recall curve (AUPRC) to evaluate the prognostic value (i.e., discriminative power) of predicting each comorbidity. Throughout the experiment, we set $K = 16$ – which is larger than C – to find the contribution of these loss functions to the number of activated clusters.

Table 2: Performance comparison with varying the balancing coefficients α, β for the UKCF dataset.

Coefficients		**Clustering Performance**				**Prognostic Value**	
α	β	Activated No.	Purity	NMI	ARI	AUROC	AUPRC
0.0	0.0	16	0.573±0.01	0.006±0.00	0.000±0.00	0.500±0.00	0.169±0.00
0.0	1.0	16	0.573±0.01	0.006±0.00	0.000±0.00	0.500±0.00	0.169±0.00
3.0	0.0	8	0.795±0.01	0.431±0.01	0.569±0.01	0.840±0.01	0.583±0.02
3.0	1.0	8.4	**0.808±0.01**	**0.468±0.01**	**0.606±0.01**	**0.852±0.00**	**0.608±0.01**

As we can see in Table 2, both auxiliary loss functions make important contributions in improving the quality of predictive clustering. More specifically, the sample-wise entropy encourages the selector to choose one dominant cluster. Thus, as we can see results with $\alpha = 0$, without the sample-wise entropy, our selector assigns an equal probability to all 16 clusters which results in a trivial solution. In addition, we observed that, by augmenting the embedding separation loss, AC-TPC identifies a smaller number of clusters owing to the well-separated embeddings.

5.6 Targeting Multiple Future Outcomes – a Practical Scenario

In this experiment, we focus on a more practical scenario where the future outcome of interest is high-dimensional and the number of classes based on all the possible combinations of future outcomes becomes intractable. For example, suppose that we are interested in the development of M comorbidities in the next year whose possible combinations grow exponentially $C = 2^M$. Interpreting such a large number of patient subgroups is a daunting task which may complicate the understanding of underlying disease progression. Since different comorbidities may share common driving factors (Ronan et al., 2017), we hope our model to be able to identify much smaller underlying (latent) clusters that govern the development of comorbidities. To incorporate M comorbidities (i.e., M binary labels), we redefine the output space as $\mathcal{Y} = \{0, 1\}^M$ and modify the predictor and loss functions, accordingly, as described in Appendix A.

Throughout this experiment, we aim at identifying subgroups of patients that are associated with the next-year development of 22 different comorbidities in the UKCF dataset. In Table 3, we reported 12 identified clusters – on average, the number of activated clusters was 13.6 – and the top three

Table 3: The top-3 frequent comorbidities developed in the next year for the 12 identified clusters. The values in parentheses indicate the corresponding frequency.

Clusters	Top-3 Frequent Comorbidities		
0	Diabetes (0.85)	Liver Enzymes (0.21)	Arthropathy (0.14)
1	Liver Enzymes (0.09)	Arthropathy (0.08)	Depression (0.07)
2	ABPA (0.77)	Osteopenia (0.21)	Intestinal Obstruction (0.11)
3	Asthma (0.89)	Liver Disease (0.87)	Diabetes (0.29)
4	Osteoporosis (0.76)	Diabetes (0.43)	Arthropathy (0.20)
5	Asthma (0.88)	Diabetes (0.81)	Osteopenia (0.28)
6	Liver Disease (0.85)	Asthma (0.03)	ABPA (0.09)
7	ABPA (0.83)	Diabetes (0.78)	Osteopenia (0.25)
8	Diabetes (0.94)	Liver Disease (0.83)	Liver Enzymes (0.43)
9	Asthma (0.89)	Osteopenia (0.26)	ABPA (0.19)
10	Osteopenia (0.82)	Diabetes (0.81)	Arthropathy (0.23)
11	Osteopenia (0.77)	Liver Enzymes (0.18)	Arthropathy (0.12)

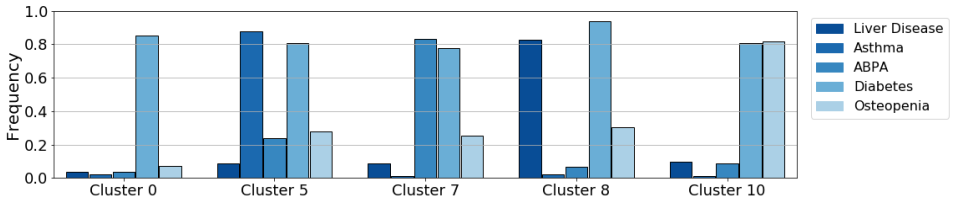


Figure 3: Clusters with high-risk of developing diabetes. We reported the cluster-specific frequencies of developing comorbidities – liver disease, asthma, ABPA, and osteopenia that are co-occurred with diabetes – in the next year.

frequent comorbidities developed in the next year since the latest observation and their corresponding frequencies; please refer to Appendix E for a full list. Here, the frequency is calculated in a cluster-specific fashion based on the true label. As we can see in Table 3, the identified clusters displayed very different label distributions; that is, the combination of comorbidities as well as their frequencies were very different across the clusters. For example, patients in Cluster 1 experienced low-risk of developing any comorbities in the next year while 85% of patients in Cluster 0 were diagnosed with diabetes in the next year.

