Flexible Transfer Learning Framework for Bayesian Optimisation

Tinu Theckel Joy(✉), Santu Rana, Sunil Kumar Gupta, and Svetha Venkatesh

Centre for Pattern Recognition and Data Analytics,
Deakin University, Geelong 3216, Australia
{ttheckel,santu.rana,sunil.gupta,svetha.venkatesh}@deakin.edu.au

Abstract. Bayesian optimisation is an efficient technique to optimise functions that are expensive to compute. In this paper, we propose a novel framework to transfer knowledge from a completed source optimisation task to a new target task in order to overcome the cold start problem. We model source data as noisy observations of the target function. The level of noise is computed from the data in a Bayesian setting. This enables flexible knowledge transfer across tasks with differing relatedness, addressing a limitation of the existing methods. We evaluate on the task of tuning hyperparameters of two machine learning algorithms. Treating a fraction of the whole training data as source and the whole as the target task, we show that our method finds the best hyperparameters in the least amount of time compared to both the state-of-art and no transfer method.

1 Introduction

Whether it is the design of new products in manufacturing, or tuning hyperparameters in machine learning algorithms, it is expensive to search for the best solution exhaustively because these functions are expensive to evaluate. Bayesian optimisation offers powerful solutions in this space [1]. It is efficient in terms of the number of function evaluations required, and is powerful to model objective functions without knowing its form [2]. Bayesian optimisation has been successfully applied in many different fields including learning optimal robot mechanics [3], sequential experimental design [4], optimal sensor placement [5], etc.

Recently, it has found popularity in tuning hyperparameters for machine learning algorithms [6,7]. A problem arises in the case of "cold start", when a new tuning task is tackled. In initial trials, many bad set of hyperparameters may be recommended before a good region is found. When data is large and model is complex, tuning hyperparameters can be excruciatingly long. Reducing the time to optimally tune remains an important problem to solve.

One solution is to induce transfer learning by leveraging the data from previous tasks. Bardenet et al. [8] build a model from past experience by biasing search in a new problem towards the part of the hyperparameter space where optimal hyper-parameters can be found. Incorporating a surrogate based ranking method, they can collaboratively optimise similar objective functions.

J. Bailey et al. (Eds.): PAKDD 2016, Part I, LNAI 9651, pp. 102–114, 2016.
DOI: 10.1007/978-3-319-31753-3_9

Yogatama and Mann [9] use a Bayesian optimisation setting to transfer knowledge from one dataset to the next by using a Gaussian process to model deviations from the per dataset mean. However, both assume that the transfer occurs where the source (previous dataset) and the target (current dataset) tasks are highly related *e.g.* [8] assumes strong similarity in terms of ranking behavior and [9] assumes strong similarity in the deviations from the respective means. *Therefore, transfer learning approach for Bayesian optimisation that can handle different relatedness among the source and the target tasks in a principled manner is still an open problem.*

Addressing this problem we propose an alternate framework to transfer knowledge across tasks. Intuitively, we do this by modeling the source data as noisy observations of the target function. We achieve this through the modification of the kernel of the Gaussian process, adding more noise variance to source observations. We start by assuming that the source and target functions lie within some envelope of each other. The width of the envelope is determined by the noise variance - smaller noise variance imply that the source observations provide a strong prior knowledge of the target function, larger noise variance imply that the source does not influence the target function. Former is required when tasks are related and the latter is desirable when the tasks are unrelated. We estimate the appropriate noise variance for the target and a source task from the data in a Bayesian setting.

We apply our algorithm for hyperparameter tuning using a novel setting. We sample a small fraction of the data, treating it as a source. This source dataset is then evaluated exhaustively on a number of different hyperparameters. Since the number of data in the source is small, the cost of exhaustive evaluation is low. This knowledge is now used to tune the hyperparameters of the original dataset. We experiment on three benchmark classification datasets for finding the best hyperparameters for two machine learning approaches - elastic net and support vector machine with RBF kernel. In all the experiments, our method is able to find the best hyperparameters in the least amount of time (considering time taken by both source sampling and the target optimisation) over both the current state-of-the-art and the usual tuning algorithm without transfer learning.

