Multi-Task Multi-Kernel Online Learning Algorithm for Data Streams (Postprint)

Authors: Pei Le, Liu Qun

Date: 2018-05-18T00:00:00+00:00

Abstract

Multi-task multi-kernel learning has gradually emerged as a focal point in online learning algorithm research. Regarding data stream processing, existing online learning algorithms exhibit certain deficiencies in accuracy; consequently, we propose a novel multi-task multi-kernel online learning model to enhance the accuracy of data stream prediction. By preserving the foundation of multi-task multi-kernel learning while extending it to the online learning paradigm, we derive a new online learning algorithm; simultaneously, we maintain a fixed-size data window for input data, thereby trading modest space overhead for data integrity. The experimental section conducts a relatively detailed analysis of kernel function selection and training sample set size. Through evaluation on both UCI datasets and real-world airport passenger flow data, the approach effectively guarantees the accuracy and real-time performance of stream data processing, thereby demonstrating certain practical application value.

Full Text

Preamble

Online Learning Algorithm Based on Multi-Task and Multi-Kernel for Stream Data

Pei Le, Liu Qun
(Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts & Telecommunications, Chongqing 400065, China)

Abstract: Multi-task and multi-kernel learning has gradually become a research focus in online learning algorithms. For data stream prediction, existing online learning algorithms exhibit certain shortcomings in accuracy. Therefore, this paper proposes a novel multi-task multi-kernel online learning model to improve the accuracy of data stream prediction. Building upon multi-task multi-kernel learning, we extend the framework to online learning, thereby obtaining

a new online learning algorithm. Simultaneously, we maintain a fixed-size data window for input data, trading a modest amount of space for data integrity. The experimental section provides a detailed analysis of kernel function selection and training sample set size. Through experiments on UCI datasets and real-world airport passenger flow data, the proposed algorithm effectively ensures both accuracy and real-time performance in stream data processing, demonstrating practical application value.

Key Words: multi-task and multi-kernel learning; online learning; streaming data; SVM

0 Introduction

In many big data application domains such as financial time series prediction, natural language processing, and network traffic control, data is generated in real-time and increases dynamically. As long as the data source remains active, the data can be considered to increase without bound. Due to the sheer volume of data, complete storage is generally infeasible, necessitating online learning models for real-time processing. In research addressing stream data processing requirements, online machine learning algorithms adopt a direct processing mode for data streams, handling one random stream data instance per iteration with simple iterative updates of learning variables, thereby achieving a balance between real-time performance and accuracy. This approach represents a promising solution to this problem.

Classical online learning algorithms include the Passive-Aggressive (PA) method, the perceptron algorithm, and the Confidence-Weighted (CW) approach. However, whether PA' s hyperplane model or its confidence-weighted derivatives, these methods assume data is nearly linearly separable. This assumption does not always hold, as data is often non-linearly distributed in the original input space, and subjective biases in data collection make it difficult for data to be perfect or conform to a preset linear distribution. Therefore, further research into non-linear models is essential. To address the problem of non-linear separability in online learning, online multi-task learning models represent a primary research direction.

Online multi-task learning models aim to simultaneously learn multiple related tasks using shared information, allowing each task to benefit from learning across all tasks. For example, references employ mixed-norm regularization to learn weights for each task, which considers inter-task correlations while achieving sparsity both within individual tasks and across different tasks. However, these methods struggle to converge quickly and train corresponding models when the number of training samples is small. To address convergence issues, Yang et al. propose a simple information-sharing strategy where all tasks share a common kernel function. This approach maps feature data through kernel functions into another feature space, avoiding complex computations on the data itself,

while kernel combination methods can better extract data features. However, the primary challenge lies in kernel function selection, as different kernel choices significantly impact results.

Offline multi-kernel learning models seek an appropriate kernel function through linear or non-linear combinations of several pre-selected candidate kernels, thereby avoiding the need to validate different kernel selections. Based on this analysis, this paper investigates multi-task multi-kernel online learning algorithms in stream data environments. We apply multi-kernel methods within an online multi-task learning framework, iteratively determining a combination kernel function most suitable for current data generation patterns through online learning. This avoids direct analysis of data samples for kernel learner updates, simplifying computation while leveraging the high convergence rate of kernel methods and the advantages of multi-task joint learning, thereby facilitating more convenient stream data processing. The contributions of this paper are twofold: First, we extend the mature multi-task multi-kernel learning framework to online learning, forming a novel algorithm that better handles stream data; second, we maintain a data window for newly generated data to ensure data integrity when immediate processing is not possible.

