Personalized Neural Architecture Search for Timeseries Data

Tianran Wei

Course: Data Science & Real-Time Big Data Analytics

July 31, 2020

Abstract

When we make predictions for Time Series Data, a high Accuracy can be achieved due to the development of Machine Learning techniques. Additionally, personalization is another important issue in especially when users of the prediction system have different preferences for the prediction. According to the preference, the prediction model could achieve a performance improvement with the help of parametrization. Providing personalized model with high accuracy and usability to individual users is the goal we pursued. In this paper, we build experimentally a personalization framework, in which two representative ranking algorithms are implemented and compared. Our experiments are meant to simulate the procedure of personalizing Neural Architecture Search based on the Timeseries Data and evaluating the performance of different Ranking Algorithms.

Contents

1 Motivation and Introduction

To provide a comprehensive analysis we will firstly give an introduction to the data. What we have is a time series dataset of power consumption in Stuttgart airport. A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time.

timestamp	value
2018-07-19 14:35:00	6074.53
2018-07-19 14:40:00	5971.69
2018-07-19 14:45:00	5700.07
⋮	⋮
2018-11-14 10:55:00	5591.39

We assume that there are three employees in Stuttgart Airport responsible for the airport's power control, one is responsible for planning the power supply within 24 hours, the second one for the next week and the last for the limit controlling of power consumption. Based on the assumption that the models have a dissimilar performance for the prediction in different periods and performances, the personalization of the prediction model is a crucial way for improving the performance with respect to the individual preference

The personalization procedure in our experiment is composed of modeling, Active Learning, Preference Learning, and Ranking.

With modeling we simulate the models with different performance by generating models with normal distribution.

Active learning (AL) aims to economically learn an accurate model by reducing the annotation cost. It is based on the premise that a model can get better performance if it is allowed to prepare its own training data, by choosing the most beneficial data points and querying their annotations from annotators[1]. Due to restrictions of the experiment, we will simulate the user preference with Mean Square Error(MSE), which means the annotation process is proceeded by calculating the MSE of each model.

In general, a preference learning system in the context of our experiment is provided with a set of models for which preferences are known, and the task is to learn a function that predicts preferences for a new set of models.

In fact, among the problems in the realm of preference learning, the task of "learning to rank" has probably received the most attention in the literature so far[2], and a number of different ranking problems have already been introduced. What we present is an empirical study in which we compare the two most common approaches to this problem: pairwise ranking and pointwise ranking.

2 Data Set and Model Generation

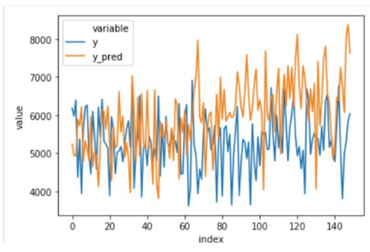
For the reason that we aim to explore the user preference to the prediction model, we generate predictions to simulate the models with different structures and their influence on the prediction. We assume that $Y = (y_1, y_2, ..., y_n)$ is normally distributed holds values describing the power consumption over time. N simulated model Prediction $\hat{Y}^1, \hat{Y}^2, ..., \hat{Y}^N$, with $(\hat{Y}^i = (\hat{y}^i_1, \hat{y}^i_2, ..., \hat{y}^i_n), i = 1, 2, ..., N$ are generated as follows:

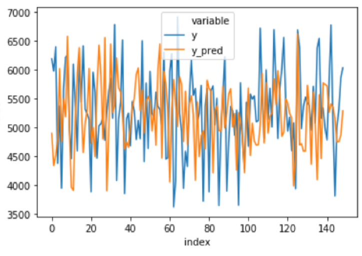
$$\hat{Y}^i \sim \mathcal{N}(\mu, \sigma^2), \mu = ((y_1, y_2, ..., y_N)), \sigma = [\sigma_Y * 0.9, \sigma_Y * 1.1]$$