In short our contributions are:

- Proposal of a new transfer learning algorithm for Bayesian optimisation in the most general setting that includes source and target tasks with different similarity. This is achieved by modeling the source as a noisy observation of the target function and automatically estimating the noise variance from data.
- Proposal of a novel setting for tuning hyperparameters that exploits the proposed transfer learning framework to improve efficiency. Using a small fraction of the training dataset as a source task, we accelerate the hyperparameter tuning for the whole training set, which is modeled as the target task.
- Evaluation of the proposed algorithm for the best hyperparameter search for two machine learning algorithms on three benchmark classification datasets. Our method is around 6 times faster than methods that tune hyperparameters without transfer learning and around 3 times faster than the state-of-art transfer learning for Bayesian optimisation [9].

2 Preliminaries

2.1 Gaussian Process

Gaussian processes (GPs) [10] are a way of specifying prior distributions over the space of smooth functions. The properties of the Gaussian distribution allow us to compute the predictive means and variances in closed form. It is specified by its mean function, $\mu(x)$ and covariance function, $k(x, x^{'})$. A sample from a Gaussian process is a function as,

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x^{'})). \tag{1}$$

where the value $f(x)$ at an arbitrary x is a Gaussian distributed random variable specified by a mean and a variance. Without any loss in generality, the prior mean function can be assumed to be a zero function making the Gaussian process fully defined by the covariance function. A popular choice of covariance function is squared exponential function,

$$k(\mathbf{x}, \mathbf{x}') = exp(-\frac{1}{2}||\mathbf{x} - \mathbf{x}'||^2) \tag{2}$$

Other choice of covariance functions include linear kernel, Matérn kernel etc.

Let us assume that we have data points $\mathbf{x}_{1:p}$ and say that the function values corresponding to those data points are sampled from the prior Gaussian process with mean zero and the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. Let us denote $\mathbf{y}_{1:p} = f(\mathbf{X}_{1:p})$ as the function values corresponding to the data points $\mathbf{X}_{1:p}$. The function values $\mathbf{y}_{1:p}$ jointly follow a multivariate Gaussian distribution $\mathbf{y}_{1:p} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_p) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_p, \mathbf{x}_1) & \dots & k(\mathbf{x}_p, \mathbf{x}_p) \end{bmatrix} \tag{3}$$

is called the kernel matrix. For a new data point $\mathbf{x}_{p+1}$, let the function value be $y_{p+1} = f(\mathbf{x}_{p+1})$. Then, by the properties of Gaussian process, $\mathbf{y}_{1:p}$ and y_{p+1} are jointly Gaussian as,

$$\begin{bmatrix} \mathbf{y}_{1:p} \\ y_{p+1} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{p+1}, \mathbf{x}_{p+1}) \end{bmatrix} \tag{4}$$

where $\mathbf{k} = [k(\mathbf{x}_1, \mathbf{x}_{p+1})\ k(\mathbf{x}_2, \mathbf{x}_{p+1})\ \dots\ k(\mathbf{x}_p, \mathbf{x}_{p+1})]$. Using Sherman-Morrison-Woodburry [10] formula, the predictive distribution of the function value at a new location $(\mathbf{x}_{p+1})$ can be written as,

$$P(y_{p+1}|\mathbf{X}_{1:p}, \mathbf{y}_{1:p}) \sim \mathcal{N}(\mu_p(\mathbf{x}_{p+1}), \sigma_p^2(\mathbf{x}_{p+1})) \tag{5}$$

where the predicted mean and the variance is given by $\mu_p(\mathbf{x}_{p+1}) = \mathbf{k}^T\mathbf{K}^{-1}\mathbf{y}_{1:p}$ and $\sigma(\mathbf{x}_{p+1}) = k(\mathbf{x}_{p+1}, \mathbf{x}_{p+1}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{y}_{1:p}$ respectively. If the observation is

a noisy estimate of the actual function value i.e. $y = f(x) + \eta$, where $\eta \sim \mathcal{N}(0, \sigma^2_{noise})$ the modified predictive distribution becomes

$$P(y_{p+1}|\mathbf{X}_{1:p}, \mathbf{y}_{1:p}) \sim \mathcal{N}(\mu_p(\mathbf{x}_{p+1}), \sigma^2_p(\mathbf{x}_{p+1}) + \sigma^2_{noise}) \tag{6}$$

where $\mu_p(\mathbf{x}_{p+1}) = \mathbf{k}^T[\mathbf{K}+\sigma^2_{noise}\mathbf{I}]^{-1}\mathbf{y}_{1:p}$ and $\sigma^2_p(\mathbf{x}_{p+1}) = k(\mathbf{x}_{p+1}, \mathbf{x}_{p+1}) - \mathbf{k}^T[\mathbf{K}+\sigma^2_{noise}\mathbf{I}]^{-1}\mathbf{y}_{1:p}$ respectively.