---

1 Multi-Task Multi-Kernel Learning Algorithm

Assume there are T tasks whose data originates from the same space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}$. These T tasks each possess different data points $\{(x_{t1}, y_{t1}), \dots, (x_{tn_T}, y_{tn_T})\}_{t=1}^{T}$, where n_t is the total number of data points for task t. For task t, its decision function is $f_t(x) = w_t \cdot \Phi(x) + b_t$, where b_t is the bias term, $\Phi(x)$ is the mapped feature vector, and coefficient w_t is the set of all kernel coefficients θ_{tm} corresponding to task t, i.e., $w_t = \{w_{t1}, w_{t2}, \dots, w_{tM}\}$. Here, θ_{tm} is the coefficient of the kernel function for each task, where k_m is a kernel function defined in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_m$, with $m = 1, 2, \dots, M$ being pre-selected kernel functions.

The objective of multi-task multi-kernel learning is to learn a decision function for each task under constraints that minimize empirical risk and weight regularization terms, where all tasks share a common sparse kernel representation. This requires establishing a learning algorithm that can create a function for each task. To achieve this goal, we transform the problem into a regularized optimization problem:

$$\begin{aligned} \min_{w_1,\dots,w_T} \quad & \sum_{t=1}^{T} \sum_{i=1}^{n_t} L(f_t(x_{ti}), y_{ti}) + C \cdot \Omega(w_1, \dots, w_T) \\ \text{s.t.} \quad & \text{constraints on kernel coefficients} \end{aligned}$$

where $L(f_t(x_{ti}), y_{ti})$ is the loss function describing the fit between data and

model (i.e., training error), C is a parameter controlling the trade-off between model complexity and loss penalty, and $\Omega(w_1, \ldots, w_T)$ represents relationships between different tasks as a regularization term on all decision coefficients w_t. Without the Ω constraint, the problem decomposes into T independent learning problems.

Multi-task multi-kernel algorithms are essentially multi-objective optimization problems. Drawing on multi-objective optimization solution methods, we obtain Pareto optimal solutions for the algorithm. We employ scalarization methods to solve this multi-task multi-kernel problem, i.e., finding optimal solutions for corresponding problems through different optimal combinations of objective functions. Therefore, problem (2) can be viewed as a Pareto Front problem in multi-objective optimization with a specific solution. Using parameter λ to scalarize problem (2), the optimization problem can be rewritten as:

$$\min_{w_1,\ldots,w_T} \sum_{t=1}^{T} \lambda_t \left(\sum_{i=1}^{n_t} L(f_t(x_{ti}), y_{ti}) + C \cdot \Omega(w_1, \ldots, w_T) \right)$$

where $\lambda_t \geq 0$ and $\sum_{t=1}^{T} \lambda_t = 1$. Thus, problem (2) can be solved by optimizing problem (3). This paper makes the following conventions: λ_t ranges in $[0, 1]$, where λ is predetermined.

The basic model of regularization term Ω is $\Omega = \frac{1}{2}\|w\|^2$, where w represents model weight coefficients. The information-sharing strategy used in this paper learns both inherent features of each task and common features across all tasks. Therefore, model weights include not only task-specific weights w_t but also kernel coefficients θ containing common features across all tasks. When discussing the weight regularization term Ω, both weight coefficients must be considered. The purpose of Ω is to maximize differences between tasks, enabling better prediction performance. Thus, it is necessary to consider inherent features of each task on top of common features. Based on literature, Ω is expressed as:

