Boosting Based Multiple Kernel Learning and Transfer Regression for Electricity Load Forecasting

Di Wu[1], Boyu Wang[2], Doina Precup[1], and Benoit Boulet[1]

[1] McGill University
H3A 0G4, Montreal, Quebec, Canada
di.wu5@mail.mcgill.ca, dprecup@cs.mcgill.ca, benoit.boulet@mcgill.ca
[2]Princeton University,
Princeton, NJ 08544, USA
boyuw@princeton.edu

Abstract. Accurate electricity load forecasting is of crucial importance for power system operation and smart grid energy management. Different factors, such as weather conditions, lagged values, and day types may affect electricity load consumption. We propose to use multiple kernel learning (MKL) for electricity load forecasting, as it provides more flexibilities than traditional kernel methods. Computation time is an important issue for short-term load forecasting, especially for energy scheduling demand. However, conventional MKL methods usually lead to complicated optimization problems. Another practical aspect of this application is that there may be very few data available to train a reliable forecasting model for a new building, while at the same time we may have prior knowledge learned from other buildings. In this paper, we propose a boosting based framework for MKL regression to deal with the aforementioned issues for short-term load forecasting. In particular, we first adopt boosting to learn an ensemble of multiple kernel regressors, and then extend this framework to the context of transfer learning. Experimental results on residential data sets show the effectiveness of the proposed algorithms.

Keywords: Electricity load forecasting, boosting, multiple kernel learning, transfer learning

1 Introduction

Electricity load forecasting is very important for the economic operation and security of a power system. The accuracy of electricity load forecasting directly influences the control and planning of power system operation. It is estimated that a 1% increase of forecasting error would bring in a 10 million pounds increase in operating cost per year (in 1984) for the UK power system [4]. Experts believe that this effect could become even stronger, due to the emergence of highly uncertain energy sources, such as solar and wind energy generation. Depending on the lead time horizon, electricity load forecasting ranges from short-term forecasting (minutes or hours ahead) to long-term forecasting (years ahead) [13]. With increasingly competitive markets and demand response energy management [15], short-term load forecasting is becoming more and more important [25]. In this paper, therefore, we will focus on tackling this problem.

Electricity load forecasting is a very difficult task since the load is influenced by many uncertain factors. Various methods have been proposed for electricity load forecasting including statistical methods, time series analysis, and machine learning algorithms [21]. Some recent work uses multiple kernels to build prediction models for electricity load forecasting. For example, in [1], Gaussian kernels with different parameters are applied to learn peak power consumption. In [8], different types of kernels are used for different features and a multi-task learning algorithm is proposed and applied on low level load consumption data to improve the aggregated load forecasting accuracy. However, all of the existing methods rely on a fixed set of coefficients for the kernels (i.e., simply set to 1), implicitly assuming that all the kernels are equally important for forecasting, which is suboptimal in real world applications.

Multiple kernel learning (MKL) [2], which learns both the kernels and their combination weights for different kernels, could be tailored to this problem. Through MKL, different kernels could have different weights according to their influence on the outputs. However, learning with multiple kernels usually involves a complicated convex optimization problem, which limits their application on large scale problems. Although some progresses have been made in improving the efficiency of the learning algorithms, most of them only focus on classification tasks [26, 23]. On the other hand, electricity load forecasting is a regression problem and the computation time is an important issue.

Another practical issue for load forecasting is the lack of data to build a reliable forecasting model. For example, consider the case of a set of newly built houses (target domain) for which we want to predict the load consumption. We may not have enough data to build a prediction model for these new houses, while we have a large amount of data or knowledge from other houses (source domain). The challenge here is to perform transfer learning [18], which relies on the assumption is that there are some common structures or factors that can be shared across the domains. The objective of transfer learning for load forecasting is to improve the forecasting performance by discovering shared knowledge and leveraging it for electricity load prediction for target buildings.

In this paper, we address both challenges within a novel boosting-based MKL framework. In particular, we first propose the *boosting based multiple kernel regression* (BMKR) algorithm to improve the computational efficiency of MKL. Furthermore, we extend BMKR to the context of transfer learning, and propose two variants of BMKR: *kernel-level boosting based transfer multiple kernel regression* (K-BTMKR) and *model-level gradient boosting based transfer multiple kernel regression* (M-BTMKR). Our contribution, from an algorithmic perspective, is two-fold: We propose a boosting based learning framework 1. to learn regression models with multiple kernels efficiently, and 2. to leverage the MKL models learned from other domains. On the application side, this work introduces the use of transfer learning for the load forecasting problem, which opens up potential future work avenues.