2.2 Bayesian Optimisation

Bayesian optimisation is an efficient method for the global optimisation of costly objective functions [2]. It is especially used in situations where one does not have access to the function form. The user only has the access to the noisy evaluations. Examples in machine learning include tuning hyperparameters of a machine learning model, where the function that relates the choice of hyperparameters to the model performance is unknown and can be very complex.

The unknown function is modeled using a Gaussian process. Bayesian optimisation then employs a simple strategy where it makes use of a surrogate utility function, which is easy to evaluate. The surrogate utility function is called *acquisition function*. The role of the acquisition function is to guide us to reach the optimum of the underlying function. Essentially, acquisition functions are defined in such a way that a high value of the acquisition function corresponds to the potential high value of the underlying function when the optimisation problem is a maxima problem. The new point (*e.g.* new hyperparameter setting) to evaluate next, is then obtained by maximizing the acquisition function.

Acquisition Functions. Acquisition function can be defined either using improvement based criteria or using confidence based criteria. Improvement based criteria such as Probability of Improvement (PI) [11] or Expected Improvement (EI) [12] results in maximizing the probability of improvement over the current best or the improvement in the expected sense. Confidence based criteria such GP-UCB [13] use the upper confidence bound of the GP predictive distribution as an acquisition function. Sometime, a mix of them can also be used as the acquisition function [4]. In this paper, we use EI as the criteria for its usefulness and simplicity. A brief description of EI is provided below.

Expected Improvement (EI). Let us assume that our optimisation problem is optimizing $\arg\max_{\mathbf{x}} f(\mathbf{x})$ and the current best is at $\mathbf{x}^+ = \arg\max_{\mathbf{x}_i \in \mathbf{X}_{1:p}} f(\mathbf{x}_i)$. The improvement function is defined as,

$$I(\mathbf{x}) = \max\{0, f(\mathbf{x}) - f(\mathbf{x}^+)\} \tag{7}$$

The acquisition function is then defined on the expected value of $I(\mathbf{x})$ [2] as,

$$\arg\max_{\mathbf{x}} \mathbb{E}(I(\mathbf{x})|D_{1:p}) \tag{8}$$

Algorithm 1. Generic Bayesian Optimisation Algorithm

1: **Input:** The initial observation $D \equiv \{\mathbf{X}_{1:p}, \mathbf{y}_{1:p}\}$.
2: **Output**: $\{x_n, y_n\}_{n=1}^{T}$
3: **for** $n = 1, 2, ..T$
4: Find $x_n = \arg\max_{\mathbf{x}} \mathbb{E}(I(\mathbf{x})|D)$ of (9).
5: Evaluate the objective function: $y_n = f(x_n)$.
6: Augment the observation set $D = D \cup (x_n, y_n)$ and update the GP.
7: **end for**

where $D_{1:p} \equiv \{\mathbf{x}_{1:p}, \mathbf{y}_{1:p}\}$. The analytic form of $E(I(\mathbf{x}))$ can be obtained as [12],

$$\mathbb{E}(I(\mathbf{x})) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(z) + \sigma(\mathbf{x})\phi(z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{9}$$

where $z = (\mu(\mathbf{x}) - f(\mathbf{x}^+))/\sigma(\mathbf{x})$. $\Phi(.)$ and $\phi(.)$ are the CDF and PDF of a standard normal distribution respectively.

The generic Bayesian optimisation algorithm is presented in Algorithm 1.

3 Proposed Method

The generic Bayesian optimisation algorithm suffers from "cold start" problem i.e. at the beginning it may take many trials before it reaches a good region. To improve the efficiency, we propose a novel Bayesian optimisation framework using transfer learning. We elaborate our framework in the context of hyperparameter optimisation.

