Gaussian Process Model Re-Use

CONFERENCE PAPER · SEPTEMBER 2015

READS
13

3 AUTHORS:

Tom Diethe
University of Bristol
48 PUBLICATIONS **76** CITATIONS
SEE PROFILE

Niall Twomey
University of Bristol
23 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Peter A. Flach
University of Bristol
224 PUBLICATIONS **4,445** CITATIONS
SEE PROFILE

Gaussian Process Model Re-Use

Tom Diethe, Niall Twomey, and Peter Flach

Intelligent Systems Laboratory, University of Bristol
{tom.diethe,niall.twomey,peter.flach}@bristol.ac.uk

Abstract. Consider the situation where we have some pre-trained classification models for bike rental stations (or any other spatially located data). Given a new rental station (deployment context), we imagine that there might be some rental stations that are more similar to this station in terms of the daily usage patterns, whether or not these stations are close by or not. We propose to use a Gaussian Process (GP) to model the relationship between geographic location and the type of the station, as determined by heuristics based on the daily usage patterns. For a deployment station, we then find the closest stations in terms of the Gaussian Process (GP) function output, and then use the models trained on these stations on the deployment station. We compare against several baselines, and show that this method is able to outperform those baselines.

1 Introduction

Automated tracking in "smart environments" (including smart homes and smart cities) is fast becoming an interesting area of research, with numerous research groups principal interests rooted in this area. Notable examples include CASAS[1] [1], the RUBICON project[2] [2], and SPHERE [3] [3,4].

There are at least two major challenges for machine learning in this setting. Firstly, the deployment context will necessarily be very different to the context in which learning occurs, due to both individual differences in typical daily patterns, and also due to sensor configurations, positioning, or layouts. Secondly, accurate labelling of training data is often an extremely time-consuming process, and the resulting labels are potentially noisy and error-prone.

From this we can see that the re-use of learnt knowledge, particularly in the form of models and model parameters, is of critical importance in the majority of knowledge-intensive application areas, particularly because of the expected deviations between the operating contexts in training and deployment. The REFRAME project[4] is particularly concerned with this issue, and is the sponsor of the Model Reuse with Bike rental Station data (MoReBikeS) challenge, which focuses on model reuse and context change.

[1] http://ailab.wsu.edu/casas/
[2] http://fp7rubicon.eu/
[3] http://www.irc-sphere.ac.uk
[4] http://reframe-d2k.org/

1.1 MoReBikeS Challenge

The task in this challenge is to predict the number of available bikes in every bike rental stations 3 hours in advance. There are at least two use cases for such predictions. Firstly, a user may plans to rent/return a bike in 3 hours time and wants to choose a bike station which is not empty or full. Secondly, the company operating the bike hire wants to avoid situations where a station is empty or full and therefore needs to move bikes between stations, meaning that they need to know which stations are more likely to be empty or full soon. In both these cases the prediction can be based on contextual information such as the time of the day, week, or year, or the current weather conditions, as well as information about the current status of the station. One might hypothesise that the quality of predictions would be the better given more historical information.

In this challenge there are 200 stations which have been running for > 2 years and 75 stations which have been open for a month. The task is to reuse the models learnt on 200 "old" stations in order to improve prediction performance on the 75 "new" stations, so the evaluation is based on these 75 stations. Pre-built classification models on the 200 stations are provided without the full training data (although full training data is available for 10 stations). Data for one month is provided, also to help in deciding how the models can be reused.

1.2 Daily patterns

In Figure 1 we can see that different stations clearly exhibit different daily patterns. Most obviously, there are stations that tend to be full in the night and emptier during the day, as in stations 26 and 28 (columns 1 and 3). Essentially these are stations that are on the outer areas of the city, and the bikes are used during the day to travel into more central parts of the city. There are also stations that exhibit the opposite pattern, such as station 29 (column 4). These stations are left empty at night, since the operators know that the will fill up during the day as people travel into the city. There are of course stations that fall between these two extremes, of which station 27 (column 2) is an example.