In Figure 3, we further investigated subgroups of patients – Cluster 0, 5, 7, 8, and 10 – who had high risk of developing diabetes in the next year. Although all these clusters displayed high risk of diabetes, the frequencies of other co-occurring comorbidities was significantly different across the clusters. In one notable example, around 89% of the patients in Cluster 5 were diagnosed with asthma in the next year while it was less than 3% of the patients in the other clusters. Interestingly, "leukotriene" – a medicine commonly used to manage asthma – and "FEV_1% predicted" – a measure of lung function – were the two most different input features between patients in Cluster 5 and those in the other clusters. We observed similar findings in Cluster 7 with ABPA, Cluster 8 with liver disease, and Cluster 10 with osteopenia.

Therefore, by grouping patients who are likely to develop a similar set of comorbidities, our method identified clusters that can be translated into actionable information for clinical decision-making.

6 Conclusion

In this paper, we introduced AC-TPC, a novel deep learning approach for predictive clustering of time-series data. We carefully defined novel loss functions to encourage each cluster to have homogeneous future outcomes (e.g., adverse events, the onset of comorbidities, etc.) and designed optimization procedures to address challenges and to avoid trivial solutions in identifying these cluster assignments and centroids. Throughout the experiments on two real-world datasets, we showed that our model achieves superior clustering performance over state-of-the-art and identifies meaningful clusters that can be translated into actionable information for clinical decision-making.

REFERENCES

L. Samal, A. Wright, B. Wong, J. Linder, and D. Bates. Leveraging electronic health records to support chronic disease management: the need for temporal data views. *Informatics in Primary Care*, 19(2):65–74, 2011.

K. J. Ramos, B. S. Quon, S. L. Heltshe, N. Mayer-Hamblett, E. D. Lease, M. L. Aitken, N. S. Weiss, and C. H. Goss. Heterogeneity in survival in adult patients with cystic fibrosis with $\text{FEV}_1 < 30\%$ of predicted in the united states. *Chest*, 151(6):1320–1328, June 2017.

Xi Zhang, Jingyuan Chou, Jian Liang, Cao Xiao, Yize Zhao, Harini Sarva, Claire Henchcliffe, and Fei Wang. Data-driven subtyping of parkinson's disease using longitudinal clinical records: A cohort study. *Scientific Reports*, 9(797):1–12, January 2019.

A. Rusanov, P. V. Prado, and C. Weng. Unsupervised time-series clustering over lab data for automatic identification of uncontrolled diabetes. *In Proceedings of the 4th IEEE International Conference on Healthcare Informatics (ICHI)*, 2016.

A. Giannoula, A. Gutierrez-Sacristán, Á. Bravo, F. Sanz, and L. I. Furlong. Identifying temporal patterns in patient disease trajectories using dynamic ping: A population-based study. *Scientific Reports*, 8(4216):1–14, March 2018.

D. T. A. Luong and V. Chandola. A k-means approach to clustering disease progressions. *In Proceedings of the 5th IEEE International Conference on Healthcare Informatics (ICHI)*, 2017.

S. Lloyd. Least squares quantization in pcm. *IEEE Transaction on Information Theory*, 28(2): 129–137, March 1982.

A. Boudier, S. Chanoine, S. Accordini, J. M. Anto, X. Basaga na, J. Bousquet, P. Demoly, J. Garcia-Aymerich, F. Gormand, J. Heinrich, C. Janson, N. Künzli, R. Matran, C. Pison, C. Raherison, J. Sunyer, R. Varraso, D. Jarvis, B. Leynaert, I. Pin, and V. Siroux. Data-driven adult asthma phenotypes based on clinical characteristics are associated with asthma outcomes twenty years later. *Allegy*, 74(5):953–963, May 2019.

W. M. Wami, F. Buntinx, S. Bartholomeeusen, G. Goderis, C. Mathieu, and M. Aerts. Influence of chronic comorbidity and medication on the efficacy of treatment in patients with diabetes in general practice. *The British Journal of General Practice*, 63(609):267–273, March 2013.

H. Blockeel, S. Dzeroski, J. Struyf, and B. Zenko. *Predictive Clustering*. Springer New York, 2017.

V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. *In Proceedings of the 13th Conference on Neural Information Processing Systems (NIPS 2000)*, 2000.

B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *In Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, 2017.

C. A. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi. A novel bit level time series representation with implications for similarity search and clustering. *In Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2005)*, 2005.

S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering – a decade review. *Information Systems*, 53:16–38, May 2015.

I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou. Patient subtyping via time-aware lstm networks. *In Proceedings of the 23rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2017)*, 2017.