Let us denote the source observations as $\{\mathbf{x}_i^s, y_i^s\}_{i=1}^{N^s}$, where $\mathbf{x}_i^s$ denotes the hyperparameter setting, y_i^s is the performance of the model built using the hyperparameters $\mathbf{x}_i^s$ and N^s is the size of source observations. The source observations are generated either on a grid or at random hyperparameter settings. No optimisation is performed at this stage. We assume that the source and the target function lies within a close proximity with each other since they only differ in the amount of training data; source models the mapping from hyperparameter setting to the model performance on a small subset of the whole training data, whereas target function maps the same for the whole training data. We model the difference between the source and target. We use source data to provide us with a rough guideline about the target function, $f^t(.)$. To accomplish this, we model source observations as a noisy measurement of the target function as,

$$y_i^s = f^t(\mathbf{x}_i^s) + \epsilon_i^s, \forall i = 1, \ldots, N^s \tag{10}$$

where $\epsilon^s \sim \mathcal{N}(0, \sigma_s^2)$ is a random noise. This implies that the source function values lies within $3\sigma_s$ ball of the target function values with a probability close to 1.

Let us denote the observations from the target task as $\{\mathbf{x}_i^t, y_i^t\}_{i=1}^{N^t}$, where N^t is the number of target observations so far. We combine data from both the source and target and create a combined observation set: $\mathbf{X} = \{\mathbf{X}^s, \mathbf{X}^t\}$ and

$\mathbf{y} = \{\mathbf{y}^s, \mathbf{y}^t\}$. The target GP is built using the combined observation. The kernel matrix for the combined data is computed and then it is updated to incorporate the noise of the source as,

$$\mathbf{K} = \mathbf{K} + \begin{bmatrix} \sigma_s^2 I_{N^s \times N^s} & \mathbf{0} \\ \mathbf{0}^T & \sigma_t^2 I_{N^t \times N^t} \end{bmatrix} \tag{11}$$

where σ_s^2 models the closeness between the source and the target function and σ_t^2 is the measurement noise for the target. The value of the σ_s^2 reflects our belief on how close the source and target functions are. If they are thought to be very close then we should set σ_s^2 small and large otherwise. The value of σ_s^2 can greatly effect the efficiency of the Bayesian optimisation. In the next section, we will provide a principled way to estimate its value from the target observations.

Estimating the Source Noise Variance ($\boldsymbol{\sigma_s^2}$). We estimate the source noise variance from the data by placing an inverse gamma distribution with parameters α_0 and β_0 as the prior distribution as,

$$\sigma_s^2 \sim InvGamma(\alpha_0, \beta_0) \tag{12}$$

We start with a wide prior and then update the posterior from the observation of output value of the target (y^t) and the source (y^s) for the same hyperparameter setting. We use the evaluated target value at the recommended settings and use the source predicted value ($\hat{y}^s$) at those settings. The source function is modeled with a Gaussian process. Since the inverse gamma is a conjugate prior to the variance, the posterior is also an inverse gamma distribution with updated parameters α_n and β_n as,

$$P(\sigma_s^2 / \{y_i^s - \hat{y}_i^s\}_{i=1}^{N^t}) \sim InvGamma(\alpha_n, \beta_n) \tag{13}$$

Assuming the mean of the difference to be zero, the parameters $\alpha_n$and β_n is updated as follows,

$$\alpha_n = \alpha_0 + n/2 \tag{14}$$

$$\beta_n = \beta_0 + \frac{\sum_{i=1}^{N^t} (y_i^t - \hat{y}_i^s)^2}{2} \tag{15}$$

We use the mode of the posterior distribution as the value of source noise variance and it is given by,

$$\sigma_s^2 = \frac{\beta_n}{\alpha_n + 1} \tag{16}$$

The kernel matrix is recomputed following (11) and the Bayesian optimisation is sequentially performed using this kernel matrix. The proposed algorithm is illustrated in Algorithm 2.

Algorithm 2. Proposed Transfer Learning Algorithm.

