Pattern Recognition Letters 108 (2018) 70–77

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

# Asymmetric kernel in Gaussian Processes for learning target variance

S.L. Pintea[a,*], J.C. van Gemert[a], A.W.M. Smeulders[b]

[a] Computer Vision Lab, Delft University of Technology, Delft, Netherlands
[b] Intelligent Sensory Information Systems, University of Amsterdam, Amsterdam, Netherlands

ARTICLE INFO

*Article history:*
Received 30 March 2017
Available online 1 March 2018

*Keywords:*
Gaussian process
Kernel metric learning
Asymmetric kernel distances
Regression

ABSTRACT

This work incorporates the multi-modality of the data distribution into a Gaussian Process regression model. We approach the problem from a discriminative perspective by learning, jointly over the training data, the target space variance in the neighborhood of a certain sample through metric learning. We start by using data centers rather than all training samples. Subsequently, each center selects an individualized kernel metric. This enables each center to adjust the kernel space in its vicinity in correspondence with the topology of the targets — a multi-modal approach. We additionally add descriptiveness by allowing each center to learn a precision matrix. We demonstrate empirically the reliability of the model.

## 1. Introduction

Departing from the standard Gaussian Process, we introduce a regression approach that incorporates the multi-modality of the data distribution. While in the Gaussian Process model we have a global kernel metric that is shared by all the samples [34], here we propose to define a set of training data centers considerably smaller than the number of training samples. Subsequently, we learn from the numerous training samples an individualized kernel metric per training data center. By doing so, we are able to use a smaller training kernel matrix computed only on the training data centers while retaining the descriptive power of the model. This is highly efficient at test-time as it limits the size of the kernel matrix.

In our method we introduce two main changes to the standard Gaussian Process regressor: (i) we define a number of centers over the training data, by clustering or sampling; (ii) we learn individual kernel metric parameters per data center, discriminatively through metric learning, giving rise to a multi-modal approach with an asymmetric kernel matrix. Fig. 1 illustrates these differences when comparing with the standard Gaussian Process trained on either data centers or on all samples: we use fewer samples in the kernel computation, while enhancing the descriptiveness of the model by learning individualized metrics per center. We additionally expand the kernel parameters to a precision matrix, also learned through metric learning per center, giving rise to a multivariate multi-modal approach.

The individual steps of our model are not specifically novel, yet their combination is what gives strength. Clustering of the data in Gaussian Processes has been previously proposed [30,40]. Asymmetric kernels have been studied in works such as [26,50]. While a multivariate lengthscale parameter has been used in [18,23,30] for improved descriptiveness. Here, we combine all these ideas into a new approach which is suitable for learning target data variance. If we consider each training sample to be a data center, and enforce that all samples share the same kernel metric, and assume a univariate lengthscale in the kernel metric, we recover the standard Gaussian Process definition. We evaluate the proposed approach by gradually enabling these changes: center-based Gaussian Process, univariate multi-modal asymmetric Gaussian Process, and multivariate multi-modal asymmetric Gaussian Process. The experiments validate our models on the regression datasets *So2* and *Temp* used in [42], the large scale *Airlines* dataset of Hoang et al. [14], and two realistic image datasets: UCSD [4] and VOC-2007 [9] for pedestrian and generic-object counting, respectively.

## 2. Related work

### 2.1. Mixtures of Gaussian Processes

Noteworthy work has been focusing on mixtures of Gaussian Processes [24,27,28,35,45,52]. In [45] a mixture of Gaussian Processes is proposed to effectively deal with large data. Meeds and Osindero [27], Rasmussen and Ghahramani [35] extend this idea to an infinite mixture of Gaussian Processes. Somewhat similarly, Li [25] splits the problem into subproblems in a divide-and-conquer fashion and solves each such problem in a Gaussian Process model. Yuan and Neubauer [52] proposes an elegant variational Bayesian

* Corresponding author.
*E-mail address:* S.L.Pintea@tudelft.nl (S.L. Pintea).

https://doi.org/10.1016/j.patrec.2018.02.026
0167-8655/© 2018 Elsevier B.V. All rights reserved.

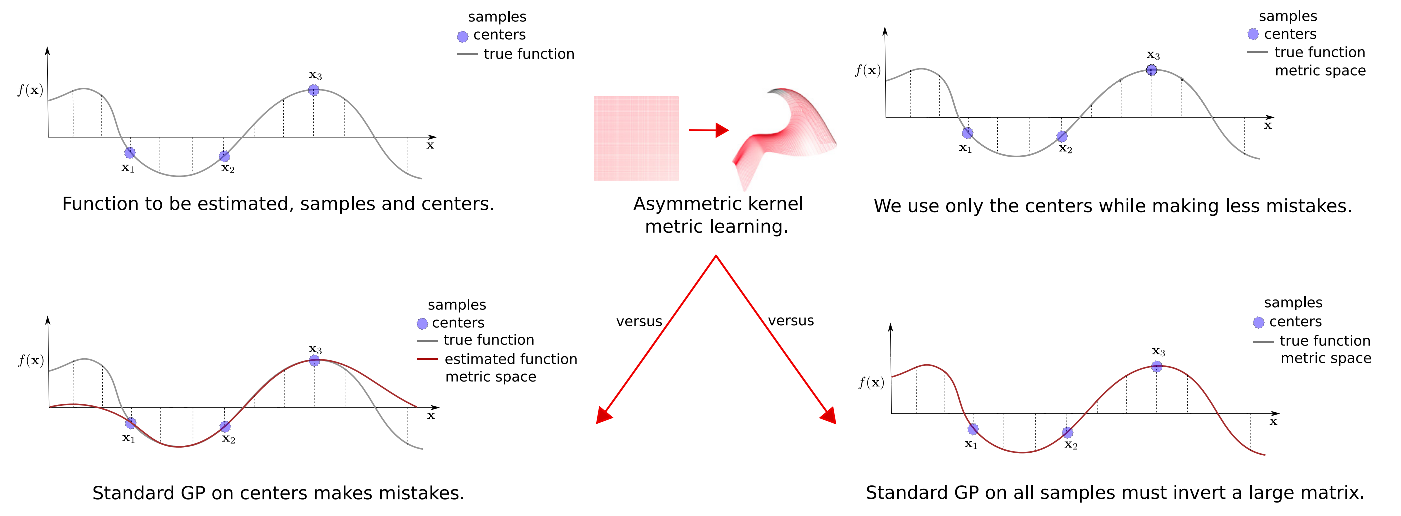

samples
centers
true function
$f(\mathbf{x})$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}$
Function to be estimated, samples and centers.
Asymmetric kernel metric learning.
metric space
We use only the centers while making less mistakes.
versus
versus
estimated function
Standard GP on centers makes mistakes.
Standard GP on all samples must invert a large matrix.

