Petrel: Personalized Trend Line Estimation with Limited Labels from One Individual

Tong-Yi Kuo and Hung-Hsuan Chen[0000−0001−5137−4449]

National Central University, Taoyuan, Taiwan
kuotony860810@gmail.com, hhchen@acm.org

Abstract. This study proposes a framework for generating customized trend lines that consider user preferences and input time series shapes. The existing trend estimators fail to capture individual needs and application domain requirements. The proposed framework obtains users' preferred trends by asking users to draw trend lines on sample datasets. The experiments and case studies demonstrate the effectiveness of the model. The code and dataset are available at https://github.com/Anthony860810/Generating-Personalized-Trend-Line-Based-on-Few-Labelings-from-One-Individual.

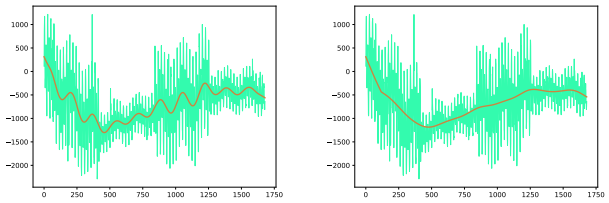
Keywords: Time series analysis · Trend estimation.

1 Introduction

Given a time series vector $\boldsymbol{x} = [x_1, \ldots, x_T]$, a trend line estimation algorithm returns a slowly varying time series $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_T]$ that aims to represent the global pattern of the original time series $\boldsymbol{x}$. However, a trend line has no precise definition: "slowly varying" and "global pattern" are both vague descriptions. As a result, even though many trend estimation algorithms are available [3, 5, 10, 20, 23], none seems to have a dominant advantage over the others.

We believe the ambiguous definition of a trend is inevitable because a proper trend should depend on not only the input time series but also the nature of an application and a user's needs. In other words, even given the same time series, different users or applications may prefer different trends. To demonstrate this, we asked different users to draw trend lines given a fixed time series. The results indeed show that different users illustrate trends with distinct patterns. An example is shown in Figure 1: given a time series, the left trend is sensitive to local turbulence, and the right trend is smoother. A simple survey shows that 30% of users preferred the left trend and 70% preferred the right one. This result motivates our study, which aims to design a framework to estimate a personalized trend given both a time series and a user's personal requirements or preferences.

Unfortunately, it is extremely challenging for a user to concretely illustrate the characteristic of a trend that meets her/his requirements or preferences. Ultimately, we decided to leverage machine learning algorithms to capture a user's needs based on the user's plotted trend lines on limited time series samples. We hope that once the machine learning algorithms determine a user's needs by



(a) Trend line 1 (b) Trend line 2

Fig. 1. Two distinct trend lines on the same time series dataset. A simple survey shows that 30% of users preferred the left one and 70% of them preferred the right. The results show that different users indeed prefer different types of trend lines.

observing these examples, the algorithms can automatically generate customized trend lines for a large number of time series that have the same requirements.

We present a scenario that is well-suited for the application of our personalized trend line estimator. Let us consider a situation where a user needs to produce trend lines for a large set of time series such as stock market prices, website log reports, or Internet traffic reports [7, 15, 1, 13]. However, the currently popular trend estimators such as ℓ_1 trend filtering or Hodrick-Prescott filtering are not able to generate satisfactory trends that align with the user's specific requirements as these estimators are not personalized. Using our work, a user only needs to draw trends on a small number of time series samples, and our algorithm will identify the characteristics of the user's preferred trend lines and generate customized trend lines for all other time series.

As we will show later in the experiments, directly learning the relationship between an input time series and the trend line tends toward overfitting given limited training samples labeled by a user. As a result, the main technical challenge of our work becomes designing a framework to learn a user's preferred shapes of trend lines from limited samples.

The main contributions of this paper include the following. First, we propose a new research problem – estimating a customized trend from a time series with personalized or application-specific requirements. We explain why this task is challenging. Second, we propose a personalized trend line estimator named Petrel to address this problem. Petrel, by design, learns a user's requirements even when the user's labeled trend line samples are limited. Third, we conducted thorough experiments to compare our proposed method with supervised algorithms and few-shot learning algorithms, both of which learn to map a time series to a personalized trend line. Additionally, we conducted case studies to demonstrate that Petrel indeed identifies a user's preferences. Finally, we release both the source code and the experimental dataset for reproducibility. The dataset includes the trends plotted by the real users we recruited. The dataset alone could be an invaluable resource for researchers studying trend line estimation.