At the same time, the MSE of for each prediction will also be calculated and saved. After the simulation, the N generated model predictions are saved in an array we obtains a data set with the structure as below. This will be considered as the training set for our preference learning model.

model	value	predicted	
model-1	$(y_1, y_2, ..., y_n)$	$(\hat{y}^1_1, \hat{y}^1_2, ..., \hat{y}^1_n)$	MSE^1
model-2	$(y_1, y_2, ..., y_n)$	$(\hat{y}^2_1, \hat{y}^2_2, ..., \hat{y}^2_n)$	MSE^2
model-3	$(y_1, y_2, ..., y_n)$	$(\hat{y}^3_1, \hat{y}^3_2, ..., \hat{y}^3_n)$	MSE^3
$\vdots$	$\vdots$	$\vdots$	$\vdots$
model-N	$(y_1, y_2, ..., y_n)$	$(\hat{y}^N_1, \hat{y}^N_2, ..., \hat{y}^N_n)$	MSE^N

The illustration describe two generated model with more short term accuracy and with long term accuracy.





3 Preference Learning

Frequently, the predicted preference relation is required to form a total order, in which case we also speak of a ranking problem. Suppose that we have a set of models $\mathcal{S}$ and $(\hat{Y}^1, \hat{Y}^2, ..., \hat{Y}^N) \subseteq \mathcal{S}$ are Prediction vectors. The goal of the ranking is to learn a ranking function that accepts a random subset of $\mathcal{S}$ as input and output a ranking $\mathcal{R}_j$, where $\mathcal{R}_j$ is the total Order of the input models.

$$\mathcal{R}_j := \hat{Y}_{j1} \succ \hat{Y}_{j2} \succ ... \succ \hat{Y}_{jN}$$

The order relation $\succ$ is typically interpreted in terms of preferences, i.e., $\hat{Y} \succ \hat{Y}'$suggests that $\hat{Y}$ is preferred to $\hat{Y}'$.

Here we have an essential question: How could we correctly represent the function in a rational way?

A ranking function is typically implemented by means of a function $U : \hat{Y} \to R_j$, so that $\hat{Y} \succ \hat{Y}'$ if $U(\hat{Y}) > U(\hat{Y}')$ for all $\hat{Y}, \hat{Y}' \in \mathcal{S}$. In other words, a ranking-valued function is implicitly represented by a real-valued function[3]. Obviously, U can be considered as a kind of utility function, and U(x) as a latent utility degree assigned to an item x. Seen from this point of view, the goal in object ranking is to learn a latent utility function on a reference set X . In the following, we shall also refer to U itself as a ranking function.

3.1 Pointwise Ranking

The pointwise Ranking generates a utility function U by fitting a regression function. The Ranking will be estimated with a Ranking function $F : R \to \mathcal{R}_j$. In the experiment, the Ranking is implemented with Support Vector Regression[4] with linear Kernelas Regressor and MSE as regressand to Approximate the set of MSE.

Generally, the User will grade each given model with a score. The score provides a quantitative user preference which is supposed to learn. The data set with attributes Y, Y, MSE is split to train and test set with a partition of 0.8 and 0.2. The MSE is considered as the target value. As the code below we train the regression model and evaluate performance. As for the result, the SVM with a linear kernel reaches a R^2 value of 0.997, which confirms a relatively high accuracy of the model.

```
class SupportVectorRegression(Ranker):
    def __init__(self, epsilon=0.2, kernel='linear', C=1):
        self.model = SVR(kernel=kernel, C=C, epsilon=epsilon)
    def fit(self, X_train, y_train):
        self.model.fit(X=X_train, y=y_train)
    def predict(self,X) -> ndarray:
        return self.model.predict(X=X)
```

3.2 Pairwise Ranking

In a pairwise ranking approach, we don't try to estimate the target value of each sample, instead, given two models we are trying to predict if one sample ranks better than another. In this sense, it resembles a binary classification approach[5] and we can apply algorithms such as Neural Networks[6,7] , Support Vector Machines[8,9], or other classification algorithms to the pair of feature vectors representing two given models.