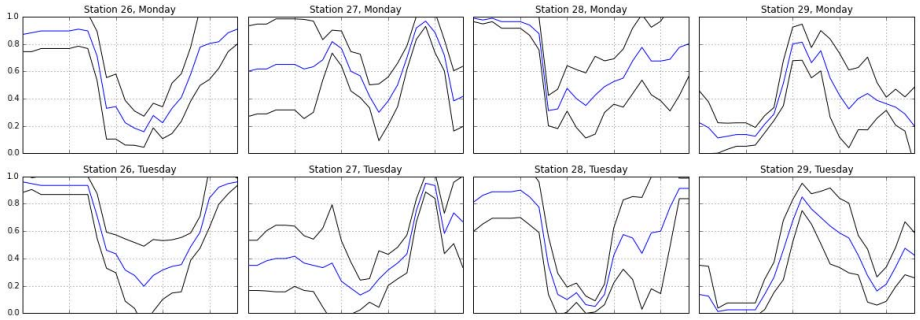


Fig. 1: Comparison of the daily patterns of several different stations. The blue line shows the mean of the daily capacity and the grey lines show $\pm$ one standard deviation.

Our proposal is that given a new station, if we are to use existing models trained on stations in known locations, the best predictions will be given by models trained on stations that exhibit a similar day/night pattern as the station in question. Note that these stations may be geographically close to each other, or they may be in a different part of the city but effectively play the same role. We will therefore leverage the locations of the deployment stations, but none of training data given in the challenge regarding the stations, simply in order to find the most similar stations.

2 Related Work

A major assumption in the majority of machine learning methods is that the training and deployment data are drawn from the same underlying distribution. For the smart-city application this assumption clearly does not hold. In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding the costly acquirement of labels. In recent years, transfer learning has emerged as a new learning framework to address this problem, and is related to areas such as domain adaptation, multi-task learning, sample selection bias, and covariate shift [5].

Definitions for domain and task have been provided by [6]. A domain $\mathscr{D}$ is a pair $(\mathscr{X}, p(\mathscr{S}))$. $\mathscr{X}$ is the feature space of $\mathscr{D}$ and $p(\mathscr{S})$ is the marginal distribution of a data-set. A task $\mathscr{T}$ is a pair $(\mathscr{Y}, f)$ for some given domain $\mathscr{D}$. $\mathscr{Y}$ is the label space of $\mathscr{D}$ and f is an unknown objective predictive function for $\mathscr{D}$ to be learnt from data, which here we write as the conditional probability distribution $p(y|\mathbf{x})$.

It is well known that the hierarchical Bayesian framework can be readily adapted to sequential decision problems [7], and it has also been shown more recently that it provides a natural formalisation of transfer learning [8]. The results of the latter of these show that a hierarchical Bayesian Transfer framework significantly improves learning speed when tasks are hierarchically related within the domain of reinforcement learning. In another study [9], the authors formulated a kernelised Bayesian transfer learning framework that is a combination of kernel-based dimensionality reduction models with task-specific projection matrices, and aims to find a shared subspace and a coupled classification model for all of the tasks in this subspace.

In our setting, since we were not the original creators of the models (*i.e.* pre-trained models have been provided), we do not have freedom to choose the overall framework in this way. However, we take the flavour of these ideas, and show how we can use kernelised Bayesian methods to re-use existing models.

3 Concepts and Notation

Here we briefly review the application of Gaussian Processes (GPs) for vector-valued regression, and then show how we will employ these for model re-use.

3.1 Gaussian Process for Vector-Valued Regression

In the vector-valued regression framework, we assume that we are given an example $\mathbf{x}_i \in \mathbb{R}^n$, and the goal is to make a prediction $\hat{\mathbf{y}}_i \in \mathbb{R}^o$ that approximates the true output

$\mathbf{y}_i \in \mathbb{R}^o$, where n is the dimensionality of the input space and o is the dimensionality of the output space. We will assume that we have m instances such that $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m$ and $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^m$.