Table 1. Notation list

Indices:	
T	Length of a time series ($t \in \{1, \ldots, T\}$)
N	Number of original time series ($n \in \{1, \ldots, N\}$)
U	Number of simulated users ($u \in \{1, \ldots, U\}$)
M	Number of user-labeled trends ($m \in \{1, \ldots, M\}$)
J	Number of test series ($j \in \{1, \ldots, J\}$)
K	Number of base trend estimators ($k \in \{1, \ldots, K\}$)
Variables for simulated trend generation and model training:	
b_k	a non-customized base trend generating function that takes a time series as the input argument
$\boldsymbol{w}_u$	the affine coefficients for a simulated user; $\boldsymbol{w}_u = [w_{u,1}, \ldots, w_{u,K}]$
$\boldsymbol{x}_n^{orig}$	a collected time series used during training; $\boldsymbol{x}_n^{orig} = \left[x_{n,1}^{orig}, \ldots, x_{n,T}^{orig}\right]$
$\hat{\boldsymbol{y}}_{n,u}^{sim}$	a simulated trend line for simulated user u on $\boldsymbol{x}_n^{orig}$; $\hat{\boldsymbol{y}}_{n,u}^{sim} = \sum_{k=1}^{K} w_{u,k} b_k(\boldsymbol{x}_n^{orig})$
Variables for inference and verification:	
$\boldsymbol{x}_j^{te}$	a test series; $\boldsymbol{x}_j^{te} = \left[x_{j,1}^{te}, \ldots, x_{j,T}^{te}\right]$
$\hat{\boldsymbol{y}}_j^{te}$	the estimated personalized trend for $\boldsymbol{x}_j^{te}$; $\hat{\boldsymbol{y}}_j^{te} = \left[\hat{y}_{j,1}^{te}, \ldots, \hat{y}_{j,T}^{te}\right]$
$\boldsymbol{y}_j^{te}$	the ground truth of the personalized trend for $\boldsymbol{x}_j^{te}$; $\boldsymbol{y}_j^{te} = \left[y_{j,1}^{te}, \ldots, y_{j,T}^{te}\right]$
$\boldsymbol{x}_m^{lab}$	a time series sample for a user to label with a trend line sample; $\boldsymbol{x}_m^{lab} = \left[x_{m,1}^{lab}, \ldots, x_{m,T}^{lab}\right]$
$\hat{\boldsymbol{y}}_m^{lab}$	the estimated trend line for $\boldsymbol{x}_m^{lab}$; $\hat{\boldsymbol{y}}_m^{lab} = \left[\hat{y}_{m,1}^{lab}, \ldots, \hat{y}_{m,T}^{lab}\right]$
$\boldsymbol{y}_m^{lab}$	the trend line sample for $\boldsymbol{x}_m^{lab}$ labeled by the user; $\boldsymbol{y}_m^{lab} = \left[y_{m,1}^{lab}, \ldots, y_{m,T}^{lab}\right]$

2 Personalized Trend Line Estimation

2.1 Task Overview

We follow the notation list shown in Table 1 in this paper.

A standard trend line estimator outputs $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_T]$ given a time series vector $\boldsymbol{x} = [x_1, \ldots, x_T]$. In this paper, we call a standard trend line estimator b_k a "non-customized base trend generator" because b_k generates $\hat{\boldsymbol{y}}$ without considering any personalized factors. Famous examples of non-customized estimators include Hodrick-Prescott filtering (HP filtering) [6, 12], ℓ_1 trend filtering [10], and seasonal-trend decomposition using LOESS (STL) [4].

In contrast, a personalized trend line estimator takes both $\boldsymbol{x}$ and the user's requirements as the input to generate $\hat{\boldsymbol{y}}$. Since it could be complicated for a user to specify the requirements directly, we ask the user to draw trend lines (called "trend line samples" below) on a small number of time series (called "time series samples" below) and let our model determine the user's preferred or required characteristics of the trend line. Specifically, let $\boldsymbol{y}_m^{lab} = \left[y_{m,1}^{lab}, \ldots, y_{m,T}^{lab}\right]$ be the trend line sample labeled by a user on a time series sample $\boldsymbol{x}_m^{lab} = \left[x_{m,1}^{lab}, \ldots, x_{m,T}^{lab}\right]$; we want to learn a customized trend line estimator f that transforms $\boldsymbol{x}_m^{lab}$ into $\boldsymbol{y}_m^{lab}$. Once f is obtained, we can estimate $\hat{\boldsymbol{y}}_j^{te}$ as a personalized trend for the time series $\boldsymbol{x}_j^{te}$ using $f(\boldsymbol{x}_j^{te})$.