1: **Input**: Source Observations: $\{\mathbf{X}^s, \mathbf{y}^s\}$, Target Observations: $\{\mathbf{X}^t, \mathbf{y}^t\}$, Combined Observation set: $\mathbf{X} = \{\mathbf{X}^s, \mathbf{X}^t\}$, $\mathbf{y} = \{\mathbf{y}^s, \mathbf{y}^t\}$, D= {**X**,**y**}.
2: **Initial Settings:** Fit a GP at the source points $\{\mathbf{X}^s, \mathbf{y}^s\}$, Fit a GP at the combined Observation Set D, Compute σ_s^2 and update $\mathbf{K}$ using (16) and (11).
3: **Output**:$\{x_n, y_n\}_{n=1}^{T}$.
4: **for** $n = 1, 2, ..T$
5: Find $x_n = \arg\max_{\mathbf{x}} \mathbb{E}(I(\mathbf{x})|D)$ of (9).
6: Evaluate the target function: $y_n = f^t(x_n)$.
8: Compute $\hat{y}_n^s$ at x_n using the GP.
9: Update α_n and β_n using (14) and (15).
10: Update source noise variance σ_s^2 and $\mathbf{K}$ using (16) and (11).
11: Augment the observation set $D = D \cup (x_n, y_n)$ and Update the GP.
12: **end for**

4 Experiments

4.1 Experimental Setup

We conduct experiments on both synthetic and real world datasets. Through two different synthetic datasets, we create two transfer learning situations by varying the similarity between the source and the target functions and analyse the behavior of the proposed algorithm in those cases. For the experiment with synthetic data, we do not tune hyperparameters of any classifier, rather we assume that the source and the target functions are known and the task is to reach the maximum of the target function. Experiments with real world dataset are performed to evaluate our algorithm with respect to the baselines on the efficiency of tuning hyperparameters for two classification algorithms. For the experiment on tuning hyperparameters, a fraction of the training data is treated as the source and whole as the target task.

We compare the proposed method with the following baselines:

- **Efficient-BO** [9]**:** In this transfer learning approach, a common function for source and target is learnt where the common function is represented as deviations from their respective means.
- **Generic-BO:** Algorithm 1 is used only on the target dataset.

We evaluate based on both the number of iteration taken and total time taken to reach the maximum performance. All timings reported in the experiments, are computed for programs running on a workstation with 3.4 GHz quad-core processor having 8 GB RAM.

4.2 Experiment with Synthetic Data

We generate two synthetic datasets: in synthetic dataset-I, we create a a target function that is highly similar to the source function and in synthetic dataset-II, we create the target function to be very different from the source function.

The source function is always fixed in both the scenarios whilst the target function is varied. The source function is a 2-variate normal probability distribution function with mean at [0,0] and covariance matrix as $I_{2\times 2}$. The target functions are also modeled by another 2-variate normal probability distribution function with the same covariance matrix but at different means. For synthetic dataset-I, the target task mean is set at $(0.1, 0.1)$ which is very close to the source function mean. The two functions are shown in Fig. 1a. For synthetic dataset-II, the target task mean is set at $(1.5, 1.5)$ which is far from the source mean. The two functions for this scenario is shown in Fig. 1b. For both the scenarios source functions are represented with 25 data points sampled randomly between $[-3, 6]$ along both the dimensions.

Figure 1c plots the percentage of the maximum value reached as a function of the iteration for the proposed method and the baselines for synthetic dataset-I. All the three methods start from the same position, but our proposed method is able to gain faster reaching 80 % of the maximum value within 7 and reaching at the maximum value by 22. In comparison, Efficient-BO reaches 80 % of the maximum value after 10, although finally reaching at the maximum around the same time as the source.

The generic Bayesian optimisation without any help from source function knowledge, reaches 80 % of the maximum value only after 15 iterations and not reaching the maximum even after 30 iterations. Figure 1e shows the noise variance estimate after each iteration for the proposed method. The variance starts with a high prior and decreases fast as the two functions are very close to each other.

Figure 1d plots the percentage of the maximum value reached as a function of the iteration for the proposed method and the baselines for synthetic dataset-II. We see that, even if all of them start from the same position, our proposed method is able to gain faster reaching close to the maximum value by 15th iteration. In comparison, the generic Bayesian optimisation reaches to a similar value only after 28th iteration. In this case, since the source and the target functions are quite different, Efficient-BO fails to reach beyond 60 % of the maximum with 35 iterations. Figure 1f shows the noise variance estimate after each iteration for the proposed method. The variance starts at the same position as it started for synthetic dataset-I, but instead of decreasing, the variance increases with the iteration as the the source and the target functions are quite different for this scenario.