The User will be given two models at a time and select one of the both according to their preference. In the Experiment, we simulate the user selection process by comparing the MSE of both models. The model with lower MSE will be classified as preferred.

The pairwise Ranker is implemented with a neural network. Because the implementation work is mostly accomplished by another college, further details about the implementation will not be described in this paper.

4 Evaluation Metrics

Because the pointwise ranking takes the numerical value as output while the pairwise ranking outputs the ordinal value, we have to find out a metric to make the results comparable.

In the experiment, we present two metrics for the purpose: Ranking-Mean-Square-Error and Top-K-Accuracy.

The Ranking-Mean-Square-Error intends to measure the dissimilarity between ordinal attributes. It could be described as three steps: numeric-to-ordinal transformation, normalization, and calculation.

- numeric-to-ordinal transformation
  We assume that attribute vector X is consisted of n numeric numbers. Let M represent the number of possible states that an ordinal attribute can have. These ordered states define the ranking $1, ..., M_f$. The value of f for the ith object is x_{if} , and f has M_f ordered states, representing the ranking $1, ..., M_f$. Replace each x_{if} by its corresponding rank,$r_{if} \in \{1, ..., M_f\}$.

- normalization
  Since each ordinal attribute can have a different number of states, it is often necessary to map the range of each attribute onto [0.0, 1.0] so that each attribute has equal weight. We perform such data normalization by replacing the rank r_{if} of the i-th object in the f-th attribute by

$$z_{if} = \frac{(r_{if} - 1)}{(M_{if} - 1)}$$

- calculation
  Let $\hat{z}_{if}$ be the ranking of object i in position f of the prediction vector $\hat{X}$. The Ranking-Mean-Square-Error between X and $\hat{X}$ is calculated as follow

$$MSE_r = \frac{(\hat{z}_{if} - z_{if})^2}{N}$$

Top-K-Accuracy measures how many of the top K models are correctly predicted. In stead calculating the whole dissimilarity, Top-K-Accuracy focus on the performance of ranking the best K Models. It can be calculated as

$$Top - K - Accuracy = \frac{R}{K}$$

R is the number of correctly ranked models in the first k best models.

5 Ranker Comparison

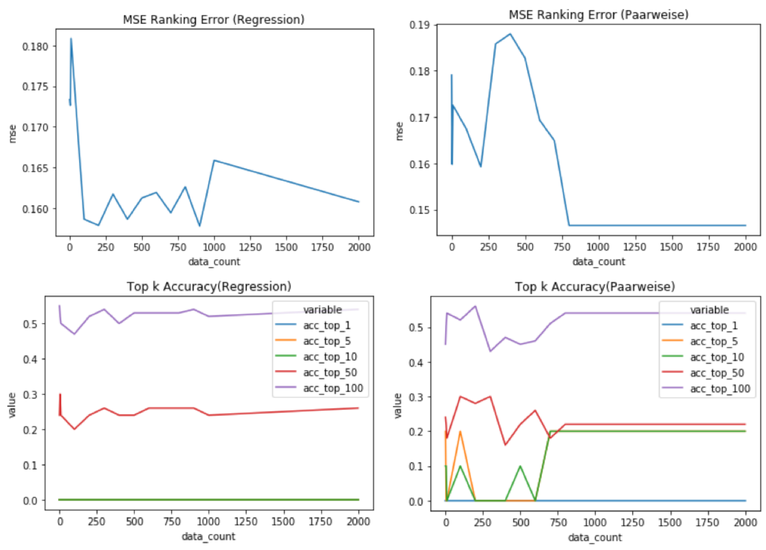
With the help of metrics, the Comparison of both ranking methods is composed of the following aspects:

- Efficiency: How many training Objects are necessary to train a model with the best quality that could be achieved.
- Accuracy: How accurate is the ranker.
- User-friendliness: Which Ranking Algorithm is considered as more user friendly during the data labeling.