**Fig. 1.** An intuitive illustration of the proposed asymmetric kernel metric optimization, when compared with the standard Gaussian Process on data centers or on all samples. Given that each data center learns both a personalized size and shape of the kernel, we obtain a more descriptive model than the standard Gaussian Process, while using a limited number of data centers in the kernel matrix computation.

algorithm for training the mixture of Gaussian Process experts. An effective model is proposed in [24], where the authors simplify the mixture of experts so no gating function is used to assign samples to components and rather, trajectory clustering is employed. Nguyen and Bonilla [28] gracefully combines the mixture of Gaussian Process experts with the idea of inducing points, providing fast approximate Gaussian Process models. Unlike these works, where the final prediction entails a combination of predictions, each obtained within the metric space of individual components, we learn the hyper-parameters associated with each training data centers in a single Gaussian Process. We do so by employing an asymmetric kernel. Therefore, at test-time for an input test sample, rather than computing $N \times M$ kernel distances, where $N$ is the number of components and $M$ is the number of training samples, we only compute $N$ distances. In our case $N$ is the number of data centers, considerably smaller than $M$.

### 2.2. Efficiency in Gaussian Processes

The work of Quiñonero-Candela and Rasmussen [31] reviews the sparse approximations of Gaussian Process from a unified perspective by analyzing the implied prior of different methods. Csató and Opper [7] proposes learning iteratively, online, the sparse set of inducing points in a Bayesian formalism by minimizing the KL divergence. Inspired from metric learning techniques, Lawrence et al. [22] uses forward selection to obtain a sparse and time-efficient model. Snelson and Ghahramani [39] proposes a graceful solution of learning a set of sparse pseudo-inputs through gradient based optimization. A combination between sparse methods based on inducing points and local regression based on a multitude of experts describing locally the target space, is proposed in [40]. In [43,44] variational approaches are used to learn sparse representations. Titsias [43] jointly learns the inducing points and the kernel hyper-parameters by minimizing a lower bound through KL divergence. The robust method of Hensman et al. [13] decomposes the Gaussian Process model, variationally, such that it is factorized based on a set of global inducing variables. Bo and Sminchisescu [2], Ranganathan et al. [33] focus on iterative updates of the Gaussian Process. Rodner et al. [36] proposes the use of parameterized histogram intersection kernels to bypass the hyper-parameter estimation. Cao et al. [3] proposes a method to speed up the hyper-parameter estimation by inducing sparsity in the model. Somewhat similar to these methods, we only retain a set of data centers as informative training samples. Yet, unlike the above approaches, we subsequently add extra information into the Gaussian Process model by treating the data centers differently.

### 2.3. Descriptiveness in Gaussian Processes

Full matrices in the kernel definition have been proposed in [30,47], to make the model more descriptive. Here, we also learn precision matrices, in the kernel metric definition. However in our work, each center has an individualized precision matrix. Paciorek and Schervish [29] proposes nonstationary covariance matrices in the Gaussian Process model, tying the kernel metrics to the input samples. However, the final kernel matrix is symmetric as it is defined using symmetric combinations of per-sample covariances, similar to RVM (Relevance Vector Machine). Kersting et al. [18], Lázaro-Gredilla and Titsias [23] propose well-founded approaches to adding descriptiveness by extending the Gaussian Process definition to a heteroscedastic approach, by modeling the noise distribution to be dependent on the training data. Kuss et al. [21] proposes EP (Expectation Propagation) as an effective manner to train these models. Similarly, we also start with the assumption that the kernel metric should be data dependent and learn an individualized kernel metric. Titsias and Lázaro-Gredilla [42] proposes a compelling method for adjusting the kernel distances by assuming the data is mapped in a feature space based on the Mahalanobis kernel distance, estimated through variational inference. Unlike this work, we learn both the shape and the scale of the kernels per data center by minimizing the predictive loss.

### 2.4. Asymmetric kernels

The use of asymmetric kernel distances is not a recent idea but, rather, a well-matured topic [20,26,46,50]. In [46] asymmetric kernels are proposed in the context of SVM classification. Wua et al. [50] shows how similarity functions commonly used in real-life applications, can be related to asymmetric kernels, and gives a formal definition for the mathematical space described by asymmetric kernels. The work in [26] proposes asymmetric kernel regression in the context of neural networks and shows that such models are better behaved around the data boundaries. The recent work of Kulis et al. [20] learns asymmetric distances for visual domain adaptation in the context of object recognition. Similar to these methods, we also use asymmetric kernel distances, as these prove to have more descriptive power when limiting the number of samples in the training kernel computation.

### 2.5. Metric learning

Rahimi and Recht [32] learns a lower dimensional mapping of data while maintaining the distances between the data samples — the kernel distances remain approximately equal to the ones of the original features. In our work, we learn the kernel metric given the targets, rather than the feature representation. In [49] the kernel metric minimizes the leave-one-out regression error. Jain et al. [16] combines kernel learning with metric learning by employing a linear transformation. In this work, we use a fixed kernel — the squared exponential kernel, we additionally expand the parameters of the kernel to full precision matrices. Globerson and Roweis [11], Weinberger et al. [48], Xing et al. [51] represent pioneering work in the field of metric learning. Xing et al. [51] is the first paper to pose metric learning as a convex optimization problem learned from similar/dissimilar pairs of points. Globerson and Roweis [11] is one of the first works to propose Mahalanobis distances for metric learning. In [48] the Mahalanobis distance is learned in a nearest neighbor classifier, which induces a large-margin separation of classes. Kedem et al. [17], Kostinger et al. [19], Weinberger et al. [48] are recent works focusing on metric learning for classification with kernels, while [15] focuses on sparse kernel learning for regression. In this work we employ metric learning rather than estimating the optimal model hyper-parameters through marginal likelihood [34]. We do so, as each data center has an associated lengthscale in the proposed model and, thus, the marginal likelihood optimization is not straightforward in our case.

