Forecasting Sparse Data Streams via Dense Data Streams: A Neural Point Process Approach

Abstract—**Forecasting occurrence of future events using time series data is crucial to many practical applications including crime, purchase or traffic prediction. In this work, we consider an important special case where we wish to forecast for *sparse* time series data streams by automatically learning similarities with related dense data streams. We propose a novel three stage model where we first extract meaningful features from a recurrent marked temporal point process model and then learn a suitable metric for clustering and finally employ a novel weight initialization routine to train a neural network based temporal model for forecasting. Our proposed algorithm *spForecast* performs extremely well on several real world applications which we demonstrate using insights gained from our extensive experiments on real world crime and product purchase datasets.**

Index Terms—**Time Series; temporal point process; RMTPP; forecasting; clustering; metric learning;**

I. Introduction

Several real world datasets have a time series nature to them comprising of information about the type of *event* and the *time* of such events; along with other allied attributes. For instance time and location of crimes happening in a city, time and location of pick-up/drop of passengers in taxis, time and quantity of purchase of products in a departmental store are some innate candidates of time series data. Forecasting future crime events, demands for taxis, sales of a product using such time series is an important problem in practice for the respective departments to optimize their resources accordingly and work efficiently. Accurate forecasting can lead to better crime patrolling in a city, improved driver schedule optimization for a taxi agency, enhanced customer sales for a departmental store and so on.

Traditionally, such time series event data has been studied using temporal point process models. Starting from the simplest Poisson process for studying the arrival of customers in a queue to inhomogeneous Poisson Processes([2]) for it's application on call center data, to models such as the TiDeH (Time Dependent Hawkes Process [3]) for predicting future profiles of re-tweet activity, the general idea is to use a parametric model to describe the event dynamics and estimate the parameters using available data to help in forecasting. While these models are simplistic in nature and make several assumptions about the data generation process, recently more sophisticated models such as [10] The Neural Hawkes Process, [13] Recurrent Marked Temporal Point Process (RMTPP) have been proposed which use a neural network to model the data.

A common requirement for most of the models discussed above is the availability of large enough data. In fact, larger the datasets, the better the forecasting ability of these models. However, in several real world applications, the data is often a mixture of dense and sparse data streams. For instance in the case of crime data, one often encounters cities where crimes happen quite often whereas in other cities the crime rate is quite low. As another example, in a departmental store, some products might sell often whereas several products might have only a few purchases over a long period of time. It is required to accurately forecast the trends for such cities or products. Previous approaches often fail to work in such cases of sparse data streams. Thus, it becomes critical to develop machine learning approaches which not only forecasts well on dense data streams but also on sparse data streams.

In this paper, we propose an algorithm *spForecast* for forecasting well on sparse data streams by making use of other dense data streams which are typically available in practice (cities with high crimes, products with high purchase rates etc.). spForecast uses a novel combination of neural point process and metric learning approaches to cluster the cities according to their 'forecast capability' and then, use a novel clustering based weight initialization routine which helps to load the weights of a point process neural network for forecasting in sparse data streams. To demonstrate the efficacy of the proposed solution, we conduct large-scale experiments on real-world datasets including crime dataset of LA and a product purchase dataset of a departmental store [25]. Our experimental results achieve an order of magnitude improvement in accuracy for forecasting of sparse data streams using spForecast when compared to alternative approaches.

II. Related Work

Time series prediction is a dynamic research area which has attracted many of researchers community over the last few decades. There is a rich literature on different methods of time series prediction, see [1] for a detailed survey of the various techniques applied for forecasting different types of time series dataset. Works such as [4], [5], [6] analyze various data mining techniques from simple Linear Regression, Gaussian Processes, Multilayer Perceptron to deep learning for the prediction of future crime "hotspots". [7] nears the problem of forecasting crime time series data by clustering using a memetic differential fuzzy approach. Other works in retail forecasting include application of simple ARIMA models, [8]. Most of these previous works do not refer to the problem of sparse time series predictions.

We not only focus to solve the problem of data sparsity, we aim to do it via state of the art solutions. [22] describes the evolution and intricacies of temporal point process. [17] defined various stages of point process modeling. In sociology, spatial-temporal point process has been used for modeling the network of criminals and the crime locations [18], [19]. In accountancy, marked temporal point process has been used to model the dynamics of high frequency stock transactions [10]. In medical science, it has been used in identifying a patient's visit to the intensive care unit [10]. In social computing, temporal point process has been used to capture the trend of the tweets [10], [20]. Recent works have taken the direction of deep reinforcement learning of marked temporal point process [23], extensions of spatio-temporal process [21].