4.3 Experiment with Real World Datasets

We experiment with three real world datasets for tuning hyperparameters for two classification algorithms: Elastic net and Support Vector Machines (SVM) with radial basis function (RBF) kernel. All three datasets are benchmark datasets used in [14]. A brief description of the datasets are provided in Table 1. For elastic net, the hyperparameters are the L_1 and L_2 penalty weights. The bounds for both of them are chosen to be within $[10^{-5}, 10^{-1}]$. For SVM with RBF kernel, the hyperparameters are the cost parameter (C) of the SVM formulation and the

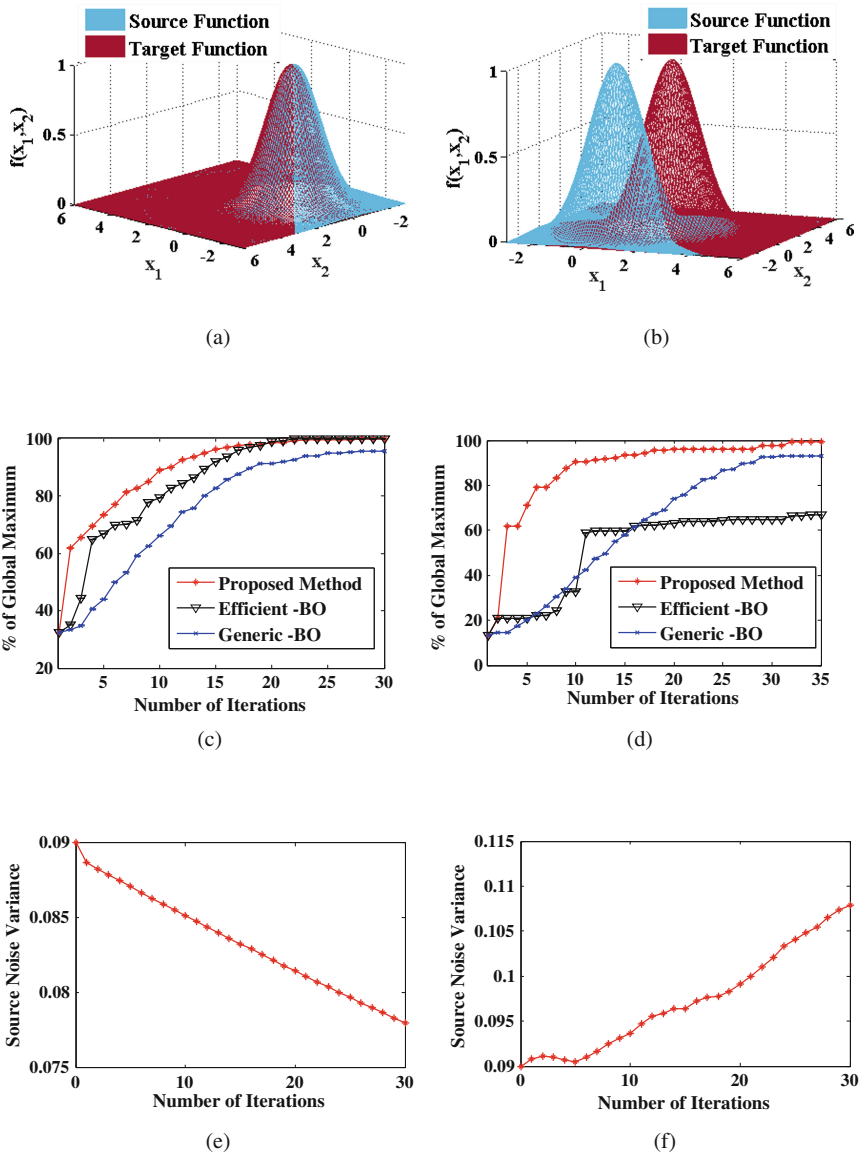

Fig. 1. Synthetic dataset-I (left column) and Synthetic dataset-II (right column): (a, b) the source (blue) and the target functions (red), (c, d) percentage of the maximum value achieved as a function of number of iterations for the three different methods and (e, f) estimated source noise variance after each iteration for the proposed method (Color figure online).

width of the RBF kernel. We choose $10^{-3}$to 10^{3} and $10^{-5}$to 10^{0} as bounds for C and the width of the kernel, respectively. As the ranges are high, we perform Bayesian optimisation on the logarithmic of the values of the hyperparameters. For each dataset, 5 separate training datasets are created by randomly sampling 70 % of the data for training. Average results over these training set are reported.

Table 1. Datasets used in the experiments.