N. S. Madiraju, S. M. Sadat, D. Fisher, and H. Karimabadi. Deep temporal clustering: Fully unsupervised learning of time-domain features. *arXiv preprint arXiv:1802.01059*, 2018.

J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. *In Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, 2017.

V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch. SOM-VAE: Interpretable discrete representation learning on time series. *In Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

R. Ranganath, A. Perotte, N. Elhadad, and D. Blei. Deep survival analysis. *In Proceedings of the 1st Machine Learning for Healthcare Conference (MLHC 2016)*, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, 2010.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(1):2837–2854, October 2010.

L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.

N. J. Ronan, J. Elborn, and B. J. Plant. Current and emerging comorbidities in cystic fibrosis. *Presse Med.*, 46(6):125–138, June 2017.

T. Warren Liao. Clustering of time series data–a survey. *Pattern Recognition*, 38(11):1857–1874, November 2005.

A VARIATIONS FOR REGRESSION AND BINARY CLASSIFICATION TASKS

As the task changes, estimating the label distribution and calculating the KL divergence in (2) must be redefined accordingly:

- For regression task, i.e., $\mathcal{Y} = \mathbb{R}$, we modify the predictor as $g_\phi : \mathcal{Z} \to \mathbb{R}$ and replace ℓ_1 by $\ell_1(y_t, \bar{y}_t) = \|y_t - \bar{y}_t\|_2^2$. Minimizing $\ell_1(y_t, \bar{y}_t)$ is equivalent to minimizing the KL divergence between $p(y_t|\mathbf{x}_{1:t})$ and $p(y_t|s_t)$ when we assume these probability densities follow Gaussian distribution with the same variance.
- For the M-dimensional binary classification task, i.e., $\mathcal{Y} = \{0, 1\}^M$, we modify the predictor as $g_\phi : \mathcal{Z} \to [0, 1]^M$ and replace ℓ_1 by $\ell_1(y_t, \bar{y}_t) = -\sum_{m=1}^{M} y_t^m \log \bar{y}_t^m + (1 - y_t^m) \log(1 - \bar{y}_t^m)$ which is required to minimize the KL divergence. Here, y_t^m and $\bar{y}_t^m$ indicate the m-th element of y_t and $\bar{y}_t$, respectively. The basic assumption here is that the distribution of each binary label is independent given the input sequence.

B DETAILED DERIVATION OF (7)

To derive the gradient of the predictive clustering loss in (7) with respect $\omega_A = [\theta, \psi]$, we utilized the ideas from actor-critic models (Konda and Tsitsiklis, 2000). The detailed derivation of the former term in (7) is described below (for notational simplicity, we omit the expectation on $\mathbb{E}_{\mathbf{x},y\sim p_{XY}}$):

$$\begin{aligned}
\nabla_\omega \left(\sum_{t=1}^{T} \mathbb{E}_{s_t \sim Cat(\pi_t)} \left[\ell_1(y_t, \bar{y}_t) \right] \right) &= \nabla_\omega \left(\sum_{t=1}^{T} \sum_{s_t \in \mathcal{K}} \pi_t(s_t) \ell_1(y_t, \bar{y}_t) \right) \\
&= \sum_{t=1}^{T} \sum_{s_t \in \mathcal{K}} \nabla_\omega \pi_t(s_t) \ell_1(y_t, \bar{y}_t) \\
&= \sum_{t=1}^{T} \sum_{s_t \in \mathcal{K}} \frac{\nabla_\omega \pi_t(s_t)}{\pi_t(s_t)} \pi_t(s_t) \ell_1(y_t, \bar{y}_t) \\
&= \sum_{t=1}^{T} \sum_{s_t \in \mathcal{K}} \pi_t(s_t) \ell_1(y_t, \bar{y}_t) \nabla_\omega \log \pi_t(s_t) \\
&= \sum_{t=1}^{T} \mathbb{E}_{s_t \sim Cat(\pi_t)} \left[\ell_1(y_t, \bar{y}_t) \nabla_\omega \log \pi_t(s_t) \right]
\end{aligned} \tag{11}$$

C DETAILS ON THE DATASETS

C.1 UKCF DATASET

UK Cystic Fibrosis registry (UKCF)[9] records annual follow-ups for 5,171 adult patients (aged 18 years or older) over the period from 2008 and 2015, with a total of 25,012 hospital visits. Each patient is associated with 89 variables (i.e., 11 static and 78 time-varying features), including information on demographics and genetic mutations, bacterial infections, lung function scores, therapeutic managements, and diagnosis on comorbidities. The detailed statistics are given in Table 4.

C.2 ADNI DATASET

Alzheimer's Disease Neuroimaging Initiative (ADNI)[10] study consists of 1,346 patients with a total of 11,651 hospital visits, which tracks the disease progression via follow-up observations at 6 months interval. Each patient is associated with 21 variables (i.e., 5 static and 16 time-varying features), including information on demographics, biomarkers on brain functions, and cognitive test results.