A major drawback of such existing works is that they do not address the problem of forecasting in sparse data stream. However, there are various works which tries to tackle similar problems. For instance [18] attempts to capture the homicide hot-spots based on dense crimes like robbery, assault. They use self-exciting process for modeling hot-spots. They also define a single variable for the contribution of other data streams; whereas our work uses a dimensional vector of *forecast weights* and *forecast features* from deep neural network as a knowledge representation of dense data streams. [9] models the prediction of sparse data series using dense data series based on simple regression; whereas we focus on modeling using deep neural network along with metric learning for forecasting sparse time series.

In this work, we propose a model that improves forecasting for sparse data streams with the help of dense data streams. Our work is based on recent machine learning advances such as recurrent marked temporal point process [13] and standard machine learning techniques such as distance metric learning approach [11]. We describe these in more detail in the following sections.

III. Problem Statement

Suppose, an input dataset is a set of data streams C = $\{S^1, S^2, \ldots S^n\}$. Each data stream $S^i = ((t_1^i, y_1^i), (t_2^i, y_2^i), \ldots (t_{n_i}^i, y_{n_i}^i)$ is a sequence of pairs where t_j^i is the time when the event of type (or marker) y_j^i has occurred to the entity i, and $t_j^i < t_{j+1}^i$. Depending on specific applications, the entity and the event type can have different meanings. For example, in crime detection, S^i can be a trace of time and crime type pairs for a city i where t_j^i is the time when the crime of type y_j^i occurs. In retail transactions, S^i can be a sequence of time and action pairs for a particular product i where t_j^i is the time when a transaction of selling ($y_j^i = 1$) has occurred. Note that in general time series have different length, but we pad the series to make them of equal length. The time series which have a significantly lesser number of events occurring over a considerable period as compared to the average number of events that occur in the dataset, we call those data stream as *sparse*. Let us assume that there are u sparse data streams in $U \in C$. In this work, we want to build models which are able to:

1) Group similar time-series together effectively based on their past trends;
2) Predict accurately the next event pair for the u sparse data streams.

Grouping problem : Given a sparsely populated similarity information of a number of time-series, the goal is to decompose the set C of data-streams into k groups such that no two dissimilar items belong to the same group. Furthermore, using the similarity and dissimilarity information of *some* data streams, we are to distance the remaining data streams; leading to groups of intermixed sparse and dense data streams. We will also look into how we retrieve the similarities between certain pairs of time-series.

Forecasting problem : Given C, a combination of sparse and dense data streams, and a natural number p, the forecasting problem for us is to predict p-step ahead values of U. In other words, we need to output u × p matrix, Û so as to minimize the error w.r.t to the actual values we observe in future: Ū . The error metric we report is the root mean square error defined as follows for a particular U_i:

$$Error_i = \sqrt{\frac{1}{p}\sum_j (\hat{U}_{i,j} - \bar{U}_{i,j})^2} \tag{1}$$

where $1 \le j \le p$ for a particular sparse data-stream in U.

In the next section, we begin describing the necessary building blocks to solve the above problems.

IV. Background

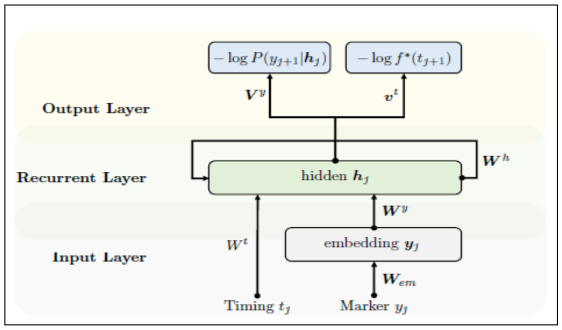
We provide some background on the two problems we focus on in this paper: clustering and forecasting. We first describe a state of the art machine learning method for event forecasting in dense data streams - Recurrent Marked Temporal Point Process model (RMTPP).

A. Recurrent Marked Temporal Point Process

Recurrent Marked Temporal Point Process is an extension of marked temporal point process using RNNs [14]. It has emerged as a powerful model for time-series forecasting as demonstrated. The distinctive feature of RMTPP is that the intensity function of a temporal point process is viewed as a nonlinear function of the history. This history at any time instant j, comprises of the inter-dependencies of the time and events prior to the point t_j in the series. The embedding layer captures the dynamics of the series in a fixed length vector - h. It has the ability to learn the dynamics without the need to know any actual parametric forms.

Additionally, RMTPP is applicable for a wide range of situations where data rich in event markers accompanied with time is encountered; portraying the scalability of the application of our method in various fields. For a given

Fig. 1: Architecture of Recurrent Marked Temporal Point Processes [13]



sequence S^i of n_i events, prediction of the next pair $(t^i_{n_i+1}, y^i_{n_i+1})$ is demonstrated by the architecture of RMTPP in Figure 1. At the j-th event, the input layer first projects the sparse one-hot vector representation of the marker y_j into a latent space. Similar to a Recurrent Neural Network, h_{j-1} is updated to h_j by taking into account the effect of the current event $(t_j,\ y_j)$ by the following equation :

$$h_j = max(W^y y_j + W^t t_j + W^h h_{j-1} + b_h, 0) \tag{2}$$

Hidden representations h_j accounts for the most robust representation in the network determining the prediction of the next pair (t^i_p, y^i_p) by the following two equations:

1) Given h_j **marker generation** is modeled with a multinomial distribution by

$$P(y_{j+1} = k|h_j) = \frac{exp(V^y_{k,:} h_j + b^y_k)}{\sum_{k=1}^{K} exp(V^y_{k,:} h_j + b^y_k)} \tag{3}$$

where K is the number of markers, and $V^y_{k,:}$ is the k-th row of matrix V^y.