A Gaussian Process (GP) is defined as a Gaussian distribution over latent functions. Given the consistency property of Gaussian distributions, whereby marginals are also Gaussian, point-wise evaluations at the data-points $\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_m)]^m$ are jointly Gaussian. By specifying a covariance function $\mathbf{C}(\mathbf{x}, \mathbf{x}')$ and a mean function $h(\mathbf{x})$, we see that $\mathbf{f} \sim \mathcal{N}(\mathbf{h}, \mathbf{K})$ where $\mathbf{h} = [h(\mathbf{x}_1, ..., \mathbf{x}_m)]$ and $\mathbf{K}$ is the covariance matrix which is equivalent to the Gram matrix in kernel methods (see subsection 3.2 below). Since the data-points only appear in the expression through the covariance matrix, and hence through inner products, any nonlinear mapping that produces valid covariances (kernels) can be used, as with other kernel methods. For regression, it is assumed that the targets are observed independently. The (intractable) posterior is then,

$$p(\mathbf{f}|\mathbf{X}, \mathbf{Y}) = \frac{1}{Z} p(\mathbf{f}|\mathbf{X}) \prod_{t=1}^{T} g(\mathbf{y}_t | f(\mathbf{x}_t)),$$

where the normalising constant $Z = p(\mathbf{Y}|\mathbf{X})$ is the marginal likelihood. Currently the most accurate deterministic approximation to this is through the use of Expectation Propagation (EP) [10]. In EP, the likelihood is approximated by an un-normalised Gaussian to give,

$$\begin{aligned} p(\mathbf{f}|\mathbf{X}, \mathbf{Y}) &= \frac{1}{Z_{EP}} p(\mathbf{f}|\mathbf{X}) \prod_{i=1}^{m} \frac{1}{Z_i} \tilde{g}(\mathbf{y}_i | f(\mathbf{x}_i)), \\ &= \frac{1}{Z_{EP}} p(\mathbf{f}|\mathbf{X}) \mathcal{N}(\mathbf{f}|\tilde{\mathbf{h}}, \tilde{\mathbf{C}}), \\ &= \mathcal{N}(\mathbf{f}|\mathbf{h}, \mathbf{C}), \end{aligned}$$

where Z_{EP} and Z_t are normalisation coefficients, $g(\mathbf{y}_i|\tilde{f}(\mathbf{x}_i))$ and $\mathcal{N}(\mathbf{f}|\tilde{\mathbf{h}}, \tilde{\mathbf{C}})$ are the Gaussian approximations to $g(\mathbf{y}_i|f(\mathbf{x}_i))$ at each site $\mathbf{x}_i$. In order to obtain the full (approximate) posterior $q(\mathbf{f}|\mathbf{X}, \mathbf{Y})$, one would start by using the prior, *i.e.* $q(\mathbf{f}|\mathbf{X}, \mathbf{Y}) = p(\mathbf{f}|\mathbf{X})$ and update each site approximation $\tilde{g}(\cdot)$ sequentially. In order to do this, the so-called cavity distribution $\tilde{q}(\mathbf{f}|\mathbf{X}, \mathbf{Y}_t)$ is used - *i.e.* the current posterior with the point $\mathbf{x}_i$ removed. In "online" methods (*c.f.* [11]) the full posterior approximation is only achieved after (at least one) full pass through the dataset.

GP inference in all cases was done using the GPy library for GP in python [12].

3.2 Kernel Methods

Briefly, a *kernel function* [13] is a function that for all $\mathbf{x}, \mathbf{z} \in \mathscr{X}$ satisfies $\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$, where $\phi : \mathscr{X} \mapsto \mathscr{H}$ is a mapping from $\mathscr{X}$ to an (inner product) Hilbert space $\mathscr{H}$. This allows inner products between *nonlinear* mappings (when $\phi(\cdot)$ is a nonlinear function), as long as the inner product $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ can be evaluated efficiently. In many cases, this inner product or kernel function can be evaluated much more efficiently than the feature vector itself, which can even be infinite dimensional

in principle. For a given kernel function the associated feature space is not necessarily unique. A commonly used kernel function for which this is the case is the Radial Basis Function (RBF) kernel, which is defined as:

$$\kappa_\gamma(\mathbf{x},\mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\gamma}\right). \tag{1}$$

The RBF kernel projects the data into a (non-unique) *infinite-dimensional Hilbert Space*. We will also use a Periodic Exponential (PE) kernel, which is defined as:

$$\kappa_{\gamma,\sigma,\omega}(x,z) = \sigma \exp\left(-2\cos\left(\frac{\pi\omega(x-z)}{\gamma}\right)\right), \tag{2}$$

where γ is the length-scale, σ is the amplitude, and ω is the period. Note that here x and z are scalars, as this kernel function is defined for scalar inputs only.

3.3 GPs for Model Re-Use

We now describe how GPs will be employed for model re-use. The key insights from subsection 1.2 are that different bike rental stations clearly exhibit different daily patterns. We need to characterise these different patterns in a way that can be provided as a target to the GP.