[9] https://www.cysticfibrosis.org.uk/the-work-we-do/uk-cf-registry

[10] https://adni.loni.usc.edu

Table 4: Summary and description of the UKCF dataset.

STATIC COVARIATES

		Type	Freq.		Type	Freq.
Demographic	Gender	Bin.	0.55			
Genetic	Class I Mutation	Bin.	0.05	Class VI Mutation	Bin.	0.86
	Class II Mutation	Bin.	0.87	DF508 Mutation	Bin.	0.87
	Class III Mutation	Bin.	0.89	G551D Mutation	Bin.	0.06
	Class IV Mutation	Bin.	0.05	Homozygous	Bin.	0.58
	Class V Mutation	Bin.	0.04	Heterozygous	Bin	0.42

TIME-VARYING COVARIATES

		Type	Mean (Freq.)	Min / Max		Type	Mean (Freq.)	Min / Max
Demographic	Age	Cont.	30.4	18.0 / 86.0	Height	Cont.	168.0	129.0 / 198.6
	Weight	Cont.	64.1	24.0 / 173.3	BMI	Cont.	22.6	10.9 / 30.0
	Smoking Status	Bin.	0.1					
Lung Func. Scores	FEV_1	Cont.	2.3	0.2 / 6.3	Best FEV_1	Cont.	2.5	0.3 / 8.0
	FEV_1% Pred.	Cont.	65.1	9.0 / 197.6	Best FEV_1% Pred.	Cont.	71.2	7.5 / 164.3
Hospitalization	IV ABX Days Hosp.	Cont.	12.3	0 / 431	Non-IV Hosp. Adm.	Cont.	1.2	0 / 203
	IV ABX Days Home	Cont.	11.9	0 / 441				
Lung Infections	B. Cepacia	Bin.	0.05		P. Aeruginosa	Bin.	0.59	
	H. Influenza	Bin.	0.05		K. Pneumoniae	Bin.	0.00	
	E. Coli	Bin.	0.01		ALCA	Bin.	0.03	
	Aspergillus	Bin.	0.14		NTM	Bin.	0.03	
	Gram-Negative	Bin.	0.01		Xanthomonas	Bin.	0.05	
	S. Aureus	Bin.	0.30					
Comorbidities	Liver Disease	Bin.	0.16		Depression	Bin.	0.07	
	Asthma	Bin.	0.15		Hemoptysis	Bin.	0.01	
	ABPA	Bin.	0.12		Pancreatitus	Bin.	0.01	
	Hypertension	Bin.	0.04		Hearing Loss	Bin.	0.03	
	Diabetes	Bin.	0.28		Gall bladder	Bin.	0.01	
	Arthropathy	Bin.	0.09		Colonic structure	Bin.	0.00	
	Bone fracture	Bin.	0.01		Intest. Obstruction	Bin.	0.08	
	Osteoporosis	Bin.	0.09		GI bleed – no var.	Bin.	0.00	
	Osteopenia	Bin.	0.21		GI bleed – var.	Bin.	0.00	
	Cancer	Bin.	0.00		Liver Enzymes	Bin.	0.16	
	Cirrhosis	Bin.	0.03		Kidney Stones	Bin.	0.02	
Treatments	Dornase Alpha	Bin.	0.56		Inhaled B. BAAC	Bin.	0.03	
	Anti-fungals	Bin.	0.07		Inhaled B. LAAC	Bin.	0.08	
	HyperSaline	Bin.	0.23		Inhaled B. SAAC	Bin.	0.05	
	HypertonicSaline	Bin.	0.01		Inhaled B. LABA	Bin.	0.11	
	Tobi Solution	Bin.	0.20		Inhaled B. Dilators	Bin.	0.57	
	Cortico Combo	Bin.	0.41		Cortico Inhaled	Bin.	0.15	
	Non-IV Ventilation	Bin.	0.05		Oral B. Theoph.	Bin.	0.04	
	Acetylcysteine	Bin.	0.02		Oral B. BA	Bin.	0.03	
	Aminoglycoside	Bin.	0.03		Oral Hypo. Agents	Bin.	0.01	
	iBuprofen	Bin.	0.00		Chronic Oral ABX	Bin.	0.526	
	Drug Dornase	Bin.	0.02		Cortico Oral	Bin.	0.14	
	HDI Buprofen	Bin.	0.00		Oxygen Therapy	Bin.	0.11	
	Tobramycin	Bin.	0.03		O_2 Exacerbation	Bin.	0.03	
	Leukotriene	Bin.	0.07		O_2 Nocturnal	Bin.	0.03	
	Colistin	Bin.	0.03		O_2 Continuous	Bin.	0.03	
	Diabetes Insulin	Bin.	0.01		O_2 Pro re nata	Bin.	0.01	
	Macrolida ABX	Bin.	0.02					

ABX: antibiotics

Table 5: Summary and description of the ADNI dataset.