Dataset	Number of data points	Number of features
Liver disorders	345	6
Heart Diseases	270	13
Breast Cancer	683	10

The results for the hyperparameter tuning of the machine learning algorithms on the different datasets are presented in Fig. 2. The results show that the proposed method achieves the maximum accuracy in the least number of evaluations compared to both the methods. The transfer learning based Efficient-BO follows closely but have never been able to reach to the optimal hyperparameters faster than the proposed method. The Generic-BO without any transfer learning performs the slowest and mostly not being able to reach to the best within 30 iterations.

In Table 2, we present the actual time taken to find the best hyperparameters in CPU seconds. On all the datasets and for both the classification algorithms, our proposed method reaches to the optimal hyperparameters the quickest. It is around 6 times faster than the no transfer method and around 3 times faster than the Efficient-BO [9]. This clearly demonstrate the usefulness of our proposed method for tuning hyperparameters.

Table 2. Time in CPU seconds to reach the best hyperparameter when 40 % of the whole training data is used as source.

Datasets	Elastic Net			SVM with RBF Kernel		
	Proposed Method	Efficient-BO	Generic-BO	Propsed Method	Efficient-BO	Generic-BO
Liver Disorders	**4.8**	12.8	30.3	**6.7**	2 3.3	54.4
Heart Diseases	**3.9**	11.3	27.8	**5.9**	19.3	49.7
Breast Cancer	**6.4**	13.8	33.8	**8.3**	23.7	62.9

In Fig. 3, we plot the total time taken to tune the hyperparameters by our proposed method with respect to the size of the source data. Plots for all three datasets are shown. When a smaller percentage of data is used as source, the source to target difference may be higher leading to more optimisation iterations for target. This amounts to higher computational demand. Increasing source percentage implies larger computational demand for source but decreasing computational requirement for target. In other extreme, a large source means time taken for evaluating source itself is very high. This leads to a nice 'U' shaped