2.2 Challenge of the Task

An obvious approach to performing the aforementioned task is to train a supervised learner to map a time series sample $\boldsymbol{x}_m^{lab}$ to a trend line sample $\boldsymbol{y}_m^{lab}$. However, such an approach requires a large number of user-labeled trends. In our scenario, since a user only draws trend line samples for few time series samples, a supervised learning algorithm tends to overfit the training data.

A possible strategy to address the issue of small training data is the pretraining and fine-tuning strategy used in few-shot learning [18, 22]. However, as we will show later in the experiments, although few-shot learning performs moderately better than supervised learners, the estimated personalized trends are still unsatisfactory. In summary, it is challenging to learn a personalized trend estimator based on a limited number of $\boldsymbol{x}_m^{lab}$ and $\boldsymbol{y}_m^{lab}$ labeled by a single user.

2.3 Petrel Model – Training

We propose the Petrel model to estimate personalized trends. Instead of directly learning to map $\boldsymbol{x}_m^{lab}$ onto $\boldsymbol{y}_m^{lab}$, Petrel consists of a two-stage training process.

Collecting a large number of time series is simple, but labeling the personalized trends for them is laborious. In stage 1 of the training, we generate simulated personalized trends from a large collection of data series. For a collected time series $\boldsymbol{x}_n^{orig}$, we use K non-customized base trend generators to generate K different trends: $b_1(\boldsymbol{x}_n^{orig}), \ldots, b_K(\boldsymbol{x}_n^{orig})$. Next, we generate a simulated personalized trend line $\hat{\boldsymbol{y}}_{n,u}^{sim}$ for a simulated user u by assuming that $\hat{\boldsymbol{y}}_{n,u}^{sim}$ is composed of an affine combination of $b_k(\boldsymbol{x}_n^{orig})$, as shown by the following equation.

$$\hat{\boldsymbol{y}}_{n,u}^{sim} = \sum_{k=1}^{K} w_{u,k} b_k(\boldsymbol{x}_n^{orig}), \tag{1}$$

where the $w_{u,k}$s are the affine coefficients representing the characteristics of user u's preferred trend. We vary the values of $[w_{u,1}, \ldots, w_{u,K}]$ to simulate different user us so that different simulated personalized trends $\hat{\boldsymbol{y}}_{n,u}^{sim}$ are generated even when $\boldsymbol{x}_n^{orig}$ is fixed. For the non-customized base trend generating functions, we selected three trend estimators: b_1 is ℓ_1 trend filtering [10], b_2 is HP filtering [6], and b_3 is STL estimation [4]. It is straightforward to include other base trend generators, e.g., the ARIMA model [9] or a local regression model.

In stage 2, we train an affine coefficient estimator f_{coef}. Given a time series $\boldsymbol{x}_n^{orig}$ and a simulated personalized trend $\hat{\boldsymbol{y}}_{n,u}^{sim}$ that was generated in stage 1 as the input features, we want the affine coefficient estimator f_{coef} to return the affine coefficients $[w_{u,1}, \ldots, w_{u,K}]$ after training. In our experiment, f_{coef} is composed of 4 layers of 1D convolutions (each with ReLU as the activation function) followed by a fully connected layer (with softmax as the activation function to ensure that the sum of the outputs is 1). The input includes 2 channels – one for the time series $\boldsymbol{x}_n^{orig}$ and the other for the simulated personalized trend $\hat{\boldsymbol{y}}_{n,u}^{sim}$.

The first stage of this training strategy can generate a large number of training instances for the second stage to train the affine coefficient estimator f_{coef}, which plays a crucial role during inference, as described below.

2.4 Petrel Model – Inference

The Petrel inference method also involves two stages. In stage 1, the user is asked to plot M trend line samples (i.e., $\boldsymbol{y}_1^{lab}, \ldots, \boldsymbol{y}_M^{lab}$) on M time series samples (i.e., $\boldsymbol{x}_1^{lab}, \ldots, \boldsymbol{x}_M^{lab}$) for a small value of M. For each pair $\left(\boldsymbol{x}_m^{lab}, \boldsymbol{y}_m^{lab}\right)$, we estimate the affine coefficients to generate the trend $\boldsymbol{y}_m^{lab}$ using Equation 2.

$$\boldsymbol{w}_m = [w_{m,1}, \ldots, w_{m,K}] = f_{\text{coef}}\left(\boldsymbol{x}_m^{lab}, \boldsymbol{y}_m^{lab}\right) \tag{2}$$