STATIC COVARIATES

		Type	Mean (Freq.)	Min/Max (Mode)		Type	Mean (Freq.)	Min/Max (Mode)
Demographic	Race	Cat.	0.93	White	Ethnicity	Cat.	0.97	No Hisp/Latino
	Education	Cat.	0.23	C16	Marital Status	Cat.	0.75	Married
Genetic	$APOE_4$	Cont.	0.44	0/2				

TIME-VARYING COVARIATES

		Type	Mean	Min / Max		Type	Mean	Min / Max
Demographic	Age	Cont.	73.6	55/92				
Biomarker	Entorhinal	Cont.	3.6E+3	1.0E+3 / 6.7E+3	Mid Temp	Cont.	2.0E+4	8.9E+3 / 3.2E+4
	Fusiform	Cont.	1.8E+5	9.0E+4 / 2.9E+5	Ventricles	Cont.	4.1E+4	5.7E+3 / 1.6E+5
	Hippocampus	Cont.	6.9E+3	2.8E+3 / 1.1E+4	Whole Brain	Cont.	1.0E+6	6.5E+5 / 1.5E+6
	Intracranial	Cont.	1.5E+6	2.9E+2 / 2.1E+6				
Cognitive	ADAS-11	Cont.	8.58	0/70	ADAS-13	Cont.	13.61	0/85
	CRD Sum of Boxes	Cont.	1.21	0/17	Mini Mental State	Cont.	27.84	2/30
	RAVLT Forgetting	Cont.	4.19	-12/15	RAVLT Immediate	Cont.	38.25	0/75
	RAVLT Learning	Cont.	4.65	-5/14	RAVLT Percent	Cont.	51.70	-500/100

Table 6: Comparison table of benchmarks.

Methods	Handling Time-Series	Clustering Method	Similarity Measure	Label Provided	Label Associated
KM-DTW	DTW	K-means	DTW	N	N
KM-E2P ($\mathcal{Z}$)	RNN	K-means	Euclidean in $\mathcal{Z}$	Y	N
KM-E2P ($\mathcal{Y}$)	RNN	K-means	Euclidean in $\mathcal{Y}$	Y	Y (direct)
DCN-S2S	RNN	K-means	Euclidean in $\mathcal{Z}$	N	N
DCN-E2P	RNN	K-means	Euclidean in $\mathcal{Z}$	N	Y (indirect)
SOM-VAE	Markov model	embedding mapping	reconstruction loss	N	N
SOM-VAE-P	Markov model	embedding mapping	prediction loss	Y	Y (direct)
Proposed	RNN	embedding mapping	KL divergence	Y	Y (direct)

The three diagnostic groups were normal brain functioning (0.55), mild cognitive impairment (0.43), and Alzheimer's disease (0.02). The detailed statistics are given in Table 5.

D Details on the Benchmarks

We compared AC-TPC in the experiments with clustering methods ranging from conventional approaches based on K-means to the state-of-the-art approaches based on deep neural networks. The details of how we implemented the benchmarks are described as the following:

- **Dynamic time warping followed by K-means**[11]: Dynamic time warping (DTW) is utilized to quantify pairwise distance between two variable-length sequences and, then, K-means is applied (denoted as **KM-DTW**).
- **K-means with deep neural networks**: To handle variable-length time-series data, we utilize our encoder and predictor that are trained based on (10) for dimensionality reduction; this is to provide fixed-length and low-dimensional representations for time-series. Then, we apply K-means on the latent encodings $\mathbf{z}$ (denoted as **KM-E2P ($\mathcal{Z}$)**) and on the predicted label distributions $\hat{y}$ (denoted as **KM-E2P ($\mathcal{Y}$)**), respectively. We implemented the encoder and predictor of KM-E2P with the same network architectures with those of our model: the encoder is a single-layer LSTM with 50 nodes and the decoder is a two-layered fully-connected network with 50 nodes in each layer.
- **Extensions of DCN**[12] (Yang et al., 2017): Since the DCN is designed for static data, we replace the encoder-decoder structure with a sequence-to-sequence network as an extension to incorporate time-series data (denoted as **DCN-S2S**). For implementing DCN-S2S, we used a single-layer LSTM with 50 nodes for both the encoder and the decoder. And, we augmented a fully-connected layer with 50 nodes is used to decode the latent representation and reconstruct the original sequence.

  In addition, since predictive clustering is associated with the label distribution, we compared a DCN whose encoder-decoder structure is replaced with our encoder and predictor (denoted as **DCN-E2P**) to focus the dimensionality reduction – and, thus, finding latent encodings where clustering is performed – on the information for predicting the label distribution. We implemented the encoder and predictor of DCN-E2P with the same network architectures with those of our model as described above.
- **SOM-VAE**[13] (Fortuin et al., 2019): We compare with SOM-VAE – though, this method is oriented towards visualization of input data via SOM – since it naturally clusters time-series data assuming Markov property (denoted as **SOM-VAE**). We replace the original CNN architecture of the encoder and the decoder with three-layered fully-connected network with 50 nodes in each layer, respectively.