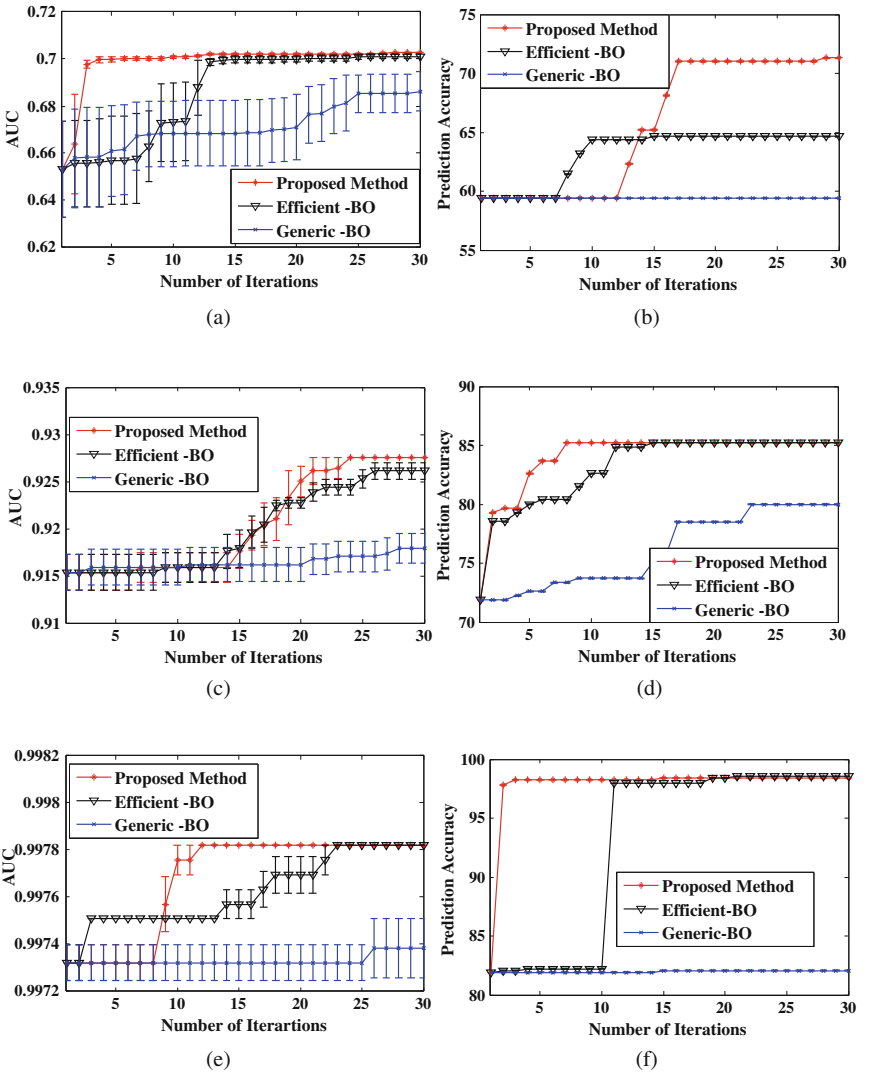

Fig. 2. Hyperparameter tuning experiment on three datasets at three rows (top: liver disorder, middle: heart disease and bottom: breast cancer): current best performance as a function of the iteration for three methods for Elastic Net (left column) SVM with RBF Kernel (right column).

efficiency curve for our proposed algorithm. The optimal lies in the middle and at around 40 % for all three datasets for both classifiers. For much larger data it may be possible to reach to the maximum even with a smaller fraction of data but from our experience 20–40 % is a good starting point.

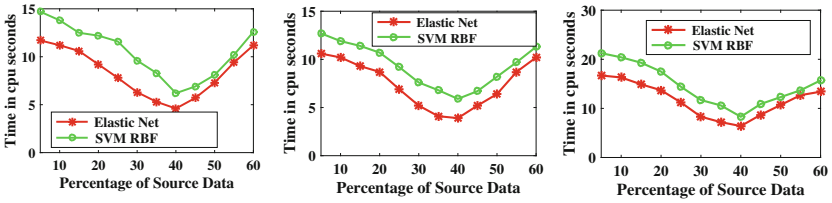

Fig. 3. Time taken to reach to the best hyperparameter vs source size as a percentage of the whole data: (a) Liver disorders, (b) Heart Diseases and (c) Breast cancer Datasets.

5 Conclusion

In this paper, we proposed a novel transfer learning framework for Bayesian optimisation. We model source task as a noisy observation of the target function and use the source observations to avoid the cold start problem for target task optimisation. The noise variance is estimated from data based on the data in a Bayesian setting. This enabled us to address the limitations of the existing methods that only work when tasks are closely related. The proposed method performs around 6 times faster than the generic Bayesian optimisation method and around 3 times faster than the current state-of-art on the task of tuning hyperparameters.

References

1. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp. 2951–2959 (2012)
2. Mockus, J.: Application of Bayesian approach to numerical methods of global and stochastic optimization. J. Global Optim. **4**(4), 347–365 (1994)
3. Lizotte, D.J., Wang, T., Bowling, M.H., Schuurmans, D.: Automatic gait optimization with Gaussian process regression. IJCAI **7**, 944–949 (2007)
4. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning (2010). arXiv preprint arXiv:1012.2599
5. Garnett, R., Osborne, M.A., Roberts, S.J.: Bayesian optimization for sensor set selection. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 209–219. ACM (2010)
6. Bergstra, J., Bardenet, R., Kégl, B., Bengio, Y.: Implementations of algorithms for hyper-parameter optimization. In: NIPS Workshop on Bayesian Optimization, p. 29 (2011)
7. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855. ACM (2013)

8. Bardenet, R., Brendel, M., Kégl, B., et al.: Collaborative hyperparameter tuning. In: Proceedings of the 30th International Conference on Machine Learning (ICML 2013), pp. 199–207 (2013)
9. Yogatama, D., Mann, G.: Efficient transfer learning method for automatic hyperparameter tuning. Transfer **1**, 1 (2014)
10. Williams, C.K., Rasmussen, C.E.: Gaussian Processes for Machine Learning, vol. 2, 3rd edn. The MIT Press, Cambridge (2006)
11. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J. Fluids Eng. **86**(1), 97–106 (1964)
12. Mockus, J., Tiesis, V., Zilinskas, A.: The application of bayesian methods for seeking the extremum. Towards Global Optim. **2**(117–129), 2 (1978)
13. Srinivas, N., Krause, A., Kakade, S., Seeger, M.: Gaussian process optimization in the bandit setting: no regret and experimental design. In: Proceedings of International Conference on Machine Learning (ICML) (2010)
14. Chang, C.-C., Lin, C.-J.: Libsvm: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011)