

Overlapping Domain Cover for Scalable and Accurate Regression Kernel Machines

Supplementary Materials

Mohamed Elhoseiny, Ahmed Elgammal
m.elhoseiny@cs.rutgers.edu, elgammal@cs.rutgers.edu
Computer Science Department Rutgers University
New Jersey, USA

Contents

Lemma 4.1 Proof	1
More Figures and Results	3
Local Kernel Machines hyper-parameters on each dataset	3
Poser Dataset	3
HumanEva Dataset	4
Human 3.6 Dataset	5
Equal Size Kmeans (EKmeans): More Details	5
Iterative Minimum-Distance Assignments (IMDA) k-means EKmeans . . .	5
Comparison between IMDA and Assign and Balance (AB) EKmeans(presented in the paper)	5
More figures on AB Ekmeans	5
Equal Size Assignment Algorithms Pseudocode	6
Overlapping Domain Cover(ODC) Generation-Algorithm	7
IWTGP integration under ODC-Framwork	7
Training	9
Prediction	9
IWTGP-ODC Experiments	10

Lemma 4.1 Proof

Lemma 4.1. Under ODC notion, as the overlap p increases, the closer the nearest model to an arbitrary test point and the more likely that model get trained on a big neighborhood of the test point.

Proof. We start by outlining the main idea behind the proof, which is directly connected to the fact that $K = N/(1 - p)M$, which indicates that the number of local

models increases as p increases given fixed N and M . Under the assumption that the local models are spatially cohesive, $p \rightarrow 1$ theoretically indicates that there is a local model centered at each point in the space (*i.e.* $K = \infty$). Hence, as p increases, the distribution of the kernel machines is the finest and the more likely a test point to find the closest kernel machines trained on a big neighborhood of it leading to more accurate prediction. Meanwhile, as p goes to 0, the distribution is the coarsest and the less likely a test point finds, the closest kernel machines, trained on a big neighborhood.

Let's assume that each kernel machine is defined on M points that are spatially cohesive, covering the space of N points with $\frac{N}{(1-p)M}$. Let's assume that center of the M points in kernel machine i is μ_i , the the Co-variance matrix of these points are Σ_i . Hence

$$\begin{aligned} p(\mathbf{x}|D_i) &= \mathcal{N}(\mu_i, \Sigma_i) \\ &= (2\pi)^{-\frac{d_X}{2}} |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)} \end{aligned} \quad (1)$$

where $\mathcal{N}(\mu_i, \Sigma_i)$ is a normal distribution of mean μ_i and Co-variance matrix Σ_i .

Let's assume that there are two ODCs, ODC_1 and ODC_2 , defined on the same N points, the first one has overlap p_1 and the second one is with overlap p_2 , such that, $p_2 > p_1$. Let's assume that the number of kernel machines in ODC_1 and ODC_2 are K_1 and K_2 , respectively. Hence,

$$K_1 = \frac{N}{(1-p_1)M}, \quad K_2 = \frac{N}{(1-p_2)M} \quad (2)$$

Since $p_2 > p_1$, $0 \leq p_1 < 1$ and $0 \leq p_2 < 1$, then $K_2 > K_1$, which indicates that the number of kernel machines in ODC_2 with higher overlap is bigger than the number of kernel machines in ODC_1 . Let's assume that there is an test point \mathbf{x}^* and define that the probability that \mathbf{x}^* is captured by the ODC to be proportional to the maximum probability of \mathbf{x}^* among the domains.

$$\begin{aligned} p(\mathbf{x}^*) &= \sum_{i=1}^K p(\mathbf{x}^*, D_i) = \sum_{i=1}^K p(\mathbf{x}^*|D_i) \delta(p(\mathbf{x}^*|D_i) - \max_{j=1}^K (p(\mathbf{x}^*|D_j))) \\ p(\mathbf{x}^*) &= \max_{i=1}^K p(\mathbf{x}^*|D_i) \\ &= (2\pi)^{-\frac{d_X}{2}} \max_{i=1}^K |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}^*-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}^*-\mu_i)} \end{aligned} \quad (3)$$

where $\delta(0) = 1$, 0 otherwise. The reason behind this definition of $p(x^*)$ is that our method select the domain of predution based on $\operatorname{argmax}_{i=1}^K p(\mathbf{x}^*|D_i)$. Hence $p_{ODC_1}(x^*) = \max_{i=1}^{K_1} p_{ODC_1}(\mathbf{x}^*|D_i)$ and $p_{ODC_2}(x^*) = \max_{i=1}^{K_2} p_{ODC_2}(\mathbf{x}^*|D_i)$.