2 Background

2.1 Multiple Kernel Regression

Let $\mathcal{S} = \{(x_n, y_n), n = 1, \ldots, N\} \in \mathbb{R}^d \times \mathbb{R}$ be the data set with N samples, $\mathcal{K} = \{k_m : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, m = 1, \ldots, M\}$ be M kernel functions. The objective of MKL

is to learn a prediction model, which is a linear combination of M kernels, by solving the following optimization problem [11]:

$$\min_{\eta \in \Delta} \min_{F \in \mathcal{H}_K} \frac{1}{2}||F||_K^2 + C \sum_{n=1}^{N} \ell(F(x_n), y_n), \tag{1}$$

where $\Delta = \{\eta \in \mathbb{R}_+ | \sum_{m=1}^{M} \eta_m = 1\}$ is a set of weights, $\mathcal{H}_K$ is the reproducing kernel Hilbert space (RKHS) induced by the kernel $K(x, x_n) = \sum_{m=1}^{M} \eta_m k_m(x, x_n)$ and $\ell(F(x), y)$ is a loss function. In this paper we use the squared loss $\ell(F(x), y) = \frac{1}{2}(F(x) - y)^2$ for the regression problem. The solution of Eq. 1 is of the form[1]

$$F(x) = \sum_{n=1}^{N} \alpha_n K(x, x_n), \tag{2}$$

where the coefficients $\{\alpha_n\}$ and $\{\eta_m\}$ are learned from samples.

Compared with single kernel approaches, MKL algorithms can provide better learning capability and alleviate the burden of designing specific kernels to handle diverse multivariate data.

2.2 Gradient Boosting and ϵ-Boosting

Gradient boosting [10, 16] is an ensemble learning framework which combines multiple hypotheses by performing gradient descent in function space. More specifically, the model learned by gradient boosting can be expressed as:

$$F(x) = \sum_{t=1}^{T} \rho^t f^t(x), \tag{3}$$

where T is the number of total boosting iterations, and the t-th base learner f^t is selected such that the distance between f^t and the negative gradient of the loss function at $F = F^{t-1}$ is minimized:

$$f^t = \arg\min_{f} \sum_{n=1}^{N} \left(f(x_n) - r_n^t\right)^2, \tag{4}$$

where $r_n^t = -\left[\frac{\partial \ell(F(x_n), y_n)}{\partial F}\right]_{F=F^{t-1}}$, and ρ^t is the step size which can either be fixed or chosen by line search. Plugging in the squared loss we have $r_n^t = y_n - F^{t-1}(x_n)$. In other words, gradient boosting with squared loss essentially fits the residual at each iteration.

Let $\mathcal{F} = \{f_1, \ldots, f_J\}$ be a set of candidate functions, where $J = |\mathcal{F}|$ is the size of the function space, and $f : \mathbb{R}^d \to \mathbb{R}^J$, $f(x) = [f_1(x), \ldots, f_J(x)]^\top$ be the mapping defined by $\mathcal{F}$. Gradient boosting with squared loss usually proceeds in a greedy way:

[1] We ignore the bias term for simplicity of analysis, but in practice, the regression function can accomodate both the kernel functions and the bias term.

the step size is simply set $\rho^t = 1$ for all iterations. On the other hand, if the step size ρ^t is set to some small constant $\epsilon > 0$, it can be shown that under the monotonicity condition, this example of gradient boosting algorithm, referred to as ϵ-boosting in [20], essentially solves an ℓ_1-regularized learning problem [12]:

$$\min_{||\beta||_1 \leq \mu} \sum_{n=1}^{N} \frac{1}{N} \ell\left(\beta^\top f(x_n), y_n\right), \tag{5}$$

where $\beta \in \mathbb{R}^J$ is the coefficient vector, and μ is the regularization parameter, such that $\epsilon T \leq \mu$. In other words, ϵ-boosting implicitly controls the regularization via the number of iterations T rather than μ.