2) Based on h_j the **conditional intensity function** is formulated as

$$\lambda^*(t) = exp(v^{t^T}.h_j + w^t(t - t_j) + b^t) \tag{4}$$

where v^t (term for past influence) is a column vector, and w^t (term for current influence), b^t (base intensity)are scalars.

The ability of RMTPP to model the above is remarkable and works best with dense time series data. To tackle the problem of a more accurate prediction for sparse data streams, we propose a novel method of modeling a relationship between the data streams and transferring knowledge from one to another. Towards this, we describe next a distance metric learning approach that helps us group similar data steams together.

B. Distance Metric Learning and Clustering:

Learning a good distance metric for time series data streams in a feature space is vital to the task of clustering. Paper [11] talks about the same, tackling the task of learning a distance metric for an input space of data, given a set of similar/dissimilar pairs, preserving the distance relation among the training data. Let $\mathscr{C} = \{x_1, x_2, ..., x_n\}$ be a collection of features associated with a set of data streams C, where n is the number of samples in the collection, where $x_i \in \mathbf{R}^m$ is a fixed length feature vector corresponding to data-stream S^i. Suppose, it is given that some pairs in x is "similar" :

$$S = (x_i, x_j) \in S \text{ if } x_i \text{ and } x_j \text{ are similar.} \tag{5}$$

Given the above, according to Eric P.Xing, we can successfully incorporate a supervised distance metric learning task. The distance between any two features x_1 and x_2 is then expressed by :

$$D_A(x_1, x_2) = ||x_1 - x_2||_A = \sqrt{(x_1 - x_2)^T A(x_1 - x_2)} \tag{6}$$

where matrix A $\in R^{m*m}$ is the distance metric to be computed. The distance metric has to obey four axioms : non-negativity, identity of indiscernibles, symmetry and subadditvity/ triangle inequality. Learning such a distance metric is also equivalent to finding a rescaling of a data that replaces each point x with $A^{1/2}$x and applying the standard Euclidean metric to the rescaled data.

After rescaling the data, clustering of the feature set x can be performed by well-known clustering methods like K-means or spectral clustering to obtain any k groups. We now dive into the detailed explanation of our method in section V followed by experimentation in section VI.

V. Proposed Method

Our algorithm *spForecast*, integrates distance metric learning with RMTPP model for prediction (talked about above) in sparse time series. *spForecast* encompasses three main steps, which is demonstrated in Algorithm 1. First, the n time-series data streams are divided based on the abundance of data into U and V (lines (4) and (5)). Then, for all n data streams, RMTPP models are trained respectively forecasting the next event and time pair for a time series. The trained models are stored from where the hidden states representations (equation 2) or the *forecast features* at the last time step of each time series and weights (from equations 2, 3, 4) or *forecast weights* from all the layers of RMTPP for all the time series models are extracted and stored.

To establish the similarity relationship between the data streams, we generate m random pairs from the set C. These m random pairs contribute to formulating the matrices Sim^{nXn} and Dis^{nXn}, where pair $(S^k, S^l) \in$ X is similar/dissimilar according to lines 10 - 15 of the algorithm. We denote a set of similar data stream pairs by the set

$$Sim = \{(x_k, x_l) \mid \text{e}(S^k, S^l) \le \theta_1\} \tag{7}$$

and the set of dissimilar pairs is denoted by

$$Dis = \{(x_k, x_l) \mid \text{e}(S^k, S^l) \ge \theta_2\} \tag{8}$$

where, e(S^k, S^l) is the root mean square error of the RMTPP model as shown in equation 1 trained on a time-series S^k and tested on S^l to predict it's next pair $(t^l_{n_{l+1}}, y^l_{n_{l+1}})$. The training is done primarily on one entity while the trained model is used to predict the next time step of another entity. This step is crucial to our algorithm as it forms a bridge between two different time series entities. The rmse is noted and chosen as the decision criterion for the constraints; two time series similar in their trends and seasonality will correspond to two closely related models learned for their respective prediction task. The closer the models are, the smaller will be the value of the e(S^k, S^l) (inter model prediction rmse). Hence, this rmse denotes the dissimilarity between the two sequences. As specified by [11], the value of m is evaluated such that the set of similar data pairs is 1% of all the data pairs.