We have Training Set T, let $|T|$ be the number of Objects of T. To compare the Efficiency and Accuracy under the condition of the different number of Training Data, the evaluation process is executed iteratively for $T \in [1, 5, 10, 25, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000]$. For each size of T, the Ranker will be trained and evaluated with Ranking-Mean-Square-Error and Top-K-Accuracy.

6 Result

The Ranking-Mean-Square-Error and Top-K-Accuracy for different numbers of training data are illusrated below.



The following conclusions can be drawn from these graphs:

- The pointwise ranking reaches the best accuracy with about 250 training objects, while the pairwise ranking needs 750. It presumes a better efficiency of pointwise ranking.
- pairwise ranking has a lower minimal value in terms of Ranking-MSE and f better performance at top 10 and top 100 Accuracy. In regard of precision, the pairwise ranking has an advantage.

- User-friendliness: There is wide agreement that the pairwise ranking is more unser friendly. Because for the user, it is more intuitive to select one from the given two models, than give a specific model a score.

7 Conclusion and Future Work

In summary, our experiment conveys a process of generating the model with different structures, simulating the user preference, and learning the preference with pointwise and pairwise ranking. It's worth noting that the analysis above about both ranking methods is highly specific to our experiment. Pointwise ranking learns the preference faster is probably because the simulated Preference - MSE, is quantifiable and can be easily approximated by regression. A Generalization test for the learning ability of the Ranker towards more complex preference should be explored in further experiments.

Besides,there are many topics in the filed of personalization, which rouses my interests during the seminar. Such as how could the learned preference benefits the prediction model training so that the quality of prediction model for the individual user improves iteratively, and the implementation of other learning to rank algorithms like PolyRank[10] and FastAP[11]. They will be considered as the future work.

Acknowledgments

I would like to thank Cedric Kulbach and Lars Böcking, who never wavered in their support during the seminar. Thank you for your patient guidance and all of the valuable advice.

References

1 Paul F Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, Dario Amodei *Deep Reinforcement Learning from Human Preferences*

2 Alyssa Glass *Explaining Preference Learning*, CS229 Final Project Stanford University, Stanford

3 Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, Klaus Brinker *Label ranking by learning pairwise preferences*, Artificial Intelligence 172(2008) pp.1897-1916

4 Steve R. Gunn *Support Vector Machines for Classification and Regression*, Technical Report,10.05.1998

5 Johannes Fürnkranz, Round robin *Round robin classification*, Journal of Machine Learning Research 2 (2002) 721–747.

6 David Price, Stefan Knerr, Léon Personnaz, Gérard Dreyfus, *Pairwise neural network classifiers with probabilistic outputs*, in: G. Tesauro, D. Touretzky, T. Leen (Eds.), Advances in Neural Information Processing Systems 7 (NIPS-94), MIT Press, 1995, pp. 1109–1116.

7 Stefan Knerr, Léon Personnaz, Gérard Dreyfus, *Handwritten digit recognition by neural networks with single-layer training*, IEEE Transactions on Neural Networks 3 (6) (1992) 962–968.

8 Ulrich H.-G. Kreßel, *Pairwise classification and support vector machines*,in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, 1999, pp. 255–268, Chapter 15.

9 Chih-Wei Hsu, Chih-Jen Lin, *A comparison of methods for multi-class support vector machines*, IEEE Transactions on Neural Networks 13 (2) (March 2002) 415–425.

10 Davidov, Ori; Ailon, Nir; Oliveira, Ivo F. D. (2018), *A New and Flexible Approach to the Analysis of Paired Comparison Data*, ournal of Machine Learning Research. 19 (60): 1–29.

11 Fatih Cakir, Kun He, Xide Xia, Brian Kulis, Stan Sclaroff, *Deep Metric Learning to Rank*, In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

8 Erklärung

Ich versichere hiermit wahrheitsgemäß, die Arbeit selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.