$$\Omega = \sum_{t=1}^{T} \|w_t - \theta\|^2 + \|\theta\|^2$$

2.2 Input Data Window Concept

Most existing stream data processing algorithms process newly generated data individually, requiring strong real-time processing capabilities. If the algorithm' s data stream processing capacity cannot meet real-time requirements, some data will be lost due to untimely processing. To ensure data integrity, this paper uses a certain amount of space to trade for data completeness. Specifically, the algorithm maintains a fixed-size input data window, where each window can store one data sample. When new data continuously arrives, samples that

cannot be processed immediately are saved in this window to prevent loss due to delayed processing, thereby somewhat reducing real-time performance requirements. For the basic model used in this paper—the SVM algorithm—the number of input samples (one or multiple) does not affect the overall running speed, making it necessary to maintain an input data window. Experiments in this paper demonstrate that optimal performance is achieved when the window size is 3.

2.3 Model Update Strategy

For model updating, given the infinite nature of stream data, each prediction error adds a new support vector, suggesting that the number of support vectors is unbounded. Directly computing the kernel matrix K for the entire dataset is computationally unrealistic. Therefore, this paper addresses this issue by setting a maximum limit on support vectors, achieved at the cost of some accuracy. Subsequent experiments discuss in detail how this maximum number setting affects prediction results.

For SVM, the algorithm needs to maintain a fixed-size training sample set for kernel matrix computation. With the introduction of online learning techniques, the data in this sample set consists of support vectors that need to be preserved. Therefore, a strategy for updating data samples in the training set is essential. Drawing on memory page scheduling algorithms from operating systems, this paper adopts a First-In-First-Out (FIFO) strategy for updating data samples. For data streams, data generation patterns may change over time, making the FIFO strategy—replacing the sample with the longest existence time—reasonable.

According to the FIFO strategy, the training sample set is updated as follows: each new sample is placed in the last row and last column, then the first row and first column are removed to complete the update. This strategy of limiting the working set size has certain limitations but represents a compromise under limited computational and storage resources.

Before updating the training model, this paper pre-updates the common kernel coefficients, penalizing each prediction task to reduce model training iteration coefficients and thereby decrease training time. During the $(j+1)$-th iteration, the update formula for θ is:

$$\theta_{j+1} = \beta \cdot \theta_j$$

where β is a random number between 0 and 1. Algorithm 1 describes the overall process of this model in detail.

Algorithm 1: Multi-Task Multi-Kernel Online Learning Algorithm

Input: Training sample set $D_j^t = \{x_1, x_2, \dots, x_{N_j^t}\}$; kernel function set $K = \{k_1, k_2, \dots, k_m\}$; sequentially input one sample x_0 or a group of samples D_0 for

each task.

Output: Prediction results and updated model.

1. Compute the corresponding kernel matrix for the task based on the new sample;
2. Obtain the existing training model $Model_j^t$ for the task;
3. Predict result y_{pred};
4. **if** prediction == true value
   - Accept next data;
5. **else**
   - Update the last row and column of the training sample set with D_{j+1}^t;
   - Update kernel combination coefficients $\theta_{j+1} = \beta \cdot \theta_j$;
   - Train a new model $Model_{j+1}^t$ for the task using SVM;
6. **end if**

Algorithm 1 provides a comprehensive description of the proposed online learning algorithm. To handle continuously arriving stream data, it combines the multi-task multi-kernel learning framework with online techniques to generate a new algorithm. Based on processing one or multiple newly arrived data instances, the data sample set is continuously updated to better obtain models for each task and improve prediction accuracy.

2.4 Time-Space Cost Analysis

Large-scale data streams feature massive data volume and extremely fast arrival rates, requiring stream mining algorithms to process data in real-time within limited memory space. This demands low time and space complexity for algorithms targeting big data streams. Since this paper maintains an input data window, real-time requirements are somewhat reduced. Below is a simple analysis of the algorithm' s time-space complexity.

Assume there are Q tasks, each with k d-dimensional training samples, totaling N training samples. For the proposed online learning algorithm MTMKOL, the time complexity per iteration is $O(Q \cdot m \cdot k)$, and $O(Q \cdot m \cdot N)$ when samples appear sequentially. For space complexity, the algorithm must continuously maintain a training sample set of size $O(Q \cdot d \cdot k)$, while also maintaining an input data window of size 3, requiring $O(3 \cdot Q \cdot d)$ space. Therefore, the total space complexity is $O(Q \cdot d \cdot k + 3 \cdot Q \cdot d) = O(Q \cdot d \cdot (k + 3))$.