In stage 2, we estimate the personal affine coefficients ($[w_1^*, \ldots, w_K^*]$) for a user and generate the personalized trend $\hat{\boldsymbol{y}}_j^{te}$ for a test series $\boldsymbol{x}_j^{te}$. We use two methods of estimating the personal affine coefficients. The first is a simple average of $\boldsymbol{w}_1$ to $\boldsymbol{w}_M$, as shown in Equation 3.

$$w_k^* = \frac{1}{M} \sum_{m=1}^{M} w_{m,k} \tag{3}$$

The second way to compute w_k^* is to take a weighted sum of $\boldsymbol{w}_1$ to $\boldsymbol{w}_M$. The weights should be inversely correlated with the distance between $\boldsymbol{y}_m^{lab}$ and $\hat{\boldsymbol{y}}_m^{lab}$ (the estimated trend line for $\boldsymbol{x}_m^{lab}$). Ultimately, we define the distance by the symmetric mean absolute percentage error (SMAPE) (defined in Equation 7). The personal affine coefficient (estimated by weighted sum) is shown in Equation 4.

$$w_k^* = \sum_{m=1}^{M} \alpha_m w_{m,k}, \tag{4}$$

where α_m is defined in Equation 5.

$$\alpha_m = \frac{1 \Big/ \text{SMAPE}\left(\boldsymbol{y}_m^{lab}, \hat{\boldsymbol{y}}_m^{lab}\right)}{\sum_{i=1}^{M} 1 \Big/ \text{SMAPE}\left(\boldsymbol{y}_i^{lab}, \hat{\boldsymbol{y}}_i^{lab}\right)} \tag{5}$$

Once w_k^* is obtained by either a simple average (Equation 3) or weighted sum (Equation 4), we estimate the personalized trend for a test series $\boldsymbol{x}_j^{te}$ by

$$\hat{\boldsymbol{y}}_j^{te} = \sum_{k=1}^{K} w_k^* b_k\left(\boldsymbol{x}_j^{te}\right). \tag{6}$$

3 Experiments

3.1 Experimental Datasets

We used two datasets for the experiments. The first dataset is the Yahoo! S5 time series dataset [17]. We asked users to plot trends on some of these time series. The second dataset includes real users' plotted trends on manually created time series. Some of these user-plotted trends will be regarded as trend line samples (y_m^{lab}), and others will be regarded as the ground truth of the personalized trend lines that will be used for evaluation.

Table 2. A comparison of trend estimation algorithms on the Yahoo! S5 series

Type	Algorithm	SMAPE	MSE
Our method	Petrel (averaged)	**0.44**	5264.34
	Petrel (weighted)	**0.44**	**5258.34**
DNN models	ConvNet	0.83	176593.87
	LSTM	1.02	497312.33
	Transformer	1.08	579188.89
DNN with pretraining and fine-tuning	P&F ConvNet	**0.44**	5425.77
	P&F LSTM	0.52	7394.09
	P&F Transformer	0.47	9311.75
	P&F MLP	0.68	31934.92

Table 3. A comparison of trend estimation algorithms on the manually created series

Type	Algorithm	SMAPE	MSE
Our method	Petrel (averaged)	0.33	6164.38
	Petrel (weighted)	**0.32**	**6002.32**
DNN models	ConvNet	0.94	166951.8
	LSTM	1.11	323712.95
	Transformer	1.20	637955.96
DNN with pretraining and fine-tuning	P&F ConvNet	1.45	241890.91
	P&F LSTM	1.23	1292454.44
	P&F Transformer	0.81	1357013.58
	P&F MLP	1.18	242234.14

3.2 Experimental Scenario

To generate personalized trends for the participants, we asked each participant to draw trend lines on 10 time series samples. These time series samples, along with the trend line samples plotted by users, were used by the algorithms to learn or infer a user's preferences regarding the shapes of the trends.

Next, Petrel and each of the compared baselines generated personalized trends for 5 time series in Yahoo! S5 ($\boldsymbol{x}_n^{orig}$) and another 5 manually created time series ($\boldsymbol{x}_j^{te}$). We asked the users to plot the trends on these 10 time series without showing the machine-generated trends. We compare the distances between a user's plotted trends and the machine-estimated personalized trends.