The second important step of our algorithm is achieved when a transformed forecast feature matrix H' is computed in line (16). This is the version of H (forecast features) rescaled by $A^{1/2}$. Line (17) clusters this H' to group together the n data streams. Lastly, the optimal combination of weights W_u, for each $u \in U$ is estimated by line (19) where the average of the weights (stored in line (8)) from the data stream models in respective clusters is taken. These weights represent the knowledge for a sparse data stream predictive model should have initially before any encounters with it's data. The u number of RMTPP is then trained based on W_u.

Algorithm 1 Proposed algorithm *spForecast*

```
1: procedure spForecast
2:     Sim ← $\mathscr{I}$
3:     Dis ← $\emptyset$
4:     $U \leftarrow$ SparseDataStreams len(U) $< \gamma$
5:     $V \leftarrow$ DenseDataStreams len(V) $> \gamma$
6:     model ← RMTPP(train,$z_{train}$,$z_{test}$), $\forall z \in U \cup V$
7:     $H \leftarrow$ model.hidden_states()
8:     $W \leftarrow$ model.weights()
9:     X ← generate_random_pairs(m,V)
10:    obj ← RMTPP(test,$v1_{train}$,$v2_{test}$) $\forall (v1, v2) \in X$
11:    e ← obj.mse()
12:    if $e(v_1, v_2) < \theta_1$ then
13:        $Sim \leftarrow Sim \cup \{(v_1, v_2)\}$
14:    if $e(v_1, v_2) > \theta_2$ then
15:        $Dis \leftarrow Dis \cup \{(v_1, v_2)\}$
16:    H' ← MetricLearning(H,Sim,Dis)
17:    clusters ← Kmeans(H',k)
18:    for each item $u$ in $U$ do
19:        $W_u \leftarrow$ Avg$(W_i)$ $\forall i \in clusters(u) \setminus U$
20:        RMTPP(train,$u_{train}$,$u_{test}$,$W_u$)
```

A. Working

This section elaborates on the intuition for our work, thus focusing more on some individual concepts regarding the features of the time series and the similarity.

Forecast Features. The focal point of our method is on capturing the forecast capabilities from the time-series to deduce a similarity relationship amongst them. To ensure that relevant forecast features are extracted from the time-series data streams, we scrutinize the RMTPP model closely. Looking at equations 3 and 4, such a metric is essentially equivalent to the hidden states at the last time-step of RMTPP model for each sequence. Since both the marker and time estimation is computed based on the hidden states of the network, acquiring the same is meaningful for us. Apart from this, h_j marks as an important characteristic of RNN, containing in it, the memory of the influence from the timings and markers of past events. Hence, for our method, we choose h_j as the forecast features resembling a particular time-series' characteristics of it's trends and seasonality as well as demonstrating the predictive capability.

Forecast Weights. Our method not only focuses on the predictive nature of a particular data stream but also recognizes the overall patterns learned in terms of parameters of a model. These parameters are the weights and biases tuned in the embedding, hidden and output layers of the RMTPP. These weights summarize the dynamics of a time series data and is hence, intuitive to conserve them for improved prediction in the case of sparse data streams similar to it. As shown in line (18) of our proposed algorithm, we take the average of the weights of entities similar to a particular sparse data stream as our forecast weights. These forecast weights describe the time-series inherently. On one hand we have the a non-linear transformation of the time series itself in terms of *forecast features* and on the other hand, we have the parameters as *forecast weights* which brought about the transformation.

Similarity. We talk about two concepts for similarity used in our algorithm here - first being the similarity matrices (7) that we generate to learn the distance metric for the forecast features. Comparison of two time-series with respect to forecasting for one time-series based on another's trained model is essential built on the fact that two time series would be called *similar* if their *trends* are said to be similar, i.e. if events with the time of occurrence of one time series depends on the other. Capturing this within-series dependence for a subset of all pairs in our dataset is essential to determine the similarity between data streams. However, we cannot accomplish this task for all pairs as building n^2 RMTPP models will not be feasible in light of the time it would take for training. Hence, we resolve to taking only a subset of such pairs to determine the similarity and dissimilarity matrices.

Post learning a distance metric and transforming the forecast features, in our algorithm we simply use K-means for clustering. Keeping in mind the predictive capability and importance of the forecast features as discussed, they are the most apt to measure similarity between time series. Simple euclidean distance between them does not result into their closeness. However, transforming with metric A, talked above, results in pushing the dissimilar forecast features. This set proves to be vital for clustering using k-means allowing euclidean distance to group sparse and