2.3 Transfer Learning from Multiple Sources

Let $\mathcal{S}_T = \{(x_n, y_n), n = 1, \ldots, N\}$ be the data set from the target domain, and $\{\mathcal{S}_1, \ldots, \mathcal{S}_S\}$ be the data sets from S source domains, where $\mathcal{S}_s = \{(x_n^s, y_n^s), n = 1, \ldots, N_s\}$ are the samples of the s-th source. Let $\{F_1, \ldots, F_S\}$ be the prediction models learned from S source domains. In this work, the s-th model F_s is trained by some MKL algorithm (e.g., BMKR), and is of the form:

$$F_s = \sum_{m=1}^{M} \eta_m^s h_m^s(x) = \sum_{m=1}^{M} \eta_m^s \sum_{n=1}^{N_s} \alpha_n^s k_m(x, x_n^s). \tag{6}$$

The objective of transfer learning is to build a model F that has a good generalization ability in the target domain using the data set $\mathcal{S}_T$ (which is typically small) and knowledge learned from sources $\{\mathcal{S}_1, \ldots, \mathcal{S}_S\}$. In this work, we assume that such knowledge has been embedded into $\{F_1, \ldots, F_S\}$, and therefore the problem becomes to explore the model structures that can be transferred to the target domain from various source domains. This type of learning approach is also referred to as *parameter transfer* [18].

3 Methods

3.1 Boosting based Multiple Kernel Learning Regression

The idea of BMKR is to learn an ensemble model with multiple kernel regressors using the gradient boosting framework. The starting point of our method is similar to multiple kernel boosting (MKBoost) [23], which adapts AdaBoost [9] for multiple kernel classification. We extend this idea to a more general framework of gradient boosting [10, 16], which allows different loss functions for different types of learning problems. In this paper, we focus on the regression problem and use the squared loss.

At the t-th boosting iteration, for each kernel $k_m, m = 1, \ldots, M$, we first train a kernel regression model such as support vector regression (SVR) by fitting the current residuals, and obtain a solution of the form:

$$f_m^t(x) = \sum_{n=1}^{N} \alpha_{t,n} k_m(x, x_n). \tag{7}$$

Algorithm 1 BMKR: Boosting based Multiple Kernel Regression

Input: Data set $\mathcal{S}$, kernel functions $\mathcal{K}$, number of iterations T

Initialize residual: $r_n^1 = y_i, \forall n \in \{1, \ldots, N\}$, and $F = 0$
for $t = 1, ..., T$ **do**
  for $m = 1, ..., M$ **do**
    Sample N' data points from $\mathcal{S}$
    Train a kernel regression model f_m^t with k_m by fitting the residuals of the selected N' samples
    Compute the loss: $e_m^t = \frac{1}{2}\sum_{n=1}^{N}\left(f_m^t(x_n) - r_n^t\right)^2$
  end for
  Select the regression model with the smallest fitting error: $f^t = \arg\min_{f_m^t} e_m^t$
  Add f^t to the ensemble: $F \leftarrow F + \epsilon f^t$
  Update residuals: $r_n^{t+1} = y_n - F(x_n), \forall n \in \{1, 2, ...N\}$
end for

Output: the final multiple kernel function $F(x)$

Then we choose from M candidates, the regression model with the smallest fitting error

$$f^t = \underset{f_m^t, m\in\{1,...,M\}}{\arg\min} e_m^t, \tag{8}$$

where $e_m^t = \frac{1}{2}\sum_{n=1}^{N}\left(f_m^t(x_n) - r_n^t\right)^2$, and add it to the ensemble F. The final hypothesis of BMKR is expressed as in Eq. 3.

The pseudo-code of BMKR is shown in Algorithm 1. For gradient boosting with squared loss, the step size ρ^t is not strictly necessary [3], and we can either simply set it to 1, or a fixed small value ϵ as suggested by ϵ-boosting. Note that at each boosting iteration, instead of fitting all N samples, we can select only N' samples for training a SVR model, as suggested in [23], which can substantially reduce the computational complexity of each iteration as $N' \ll N$.

3.2 Boosting based Transfer Regression

As explained in Section 1, as we typically have very few data in the target domain, and therefore the model can easily overfit, especially if we train a complicated MKL model, even with the boosting approach. To deal with this issue, we can implicitly regularize the candidate functions at each boosting iteration by constraining the learning process within the function space spanned by the kernel functions trained on the source domains, rather than training the model in the function space spanned by arbitrary kernels. On the other hand, however, the underlying assumption of this approach is that at least one source domain is closely related to the target domain and therefore the kernel functions learned from the source domains can be reused. If this assumption does not hold, *negative transfer* could hurt the prediction performance. To avoid this situation, we also keep a MKL model which is trained only on the target domain. Consequently, the challenge becomes how to balance the knowledge embedded in the model learned from the source domains and the data fitting in the target domain.

Algorithm 2 BTMKR: Boosting based Transfer Multiple Kernel Regression

Input: Data set $\mathcal{S}_T$ from the target domain, number of iterations T, regularization parameter λ, multiple kernel functions $\{F_1, \ldots, F_S\}$ learned from S source domains, where each F_s is given by Eq. 6.