Although the time series in Yahoo! S5 (i.e., $\boldsymbol{x}_n^{orig}$) are available in the beginning, the ground truths of the corresponding personalized trends are not given. In particular, Petrel uses the simulated personalized trend $\hat{\boldsymbol{y}}_{n,u}^{sim}$ as part of the input feature; the pretraining and fine-tuning models (the baseline models that will be introduced in the next section) use $\hat{\boldsymbol{y}}_{n,u}^{sim}$ as the target during the pretraining step, but the $\hat{\boldsymbol{y}}_{n,u}^{sim}$ values are not the ground truths of the personalized

trends. To compare the generalizability of the Petrel model and the baseline models, we also tested each model on the manually generated time series $\boldsymbol{x}_j^{te}$ that differed from any time series in $\boldsymbol{x}_n^{orig}$ or $\boldsymbol{x}_m^{lab}$.

3.3 Baseline Methods

We compare Petrel with deep neural network (DNN) models and DNN models with pretraining and fine-tuning strategies. We do not include traditional trend estimators (e.g., ℓ_1 trend filter or HP filter) because they are non-personalized.

The first type of baseline model (DNNs) includes the convolutional neural network (ConvNet), long short-term memory (LSTM), and Transformer. Each model learns to use each of the 10 time series samples as the input features to predict the corresponding trend line sample drawn by a user. After training, each model estimates the personalized trend line for each of the test series.

The second type of baseline model applies the pretraining and fine-tuning strategy to the ConvNet (denoted as P&F ConvNet), LSTM (denoted as P&F LSTM), Transformer (denoted as P&F Transformer), and multilayer perceptron (denoted as P&F MLP). The network structure of each model is the same as that used in the first type. However, during the pretraining step, we use $\boldsymbol{x}_n^{orig}$, the collected time series, and $\hat{\boldsymbol{y}}_{n,u}^{sim}$, the simulated trend line, as the training feature and target. In the fine-tuning step, we use $\boldsymbol{x}_m^{sam}$, the time series samples, and $\boldsymbol{y}_m^{sam}$, the user-plotted trend line samples, as the features and targets to fine-tune the parameters in the last layer.

3.4 The Quality of the Estimated Personalized Trends

We quantify the quality of a trend line generating algorithm by comparing its generated trends with users' plotted trends based on the symmetric mean absolute percentage error (SMAPE) and mean squared error (MSE). Let $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_T]$ and $\boldsymbol{y} = [y_1, \ldots, y_T]$ be the algorithm-generated and user-plotted trend lines for a time series with T time steps. The SMAPE and MSE between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ are defined by Equation 7 and Equation 8, respectively.

$$\text{SMAPE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{T}\sum_{t=1}^{T} 2\frac{|\hat{y}_t - y_t|}{|\hat{y}_t| + |y_t|} \tag{7}$$

$$\text{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{T}\sum_{t=1}^{T} (\hat{y}_t - y_t)^2 \tag{8}$$

While the MSE is the most widely used metric to measure the difference between the predictions and observed targets when the target variables are numeric, the MSE is scale dependent, i.e., the score depends on the scale of the time series. On the other hand, the SMAPE scales the value by considering the magnitude of each predicted $\hat{y}_t$ and the observed y_t, so the SMAPE score is always between 0 and 2 (a smaller SMAPE means that the prediction is more accurate). Unfortunately, given two estimated trends $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_2$, $\text{SMAPE}(\hat{\boldsymbol{y}}_1, \boldsymbol{y}) > \text{SMAPE}(\hat{\boldsymbol{y}}_2, \boldsymbol{y})$

does not imply $\text{MSE}(\hat{\boldsymbol{y}}_1, \boldsymbol{y}) > \text{MSE}(\hat{\boldsymbol{y}}_2, \boldsymbol{y})$ (and vice versa). For a fair comparison, we report both the SMAPE and MSE.

Table 2 gives the SMAPEs and MSEs of various methods in predicting personalized trends on the Yahoo! S5 dataset. These time series appear in the training step of Petrel and the pretraining steps of all the pretraining and fine-tuning models. However, the ground-truth labels of the personalized trends are not given. Petrel (averaged) and Petrel (weighted) refer to the strategies used to determine the personal affine coefficients w_k^*: either using Equation 3 (averaged) or Equation 4 (weighted). The results show that Petrel outperforms all the baseline methods in terms of both SMAPE and MSE on Yahoo! S5. The DNN models perform poorly, likely because of the limited number of training instances from the time series samples and trend line samples. For the pretraining and fine-tuning strategies, although the trends in the pretraining step are synthesized by Equation 1, these synthetic trends are still helpful in alleviating overfitting.