A rough way to characterise these differences is to simply assume that there is a periodic daily pattern which is sinusoidal, but varies in amplitude and phase. We propose therefore to fit a sinusoidal function to the first week of data of each of the training stations using a GP with a PE kernel, with ω fixed to 24 (the number of hours in the day. The input $\mathbf{X}$ to the GP then is simply $(24d)+h$ where d is the day and h is the hour of day, and the targets $\mathbf{Y}$ are the number of bikes (so in this case the dimensionality of the output space $o = 1$). For each of the stations, we then extract the amplitude and the phase as learnt by the GP.

Two examples of functions fitted (posterior means of the GP) in this manner are given in Figure 2. Note that the phase of the sinusoid for station 21 is almost the opposite of that of station 22, indicating that this has accurately captured the differences between these stations along this axis. In this figure the y-axis was scaled to the range $[0,1]$ which obscures any differences in amplitude, but we are interested in this value too: the phase indicates whether the station is one that is regularly "commuted to" or "commuted from", and the amplitude indicates the daily business of the station.

We then fit a further GP, using the decimal values of the latitude and longitude of the locations of the stations as the inputs $\mathbf{X}$, and the targets $\mathbf{Y}$ are the amplitude and the phase as learnt by the previous GP (so here the dimensionality of the output space $o = 2$). In Figure 3 we have plotted the functions fitted (posterior means of the GP) in each of the output dimensions, where Figure 3a shows the magnitude and Figure 3b shows the phase.

In order to decide which models we want to re-use, we need to characterise the similarity of a given deployment station to the training stations. Here we use the GP to compute the marginal likelihood of the magnitude and phase of the sinusoid of the daily pattern, given the station's location. We then compute the similarity between the

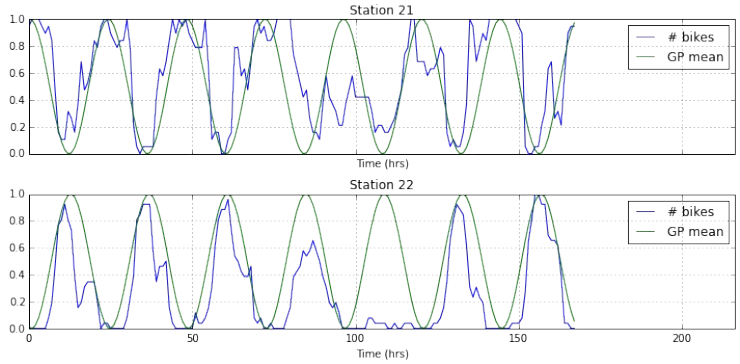


Fig. 2: Results of fitting the GP using a PE kernel to the hour of day of the bike rental data for two of the stations in Valencia.

marginal likelihood and the posterior means of the magnitude and phase of the sinusoidal functions of the daily patterns of the training stations. The similarity metric that we employ is again the RBF function as given in Equation 1. Since we are simply using this function to map differences in magnitude and phase to a weighting for the stations, we are free to choose the γ to adjust the effective number of stations to include in deployment. As a heuristic, we selected the γ that meant that 90% of the weight was given to the first 20 stations. Note that after rescaling such that they have the same range, here we treat the magnitude and the phase equally.

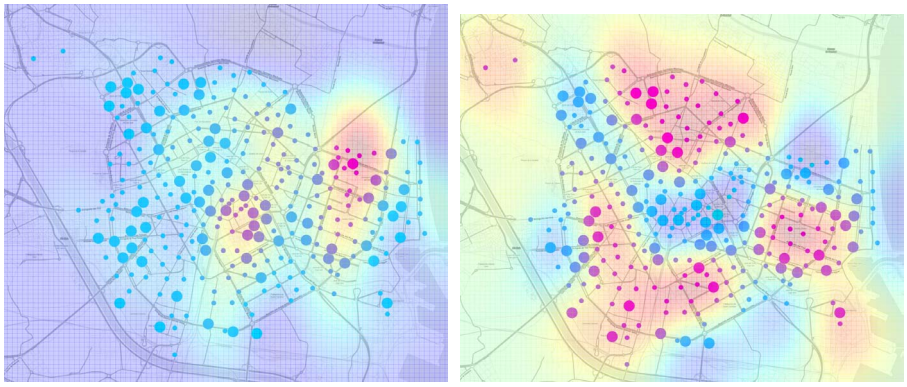
Figure 4 shows the similarity function as computed for two example deployment stations (221 and 222). Note that these stations are in some sense "opposite", in that their similarity patterns are almost inverted. Also note that the regions of highest similarity are not restricted to a single geographical region, and are not spherical in nature.