1: Initialize residual: $r_n^1 = y_n, \forall n \in \{1, \ldots, N\}$, and $F = 0$
2: **for** $t = 1, ..., T$ **do**
3: Compute the regression model f^* and h^* (line 8 – 21)
4: Select the base learner: $f^t = \begin{cases} f^*, & \text{if } \frac{\sum_{n=1}^N r_n^t f^*(x_n)}{\lambda} > \sum_{n=1}^N r_n^t h^*(x_n) \\ h^*, & \text{otherwise.} \end{cases}$
5: Add f^t to the ensemble: $F \leftarrow F + \epsilon f^t$
6: Update residuals: $r_n^{t+1} = y_n - F(x_n),\ \forall n \in \{1, 2, ...N\}$
7: **end for**
Output: the final multiple kernel function $F(x)$

K-BTMKR

8: **for** $s = 1, ..., S$ **do**
9: **for** $m = 1, ..., M$ **do**
10: Fit the current current residuals: $\gamma_{s,m}^t = \frac{\sum_{n=1}^N r_n^t h_m^s(x_n)}{\sum_{n=1}^N h_m^s(x_n)^2}$
11: Compute the loss of h_m^s: $e_{s,m}^t = \frac{1}{2}\sum_{n=1}^N \left(\gamma_{s,m}^t h_m^s(x_n) - r_n^t\right)^2$
12: **end for**
13: **end for**
14: Fit the residuals by training a kernel regressor: $f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{2}\sum_{n=1}^N \left(f(x_n) - r_n^t\right)$
15: Return the regression models: f^* and $h^* = \arg\min_{\{h_m^s\}} e_{s,m}^t$

M-BTMKR

16: **for** $s = 1, ..., S$ **do**
17: Fit the current current residuals: $\gamma_s^t = \frac{\sum_{n=1}^N r_n^t F_s(x_n)}{\sum_{n=1}^N F_s(x_n)^2}$
18: Compute the loss of F_s: $e_s^t = \frac{1}{2}\sum_{n=1}^N \left(\gamma_s^t F_s(x_n) - r_n^t\right)^2$
19: **end for**
20: Fit the residuals by training a kernel regressor: $f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{2}\sum_{n=1}^N \left(f(x_n) - r_n^t\right)$
21: Return the regression models: f^* and $h^* = \arg\min_{\{F_s\}} e_s^t$

To address this issue in a principled manner, we follow the idea of ϵ-boosting [20, 6] and propose the BTMKR algorithm, which is aimed towards transfer learning. There are two levels of transferring the knowledge of models: kernel-level transfer and model-level transfer, denoted by K-BTMKR and M-BTMKR respectively. At each iteration, K-BTMKR selects a single kernel function from $S \times M$ candidate kernels, while M-BTMKR selects a multiple kernel model from S domains. Therefore, K-BTMKR has higher "resolution" and more flexibility, at the price of higher risk of overfitting, as the dimension of its search space is M higher than that of M-BTMKR.

Kernel-Level Transfer (K-BTMKR) Let $\mathcal{H} = \{h_1^1, \ldots, h_M^1, \ldots, h_1^S, \ldots, h_M^S\}$ be the set of MS candidate kernel functions learned from S source domains, and $\mathcal{F} = \{f_1, \ldots, f_J\}$ be the set of J candidate kernel functions from the target domain. Note

that as the kernel functions from the source domains are fixed, the size of $\mathcal{H}$ is finite, while the size of the function space of the target domain is infinite, since the weights learned by SVR can be arbitrary (i.e., Eq. 7). For simplicity of analysis, we assume J is also finite. Given the mapping $h : \mathbb{R}^d \to \mathbb{R}^{MS}, h(x) = [h_1^1(x), \ldots, h_M^S]^\top$ defined by $\mathcal{H}$ and the mapping f defined by $\mathcal{F}$, we formulate the transfer learning problem as:

$$\min_{\beta_\mathcal{S}, \beta_\mathcal{T}} \mathcal{L}\left(\beta_\mathcal{S}, \beta_\mathcal{T}\right) \quad \text{s.t. } ||\beta_\mathcal{S}||_1 + \lambda||\beta_\mathcal{T}||_1 \leq \mu, \tag{9}$$

where $\mathcal{L}(\beta_\mathcal{S}, \beta_\mathcal{T}) \triangleq \sum_{n=1}^N \ell(\beta_\mathcal{S}^\top h(x_n) + \beta_\mathcal{T}^\top f(x_n), y_n)$, $\beta_\mathcal{S} \triangleq [\beta_1^1, \ldots, \beta_M^S]^\top \in \mathbb{R}^{MS}$, $\beta_\mathcal{T} \triangleq [\beta_1, \ldots, \beta_J]^\top \in \mathbb{R}^J$ are the coefficient vectors for the source domains and the target domain respectively, and λ is a parameter that controls how much we penalize $\beta_\mathcal{T}$ against $\beta_\mathcal{S}$. Intuitively, if the data from target domain is limited, we should set $\lambda \geq 1$ to favor the model learned from the source domains, in order to avoid overfitting.