Table 3 shows the quality of the predicted personalized trends for Petrel and the baseline models on the manually created time series that do not appear during all training (or pretraining) steps. The Petrel model shows a more obvious advantage. Both types of baseline models (DNNs and DNNs with pretraining and fine-tuning) generate trends that are very different from a user's plotted trends. When comparing the results with the results shown in Table 2, the performance of the pretraining and fine-tuning strategies becomes much worse. This is likely because the pretraining and fine-tuning strategy works only when the test series appear in the pretraining step but fails if the test series differ greatly from those appearing in the pretraining step.

3.5 Case Study

This section presents a case study of two users to demonstrate that Petrel indeed identifies the users' preferred trend shapes. We only show the averaged version of Petrel here because the weighted version gives similar results.

Figure 2 shows three trend samples plotted by users A and B: user A prefers the trends to be sensitive to local turbulence, but user B prefers smooth trends.

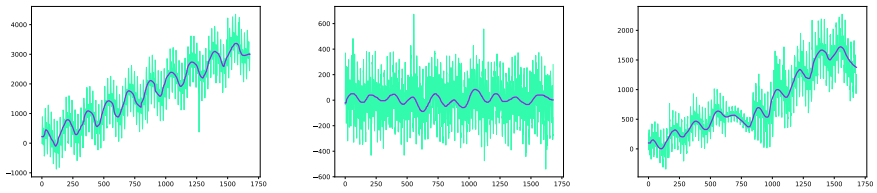
Figure 3 shows the personalized trends generated by Petrel on three series presented in the Yahoo! S5 dataset (each column represents the same series). The personalized trend for user A appears sensitive to local turbulence, while user B's trend is smoother. This matches our assessment of their preferences.

Figure 4 illustrates the personalized trends generated by Petrel on three manually crated series, i.e., series that do not appear in any training phase. Again, when given the same time series, the estimated personalized trend for user B is smoother than the one generated for user A.

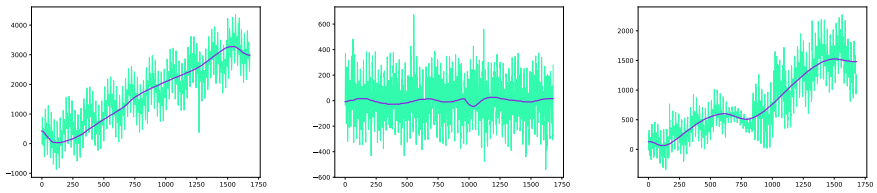
4 Related Work

4.1 Trend Estimation

Trend line estimation is a fundamental problem in time series analysis. It aims to find a slowly varying curve that represents a given sequence well. It has been



(a) Trend 1 by user A (b) Trend 2 by user A (c) Trend 3 by user A



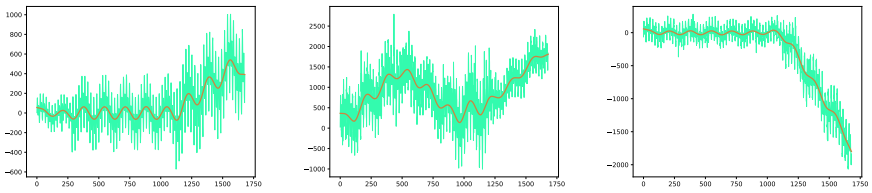
(d) Trend 1 by user B (e) Trend 2 by user B (f) Trend 3 by user B

Fig. 2. Three trend line samples plotted by user A and user B. It appears that user A prefers the trends to be sensitive to local turbulence, but user B prefers smooth trends.

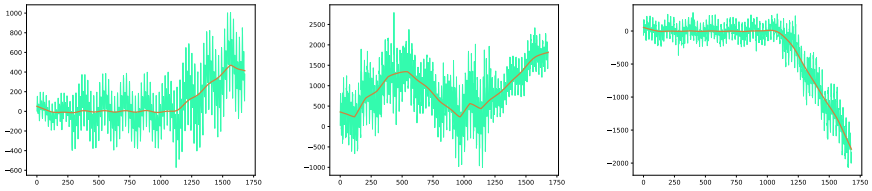
shown that estimating trend lines improves the performance of time series analysis tasks. For example, removing trends from a time series as a preprocessing step may improve the prediction quality of various target tasks [11, 16, 24].