  In addition, we compare with a variation of SOM-VAE by replacing the decoder with the predictor to encourage the latent encoding to capture information for predicting the label distribution (denoted as **SOM-VAE-P**). For the implementation, we replaced the decoder of SOM-VAE with

[11] `https://github.com/rtavenar/tslearn`
[12] `https://github.com/boyangumn/DCN`
[13] `https://github.com/ratschlab/SOM-VAE`

our predictor which is a two-layered fully-connected layer with 50 nodes in each layer to predict the label distribution.

For both cases, we used the default values for balancing coefficients of SOM-VAE and the dimension of SOM to be equal to K.

We compared and summarized major components of the benchmarks in Table 6.

E Additional Results on the Experiments

E.1 Targeting Multiple Future Outcomes

Throughout the experiment in Section 5.6, we identified 12 subgroups of patients that are associated with the next-year development of 22 different comorbidities in the UKCF dataset. In Table 7, we reported 12 identified clusters and the full list of comorbidities developed in the next year since the latest observation and the corresponding frequency. Here, the frequency is calculated in a cluster-specific fashion based on the true label.

Table 7: The comorbidities developed in the next year for the 12 identified clusters. The values in parentheses indicate the corresponding frequency.

Clusters	Comorbidities and Frequencies			
Cluster 0	Diabetes (0.85)	Liver Enzymes (0.21)	Arthropathy (0.14)	Depression (0.10)
	Hypertens (0.08)	Osteopenia (0.07)	Intest. Obstruction (0.07)	Cirrhosis (0.04)
	ABPA (0.04)	Liver Disease (0.04)	Osteoporosis (0.03)	Hearing Loss (0.03)
	Asthma (0.02)	Kidney Stones (0.01)	Bone fracture (0.01)	Hemoptysis (0.01)
	Pancreatitis (0.01)	Cancer (0.00)	Gall bladder (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 1	Liver Enzymes (0.09)	Arthropathy (0.08)	Depression (0.07)	Intest. Obstruction (0.06)
	Diabetes (0.06)	Osteopenia (0.05)	ABPA (0.04)	Asthma (0.03)
	Liver Disease (0.03)	Hearing Loss (0.03)	Osteoporosis (0.02)	Pancreatitis (0.02)
	Kidney Stones (0.02)	Hypertension (0.01)	Cirrhosis (0.01)	Gall bladder (0.01)
	Cancer (0.01)	Hemoptysis (0.00)	Bone fracture (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 2	ABPA (0.77)	Osteopenia (0.21)	Intest. Obstruction (0.11)	Hearing Loss (0.10)
	Liver Enzymes (0.07)	Diabetes (0.06)	Depression (0.05)	Pancreatitis (0.05)
	Liver Disease (0.04)	Arthropathy (0.04)	Asthma (0.03)	Bone fracture (0.02)
	Osteoporosis (0.02)	Hypertension (0.01)	Cancer (0.01)	Cirrhosis (0.01)
	Kidney Stones (0.01)	Gall bladder (0.01)	Hemoptysis (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 3	Asthma (0.89)	Liver Disease (0.87)	Diabetes (0.29)	Osteopenia (0.28)
	Liver Enzymes (0.24)	ABPA (0.15)	Osteoporosis (0.11)	Hearing Loss (0.06)
	Arthropathy (0.05)	Intest. Obstruction (0.05)	Depression (0.04)	Hypertension (0.03)
	Cirrhosis (0.02)	Kidney Stones (0.02)	Pancreatitis (0.02)	Gall bladder (0.02)
	Cancer (0.01)	Bone fracture (0.00)	Hemoptysis (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 4	Osteoporosis (0.76)	Diabetes (0.43)	Arthropathy (0.20)	Liver Enzymes (0.18)
	Osteopenia (0.15)	Depression (0.13)	Intest. Obstruction (0.11)	ABPA (0.11)
	Hearing Loss (0.09)	Liver Disease (0.08)	Hypertension (0.07)	Cirrhosis (0.07)
	Kidney Stones (0.03)	Asthma (0.02)	Hemoptysis (0.02)	Bone fracture (0.02)
	Gall bladder (0.01)	Pancreatitis (0.01)	Cancer (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 5	Asthma (0.88)	Diabetes (0.81)	Osteopenia (0.28)	ABPA (0.24)