We start by the case where the points are uniformly distributed in the space. Under this condition and assuming that spatially cohesive domain cover, this leads to that $p(\mathbf{x}^*|D_i) \approx \mathcal{N}(\mu_i, \Sigma) \forall i$, where $\Sigma_1 = \Sigma_2 \cdots = \Sigma_K = \Sigma$. Hence

$$\begin{aligned} p(\mathbf{x}^*|D_i) &\propto e^{-\frac{1}{2}(\mathbf{x}^*-\mu_i)^\top \Sigma^{-1}(\mathbf{x}^*-\mu_i)} \\ \ln(p(\mathbf{x}^*|D_i)) &\propto -(\mathbf{x}^*-\mu_i)^\top \Sigma^{-1}(\mathbf{x}^*-\mu_i) \end{aligned} \quad (4)$$

Then

$$\begin{aligned}
p(\mathbf{x}^*) &= \max_{i=1}^K p(\mathbf{x}^* | D_i) \\
&= (2\pi)^{-\frac{d_X}{2}} |\Sigma|^{-\frac{1}{2}} \max_{i=1}^K |e^{-\frac{1}{2}(\mathbf{x}^* - \mu_i)^\top \Sigma^{-1}(\mathbf{x}^* - \mu_i)}| \\
&\propto \max_{i=1}^K e^{-\frac{1}{2}(\mathbf{x}^* - \mu_i)^\top \Sigma^{-1}(\mathbf{x}^* - \mu_i)} \\
\ln(p(\mathbf{x}^*)) &\propto \max_{i=1}^K -(\mathbf{x}^* - \mu_i)^\top \Sigma^{-1}(\mathbf{x}^* - \mu_i)
\end{aligned} \tag{5}$$

Hence, $p(\mathbf{x}^*)$ gets maximized as it get closer to one of the centers of the domains μ_i , defined by the ODC. It is not hard to seen that that chances of \mathbf{x}^* to be closer to one of the centers covered by ODC_2 is higher than ODC_1 , especially when $p_2 \gg p_1$. This is since $K_1 = \frac{N}{(1-p_1)M}$, $K_2 = \frac{N}{(1-p_2)M}$. Hence $K_2 \gg K_1$ when $p_2 \gg p_1$. For instance, when $p_1 = 0$ and $p_2 = 0.9$, this leads to that ODC_1 will generate $K_1 = \frac{N}{M}$ domains, while ODC_2 will generate $K_2 = \frac{10 \cdot N}{M} = 10K_1$, which is ten times more domains and centers. The fact that there are much more domains if $K_2 \gg K_1$ together with that there domains are spatially cohesive leads to $\max_{i=1}^{K_1} -(\mathbf{x}^* - \mu_i^1)^\top \Sigma_1^{-1}(\mathbf{x}^* - \mu_i^1) \gg \max_{i=1}^{K_2} -(\mathbf{x}^* - \mu_i^2)^\top \Sigma_2^{-1}(\mathbf{x}^* - \mu_i^2)$. The proof of this statement derives from the fact that $\max_{i=1}^{K_1} -(\mathbf{x}^* - \mu_i^1)^\top \Sigma_1^{-1}(\mathbf{x}^* - \mu_i^1)$ is could maximized by (1) if \mathbf{x}^* gets very close to one of $\mu_i, i = 1 : K$, and (2) smaller variance $|\Sigma|$, which is minimized by the nature by which ODC is created, since each domain i is created by neighboring points to its center (*i.e.* $|\Sigma_1| \gg |\Sigma_2|$). This directly leads to that if $K_2 \gg K_1$ then $\max_{i=1}^{K_1} -(\mathbf{x}^* - \mu_i^1)^\top \Sigma_1^{-1}(\mathbf{x}^* - \mu_i^1) \gg \max_{i=1}^{K_2} -(\mathbf{x}^* - \mu_i^2)^\top \Sigma_2^{-1}(\mathbf{x}^* - \mu_i^2)$. Hence, $p_{ODC_2}(x^*) \gg p_{ODC_1}(x^*)$.

Even if the points are not uniformly distributed, it is still more likely that an ODC with higher overlap would have higher $p(x^*)$, since x^* is close under expectation to one of the centers if more spatially cohesive domains are generated which increases with higher overlap. Our experiments also proves that the ODC concept generalizes on three real dataset where the training points are not distributed uniformly. \square

More Figures and Results

figure 1 shows our analysis on ODC for $M = 400$. We noticed sigificant drop in the performance as M decreases. For instance when $M = 200$, The error for TGP best performance increased to 43.88mm instead of 38mm.

Local Kernel Machines hyper-parameters on each dataset

The hyper parameters were learnt using cross validation on the training set for GPR, TGP and IWTGP that we are interested in. The following subsection present the learnt hyper-parameters and the error measures on each dataset in case of TGPs.

Poser Dataset

The parameters $2\rho_x^2$, $2\rho_y^2$, λ_X , and λ_Y were assigned to 5, 5000, 10^{-4} , and 10^{-4} , respectively.