City Name	RMSE	City Name	RMSE
ALEXANDRIA	0.004382	BENTON	0.006681
AMARILLO	0.001018	BILLINGS	0.001347
ANDERSON	0.001804	BISMARCK	0.002758
ANN ARBOR	0.002126	BOISE	0.001123
AURORA	0.000406	BOWLING GREEN	0.002477
BATTLE CREEK	0.001703	BREMERTON	0.009486
BEAUFORT	0.003245	CHARLESTON	0.005093
BECKLEY	0.006714	CAMBRIDGE	0.009146
BELLINGHAM	0.004879	CHARLOTTESVILLE	0.004207
’BEND’	0.002754	CEDAR RAPIDS	0.002170

TABLE I: Simple RMTPP For High Crime Cities. RMSE is given for regularization parameter 0.0001

City Name	RMSE (0.0)	RMSE (0.1)	RMSE (0.01)	RMSE (0.001)	RMSE (0.0001)
DUBUQUE	0.019274	0.018376	**0.018331**	0.023049	0.019063
GRAND FORKS	**0.027186**	0.039014	0.031842	0.031696	0.027343
MARION	0.005252	0.011255	0.007947	**0.00508**	0.005106
MOSES LAKE	0.011285	0.012806	**0.008993**	0.010224	0.010219
MOUNT VERNON	**0.014611**	0.014727	0.016353	0.016952	0.014771
OLYMPIA	**0.030224**	0.030821	0.031357	0.031196	0.032295
RICHMOND	0.030224	0.009117	0.007216	**0.006915**	0.007229
SANDUSKY	0.030224	0.02371	**0.023571**	0.023677	0.023598
STILLWATER	0.030224	0.022609	0.023034	**0.022281**	0.023638
WALLA WALLA	**0.030224**	0.038971	0.038158	0.03978	0.035028

TABLE II: Simple RMTPP prediction error for 10 low crime cities.

dense data streams together.

Further in the experimentation section, step by step explanation of our proposed algorithm is explained.

VI. Experiments

We evaluate our method largely on crime events of 141 cities in United States. The criminal records of these cities describe the crime type and the time stamp of the crime. The records are available for a period of 5 years from 2007 to 2014. Different crime types like assault, robbery, rape, murder is accounted for, in this crime registry. Some crime events are distributed over days whereas some like assault occur multiple times a day. Neglecting crime types like assault, our focus is more towards those crime types whose occurrence is distributed over days and where prediction of these crime types is a key to optimizing resources and allocating them effectively. Hence, we look into crime types like murder and rape to investigate their trends.

A day-wise time stamp along with the crime type is provided to our algorithm for prediction. Dividing the dataset into train and test sets and then normalizing the train set for time stamp (ranging from 0-1) and test set for time stamp (from 1 onwards) is the first preprocessing step performed by our algorithm. The last six months worth data is kept aside for testing.

Next, the RMTPP code [24] already available is used for predictive analysis on all the cities with some modifications to suit our needs. The hidden units in the embedding layer to encode the time stamps was set to be 8 while in the hidden layer, they were set as 16. A network with an intensity loss function is trained with a learning rate of 0.00001. The model learning of each city was done until the validation error dipped to a point from where it started to increase. The test accuracies for 20 high crime cities from those models are listed in Table I. 10 out of 141 cities were found to be sparsely distributed data streams which had an error more than the average of the rest. These are listed in table II. Crimes as low as 10 per city was also observed for these low crime cities.

K-means clustering was performed on the forecast features producing 4 clusters of cities. Since the clustering is based on the crime pattern of different cities, the crime trends of a high crime city belonging to the same cluster as a low crime city is essentially similar in nature. Figure 2.a shows the clusters. Crime patterns captured in the weights of the RMTPP model is fetched and averaged over all the high crime cities in a cluster. Initialization of a low crime RMTPP model with these set of weights prove to result in low error rates as shown in table III.

For metric learning based experiments, 1000 random pairs of cities were taken and their similarity found out by comparing the errors as per equation 7 above. Similarity threshold was set to 0.003 and the dissimilarity threshold was 0.006. The transformed hidden forecast features (H') is plotted with it’s clusters. As we see from the two plots 2a and 2b, the similar forecast features are brought together whereas the dissimilar ones have separated out leading to well defined clusters in 2.b. Eventually, the errors using the weight initialization method are noted in table III

Comparing the two approaches in table III, we see that our weight initialization scheme performs better for sparsely distributed time series data. All the experiments have been varied with various L2 regularization parameters. Since a particular trend is not identified throughout the columns of the table, it supports the fact that different time-series would have different properties.

We perform the same set of experiments on another open dataset [25] which contains all online transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. Predicting the next purchase of a product is the primary concern here. We notice a similar distribution for transaction patterns of each product. Some products have continuous buyers whereas a few have infrequent sales. Looking at the opportunity for predicting accurate sales for these infrequent sales products, we perform the experiments to behold the results.

The dataset has the time stamp which is converted into a minute scale and the event in this case is occurrence of a transaction. A RMTPP model with a same set of parameter specification is run on all high and low seller products. Table IV lists the error in prediction for high sales data and table V shows the error in prediction for low transactional products.