Trend estimating algorithms can be parametric or nonparametric. Parametric models aim to find a function that transforms the observed time series into a trend. The simplest model of this form is probably the linear regression model – given a time series $\boldsymbol{x} = [x_1, \ldots, x_T]$, a linear regression model assumes a linear relationship between each x_j $(j \leq t)$ and $\hat{y}_t$ and looks for parameters that minimize a given objective. This concept can be easily extended to high-order polynomial regressions or even more complex functions. However, to ensure the "slowly varying" property that is usually required for a trend line, extremely high-order polynomial regressions or overcomplex functions are rarely used in practice for trend estimation. The famous autoregressive integrated moving average (ARIMA) model [8, 9] also falls into the family of parametric models. ARIMA assumes that the difference between neighboring $\hat{y}_t$s is linearly correlated with both their lagged values and the previous error terms.

Trend estimating algorithms can also be nonparametric. These algorithms usually need an objective function to define the quality of an estimated trend, but they do not assume a fixed relationship between $\boldsymbol{x}$ and $\hat{\boldsymbol{y}}$. The objective function usually involves two parts – the estimated $\hat{y}_t$ should be close to x_t, and the difference between neighboring $\hat{y}_t$s should be small. Famous nonparametric algorithms for trend estimation include ℓ_1-trend filtering [10] and HP filtering [6]. The main difference between ℓ_1-trend filtering and HP filtering is their definitions of closeness between neighboring $\hat{y}_t$s. Local regression methods,



(a) Personalized trend 1 estimated by Petrel for user A (b) Personalized trend 2 estimated by Petrel for user A (c) Personalized trend 3 estimated by Petrel for user A



(d) Personalized trend 1 estimated by Petrel for user B (e) Personalized trend 2 estimated by Petrel for user B (f) Personalized trend 3 estimated by Petrel for user B

Fig. 3. The personalized trends for user A and user B generated by Petrel (averaged) on three Yahoo! S5 time series (the time series used to generate the simulated trends).

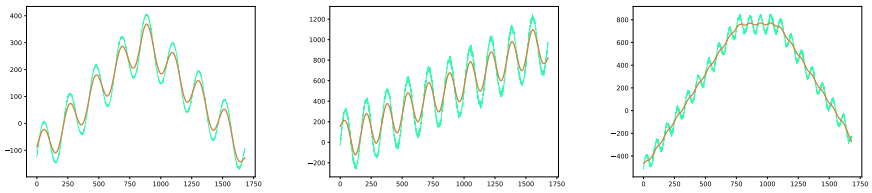
e.g., locally estimated scatterplot smoothing (LOESS) and locally weighted scatterplot smoothing (LOWESS), can also be used to estimate trends by treating the x_ts as the input features. When extending trend estimation to 2-dimensional input, a graph trend can be generated [21, 23].

4.2 Few-shot Learning

Few-shot learning (FSL) refers to a scenario in which machine learning is applied with a limited number of training instances [19, 22]. In many cases, FSL tries to embed prior knowledge into the model and update the model based on the few training instances. A typical way to embed prior knowledge is to use a pretraining strategy, which usually leverages a relevant dataset with many training instances to train a model. The fine-turning stage uses the few training instances to update all or part of the model parameters. The FSL strategy has been successfully applied to various application domains [2, 14].

5 Conclusion

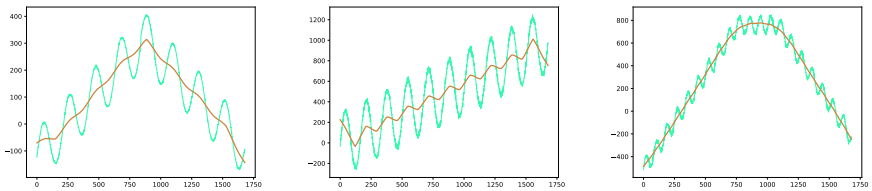
In this paper, we introduce a new and challenging task: estimating personalized trends for a large number of time series based on limited samples labeled by a user. We propose a novel algorithm named Petrel to perform this task. We recruited users to plot personalized trends on synthetic and open datasets. The



(a) Personalized trend 1 for unseen time series estimated by Petrel for user A

(b) Personalized trend 2 for unseen time series estimated by Petrel for user A

(c) Personalized trend 3 for unseen time series estimated by Petrel for user A



(d) Personalized trend 1 for unseen time series estimated by Petrel for user B

(e) Personalized trend 2 for unseen time series estimated by Petrel for user B

(f) Personalized trend 3 for unseen time series estimated by Petrel for user B

Fig. 4. The personalized trends for user A and user B generated by Petrel (averaged) on three manually created series (which did not appear during training).

experimental results show that, compared to DNN models and DNN models with the pretraining and fine-tuning strategy, the Petrel model generates trends closer to those plotted by the users. The case studies also confirm that Petrel can adapt to different users' preferences to generate personalized trends.