Following the idea of ϵ-boosting [12, 20], Eq. 9 can be solved by slowly increasing the value of μ by ϵ, from 0 to a desired value. More specifically, let $g(x) = [h(x)^\top, f(x)^\top]^\top$, and $\beta = \left[\Delta\beta_\mathcal{S}^\top, \Delta\beta_\mathcal{T}^\top\right]^\top$. At the t-th boosting iteration, the coefficient vector β is updated to $\beta + \Delta\beta$ by solving the following optimization problem:

$$\min_{\Delta\beta} \mathcal{L}\left(\beta + \Delta\beta\right) \quad \text{s.t.} \quad ||\Delta\beta_\mathcal{S}||_1 + \lambda||\Delta\beta_\mathcal{T}||_1 \leq \epsilon \tag{10}$$

As ϵ is very small, the objective function of Eq. 10 can be expanded by first-order Taylor expansion, which gives

$$\mathcal{L}\left(\beta + \Delta\beta\right) \approx \mathcal{L}\left(\beta\right) + \nabla\mathcal{L}\left(\beta\right)^\top \Delta\beta, \tag{11}$$

where

$$\frac{\partial\mathcal{L}}{\partial\beta_j} = \sum_{n=1}^N -r_n^t g_j(x_n), \quad \forall j \in \{1, \ldots, MS + J\}. \tag{12}$$

By changing the coefficients $\tilde{\beta}_\mathcal{T} \leftarrow \lambda\beta_\mathcal{T}$, it can be shown that minimizing Eq. 10 can be (approximately) solved by

$$\Delta\beta_j = \begin{cases} \epsilon, & \text{if } j = \arg\max_j \frac{\sum_{n=1}^N r_n^t g_j(x_n)}{\lambda_j} \\ 0, & \text{otherwise} \end{cases}, \tag{13}$$

where $\lambda_j = 1, \forall j \in \{1, \ldots, MS\}$, and $\lambda_j = \lambda$, otherwise. In practice, as the size of function space of target domain is infinite, the candidate functions are actually computed by fitting the current residuals, as shown in Algorithm 2.

Model-Level Transfer (M-BTMKR) The derivation of M-BTMKR is similar to that of K-BTMKR, and therefore is omitted here.

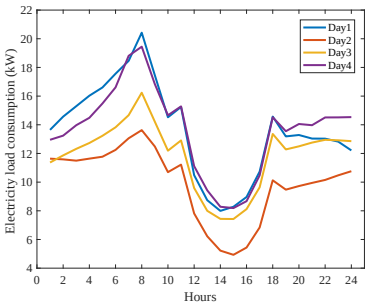

Fig. 1: Load data for four winter days

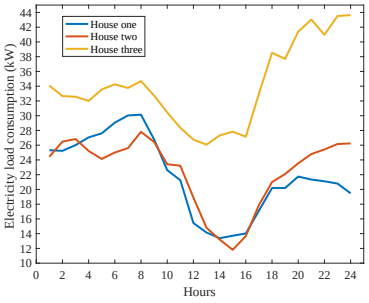

Fig. 2: Load data for three houses

3.3 Computational Complexity

The computational complexity of BMKR, as analyzed in [23], is $\mathcal{O}(TM\xi(N))$, where $\xi(N)$ is the computational complexity of training a single SVR with N samples. Standard learning approaches formulate SVR as a quadratic programming (QP) problem and therefore $\xi(N)$ is $\mathcal{O}(N^3)$. Lower complexity (e.g., about $\mathcal{O}(N^2)$) can be achieved by using other solvers (e.g., LIBSVM [5]). More important, BMKR can adopt stochastic learning approach, as suggested in [23], which only selects N' samples for training a SVR at each boosting iteration. This approach yields a complexity of $\mathcal{O}(TM(N + \xi(N')))$, which makes the algorithm tractable for large-scale problems by choosing $N' \ll N$. The computational complexity of the BTMKR algorithms is $\mathcal{O}(TM(SN + \xi(N)))$. Note that in the context of transfer learning, we use all the samples from the target domain, as the size of data set is usually small.