## 3. Asymmetric kernel for Gaussian Processes

We redefine the Gaussian Process model by allowing each training data center to learn an individualized kernel metric. This entails that the kernel matrix ceases to be symmetric in our case. However, this comes at extra gain in descriptive power, as despite using a small set of samples in the training kernel matrix, we optimize the individualized kernel metrics over the numerous available training samples.

### 3.1. Standard Gaussian Process revisited

We shortly revisit the standard Gaussian Process formulation, to unify the notations. The mean of the predictive distribution is [34]:

$$f(\mathbf{x}^*) = k(\mathbf{X}, \mathbf{x}^*)^T \left(k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}, \tag{1}$$

Where $\mathbf{x}^*$ represents an input test sample, $\mathbf{X}$ represents the training samples used for the training kernel matrix computation, $\mathbf{y}$ represents the training targets, $f(\mathbf{x}^*)$ is the prediction over the input $\mathbf{x}^*$ and $k(\cdot, \cdot)$ is the kernel metric used for estimating sample distances, and $\sigma$ is the noise hyper-parameter.

### 3.2. Center-based Gaussian Process

As Eq. (1) indicates, the training procedure requires the computation of the inverse of the training kernel-matrix, $k(\mathbf{X}, \mathbf{X})$, which is prohibitive on larger datasets. The first alteration of the Gaussian Process model that we investigate, is considering a set of data centers rather than individual training samples. We do so by either sampling the data or clustering it into a set of centers, $\overline{\mathbf{X}}$. Despite its simplicity, this is very effective in getting a fair overview over the variation in the training data while not having to use all samples during training.

More principled manners of defining data centers such as effective sampling techniques are possible. However, the focus here is not on the center definition, which is just meant as a first step towards reducing the size of the training kernel matrix. The strength of our model comes from allowing these centers to learn individualized metrics.

### 3.3. Multi-modal asymmetric kernel

Given that we have sparsified the training data by keeping only the training data centers, we lost information regarding the smoothness or variability of the target function in different regions of the data space. Therefore, we allow each training center, $\overline{\mathbf{x}}_i \in \overline{\mathbf{X}}$, to define individualized kernel metrics in its data neighborhood. The lengthscale hyper-parameter is the one defining the size of the kernel space, thus, we propose individualized lengthscale hyper-parameters, $l_i$ for each center $\overline{\mathbf{x}}_i$. This entails the second alteration of the standard Gaussian Process model. In this case the prediction function uses training and test kernel terms with per-center metrics:

$$f(\mathbf{x}^*) = \hat{k}(\overline{\mathbf{X}}, \mathbf{x}^*)^T \left(\hat{k}(\overline{\mathbf{X}}, \overline{\mathbf{X}}) + \sigma^2 I\right)^{-1} \mathbf{y}, \tag{2}$$

where $\hat{k}(\cdot, \cdot)$ is a non-symmetric kernel whose size depends on its corresponding training center — $\hat{k}(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j) = k_i(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j)$, and $\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j \in \overline{\mathbf{X}}$ are data centers. Thus, the distance from a training center to the others is computed within the associated kernel space of that center. At test-time $\hat{k}(\overline{\mathbf{X}}, \mathbf{x}^*) = (k_1(\overline{\mathbf{x}}_1, \mathbf{x}^*), k_2(\overline{\mathbf{x}}_2, \mathbf{x}^*), ..k_N(\overline{\mathbf{x}}_N, \mathbf{x}^*))$, where $N$ is the number of training centers, and $\mathbf{x}^*$ is a test sample. In this work we restrain our focus to the squared exponential kernel distance:

$$\hat{k}(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j) = k_i(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j) = \exp\left(-\frac{1}{2l_i^2}(\overline{\mathbf{x}}_i - \overline{\mathbf{x}}_j)(\overline{\mathbf{x}}_i - \overline{\mathbf{x}}_j)^T\right). \tag{3}$$

where $l_i$ is the lengthscale associated with data center $\overline{\mathbf{x}}_i$.

Given the use of individualized metrics per data center, the kernel ceases to be symmetric. Therefore, we can no longer employ the standard Cholesky decomposition for estimating the kernel-matrix inverse. We compute the kernel matrix inversion through SVD (Singular Value Decomposition). Despite this drawback, the individualized kernel metrics allow us to optimize the scale of the kernel locally, in the neighborhood of each data center.

### 3.4. Multivariate multi-modal asymmetric kernel

By allowing each center to define its own kernel metric, we change the model such that we can locally resize the kernel space to better map the target space. As highlighted in [6], not only the size of the kernel space is important but also the shape. Therefore, we also consider a multivariate extension of the model that allows for optimizing also the kernel shape per training center.

$$\hat{k}(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j) = k_i(\overline{\mathbf{x}}_i, \overline{\mathbf{x}}_j) = \exp\left(-\frac{1}{2}(\overline{\mathbf{x}}_i - \overline{\mathbf{x}}_j)\boldsymbol{P}_i(\overline{\mathbf{x}}_i - \overline{\mathbf{x}}_j)^T\right). \tag{4}$$

where $\boldsymbol{P}_i$ is the precision matrix associated with the data center $\overline{\mathbf{x}}_i$, to be learned from training data samples other than the ones defining data centers.

### 3.5. Kernel metric optimization

Standardly in the literature, the Gaussian Process hyper-parameters are learned through gradient methods by maximizing the marginal likelihood over the weights. Given that we aim to optimize a kernel metric, following the metric learning literature [6,11,48,51] we approach this problem discriminatively, and optimize the hyper-parameters with respect to the squared loss.