	Liver Enzymes (0.22)	Depression (0.15)	Osteoporosis (0.14)	Intest. Obstruction (0.12)
	Hypertension (0.10)	Cirrhosis (0.10)	Liver Disease (0.09)	Arthropathy (0.08)
	Bone fracture (0.01)	Hemoptysis (0.01)	Pancreatitis (0.01)	Hearing Loss (0.01)
	Cancer (0.01)	Kidney Stones (0.01)	GI bleed – var. (0.01)	Gall bladder (0.00)
	Colonic stricture (0.00)	GI bleed – no var. (0.00)		
Cluster 6	Liver Disease (0.85)	Liver Enzymes (0.37)	Osteopenia (0.27)	ABPA (0.09)
	Arthropathy (0.07)	Diabetes (0.06)	Intest. Obstruction (0.06)	Osteoporosis (0.05)
	Depression (0.03)	Asthma (0.03)	Hearing Loss (0.03)	Cirrhosis (0.02)
	Hemoptysis (0.02)	Hypertension (0.01)	Kidney Stones (0.01)	Pancreatitis (0.00)
	Gall bladder (0.00)	Bone fracture (0.00)	Cancer (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 7	ABPA (0.83)	Diabetes (0.78)	Osteopenia (0.25)	Osteoporosis (0.24)
	Liver Enzymes (0.15)	Intest. Obstruction (0.12)	Liver Disease (0.09)	Hypertension (0.07)
	Hearing Loss (0.07)	Arthropathy (0.06)	Depression (0.04)	Cirrhosis (0.02)
	Asthma (0.01)	Bone fracture (0.01)	Kidney Stones (0.01)	Hemoptysis (0.01)
	Cancer (0.00)	Pancreatitis (0.00)	Gall bladder (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 8	Diabetes (0.94)	Liver Disease (0.83)	Liver Enzymes (0.43)	Osteopenia (0.30)
	Hearing Loss (0.11)	Osteoporosis (0.10)	Intest. Obstruction (0.09)	Cirrhosis (0.08)
	Depression (0.08)	ABPA (0.07)	Arthropathy (0.06)	Hypertension (0.05)
	Kidney Stones (0.05)	Asthma (0.02)	Hemoptysis (0.01)	Bone fracture (0.01)
	Cancer (0.00)	Pancreatitis (0.00)	Gall bladder (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 9	Asthma (0.89)	Osteopenia (0.26)	ABPA (0.19)	Arthropathy (0.14)
	Intest. Obstruction (0.11)	Depression (0.08)	Osteoporosis (0.08)	Diabetes (0.06)
	Liver Enzymes (0.06)	Hemoptysis (0.03)	Hypertension (0.02)	Liver Disease (0.02)
	Pancreatitis (0.02)	Bone fracture (0.01)	Cancer (0.01)	Cirrhosis (0.01)
	Gall bladder (0.01)	Hearing Loss (0.01)	Kidney Stones (0.00)	Colonic stricture (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		
Cluster 10	Osteopenia (0.82)	Diabetes (0.81)	Arthropathy (0.23)	Depression (0.19)
	Liver Enzymes (0.18)	Hypertension (0.16)	Hearing Loss (0.10)	Liver Disease (0.10)
	Osteoporosis (0.10)	Intest. Obstruction (0.09)	ABPA (0.09)	Kidney Stones (0.07)
	Cirrhosis (0.05)	Asthma (0.01)	Cancer (0.00)	GI bleed – var. (0.00)
	Bone fracture (0.00)	Hemoptysis (0.00)	Pancreatitis (0.00)	Gall bladder (0.00)
	Colonic stricture (0.00)	GI bleed – no var. (0.00)		
Cluster 11	Osteopenia (0.77)	Liver Enzymes (0.18)	Arthropathy (0.12)	Depression (0.09)
	Hypertension (0.06)	Diabetes (0.06)	Hearing Loss (0.06)	ABPA (0.05)
	Liver Disease (0.05)	Osteoporosis (0.04)	Intest. Obstruction (0.04)	Cirrhosis (0.02)
	Asthma (0.02)	Pancreatitis (0.02)	Bone fracture (0.01)	Cancer (0.01)
	Kidney Stones (0.00)	Gall bladder (0.00)	Colonic stricture (0.00)	Hemoptysis (0.00)
	GI bleed – no var. (0.00)	GI bleed – var. (0.00)		

F PSEUDO-CODE OF AC-TPC

As illustrated in Section 3.2, AC-TPC is trained in an iterative fashion. We provide the pseudo-code for optimizing our model in Algorithm 1 and that for initializing the parameters in Algorithm 2.

Algorithm 1 Pseudo-code for Optimizing AC-TPC

Input: Dataset $\mathcal{D} = \{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^{N}$, number of clusters K, coefficients (α, β), learning rate (η_A, η_C, η_E), mini-batch size n_{mb}, and update step M
Output: AC-TPC parameters (θ, ψ, ϕ) and the embedding dictionary $\mathcal{E}$
Initialize parameters (θ, ψ, ϕ) and the embedding dictionary $\mathcal{E}$ via `Algorithm 2`