4 Experiments and Simulation Results

In this section, we evaluate the proposed algorithms on the problem of short-term electricity load forecasting for residential houses. Several factors including day types, weather conditions, and the lagged load consumption itself may affect the load profile of a given house. In this paper, we use three kinds of features for load forecasting: lagged load consumption, i.e., electricity consumed in the last three hours, temperature in the last three hours, and weekday/weekend information.

4.1 Data Description

The historical temperature data are obtained from [14], and the residential house load consumption data are provided by the US Energy department [17]. The data set includes hourly residential house load consumption data for 24 locations in New York state in 2012. For each location, it provides data for three types of houses, based on the house size: low, base, and high. Fig. 1 shows load consumption for a base type house for four consecutive winter days. We can see that the load consumption starts to decrease from 8 am and increases very quickly from 4 pm. Fig. 2 shows the load consumption for

Table 1: MAPE (%) performance (mean $\pm$ std dev) for high load consumption houses

Method	Spring	Summer	Fall	Winter	Average
Linear	10.42 ± 0.10	7.78 ± 0.13	9.21 ± 0.22	5.81 ± 0.13	8.30 ± 0.15
SVR	10.95 ± 0.21	7.73 ± 0.11	8.82 ± 0.21	5.88 ± 0.12	8.34 ± 0.16
BMKR	**10.31** ± 0.17	**7.64** ± 0.02	**8.42** ± 0.11	**5.73** ± 0.07	**8.02** ± 0.10

three high load consumption houses in nearby cities for the same winter day. It can be observed that the load consumption for house 1 is similar to house 2 and both are different from house 3.

4.2 BMKR for Electricity Load Forecasting

To test the performance of BMKR, we use the data of a high energy consumption house in New York City in 2012. We test the performance of BMKR separately for different seasons, and compare it with single kernel SVR and linear regression. We set the number of boosting iterations for the proposed algorithms to 100, the step-size of ϵ to 0.05, and the sampling ratio to 0.9. In order to accelerate the learning process, we initialize the model with linear regression. The candidate kernels for BMKR are: Gaussian kernels with 10 different widths $(2^{-4}, 2^{-3}, .., 2^{5})$ and a linear kernel. We repeat the simulation for 10 times, and each time we randomly choose 50% of the data in the season as training data and 50% of the data as testing data.

Table 1 shows the mean and standard deviation (std dev) of the Mean Average Percentage Error (MAPE) measurement for BMKR and the other two baselines. We can see that BMKR achieves the best forecasting performance for all seasons, obtaining 3.3% and 3.8 % average MAPE improvements over linear regression and single kernel SVR respectively.

4.3 Transfer Regression for Electricity Load Forecasting

We evaluate the proposed transfer regression algorithms: M-BTMKR and K-BTMKR on high load consumption houses. We randomly pick 6 high load consumption houses as target house and use the remaining 18 high consumption houses as source houses. We repeat the simulation 10 times for each house, and each time we randomly choose 36 samples as the training data, and 100 samples as the testing data for the target house. For source houses, we randomly chose 600 data samples as the training data in each simulation. For K-BTMKR and M-BTMKR, λ is chosen by cross validation to balance the model leaned from source house data and the model learned from target house data.

Performance of M-BTMKR and K-BTMKR are compared with linear regression, single kernel SVR and BMKR. The candidate kernels and boosting setting are the same as in Section 4.2. For the baselines, the forecasting models are trained only with data from target houses, and the results are shown in Table 2[2], from which it can be observed

[2] Due to the space limitation, we only report the results for high load consumption houses. The results for low and base load consumption houses are similar to the high load consumption houses.

Table 2: Transfer learning MAPE (%) performance for high load consumption houses

Method	Location 1	Location 2	Location 3	Location 4	Location 5	Location 6	Average
Linear	8.02 ± 0.05	9.11 ± 0.70	17.39 ± 1.62	6.05 ± 0.02	11.43 ± 0.15	9.42 ± 0.65	10.24 ± 0.53
SVR	11.53 ± 0.34	6.82 ± 0.39	25.90 ± 0.72	8.24 ± 0.08	26.31 ± 1.97	14.00 ± 0.65	15.47 ± 0.69
BMKR	8.06 ± 0.03	6.64 ± 0.54	17.85 ± 1.31	5.29 ± 0.01	12.82 ± 0.21	9.05 ± 0.57	9.95 ± 0.45
M-BTMKR	**5.35** ± 0.01	5.99 ± 0.02	**5.63** ± 0.19	**5.01** ± 0.01	9.13 ± 0.01	**5.69** ± 0.01	**6.13** ± 0.04
K-BTMKR	5.38 ± 0.02	**5.46** ± 0.30	6.97 ± 0.26	5.55 ± 0.09	**8.96** ± 0.14	7.31 ± 0.21	6.60 ± 0.17

that the proposed transfer algorithms significantly improve the forecasting performance. For each individual location, the best results are achieved by either K-BTMKR or M-BTMKR, and M-BTMKR shows the best performance on average. The forecasting accuracies of M-BTMKR and K-BTMKR are very close to each other and both are much better than the baseline algorithms without transfer. In other words, with the proposed transfer algorithms, the knowledge learned from the source houses is properly transferred to the target house.