Thus, we learn the appropriate kernel metric for each data center, $\overline{\mathbf{x}}_i \in \overline{\mathbf{X}}$, from training samples other than the data centers, $\mathbf{x}_n \in \mathbf{X} \setminus \overline{\mathbf{X}}$. We add a regularization term to the squared loss weighted by $\mu$. We use the regularized squared loss over the targets as the function to be minimized and we employ SGD (Stochastic Gradient Descent) by estimating the gradients with respect to each per-center lengthscale, $l_i$ in the univariate case and $\mathbf{P}_i$ in the multivariate case.

$$\mathcal{L}(f, \mathbf{y}^*) = \begin{cases} \sum_{i=1}^{N} \left( \sum_{\mathbf{x}_n \in \mathbf{X} \setminus \overline{\mathbf{X}}} (f(\mathbf{x}_n) - y_n^*)^2 + \mu l_i^2 \right), & \text{if univariate;} \\ \sum_{i=1}^{N} \left( \sum_{\mathbf{x}_n \in \mathbf{X} \setminus \overline{\mathbf{X}}} (f(\mathbf{x}_n) - y_n^*)^2 + \mu \mid\mid \boldsymbol{P}_i \mid\mid \right), & \text{multivariate.} \end{cases} \tag{5}$$

We denote by $\mathbf{y}^*$ the training target vector composed of values $y_n^*$ for input training samples $\mathbf{x}_n$, where $\mathbf{x}_n \in \mathbf{X} \setminus \overline{\mathbf{X}}$ are not data centers, and $N$ is the number of data centers, $||\cdot||$ denotes the Frobenius norm in the multivariate case, and $f(\cdot)$ is the predictive function following Eq. (2). At each iteration we perform one gradient update step for all hyper-parameters, therefore, allowing them to be jointly learned.

### 3.5.1. Univariate multi-modal kernel optimization

The derivative of the loss with respect to the lengthscale per center, $l_i$, for the univariate case is given by the following formulation:

$$\frac{\partial \mathcal{L}(f, \mathbf{y}^*)}{\partial l_i} = \sum_{i=1}^{N} \left\{ \sum_{\mathbf{x}_n \in \mathbf{X} \setminus \overline{\mathbf{X}}} 2(f(\mathbf{x}_n) - y_n^*) \left[ \frac{\partial \hat{k}(\cdot, \mathbf{x}_n)}{\partial l_i} \alpha_i + \hat{k}(\cdot, \mathbf{x}_n) \left( -\hat{\mathbf{K}}^{-1} \frac{\partial \hat{\mathbf{K}}}{\partial l_i} \hat{\mathbf{K}}^{-1} \mathbf{y} \right) \right] + 2\mu l_i \right\}, \tag{6}$$

$$\frac{\partial \hat{k}(\cdot, \mathbf{x}_n)}{\partial l_i} = \begin{cases} \frac{1}{l_i^3} (\overline{\mathbf{x}}_j - \mathbf{x}_n)(\overline{\mathbf{x}}_j - \mathbf{x}_n)^T \hat{k}(\overline{\mathbf{x}}_j, \mathbf{x}_n), & \overline{\mathbf{x}}_j \in \overline{\mathbf{X}}, j = i; \\ 0, & \overline{\mathbf{x}}_j \in \overline{\mathbf{X}}, j \neq i. \end{cases} \tag{7}$$

$$\frac{\partial \hat{\mathbf{K}}}{\partial l_i} = \left( \frac{\partial \hat{k}(\cdot, \overline{\mathbf{x}}_m)}{\partial l_i} \right)_{\overline{\mathbf{x}}_m \in \overline{\mathbf{X}}}, \tag{8}$$

$$\hat{\mathbf{K}}^{-1} = \left( \hat{k}(\overline{\mathbf{X}}, \overline{\mathbf{X}}) + \sigma^2 \mathbf{I} \right)^{-1}, \tag{9}$$

$$\boldsymbol{\alpha} = \left( \hat{k}(\overline{\mathbf{X}}, \overline{\mathbf{X}}) + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{y}, \tag{10}$$

where we denote by $\overline{\mathbf{X}}$ the data centers, $\hat{\mathbf{K}}$ represents the asymmetric training kernel matrix, and $\mathbf{y}^*$ is a vector of training targets $y_n^*$ for training samples $\mathbf{x}_n \in \mathbf{X} \setminus \overline{\mathbf{X}}$ that are not data centers. The lengthscale hyper-parameters, $l_i$, are estimated per training data center rather than globally.[1]

### 3.5.2. Multivariate multi-modal kernel optimization

In the multivariate case we learn a precision matrix, $\boldsymbol{P}_i$, rather than a scalar lengthscale per data center, $\overline{\mathbf{x}}_i \in \overline{\mathbf{X}}$. Therefore, we have to ensure that the precision matrix learned is symmetric. For this, in the gradient computation, we apply the derivations for symmetric matrices.

$$\frac{\partial \mathcal{L}(f, \mathbf{y}^*)}{\partial \boldsymbol{P}_i} = \left[ \frac{\partial \mathcal{L}(f, \mathbf{y}^*)}{\partial \boldsymbol{P}_i} \right] + \left[ \frac{\partial \mathcal{L}(f, \mathbf{y}^*)}{\partial \boldsymbol{P}_i} \right]^T - \operatorname{diag}\left[ \frac{\partial \mathcal{L}(f, \mathbf{y}^*)}{\partial \boldsymbol{P}_i} \right] \tag{11}$$

[1] A simple univariate torch implementation can be found at: https://silvialaurapintea.github.io/code/gp.lua.

For gradient computation, the only change in the multivariate case is in the derivative of the kernels with respect to the per-center precision matrices, $\mathbf{P}_i$:

$$\frac{\partial \hat{k}(\cdot, \mathbf{x}_n)}{\partial \boldsymbol{P}_i} = \begin{cases} -\frac{1}{2} (\overline{\mathbf{x}}_j - \mathbf{x}_n)^T (\overline{\mathbf{x}}_j - \mathbf{x}_n) \hat{k}(\overline{\mathbf{x}}_j, \mathbf{x}_n), & \overline{\mathbf{x}}_j \in \overline{\mathbf{X}}, j = i; \\ 0, & \overline{\mathbf{x}}_j \in \overline{\mathbf{X}}, j \neq i. \end{cases} \tag{12}$$

$$\frac{\partial \hat{\mathbf{K}}}{\partial \boldsymbol{P}_i} = \left( \frac{\partial \hat{k}(\cdot, \overline{\mathbf{x}}_m)}{\partial \boldsymbol{P}_i} \right)_{\overline{\mathbf{x}}_m \in \overline{\mathbf{X}}}. \tag{13}$$

In this case, an additional cone projection step [48] is applied after each update to ensure that the precision matrix remains positive definite.