After k-means clustering on the hidden states of the RMTPP model for all the products, we observe the two plots figure 2c and 2d revealing the formation of well-defined

TABLE III: Prediction errors for low crime dataset using our proposed algorithm

Methods	**RMSE with variation in L2 parameters (weights initialization)**					**RMSE with variation in L2 parameters (metric learning + weights initialization)**				
City Name	0.0	0.1	0.01	0.001	0.0001	0.0	0.1	0.01	0.001	0.0001
DUBUQUE	0.019987	**0.012677**	0.018177	0.017219	0.018636	0.019065	0.019815	0.018248	0.014779	**0.013372**
GRAND FORKS	0.030535	**0.018205**	0.020515	0.033441	0.029107	0.028695	0.025954	0.028591	**0.017503**	0.03389
MARION	**0.005076**	0.010655	0.005088	0.005082	0.007523	0.007316	0.011068	0.007585	0.006041	**0.005078**
'MOSES LAKE'	0.010105	0.010806	**0.008063**	0.009882	0.011441	**0.007405**	0.011896	0.009217	0.011308	0.010016
'MOUNT VERNON'	0.014935	**0.014235**	0.014556	0.014775	0.014957	0.015085	0.013969	**0.01283**	0.014958	0.016317
OLYMPIA	0.030727	0.031337	**0.029522**	0.031262	0.031983	0.031114	0.030323	0.030212	0.032006	**0.029996**
RICHMOND	0.006776	0.009026	0.006941	0.006772	**0.006654**	**0.006395**	0.009294	0.00693	0.006805	0.006837
SANDUSKY	0.023332	0.021431	0.022678	0.021901	**0.01413**	0.022892	0.0223	0.02305	**0.021455**	0.023065
STILLWATER	0.023104	0.023113	0.023033	0.022734	**0.022353**	0.022736	0.022488	**0.022282**	0.022692	0.022329
'WALLA WALLA'	0.040305	0.039963	0.040809	**0.035921**	0.03787	0.039979	0.038968	0.040893	0.034763	**0.032907**

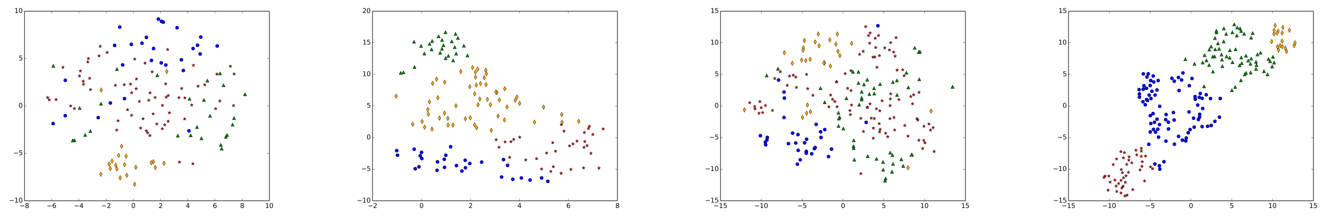

(a) (b) (c) (d)

Fig. 2: Scatter Plot showing the 2-dimensional forecast features along with their clusters. Dimensionality reduction method used was T-SNE. a) H for crime dataset b) H' features for crime data set c) H for online retail d) H' for online retail.

Product Name	**RMSE**
6 RIBBONS RUSTIC CHARM	0.001533
60 CAKE CASES VINTAGE CHRISTMAS	0.001109
BLUE HARMONICA IN BOX	0.001224
CHILLI LIGHTS	0.001771
DOORMAT UNION FLAG	0.002485
GUMBALL COAT RACK	0.001576
HOT BATHS METAL SIGN	0.001991
LUNCH BAG BLACK SKULL	0.002738
PAPER CHAIN KIT 50'S CHRISTMAS	0.002032
POPCORN HOLDER	0.001291
STRAWBERRY CERAMIC TRINKET BOX	0.002093
ANTIQUE SILVER T-LIGHT GLASS	0.001665
BATHROOM METAL SIGN	0.002313
BOX OF 24 COCKTAIL PARASOLS	0.001972
CREAM HEART CARD HOLDER	0.002593
GARDENERS KNEELING PAD CUP OF TEA	0.002189
HAND WARMER OWL DESIGN	0.001053
JUMBO BAG TOYS	0.003823
METAL HANGER FRENCH CHATEAU	0.002763
PHOTO FRAME CORNICE	0.001638

TABLE IV: Simple RMTPP For High Sales Products. RMSE noted for regularization parameter 0.0001

Product ID	**RMSE (0.0)**	**RMSE (0.1)**	**RMSE (0.01)**	**RMSE (0.001)**	**RMSE (0.0001)**
1	**0.04822**	0.049651	0.048256	0.048274	0.04832
2	0.014484	0.015139	0.014496	0.014424	**0.013928**
3	0.008239	**0.008154**	0.008835	0.008863	0.008787
4	0.025133	**0.024978**	0.025108	0.025048	0.025053
5	**0.059203**	0.059317	0.059459	0.059536	0.059399
6	0.004419	0.00488	0.004994	**0.004416**	0.005177
7	0.012814	**0.012754**	0.013104	0.01292	0.013018
8	**0.024799**	0.024924	0.02519	0.025211	0.024948
9	0.003112	0.003447	**0.003111**	0.003112	0.003112

TABLE V: Simple RMTPP prediction error for low sales products with variation in L2 params.