3.4 Baseline Method

As a baseline method, we also compute similarities between stations simply using the RBF function with latitudes and longitudes as inputs. This has the effect of putting a spherical Gaussian "blob" around the deployment station, and then choosing stations according distance in this space. Note that in many cases, the GP method will coincide with this baseline method, since close-by stations will often exhibit similar patterns. However the GP method also has the ability to select stations that are far away geographically, but still exhibit similar daily usage patterns. We note, however, that this is quite a strong baseline.

3.5 Base Models

For each training station there are 6 linear models, all built using R function rlm from the package MASS, with missing value imputation using function na.roughfix from package randomForest. The models use a subset of the following features (plus an intercept term):



(a) Magnitude (b) Phase

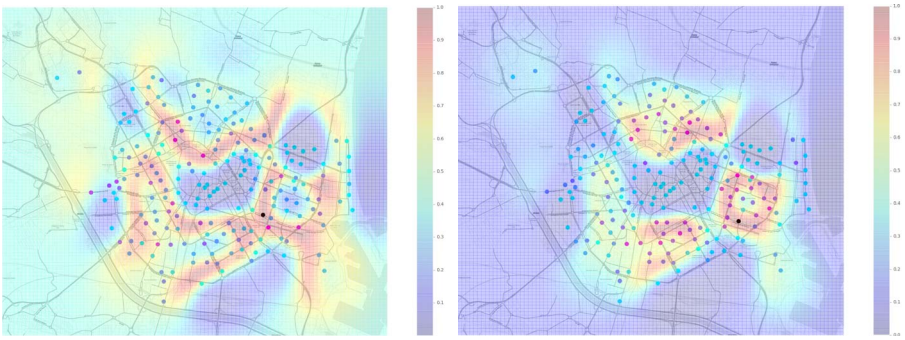
Fig. 3: Results of fitting the GP to the bike rental data of Valencia, where the inputs **X** are the station locations, and the targets **Y** are the magnitude and phase (show in seaparte sub-plots) of the sinusoidal functions fitted by the first GP, overlaid on a map of the area. Training stations are shown as small dots, and deployment stations as large dots, shaded by the GP function value at that location.

- bikes-3h-ago: number of bikes in the station 3 hours ago
- full-profile-bikes: arithmetic average of the target variable 'bikes' during all past time-points with the same week-hour, in the same station
- full-profile-3h-diff-bikes: arithmetic average of the calculated feature 'bikes-bikes-3h-ago' during all past time-points with the same week-hour, in the same station
- short-profile-bikes: arithmetic average of the target variable 'bikes' during the past 4 time-points with the same week-hour, in the same station
- short-profile-3h-diff-bikes: arithmetic average of the calculated feature 'bikes-bikes-3h-ago' during the past 4 time-points with the same week-hour, in the same position
- temperature.C: temperature in degrees C

In this study we chose only to use the last of the 6 models, that uses all of the features.

3.6 Combining Models for Classification

Once we have computed similarities between the deployment stations and all of the training stations, we still require a method for combining the trained models. Since all of the trained models have a linear functional form, we can take any convex combination of the outputs of the models and produce a valid (in terms of scaled outputs) model. We performed experiments using a uniform combination of the closest 20 models in terms of the RBF similarity, and also a convex combination using the RBF similarities as weights, scaled to sum to 1. Since there was little difference between the performance of these two methods, here we report only the results of using a uniform combination.



(a) Station 221 (b) Station 222

Fig. 4: Similarity of deployment stations 221 and 222, as computed using the RBF function with respect to the training stations (in terms of magnitude and phase of the fitted sinusoid), overlaid on a map of the area. The γ value for the RBF function as selected by the heuristic method was 0.1 in both cases. The deployment station is shown as a black dot, and the training stations are shown as dots coloured by the similarity.