### 3.6. Model properties

We analyze the properties considered in [1], from the metric learning perspective:

- *Learning paradigm.* Fully supervised, as we learn from training data the best lengthscale hyper-parameters with respect to the $L_2$ prediction loss.
- *Form of metric.* Non-linear and local with respect to the predictive function. Different metrics are learned for different regions in the target space.
- *Scalability.* Scales with the number of data centers in the univariate case, because the kernel matrix is computed over the data centers. Thus, is more efficient than using all training samples. In the multivariate case, we learn a precision matrix, and the method scales also with the number of data dimensions, making it more difficult to optimize.
- *Optimality of the solution.* Our learning formulation, given in Eqs. (6) and (11), is a non-convex function with respect to the hyper-parameters and a global optimum is not guaranteed. For this reason we use an additional validation set during training on which we select among the local optima.

## 4. Illustrative results

Fig. 2 illustrates the need for the asymmetric model. If all samples share the same kernel metric, Fig. 2(ii) and (iii) would not be possible. When restricting our attention to few training samples, we lose the information of how the target function varies between the samples, yet the per-center kernel metrics help recover this information. In this illustration, we fix the training centers to the centers of the two ellipses. The [0, 1]-normalized pixel coordinates are the input features, and the pixel intensities are the targets. We visualize three models: (i) *center-GP* — standard GP on the 2 data centers, and the optimal lengthscale hyper-parameter is found through cross validations over 100 randomly sampled training pixels; (ii) *univariate-AGP* — using Eq. (3) with the optimal lengthscale per center estimated as in Section 3.5.1, and (iii) *multivariate-AGP* — using Eq. (4). In Fig. 2 the univariate asymmetric model estimates better the sizes of the two blobs, when compared to its center-based counterpart, while the multivariate asymmetric model has the lowest error, as it learns both the sizes and the shapes of the blobs. We also show the training and validation losses in Fig. 2(iv). We additionally analyze the consistency of our approach, numerically. For this, we sample on a uniform grid a standard 1$D$ Gaussian Process with a fixed lengthscale set to 13.5. The aim is to test if the lengthscales estimated by our method tend to the true lengthscale of the Gaussian Process from which we have sampled, as we add more training data. We use two data centers for the kernel computation. Fig. 3 shows in red the Euclidian distance between the two estimated hyper-parameters by

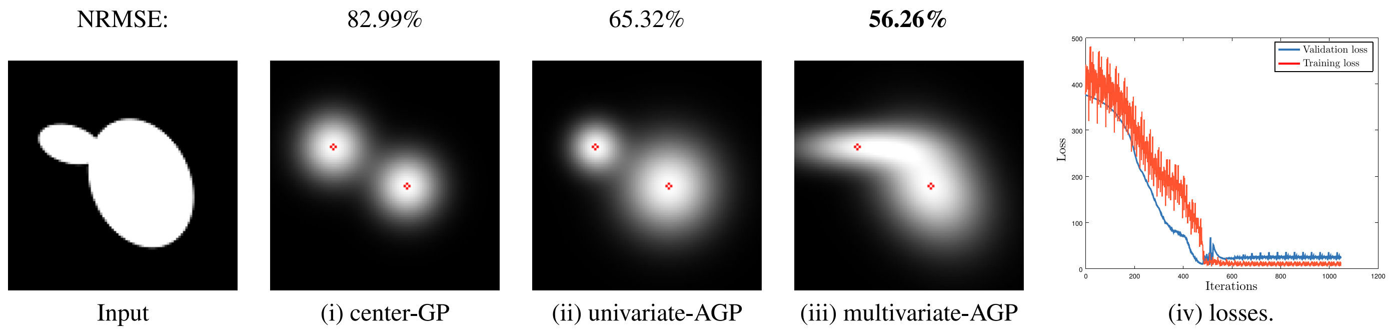


**Fig. 2.** Pixel intensity prediction from input normalized pixel location: (i) center-GP — standard GP trained on training centers; (ii) univariate AGP — proposed asymmetric model using per-center univariate lengthscale in the kernel — Eq. (3); (iii) multivariate AGP — proposed asymmetric model using per-center multivariate legnthscale — Eq. (4); (iv) losses — training and validation losses on this data.

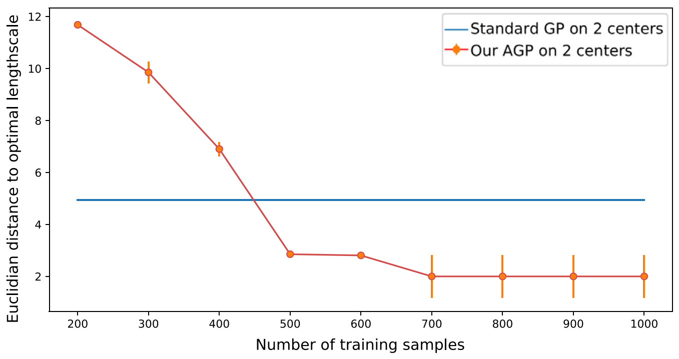


**Fig. 3.** Numeric consistency: In red, the Euclidian distance between the estimated hyper-parameters by our method, using two data centers, and the optimal hyper-parameter of the Gaussian Process we have sampled from. We plot standard deviation over 3 repetitions. In blue, the distance between the optimal hyper-parameter and the one of the standard GP on the two centers. With more data, our estimated lengthscales approach the optimal value. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 1**
Datasets statistics.

| Baseline | #Trainval | #Test | #Features |
|---|---|---|---|
| Temp [42] | 7117 | 3558 | 106 |
| So2 [42] | 15,304 | 7652 | 27 |
| Airlines [14] | 2,055 K | 102 K | 8 |
| UCSD [4] | 1200 | 2800 | 1000 |
| VOC-2007 [5] | 5011 | 4952 | 1000 |

our method and the optimal one. We report standard deviation over 3 repetitions. In blue, we show the distance between the optimal hyper-parameter and the one estimated by the standard GP through grid search, using two data centers. With more training samples, our estimated lengthscale parameters approach the optimal value. The distance does not converge to 0 as we use only two data centers in the training kernel matrix.

## 5. Experiments

### 5.1. Experimental setup

**Data splitting and center selection.** Table 1 depicts the specifics of each dataset used. Each dataset is split into trainval and test, following the standard way. Given the non-convexity of the problem, we evaluate the hyper-parameters on a small validation set after each training epoch, and keep the best. For all datasets the validation set is obtained as 100 random samples taken from the trainval dataset. These samples are not used for defining the data centers or for training data statistics. Different center selection approaches are considered in Section 5.2. We standardize the data to zero mean and unit variance per dimension, by extracting statistics over the training data excluding the validation set.