In contrast, algorithm ADA-MTL has time complexity $O(Q \cdot d \cdot N)$, algorithm BMKOL has time complexity $O(Q \cdot k \cdot N)$, and batch processing algorithm MTMKL has time complexity $O(Q \cdot m \cdot N^2)$. Based on this analysis, the proposed algorithm and the other two online learning algorithms all have time complexity $O(N)$, satisfying the time complexity requirements for online learning algorithms.

2.5 Kernel Function Selection

For multi-kernel learning problems, there is currently no unified theoretical standard for selecting the number and types of kernels, making "kernel function selection" the greatest variable in support vector machines. This paper provides a brief analysis of commonly used kernel functions and selects appropriate kernels through experiments.

Common kernel functions include polynomial kernels, linear kernels, and Gaussian kernels. This paper selects 1 polynomial kernel, 1 linear kernel, and multiple Gaussian kernels, where the number of Gaussian kernels is determined experimentally. Using the robot dataset with an input data window size of 3 and a training sample set size equal to 10% of the total samples, Table 1 shows the classification accuracy and runtime corresponding to different numbers of Gaussian kernels. Classification accuracy is the ratio of correctly predicted samples to total samples, and runtime is the total time to complete the entire dataset experiment. All values are averages over 10 runs.

Table 1: Classification Accuracy and Runtime for Different Numbers of Gaussian Kernels

Polynomial	Linear Kernel	Gaussian Kernels	Accuracy	Runtime

The results demonstrate that different numbers of Gaussian kernels yield varying classification prediction accuracy and runtime. The optimal performance occurs with 5 Gaussian kernels, achieving the shortest runtime and classification accuracy of 0.98. Therefore, the optimal kernel function combination selected in this paper consists of 1 polynomial kernel, 1 linear kernel, and 5 Gaussian kernels as the basic kernel function set.

3.1 UCI Data Experiment Analysis

To evaluate the classification performance of the proposed online learning algorithm, this paper selects two multi-task datasets from the UCI repository: robot (6 tasks, 500 samples per task, 4 attributes per task) and letter (handwritten words, 8 tasks, 500 samples per task, 16 attributes per task). This section uses classification accuracy and runtime to evaluate algorithm performance.

3.1.1 Training Sample Set Size Selection

The algorithm requires maintaining a storage space of size N for the training sample set, where N directly determines memory usage. To determine the

optimal N, experiments are conducted on the robot dataset with an input data window size of 3. Figure 1 [Figure 1: see original paper] compares classification accuracy and runtime for different training sample set sizes, with all values averaged over 10 random runs.

As shown in Figure 1, classification prediction accuracy generally improves as training sample size increases, while runtime generally decreases. The best performance occurs with 75 samples, achieving optimal accuracy and runtime. When sample size is between 50-75, both accuracy and runtime are favorable. Therefore, all experiments in this paper use training sample set sizes between 50-75. The classification accuracy reaches 0.99 because SVM inherently performs exceptionally well on classification problems, and continuous model updates and corrections substantially reduce error rates.

3.1.2 Input Window Size Selection

The algorithm requires maintaining a storage space of size n as an input data window, where n affects memory usage. To determine the optimal n, experiments are conducted on the robot dataset with 75 training samples. Figure 2 [Figure 2: see original paper] compares accuracy and runtime for different window sizes, with all values averaged over 10 random runs.

As shown in Figure 2, classification prediction accuracy remains stable initially then drops sharply as window size increases, while runtime first decreases then suddenly increases. The optimal performance occurs with a window size of 3, achieving the best accuracy and runtime. Therefore, all experiments use an input window size of 3. The window size of 3 represents an inflection point: when the window is too large, it tends to include incorrectly predicted samples, requiring the program to search through all samples in the current window to find errors before updating the model. This process incurs additional time overhead and reduces accuracy.