Product ID	**Product Name**
1	4 PURPLE FLOCK DINNER CANDLES
2	50'S CHRISTMAS GIFT BAG LARGE
3	DOLLY GIRL BEAKER
4	I LOVE LONDON MINI BACKPACK
5	NINE DRAWER OFFICE TIDY
6	OVAL WALL MIRROR DIAMANTE
7	RED SPOT GIFT BAG LARGE
8	TRELLIS COAT RACK
9	10 COLOUR SPACEBOY PEN

TABLE VI: Product dictionary

clusters in 2d.

According to the weight initialization module, the weights from the RMTPP model of similar products are averaged and set to initialize the RMTPP for low selling products. The errors reported are shown in table VII. Again with the transformed forecast feature clusters, the weight initialization module is run and the output errors is noted as shown in table VII.

Experimental Results. The tables related to both the experiments show that our proposed approach decreases the forecast errors for sparse data streams. The transformation of the forecast features into a much 'closer' space (fig 2b and 2d) by inducing a similarity information to the metric learning module led to well demarcated clusters. Error

Methods	RMSE with variation in L2 parameters (weights initialized)					RMSE with variation in L2 parameters (metric learning + weights initialized)				
Product ID	0.0	0.1	0.01	0.001	0.0001	0.0	0.1	0.01	0.001	0.0001
1	0.0481	0.049651	0.048236	0.048021	**0.047989**	**0.047974**	0.049654	0.048171	0.048034	0.048002
2	0.01467	0.014386	0.014757	**0.014268**	0.014809	0.014262	0.014493	**0.014134**	0.014173	0.014319
3	0.008156	0.007546	0.009239	0.00789	**0.007262**	0.007606	0.00803	0.008358	**0.007021**	0.008299
4	0.025086	0.025099	**0.024944**	0.025008	0.025065	**0.024962**	0.024985	0.025123	0.025	0.025121
5	0.059466	**0.059315**	0.059474	0.059362	0.05934	0.059517	0.059551	0.059326	0.059328	**0.059214**
6	0.005084	0.004959	**0.003788**	0.005172	0.004997	0.005051	0.005527	**0.004384**	0.00508	0.004992
7	0.013341	0.012824	0.012779	**0.011777**	0.012889	0.012884	**0.012605**	0.012985	0.013011	0.013057
8	**0.024735**	0.025012	0.024824	0.024861	0.024752	0.025348	**0.024738**	0.025019	0.024792	0.02501
9	**0.00282**	0.003476	0.003111	0.003111	0.003111	**0.003101**	0.00334	0.003112	0.003112	0.003113

TABLE VII: Prediction errors for low sales products using our proposed algorithm

deduction in the weight initialization method essentially measures to the decrease in error of time of crime/sales predicted by our system. For the crime dataset - since the data is a 7 year (approximately 7*365 days) long data, the decrease in error (for instance DUBUQUE) is 0.005599 and is equivalent to an improvement of approximately 14 days of prediction error. Similarly, the reduction in error observed in the retail data is quite of importance. Since the data is a 1 year long data (365*24*60 minutes), calculating the reduction of error (for Product ID 6) is 0.000628, which corresponds to approximately an improvement of 330 minutes.

The **time complexity** of the algorithm can be described as O(n) (time taken to train n RMTTP models in step 6) + O(m^2) (the time taken to train some m^2 RMTPP models for in step 10) which can be approximated to O(m^2). For example, in the case of 141 crime cities, the training of 141 RMTPPs takes a few hours but the training of some 30^2 cities could take up to a few days. However, since our task is to determine the similarity matrix, the time can also be reduced to a few hours by optimizing the code and storing the models for all 141 cities. Hence, the time taken for this algorithm is essentially a few hours as contrasted to different deep learning algorithms which take days. Online forecasting platforms could also incorporate this method as the saved models can be used for day-wise predictions and updating the models at the end of a day or in a regular interval of days. Since, the retail dataset contains of lot many products, the RMTPP training takes around 1 day and hence, the models in the back-end can be updated on an appropriate interval. The timings can be improved if the tasks are further run on a GPU system as all our experiments have been run on a CPU system, our reports of time are in commensurate.