**Parameter setting.** In the SGD, the initial learning rate is set by looking at the plots of validation and training losses during training. We use a starting learning rate of $1.0e-5$ for *So2, Temp* and *Airlines* datasets, and an initial learning rate of $1.0e-11$ for *UCSD* and *VOC-2007*, where the data dimensionality is considerably higher. We use batch-SGD with mini-batches of 64 randomly selected training samples. Given the non-convexity of the solved problem, we make use of momentum and set it to 0.9 as advised in [41]. The regularization term in the loss function, $\mu$, is set to $1.0e-5$. For the standard GP models as well as for *center-GP* — Gaussian Process trained on data centers only — we estimate the model hyper-parameters by performing grid-search and evaluating on the validation samples. We use the same procedure for initializing our per-center lengthscales, before optimizing them in the SGD.

**Evaluation metric.** For comparison with existing work we report MSE (Mean Squared Error), RMSE (Root Mean Squared Error), or NRMSE (Normalized Root Mean Squared Error) defined as:

$$\text{NRMSE}(\mathbf{y}, \mathbf{y}^*) = \sqrt{\frac{1}{N}\sum_{n}^{N}\frac{(y_n - y_n^*)^2}{\text{var}(\mathbf{y}_{\text{train}})}}, \tag{14}$$

where $\text{var}(\mathbf{y}_{\text{train}})$ is the label variance on the training data, $\mathbf{y}^*$ are the test targets, and $\mathbf{y}$ the predictions.

**Table 2**
The effect of the center selection method when considering: K-means, random sampling, spectral clustering and GMM center definition on the *Temp* dataset.

| # Centers | NRMSE scores | | | |
|---|---|---|---|---|
| | K-Means | Sampling | Spectral | GMM |
| 10 | 0.602 | 0.557 | 0.579 | 0.606 |
| 50 | 0.482 | 0.467 | 0.482 | 0.523 |

### 5.2. Center selection

Here we analyze the effect of the center selection method on the overall performance of our method. For this we use the *Temp* dataset which has 106 dimensions per sample. We consider four center selection methods: K-Means, random sampling, spectral clustering and GMM (Guassian Mixture Model). We test on our *univariate-AGP* — using univariate individualized lengthscales — with 10 and 50 centers, respectively.

Table 2 indicates that the choice of the centers is not essential as all methods perform similar. Given that the strength of the

**Table 3**
Evaluation of the proposed models — *univariate-AGP* and *multivariate-AGP* on the large scale dataset used in [14]. Our proposed models are trained on 50 data centers. We compare our results with the methods analyzed in [14]: PIC, FITC, DTC.

| AGP | | PIC | FITC | DTC |
|---|---|---|---|---|
| Univariate | Multivariate | | | |
| **30.093** (± 2.285) | **30.805** (± 2.950) | 33.351 | 39.530 | 39.531 |

model is in learning individualized hyper-parameters and less in the method used for defining centers, in our subsequent experiments we rely on k-means clustering.

### 5.3. Multi-modal approach evaluation

Table 4 depicts the results of our approaches on the *So2* and *Temp* datasets when compared to Titsias and Lázaro-Gredilla [42] and with the standard Gaussian Process model. The gain brought by the multi-modal asymmetric methods over the center-based Gaussian Process and the standard Gaussian Process is more obvious for the *Temp* dataset. This can be explained by the larger number of dimensions to learn from, in the multivariate case. On both datasets, the proposed models outperform [42], while using only 50 data centers.

The performance decreases slightly with the increase in the number of data centers for the multivariate asymmetric models. This is due to the model being trained for the same number of iterations as the univariate case, while having to learn a larger number of parameters — a precision matrix of size $D \times D$. With the increase in data dimensionality, the multivariate model becomes considerably slower and harder to optimize. This represents a drawback of the proposed multivariate approach. The variability in the target space also affects the ease with which a good solution is found.

### 5.4. State-of-the-art comparison

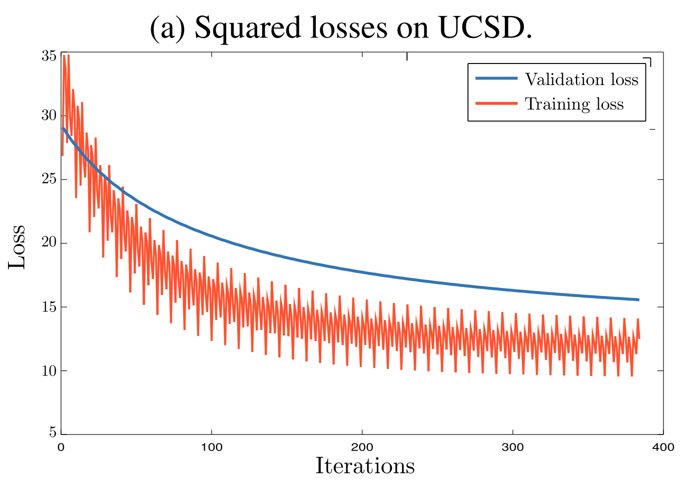
#### 5.4.1. Large scale regression problem

Here, we consider a more challenging task where the number of training samples is markedly high. We compare against effective state-of-the-art methods that focus on the same problem as we do — representing the data using an informative subset while retaining the descriptiveness of the model [14]. We use the *Airlines* dataset of Hoang et al. [14] containing over 2,000,000 training samples. The models considered are: DTC (Deterministic Training Conditional) [38], FITC (Fully Independent Training Conditional) [39], and PIC (Partially Independent Conditional) [40]. For evaluating our models we repeat each experiment three times and report the mean RMSE (Root Mean Squared Error) together with the standard deviation. Table 3 shows that our proposed *AGP* models perform well when dealing with a prohibitive number of training samples. Our approach outperforms existing methods [38–40] while using only a limited number of data centers.