3.1.3 Algorithm Scalability Analysis

To evaluate the scalability of the proposed stream data online algorithm, experiments are conducted on two multi-task datasets (robot and letter) with an input window size of 3. The experimental setup uses training samples comprising different proportions of total samples: {5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%}, with remaining samples used for evaluation. Figure 3 [Figure 3: see original paper] compares prediction results of the proposed algorithm against three other algorithms under various settings.

Figure 3(a) uses the robot dataset, while **Figure 3(b)** uses the letter dataset. The following conclusions can be drawn:

a) The proposed algorithm MTMKOL consistently achieves higher classification accuracy than the other three algorithms, with very high accuracy values. This is because SVM is inherently excellent for classification,

and maintaining an input set reduces real-time requirements, allowing the algorithm to focus primarily on accuracy. Moreover, MTMKOL shows minimal accuracy variation across different training sample size settings, demonstrating good scalability and low requirements for training set sample numbers, thus satisfying large-scale stream data demands.

b) Algorithm BMKOL exhibits significantly lower classification accuracy than the other three algorithms, likely due to its limit on support vector count and its trade-off of some accuracy for real-time performance.

c) Algorithm ADA-MTL shows unstable classification performance because it does not maintain historical data, directly computing weights that affect results.

d) Algorithm Conic MTL is a batch processing algorithm that learns a model from the given training samples for new data prediction. Its results become more accurate as training sample size increases.

3.2 Alibaba Tianchi Airport Passenger Flow Stream Data Experiment Analysis

To evaluate the regression performance of the proposed online learning algorithm in practical applications, we use airport passenger flow spatiotemporal distribution prediction data to measure model and algorithm performance through prediction error (RMSE) and data throughput, thereby verifying MTMKOL' s characteristics and advantages in airport passenger flow prediction.

The dataset comes from the Tianchi competition' s airport passenger flow spatiotemporal distribution prediction preliminary round, providing massive airport WiFi data and security check-in boarding data for Baiyun Airport terminal passenger flow analysis and prediction. Original time data is precise to the second; we simplify it by using 10-minute intervals as time slices and re-aggregating the data. Treating passenger flow prediction for each WiFi point as a task yields 749 tasks. For each task, the airport' s daily schedule is relatively stable, user walking patterns in the airport are basically stable, and time series have a certain degree of continuity. Situations at a given time point continue to some extent from previous hours and from the same time two days prior. Therefore, we use passenger counts from different time periods as task data features, with specific information shown in Table 3 .

Table 3: Data Feature Information

Feature	Description
Previous 10-minute count	Passenger count in previous 10 minutes
Previous 20-minute count	Passenger count in previous 20 minutes

Feature	Description
Same time 2 days prior	Passenger count at same time two days earlier

3.2.1 Regression Performance Analysis

To evaluate the regression performance of the proposed stream data online algorithm, we select the first 100 task points from the dataset for experiments. The training set uses 12 hours of data (72 samples), and the test set uses the remaining 1510 samples. Figure 4 [Figure 4: see original paper] compares the prediction error RMSE (root mean square error) of different algorithms on this dataset, where RMSE is the root mean square of errors between true and predicted values.

As shown in Figure 4, different algorithms yield different RMSE values. The proposed algorithm MTMKOL achieves the smallest RMSE, primarily because SVM can train good models with minimal data, making prediction errors very small. The batch algorithm Conic MTL has the largest error because it trains the model only once for testing, and this model does not well match the data characteristics, leading to larger errors. The other two algorithms also have relatively larger errors because the given training sample set is small, and the model cannot initially match the data well. Only through continuous model updates do error values gradually decrease.

3.2.2 Regression Analysis with Different Task Numbers

To evaluate the regression performance of the proposed stream data online algorithm with different numbers of tasks, experiments are conducted on all 749 task points in the dataset. The experimental setup uses task numbers {2, 100, 200, 300, 400, 500, 600, 749} to verify prediction error and data throughput under different task quantities. The training set uses 12 hours of data (72 samples), and the test set uses the remaining 1510 samples. Figure 5 [Figure 5: see original paper] compares prediction error RMSE and data throughput (the amount of data processed per second) for different task numbers.