VII. Conclusion

In this paper, we have proposed *spForecast*, a novel method for forecasting in sparse data streams by learning similarities with dense data streams. *spForecast* is based on a deep neural point process model and uses novel weight initialization using metric learning. The representative study in this paper has clearly suggested its high potential to real-world problems, even when we have no domain knowledge on the problem at hand. We demonstrated the efficacy of the proposed algorithm on real world crime and customer purchase datasets where the proposed algorithm shows excellent performance. Future work includes handling cases with combinatorially large number of markers (event types).

References

[1] J. G. De Gooijer and R. J. Hyndman, *"25 years of time series forecasting,"* International journal of forecasting, vol. 22, no. 3, pp. 443–473, 2006. A survey on forecasting of time series data by G. Mahalakshmi ; S. Sridevi ; S. Rajaram

[2] Haipeng Shen1 and Jianhua Z. Huang *Forecasting Time Series Of Inhomogeneous Poisson Process With Application To Call Center Workforce Management*

[3] Ryota Kobayashi, Renaud Lambiotte *TiDeH: Time-Dependent Hawkes Process for Predicting Retweet Dynamics*

[4] Charlie S. Marzan, Maria Jeseca C. Baculo, *Time Series Analysis and Crime Pattern Forecasting of City Crime Data* ICACS '17 Proceedings of the International Conference on Algorithms, Computing and Systems

[5] Bao Wang, Penghang Yin, *Deep Learning for Real-Time Crime Forecasting and Its Ternarization*

[6] Chung-Hsien Yu, Max W. Ward, Melissa Morabito, Wei Ding, *Crime Forecasting Using Data Mining Techniques* 2011 IEEE 11th International Conference on Data Mining Workshops

[7] Cristian D. Rodríguez Rodríguez, Diego Mayorga Gomez, Miguel A. Melgarejo Rey, *Forecasting time series from clustering by a memetic differential fuzzy approach: An application to crime prediction*, 2017 IEEE Symposium Series on Computational Intelligence (SSCI)

[8] M.W.T. Gemmink, University of Twente *A Model to Prevent Stockouts in Retail using Time Series Sales Forecasting*

[9] Abhay Jha, Shubhankar Ray, Brian Seaman Inderjit S. Dhillon, *"Clustering to Forecast Sparse Time-Series Data"*, ICDE, 2015

[10] Hongyuan Mei, Jason Eisner, *"The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process"*,submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[11] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan and Stuart Russell, *"Distance metric learning, with application to clustering with side-information"*, submitted to Advances in Neural Information Processing Systems, vol. 15, 2003

[12] Wenjie Pei, David M.J. Tax, Laurens van der Maaten, *"Modeling Time Series Similarity with Siamese Recurrent Networks"*, arXiv preprint arXiv: 1603.04713, 2016.

[13] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, Le Song, *"Recurrent Marked Temporal Point Processes: Embedding Event History to Vector"*, KDD '16, August 13-17, 2016, San Francisco, CA, USA.

[14] Shuai Xiao, Junchi Yan, Stephen M. Chu, Xiaokang Yang, Hongyuan Zh *Modeling The Intensity Function Of Point Process Via Recurrent Neural Networks*

[15] J. G. Rasmussen, *"Temporal point processes: the conditional intensity function"*, http://people.math.aau.dk/fgjgr/teaching/punktproc11/tpp.pdf, 2009.

[16] T. Kanungo, D.M. Mount, N.S. Netanyahu, *"An efficient k-means clustering algorithm: analysis and implementation"*, published in IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 24, Issue: 7, Jul 2002).

[17] E. Bacry, T. Jaisson, and J.-F. Muzy, *"Estimation of slowly decreasing hawkes kernels: Application to high frequency order book modelling"*, 2015

[18] George Mohler, *"Marked point process hotspot maps for homicide and gun crime prediction in Chicago"*, Preprint submitted to Elsevier, June 5, 2014

[19] Tahani Almanie, Rsha Mirza and Elizabeth Lor, *"Crime prediction based on crime types and using spatial and temporal criminal hotspots"*, International Journal of Data Mining and Knowledge Management Process (IJDKP) Vol.5, No.4, July 2015.

[20] Marian-Andrei Rizoiu, Young Lee, Swapnil Mishra, Lexing Xie, *"A Tutorial on Hawkes Processes for Events in Social Media"*, ArXive-prints, 2017.

[21] Alex Reinhart, *"A Review of Self-Exciting Spatio-Temporal Point Processes and Their Applications"*, ArXive-prints, 2018.

[22] D.J. Daley, D. Vere-Jones, *"An Introduction to the Theory of Point Processes"*, Springer, 2003.

[23] Utkarsh Upadhyay, Abir De, Manuel Gomez-Rodriguez, *"Deep Reinforcement Learning of Marked Temporal Point Processes"*, Springer, 2003.

[24] RMTPP Code - https://github.com/dunan/NeuralPointProcess

[25] Retail Dataset - https://archive.ics.uci.edu/ml/datasets/online+retail