#### 5.4.2. Realistic data: counting from images

We test our regression approach on two realistic image datasets — UCSD [4] and VOC-2007 [9] — for people and generic object counting, respectively. Given the image data, we rely on deep learning features. We extract 1,000 dimensional features as the output of the fully-connected layer of the ResNet-50 [12] pretrained on ImageNet [37]. For computational efficiency here we use only the univariate version of our approach with 25 centers, since the multivariate version requires optimizing a precision matrix of size $1,000 \times 1,000$.

**Pedestrian counting.** Fig. 4 shows the results of our approach on the realistic problem of pedestrian counting from images on the UCSD dataset, together with the training and validation losses.



(b) MSE on UCSD.

| | |
|---|---|
| Dalal and Triggs (2005) | 39.75 |
| Felzenszwalb et al. (2008) | 24.72 |
| Chan and Vasconcelos (2012) | 9.95 |
| AGP-10 Univariate | 16.37 (± 0.41) |
| AGP-25 Univariate | 13.90 (± 0.05) |

**Fig. 4.** (a) MSE results on the UCSD pedestrians counting dataset when compared with three prior works [4,8,10]. We obtain comparable performance with prior work, though we use only global deep learning features, while [4] relies on motion segmentation masks. (b) The training and validation squared losses on the UCSD pedestrian counting dataset.

We compare with [8] that uses low level image features, [10] relying on a person detection method specifically trained for the task, and [4] which employs motion segmentation masks. Unlike these methods, we do not use either motion segmentation masks or class specific detectors. We use only global image features extracted from a pretrained deep network, and we manage to obtain comparable performance with [4], while greatly outperforming [8,10].

**Generic object counting.** In Table 5 the goal is generic object counting on the VOC-2007 generic object dataset. We compare with the set of models proposed in the very recent deep learning method of [5]. Our features are extracted from a pretrained deep learning model, while theirs are specifically fine-tuned for this counting task. We outperform their *"glance"* models, which similar to us, rely on global image features. We additionally obtain comparable performance to *aso-sub-ft-1L-3* $\times$ *3* and *seq-sub-ft-3* $\times$ *3* which rely on local image information, as they divide each image into a $3 \times 3$ grid and extract features from each cell. It is worthwhile noting that we use only 25 data centers for computing the kernel matrix, and achieve comparable performance with methods relying on stronger features. These results support our approach.

## 6. Conclusions

This work brings forth an asymmetric kernel for the Gaussian Process model. This encompasses three components: (i) training on training centers only, (ii) learning individualized kernel metrics per center and, (iii) extending the lengthscale hyper-parameter to a precision matrix, thus learning not only the appropriate size but also the shape in the kernel metric. Due to the limitations imposed by the dependency between per-center hyper-parameters, we dis-

**Table 4**
NRMSE on the *So2* and *Temp* datasets for the 3 methods: *center-GP* — trained on training centers only, *univariate-AGP* — the univariate asymmetric model using the kernel metrics defined in Eq. (3) and, *multivariate-AGP* — the multivariate asymmetric model with kernel metrics as defined in Eq. (4). Results compared with Titsias and Lázaro-Gredilla [42] and *GP* — standard GP trained on randomly sampled examples. We show in bold the results outperforming the baseline and underline the best result.

(a) *So2* data evaluation.

| | # Samples | NRMSE |
|---|---|---|
| Titsias and Lázaro-Gredilla [42] | 100 | 1.004 |
| GP | 100 | 0.985 |
| center-GP | 50 | **0.984** |
| univariate-AGP | 50 | **0.846** |
| multivariate-AGP | 50 | **0.863** |
| center-GP | 10 | 0.985 |
| univariate-AGP | 10 | **0.818** |
| multivariate-AGP | 10 | **<u>0.808</u>** |

(b) *Temp* data evaluation.

| | # Samples | NRMSE |
|---|---|---|
| Titsias and Lázaro-Gredilla [42] | 100 | 0.489 |
| GP | 100 | 0.533 |
| center-GP | 50 | 0.559 |
| univariate-AGP | 50 | **0.482** |
| multivariate-AGP | 50 | **<u>0.445</u>** |
| center-GP | 10 | 0.642 |
| univariate-AGP | 10 | 0.602 |
| multivariate-AGP | 10 | 0.493 |

**Table 5**
RMSE results on the VOC-2007 general object counting dataset. The first two "glance" models use global image features learned in a deep learning framework, which is similar to us. The last two models use local information by dividing the image into a $3 \times 3$ grid and extracting deep learning features from each cell. Our method outperforms the models relying on global image features.

| | |
|---|---|
| Chattopadhyay et al. [5] glance-noft-2L | 0.50 ($\pm$ 0.02) |
| Chattopadhyay et al. [5] glance-sos-2L | 0.51 ($\pm$ 0.02) |
| Chattopadhyay et al. [5] aso-sub-ft-1L-$3 \times 3$L | 0.43 ($\pm$ 0.01) |
| Chattopadhyay et al. [5] seq-sub-ft-$3 \times 3$ | 0.42 ($\pm$ 0.01) |
| AGP-25 Univariate | 0.43 ($\pm$ 0.002) |

criminatively solve the problem through metric learning. Individualized kernel metrics entail the loss of the symmetry in the kernel matrix. However, this has the gain of better describing the target function in the neighborhood of the center points, used for kernel computation.

## Acknowledgements

Research supported by the Dutch national program COMMIT. Thanks to dr. Mijung Park and dr. Marco Loog for the insightful suggestions and advice.