Acknowledgements. This work is partially supported by the National Science and Technology Council of Taiwan under grant 110-2222-E-008-005-MY3.

References

1. Bai, G.J., Lien, C.Y., Chen, H.H.: Co-learning multiple browsing tendencies of a user by matrix factorization-based multitask learning. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 253–257 (2019)
2. Bansal, T., Jha, R., McCallum, A.: Learning to few-shot learn across diverse natural language classification tasks. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 5108–5123 (2020)
3. Bianchi, M., Boyle, M., Hollingsworth, D.: A comparison of methods for trend estimation. Applied Economics Letters **6**(2), 103–109 (1999)
4. Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I.: STL: A seasonal-trend decomposition. J. Off. Stat **6**(1), 3–73 (1990)
5. Gray, K.L.: Comparison of trend detection methods. University of Montana (2007)

6. Hodrick, R.J., Prescott, E.C.: Postwar us business cycles: an empirical investigation. Journal of Money, credit, and Banking pp. 1–16 (1997)
7. Hsu, C.Y., Chen, T.R., Chen, H.H.: Experience: Analyzing missing web page visits and unintentional web page visits from the client-side web logs. ACM Journal of Data and Information Quality (JDIQ) **14**(2), 1–17 (2022)
8. Hsu, C.J., Chen, H.H.: Taxi demand prediction based on lstm with residuals and multi-head attention. In: VEHITS. pp. 268–275 (2020)
9. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (2018)
10. Kim, S.J., Koh, K., Boyd, S., Gorinevsky, D.: ℓ_1 trend filtering. SIAM review **51**(2), 339–360 (2009)
11. Laptev, N., Yosinski, J., Li, L.E., Smyl, S.: Time-series extreme event forecasting with neural networks at uber. In: International conference on machine learning. vol. 34, pp. 1–5 (2017)
12. Leser, C.E.V.: A simple method of trend construction. Journal of the Royal Statistical Society: Series B (Methodological) **23**(1), 91–107 (1961)
13. Lien, C.Y., Bai, G.J., Chen, H.H.: Visited websites may reveal users' demographic information and personality. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 248–252 (2019)
14. Lifchitz, Y., Avrithis, Y., Picard, S., Bursuc, A.: Dense classification and implanting for few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9258–9267 (2019)
15. Lin, T.H., Zhang, X.R., Chen, C.P., Chen, J.H., Chen, H.H.: Learning to identify malfunctioning sensors in a large-scale sensor network. IEEE Sensors Journal **22**(3), 2582–2590 (2021)
16. Liu, Z., Hauskrecht, M.: Learning adaptive forecasting models from irregularly sampled multivariate clinical data. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. p. 1273–1279. AAAI'16, AAAI Press (2016)
17. Marlin, B.M., Zemel, R.S.: Collaborative prediction and ranking with non-random missing data. In: Proceedings of the third ACM conference on Recommender systems. pp. 5–12 (2009)
18. Shen, Z., Liu, Z., Qin, J., Savvides, M., Cheng, K.: Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021. pp. 9594–9602. AAAI Press (2021)
19. Tao, R., Zhang, H., Zheng, Y., Savvides, M.: Powering finetuning in few-shot learning: Domain-agnostic bias reduction with selected sampling. In: Thirty-Sixth Conference on Artificial Intelligence, AAAI 2022. pp. 8467–8475. AAAI Press (2022)
20. Tibshirani, R.J.: Adaptive piecewise polynomial estimation via trend filtering. The Annals of Statistics **42**(1), 285–323 (2014)
21. Varma, R., Lee, H., Kovačević, J., Chi, Y.: Vector-valued graph trend filtering with non-convex penalties. IEEE transactions on signal and information processing over networks **6**, 48–62 (2019)
22. Wang, Y., Yao, Q., Kwok, J.T., Ni, L.M.: Generalizing from a few examples: A survey on few-shot learning. ACM computing surveys **53**(3), 1–34 (2020)
23. Wang, Y.X., Sharpnack, J., Smola, A., Tibshirani, R.: Trend filtering on graphs. In: Artificial Intelligence and Statistics. pp. 1042–1050. PMLR (2015)
24. Zhang, G.P., Qi, M.: Neural network forecasting for seasonal and trend time series. European journal of operational research **160**(2), 501–514 (2005)