4 Results

The task is to predict the number of bikes in the stations 3 hours in advance. The metric for performance we will be using is the mean absolute error (MAE), defined as follows:

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i| = \frac{1}{m}\sum_{i=1}^{m}|e_i|\,, \tag{3}$$

where $\hat{y}_i$ is the prediction and y_i the true value. We report the mean μ and standard deviation σ of the MAE over the 75 deployment stations in Table 1 below.

Table 1: mean absolute error (MAE) obtained by the three methods investigated.

	MAE	
	μ	σ
Gaussian Process	0.15	0.03
Euclidean Distance	0.15	0.04
Random Station	0.15	0.04

We note that the performance of the GP-based method and the baseline methods are basically equivalent. We surmise that there are two reasons for this. Firstly, we note that the base models are all fairly similar in terms of their performance. This is due to the fact that the base models only have a few features (see subsection 3.5) and are all linear models, so may not be capturing non-linearity (particularly in the changes in bike profile throughout the day and week).

5 Conclusions and Future Work

We have examined a situation where we have some pre-trained classification models for bike rental stations, for which we know the geographical location as well some information about the daily patterns of activity. Given a new rental station (deployment context), we imagine that there might be some rental stations that are more similar to this station in terms of the daily usage patterns, whether or not these stations are close by or not. We proposed to use two Gaussian Processes (GPs), the first to model the characteristic daily usage pattern of the station, and the second to model the relationship between geographic location and a characterisation of the type of the station, as determined by the first GP. For a deployment station, we then find the closest stations in terms of the GP function output, and then use the models trained on these stations on the deployment station. We compared against several baselines, and showed that this method is able to outperform those baselines.

As further work, we would like to consider different measures of the daily usage patterns, since the single sinusoid that we fit using the first GP is clearly a rough approximation. We could also imagine computing similarities based on other contextual information, such as the season, or day of week (weekday versus weekend). This could allow a more refined selection from the pool of training models. Furthermore, we would like to explore whether this method might work better with more varied base models (which we do not have access to here), such as non-linear GP, as we surmise that the base models are too similar to highlight the differences between the approaches.

Acknowledgement

This work was performed under the Sensor Platform for HEalthcare in Residential Environment (SPHERE) Interdisciplinary Research Collaboration (IRC) funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

1. Diane J Cook and M Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods Inf Med*, 48(5):480–5, 2009.
2. Davide Bacciu, Paolo Barsocchi, Stefano Chessa, Claudio Gallicchio, and Alessio Micheli. An experimental characterization of reservoir computing in ambient assisted living applications. *Neural Computing and Applications*, 24(6):1451–1464, 2014.
3. P. Woznowski, X. Fafoutis, T. Song, S. Hannuna, M. Camplani, L. Tao, A. Paiement, E. Mellios, M. Haghighi, N. Zhu, G. Hilton, D. Damen, T. Burghardt, M. Mirmehdi, R. Piechocki, D. Kaleshi, and I. Craddock. A multi-modal sensor infrastructure for healthcare in a residential environment. In *IEEE International Conference on Communications (ICC), Workshop on ICT-enabled services and technologies for eHealth and Ambient Assisted Living*, 2015.
4. N. Zhu, T. Diethe, M. Camplani, L. Tao, A. Burrows, N. Twomey, D. Kaleshi, M. Mirmehdi, P. Flach, and I. Craddock. Bridging eHealth and the internet of things: The SPHERE project. *IEEE Intelligent Systems*, 30, 2015.
5. Sinno Jialin Pan. Transfer learning. In *Data Classification: Algorithms and Applications*, pages 537–570. 2014.

6. Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.
7. Manfred Opper. On-line learning in neural networks. chapter A Bayesian Approach to On-line Learning, pages 363–378. Cambridge University Press, New York, NY, USA, 1998.
8. Aaron Wilson, Alan Fern, and Prasad Tadepalli. Transfer learning in sequential decision problems: A hierarchical Bayesian approach. In *Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011*, pages 217–227, 2012.
9. Mehmet Gönen and Adam A. Margolin. Kernelized Bayesian transfer learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1831–1839, 2014.
10. Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
11. Ricardo Henao and Ole Winther. PASS-GP: Predictive Active Set Selection for Gaussian Processes. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, 02 2010.
12. The GPy authors. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, 2012–2014.
13. John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, U.K., 2004.