## References

[1] A. Bellet, A. Habrard, M. Sebban, Metric learning, Syn. Lect. Artif. Intell. Mach. Learn. 9 (1) (2015) 1–151.
[2] L. Bo, C. Sminchisescu, Greedy block coordinate descent for large scale gaussian process regression, CoRR abs/1206.3238 (2012).
[3] Y. Cao, M.A. Brubaker, D. Fleet, A. Hertzmann, Efficient optimization for sparse Gaussian process regression, in: NIPS, 2013, pp. 1097–1105.
[4] A.B. Chan, N. Vasconcelos, Counting people with low-level features and Bayesian regression, TIP 21 (4) (2012) 2160–2177.
[5] P. Chattopadhyay, R. Vedantam, R. Selvaraju, D. Batra, D. Parikh, Counting everyday objects in everyday scenes, CVPR (2017).
[6] L. Chen, A. Buja, Stress functions for nonlinear dimension reduction, proximity analysis, and graph drawing, JMLR 14 (1) (2013) 1145–1173.
[7] L. Csató, M. Opper, Sparse on-line gaussian processes, Neural Comput. 14 (3) (2002) 641–668.
[8] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, 1, 2005, pp. 886–893.
[9] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The PASCAL visual object classes challenge 2007 (VOC2007) results, http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
[10] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: CVPR, 2008, pp. 1–8.
[11] A. Globerson, S.T. Roweis, Metric learning by collapsing classes, NIPS, 2005.
[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.
[13] J. Hensman, N. Fusi, N.D. Lawrence, Gaussian processes for big data, CoRR abs/1309.6835 (2013).
[14] T.N. Hoang, Q.M. Hoang, K.H. Low, A unifying framework of anytime sparse Gaussian process regression models with stochastic variational inference for big data, in: ICML, 2015, pp. 569–578.
[15] R. Huang, S. Sun, Kernel regression with sparse metric learning, J. Intell. Fuzzy Syst. 24 (4) (2013) 775–787.
[16] P. Jain, B. Kulis, J.V. Davis, I.S. Dhillon, Metric and kernel learning using a linear transformation, JMLR 13 (1) (2012) 519–547.
[17] D. Kedem, S. Tyree, F. Sha, G.R. Lanckriet, K.Q. Weinberger, Non-linear metric learning, in: NIPS, 2012, pp. 2573–2581.
[18] K. Kersting, C. Plagemann, P. Pfaff, W. Burgard, Most likely heteroscedastic Gaussian process regression, in: NIPS, 2007, pp. 451–458.
[19] M. Kostinger, M. Hirzer, P. Wohlhart, P.M. Roth, H. Bischof, Large scale metric learning from equivalence constraints, in: CVPR, 2012, pp. 2288–2295.
[20] B. Kulis, K. Saenko, T. Darrell, What you saw is not what you get: Domain adaptation using asymmetric kernel transforms, in: CVPR, 2011, pp. 1785–1792.
[21] M. Kuss, T. Pfingsten, L. Csató, C.E. Rasmussen, Approximate inference for robust Gaussian process regression, Technical Report, Max Planck Institute on Biological Cybernetics, Tubingen, Germany, 2005.
[22] N. Lawrence, M. Seeger, R. Herbrich, Fast sparse gaussian process methods: the informative vector machine, in: NIPS, 2003, pp. 609–616.
[23] M. Lázaro-Gredilla, M.K. Titsias, Variational heteroscedastic Gaussian process regression, in: ICML, 2011, pp. 841–848.
[24] M. Lázaro-Gredilla, S. Van Vaerenbergh, N.D. Lawrence, Overlapping mixtures of gaussian processes for the data association problem, Pattern Recognit. 45 (4) (2012) 1386–1395.
[25] W. Li, Gaussian process learning: a divide-and-conquer approach, in: ANN, 2014, pp. 262–269.
[26] M. Mackenzie, A.K. Tieu, Asymmetric kernel regression, Trans. Neural Networks 15 (2) (2004) 276–282.
[27] E. Meeds, S. Osindero, An alternative infinite mixture of Gaussian process experts, in: NIPS, 18, 2006, p. 883.
[28] T. Nguyen, E. Bonilla, Fast allocation of Gaussian process experts, in: ICML, 2014, pp. 145–153.
[29] C. Paciorek, M. Schervish, Nonstationary covariance functions for Gaussian process regression, in: NIPS, 16, 2004, pp. 273–280.
[30] S.L. Pintea, J.C. van Gemert, A.W.M. Smeulders, Large scale Gaussian process for overlap-based object proposal scoring, CVIU (2016).
[31] J. Quiñonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, JMLR 6 (2005) 1939–1959.
[32] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: NIPS, 2007, pp. 1177–1184.
[33] A. Ranganathan, M.H. Yang, J. Ho, Online sparse Gaussian process regression and its applications, TIP 20 (2) (2011) 391–404.
[34] C.E. Rasmussen, Gaussian processes for machine learning, Citeseer, 2006.
[35] C.E. Rasmussen, Z. Ghahramani, Infinite mixtures of Gaussian process experts, in: NIPS, 2, 2002, pp. 881–888.
[36] E. Rodner, A. Freytag, P. Bodesheim, J. Denzler, Large-scale Gaussian process classification with flexible adaptive histogram kernels, in: ECCV, 2012, pp. 85–98.
[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, IJCV 115 (3) (2015) 211–252.
[38] M. Seeger, C. Williams, N. Lawrence, Fast forward selection to speed up sparse Gaussian process regression, AISTATS, 2003.
[39] E. Snelson, Z. Ghahramani, Sparse gaussian processes using pseudo-inputs, in: NIPS, 2005, pp. 1257–1264.
[40] E. Snelson, Z. Ghahramani, Local and global sparse gaussian process approximations, in: AISTATS, 2007, pp. 524–531.
[41] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: ICML, 2013, pp. 1139–1147.
[42] M. Titsias, M. Lázaro-Gredilla, Variational inference for mahalanobis distance metrics in Gaussian process regression, in: NIPS, 2013, pp. 279–287.
[43] M.K. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: AISTATS, 2009, pp. 567–574.
[44] M.K. Titsias, N.D. Lawrence, Bayesian Gaussian process latent variable model, in: AISTATS, 2010, pp. 844–851.
[45] V. Tresp, Mixtures of Gaussian processes, in: NIPS, 2000, pp. 654–660.
[46] K. Tsuda, Support vector classifier with asymmetric kernel functions, European Symposium on Artificial Neural Networks, 1999.
[47] F. Vivarelli, C.K.I. Williams, Discovering hidden features with Gaussian processes regression, in: NIPS, 1999, pp. 613–619.

[48] K.Q. Weinberger, J. Blitzer, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, JMLR 10 (2009) 207–244.
[49] K.Q. Weinberger, G. Tesauro, Metric learning for kernel regression, in: AISTATS, 2007, pp. 612–619.
[50] W. Wua, J. Xub, H. Lib, S. Oyamac, Asymmetric kernel learning, 2010.
[51] E.P. Xing, M.I. Jordan, S. Russell, A.Y. Ng, Distance metric learning with application to clustering with side-information, NIPS 15 (2002) 505–512.
[52] C. Yuan, C. Neubauer, Variational mixture of Gaussian process experts, in: NIPS, 2009, pp. 1897–1904.