repeat
Optimize the Encoder, Selector, and Predictor
for $m = 1, \cdots, M$ **do**
Sample a mini-batch of n_{mb} data samples: $\{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^{n_{mb}} \sim \mathcal{D}$
for $n = 1, \cdots, n_{mb}$ **do**
Calculate the assignment probability: $\pi_t^n = [\pi_t^n(1) \cdots \pi_t^n(K)] \leftarrow h_\psi(f_\theta(\mathbf{x}_{1:t}^n))$
Draw the cluster assignment: $s_t^n \sim Cat(\pi_t^n)$
Calculate the label distributions: $\bar{y}_t^n \leftarrow g_\phi(\mathbf{e}(s_t^n))$ and $\hat{y}_t^n \leftarrow g_\phi(f_\theta(\mathbf{x}_{1:t}^n))$
end for
Update the encoder f_θ and selector h_ψ:

$$\theta \leftarrow \theta - \eta_A \left(\frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \ell_1(y_t^n, \bar{y}_t^n) \nabla_\theta \log \pi_t^n(s_t^n) - \alpha \nabla_\theta \sum_{k=1}^{K} \pi_t^n(k) \log \pi_t^n(k) \right)$$

$$\psi \leftarrow \psi - \eta_A \left(\frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \ell_1(y_t^n, \bar{y}_t^n) \nabla_\psi \log \pi_t^n(s_t^n) - \alpha \nabla_\psi \sum_{k=1}^{K} \pi_t^n(k) \log \pi_t^n(k) \right)$$

Update the predictor g_ϕ:

$$\phi \leftarrow \phi - \eta_C \frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \nabla_\phi \ell_1(y_t^n, \bar{y}_t^n)$$

end for

Optimize the Cluster Centroids
for $m = 1, \cdots, M$ **do**
Sample a mini-batch of n_{mb} data samples: $\{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^{n_{mb}} \sim \mathcal{D}$
for $n = 1, \cdots, n_{mb}$ **do**
Calculate the assignment probability: $\pi_t^n = [\pi_t^n(1) \cdots \pi_t^n(K)] \leftarrow h_\psi(f_\theta(\mathbf{x}_{1:t}^n))$
Draw the cluster assignment: $s_t^n \sim Cat(\pi_t^n)$
Calculate the label distributions: $\bar{y}_t^n \leftarrow g_\phi(\mathbf{e}(s_t^n))$
end for
for $k = 1, \cdots, K$ **do**
Update the embeddings $\mathbf{e}(k)$:

$$\mathbf{e}(k) \leftarrow \mathbf{e}(k) - \eta_E \left(\frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \nabla_{\mathbf{e}(k)} \ell_1(y_t^n, \bar{y}_t^n) - \gamma \sum_{\substack{k'=1 \\ k' \neq k}}^{K} \nabla_{\mathbf{e}(k)} \ell_1\big(g_\phi(\mathbf{e}(k)), g_\phi(\mathbf{e}(k'))\big) \right)$$

end for
Update the embedding dictionary: $\mathcal{E} \leftarrow \{\mathbf{e}(1), \ldots \mathbf{e}(K)\}$
end for
until convergence

Algorithm 2 Pseudo-code for pre-training AC-TPC

Input: Dataset $\mathcal{D} = \{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^N$, number of clusters K, learning rate η, mini-batch size n_{mb}
Output: AC-TPC parameters (θ, ψ, ϕ) and the embedding dictionary $\mathcal{E}$
Initialize parameters (θ, ψ, ϕ) via Xavier Initializer

Pre-train the Encoder and Predictor
repeat
  Sample a mini-batch of n_{mb} data samples: $\{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^{n_{mb}} \sim \mathcal{D}$
  for $n = 1, \cdots, n_{mb}$ **do**
    Calculate the label distributions: $\hat{y}_t^n \leftarrow g_\phi(f_\theta(\mathbf{x}_{1:t}^n))$
  end for

$$\theta \leftarrow \theta - \eta \frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \nabla_\theta \ell_1(y_t^n, \hat{y}_t^n) \qquad \phi \leftarrow \phi - \eta \frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \nabla_\phi \ell_1(y_t^n, \hat{y}_t^n)$$

until convergence

Initialize the Cluster Centroids
Calculate the embedding dictionary $\mathcal{E}$ and initial cluster assignments c_t^n

$$\mathcal{E}, \{\{c_t^n\}_{t=1}^{T^n}\}_{n=1}^N \leftarrow \texttt{K-means}(\{\{\mathbf{z}_t^n\}_{t=1}^{T^n}\}_{n=1}^N, K)$$

Pre-train the Selector
repeat
  Sample a mini-batch of n_{mb} data samples: $\{(\mathbf{x}_t^n, y_t^n)_{t=1}^{T^n}\}_{n=1}^{n_{mb}} \sim \mathcal{D}$
  for $n = 1, \cdots, n_{mb}$ **do**
    Calculate the assignment probability: $\pi_t^n = [\pi_t^n(1) \cdots \pi_t^n(K)] \leftarrow h_\psi(f_\theta(\mathbf{x}_{1:t}^n))$
  end for
  Update the selector h_ψ:

$$\psi \leftarrow \psi + \eta \frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{t=1}^{T^n} \sum_{k=1}^{K} c_t^n(k) \log \pi_t^n(k)$$

until convergence