4.4 Negative Transfer Analysis

Sometimes the consumption pattern for source houses and target houses can be quite different. We would prefer that the transfer algorithms prevent potential negative transfer for such scenarios. Here we present a case study to show the importance of balancing the knowledge learned from source domains and data fitting in the target domain. We use the same high load target houses as described in Section 4.3, but for the source houses, we randomly chose eighteen houses from the low type houses. We repeat the simulation for 10 times and the results are shown in Table 3.

The proposed algorithms are compared with linear regression, single kernel SVR, BMKR, M-BTMKR$_{woT}$, and K-BTMKR$_{woT}$, where M-BTMKR$_{woT}$ and K-BTMKR$_{woT}$ denote the BTMKR algorithms that we do not keep a MKL model trained on the target domain when we learn BTMKR models (i.e., we do not train f^* in Algorithm 2.). Simulation results show that, if we do not keep a MKL model trained on the target domain, we would encounter severe negative transfer problem, and the forecasting accuracy would be even much worse than the models learned without transfer. Meanwhile, we can see that the proposed M-BTMKR and K-BTMKR could successfully avoid such negative transfer. In this case, M-BTMKR and K-BTMKR still show better performance than other algorithms, though the forecasting accuracy of K-BTMKR is very close to BMKR. M-BTMKR achieves the best average forecasting performance and provides 14.37 % average forecasting accuracy improvements over BMKR. In summary, the BTMKR algorithms can avoid the negative transfer when the data distributions of source domain and target domain are quite different.

5 Related Work

Various techniques have been proposed to efficiently learn MKL models [11], and our BMKR algorithm is originally inspired by [23], which applies the idea of AdaBoost to train a multiple kernel based classifier. BMKR is a more general framework which can

Table 3: Transfer learning MAPE (%) performance for high load consumption target houses with low load consumption source houses

Method	Location 1	Location 2	Location 3	Location 4	Location 5	Location 6	Average
Linear	8.02 ± 0.05	9.11 ± 0.70	17.39 ± 1.62	6.05 ± 0.02	11.43 ± 0.15	9.42 ± 0.65	10.24 ± 0.53
SVR	11.53 ± 0.34	6.82 ± 0.39	25.90 ± 0.72	8.24 ± 0.08	26.31 ± 1.97	14.00 ± 0.65	15.47 ± 0.69
BMKR	8.06 ± 0.03	**6.64** ± 0.54	17.85 ± 1.31	5.29 ± 0.01	12.82 ± 0.21	9.05 ± 0.57	9.95 ± 0.45
M-BTMKR	**7.71** ± 0.01	8.74 ± 0.27	**8.65** ± 1.39	6.51 ± 0.52	**11.08** ± 0.21	**8.42** ± 0.86	**8.52** ± 0.54
M-BTMKR$_{woT}$	57.64 ± 0.05	59.02 ± 0.16	59.71 ± 0.53	46.25 ± 0.81	38.52 ± 0.02	56.71 ± 0.30	52.98 ± 0.31
K-BTMKR	7.80 ± 0.06	8.60 ± 0.74	16.27 ± 2.48	**5.77** ± 0.16	11.42 ± 0.15	9.33 ± 0.67	9.87 ± 0.71
K-BTMKR$_{woT}$	54.81 ± 0.05	58.31 ± 0.17	59.00 ± 0.11	43.95 ± 0.25	37.49 ± 0.12	56.81 ± 0.03	51.73 ± 0.12

adopt different loss functions for different learning tasks. Furthermore, the boosting approach provides a natural approach to solve small sample size problems by leveraging transfer learning techniques. The original work on boosting based transfer learning proposed in [7] introduces a sample-reweighting mechanism based on AdaBoost for classification problem. Later, this approach is generalized to the cases of regression [19], and transferring knowledge from multiple sources [24]. In [6], a gradient boosting based algorithm is proposed for multitask learning, where the assumption is that the model parameters of all the tasks share a common factor. In [22], the transfer boosting and multitask boosting algorithms are generalized to the context of online learning. While both multiple kernel learning and transfer learning have been studied extensively, the effort in simultaneously dealing with these two issues is very limited. Our BTMKR algorithm distinguishes itself from these methods because it deals with these two learning problems in a unified and principled approach. To our best knowledge, this is the first attempt to transfer MKL for regression problem.