The following phenomena can be observed from Figure 5:

- Multi-task learning approaches perform significantly better in regression prediction than individual task learning, and both RMSE and throughput improve as task numbers increase.
- More tasks yield smaller prediction errors, with predicted values closer to true values.
- As task numbers increase, the amount of data processed per second continuously increases, with algorithm throughput reaching up to 320 samples per second, fully satisfying real-time requirements.

4 Conclusion

This paper proposes a novel multi-task multi-kernel online algorithm for stream data. First, optimal kernel function combinations are selected through experiments. Since no unified theoretical standard currently exists for kernel selection, this paper makes experimental choices based on existing experience. Second, input sample set size is selected through experiments to achieve good results in both prediction accuracy and time consumption. Comparisons with existing online and batch algorithms demonstrate superior performance across various experiments. Third, while existing online learning algorithms process newly arrived data directly, requiring high real-time processing capabilities, the proposed algorithm maintains a data input window for new data, effectively preventing data loss and reducing real-time requirements. Experimental results demonstrate the significance of this setting. Finally, analysis and prediction of real airport passenger flow data achieve expected results.

This algorithm focuses on stream data processing, with a relatively simple kernel coefficient update algorithm. Future work will carefully investigate this update algorithm to make model updates simpler and faster.

References

[1] Li Zhijie, Li Yuanxiang, Wang Feng, et al. Survey of online learning algorithms for big data analysis [J]. Journal of Computer Research and Development, 2015, 52(8): 1707-1721.

[2] Pan Zhisong, Tang Siqi, Qiu Junyang, et al. Survey of online learning algorithms [J]. Journal of Data Acquisition and Processing, 2016, 31(6): 1067-1082.

[3] Wang Z, Vucetic S. Online passive-aggressive algorithms on a budget [J]. Journal of Machine Learning Research, 2010, 9(9): 908-915.

[4] Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain [J]. Psychological Review, 1958, 65(6): 386.

[5] Crammer K, Dredze M, Pereira F. Confidence-weighted linear classification for text categorization [J]. Journal of Machine Learning Research, 2012, 13(1): 1891-1926.

[6] Rakotomamonjya, Flamary R, Gasso G, et al. lp-lq penalty for sparse linear and sparse multiple kernel multitask learning [J]. IEEE Trans on Neural Networks, 2011, 22(8): 1307-1320.

[7] Li Zhijie, Li Yuanxiang, Wang Feng, et al. Multi-task accelerated online learning algorithm for big data streams [J]. Journal of Computer Research and Development, 2015, 52(11): 2545-2554.

[8] Yang H, Lyu M R, King I. Efficient online learning for multitask feature selection [J]. ACM Trans on Knowledge Discovery from Data, 2013, 7(2): 1-24.

[9] Li C, Georgiopoulos M, Anagnostopoulos G C. Pareto-path multitask multiple kernel learning [J]. IEEE Trans on Neural Networks & Learning Systems, 2015, 26(1): 51-61.

[10] Zhou Zhihua, Wang Jue. Machine Learning and Its Applications [M]. Beijing: Tsinghua University Press, 2007.

[11] Zou Hengming. The Mind of Computers: Philosophical Principles of Operating Systems [M]. Beijing: Mechanical Industry Press, 2012: 100-102.

[12] Zhang Gang, Xie Xiaoshan, Huang Ying, et al. Semi-supervised online multi-kernel learning algorithm for big data streams [J]. CAAI Transactions on Intelligent Systems, 2014, 9(3): 355-363.

[13] Jian L, Shen S, Li J, et al. Budget online learning algorithm for least squares SVM [J]. IEEE Trans on Neural Networks & Learning Systems, 2016, 28(9): 2076-2087.

[14] Li C, Georgiopoulos M, Anagnostopoulos G C. Conic multi-task classification [C]// Machine Learning and Knowledge Discovery in Databases. 2014: 193-208.

[15] UCI Machine Learning Repository [EB/OL] http://archive.ics.uci.edu/ml/DOI.

[16] Airport Passenger Flow Spatiotemporal Distribution Prediction [EB/OL]. https://tianchi.aliyun.com/competition/introduction.htm?spm=5176.100066.333.4.6YizCQ&raceId=231588D

Note: Figure translations are in progress. See original paper for figures.