6 Conclusion

In this paper, we first propose BMKR, a gradient boosting based multiple kernel learning framework for regression, which is suitable for short-term electricity load forecasting problems. Different from the traditional methods for MKL, the proposed BMKR algorithm learns the combination weights for each kernel using a boosting-style algorithm. Simulation results on residential data show that the short-term electricity load forecasting could be improved with BMKR. We further extend the proposed boosting framework to the context of transfer learning and propose two boosting based transfer multiple kernel regression algorithms: K-BTMKR and M-BTMKR. Empirical results suggest that both algorithms can efficiently transfer the knowledge learned from source houses to the target houses and significantly improve the forecasting performance when the target houses and source houses have similar electricity load consumption pattern. We also investigate the effects of negative transfer and show that the proposed algorithms could prevent potential negative transfer when the source houses are quite different from the target houses.

References

1. Atsawathawichok, P., Teekaput, P., Ploysuwan, T.: Long term peak load forecasting in Thailand using multiple kernel Gaussian Process. In: ECTI-CON. pp. 1–4 (2014)
2. Bach, F.R., Lanckriet, G.R., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML. pp. 6–13 (2004)
3. Bühlmann, P., Hothorn, T.: Boosting algorithms: Regularization, prediction and model fitting. Statistical Science pp. 477–505 (2007)
4. Bunn, D., Farmer, E.D.: Comparative models for electrical load forecasting (1985)
5. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intelligent Systems and Technology 2(3), 27 (2011)
6. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Boosted multi-task learning. Machine Learning 85(1-2), 149–173 (2011)
7. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: ICML. pp. 193–200 (2007)
8. Fiot, J.B., Dinuzzo, F.: Electricity demand forecasting by multi-task learning. IEEE Trans. Smart Grid (2016)
9. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: ICML. pp. 148–156 (1996)
10. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics pp. 1189–1232 (2001)
11. Gönen, M., Alpaydın, E.: Multiple kernel learning algorithms. Journal of Machine Learning Research 12, 2211–2268 (2011)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer New York (2009)
13. Hippert, H.S., Pedreira, C.E., Souza, R.C.: Neural networks for short-term load forecasting: A review and evaluation. IEEE Trans. on Power Systems 16(1), 44–55 (2001)
14. IEM: `https://mesonet.agron.iastate.edu/request/download.phtml`
15. Kamyab, F., Amini, M., Sheykhha, S., Hasanpour, M., Jalali, M.M.: Demand response program in smart grid using supply function bidding mechanism. IEEE Trans. Smart Grid 7(3), 1277–1284 (2016)
16. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent in function space. In: NIPS. pp. 512–518 (2000)
17. OPENEI: `http://en.openei.org/doe-opendata/dataset`
18. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowledge and Data Engineering 22(10), 1345–1359 (2010)
19. Pardoe, D., Stone, P.: Boosting for regression transfer. In: ICML. pp. 863–870 (2010)
20. Rosset, S., Zhu, J., Hastie, T.: Boosting as a regularized path to a maximum margin classifier. Journal of Machine Learning Research 5, 941–973 (2004)
21. Soliman, S.A.h., Al-Kandari, A.M.: Electrical Load Forecasting: Modeling and Model Construction. Elsevier (2010)
22. Wang, B., Pineau, J.: Online boosting algorithms for anytime transfer and multitask learning. In: AAAI. pp. 3038–3044 (2015)
23. Xia, H., Hoi, S.C.: MKBoost: A framework of multiple kernel boosting. IEEE Trans. on Knowledge and Data Engineering 25(7), 1574–1586 (2013)
24. Yao, Y., Doretto, G.: Boosting for transfer learning with multiple sources. In: CVPR. pp. 1855–1862 (2010)
25. Zhang, R., Dong, Z.Y., Xu, Y., Meng, K., Wong, K.P.: Short-term load forecasting of australian national electricity market by an ensemble model of extreme learning machine. IET Generation, Transmission & Distribution 7(4), 391–397 (2013)
26. Zhuang, J., Tsang, I.W., Hoi, S.C.: Two-layer multiple kernel learning. In: AISTATS. pp. 909–917 (2011)