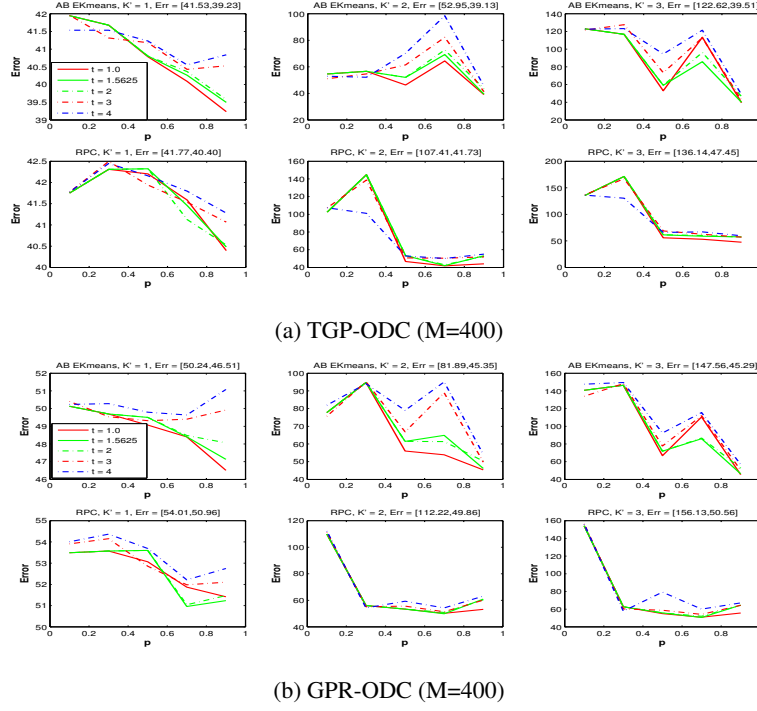


Figure 1: Overlapping Domain Cover Parameter Analysis of GPR and TGP on Human Eva Dataset (best seen in color) (M=400)

Table 1: Error and Time for Poser and Human Eva datasets (on 2.6GHZ intel core i7), M = 800

		Poser			HumanEva		
		Error (deg)	Training Time	Prediction Time	Error (mm)	Training Time	Prediction Time
TGP	NN	5.43	-	188.99 sec	38.1	-	6364 sec
	ODC ($p = 0.9, t = 1, K' = 1$)-Ekmeans	5.4	(3.7 +25.1) sec	16.5 sec	38.99	(2001 + 45.4) sec	298 sec
	ODC ($p = 0.9, t = 1, K' = 2$)-Ekmeans	5.53	(3.7 +29.4) sec	47.04 sec	39.2	(2001 + 45.24) sec	569.6946 sec
	ODC ($p = 0.9, t = 1, K' = 3$)-Ekmeans	5.4	(3.7 +28.8) sec	71.4 sec	40.9	(2001 + 45.7) sec	721.0 sec
	ODC ($p = 0, t = 1, K' = 1$)-Ekmeans	7.6	(3.9 + 1.33) sec	14.8 sec	41.87	(240 + 4.9832) sec	256.7
	ODC ($p = 0, t = 1, K' = 2$)-Ekmeans	12.3	(3.9 + 2.69) sec	42.25 sec	136.52	(240 + 4.7790) sec	514.93
	ODC ($p = 0, t = 1, K' = 3$)-Ekmeans	12.52	(3.9 + 1.86) sec	72.38 sec	187.72	(240 + 4.75) sec	771
	ODC ($p = 0.9, t = 1, K' = 1$)-RPC	5.6	(0.23 +41.6) sec	15.8 sec	39.9	(0.45 + 49.05) sec	277.25 sec
	ODC ($p = 0.9, t = 1, K' = 2$)-RPC	5.52	(0.23 +43.80) sec	43.802 sec	40.41	(0.45 + 46.77) sec	677.52 sec
	ODC ($p = 0.9, t = 1, K' = 3$)-RPC	5.59	(0.23 +43.05) sec	67.11 sec	41.21	(0.45 + 47.63) sec	883 sec
	ODC ($p = 0, t = 1, K' = 1$)-RPC	7.7	(0.15 + 1.7) sec	13.89 sec	42.32	(0.19 + 5.3) sec	241.64 sec
	ODC ($p = 0, t = 1, K' = 2$)-RPC	9.29	(0.15 + 1.8) sec	41.86 sec	58.99	(0.19 + 5.16) sec	475.14 sec
	ODC ($p = 0, t = 1, K' = 3$)-RPC	12.47	(0.15 + 1.80) sec	66.42 sec	136	(0.19 + 5.2) sec	721.49 sec
GPR	NN	6.77	-	24 sec	54.8	-	618 sec
	ODC ($p = 0.9, t = 1, K' = 1$)-Ekmeans	6.27	(3.7 +11.1) sec	0.56 sec	49.3	(2001 + 42.85)sec	78.85 sec
	ODC($p = 0.0, t = 1, K' = 1$)-Ekmeans	7.54	(3.9 + 1.38) sec	0.35 sec	49.6	(240 + 6.4) sec	48.1 sec
	ODC ($p = 0.9, t = 1, K' = 1$)-RPC	6.45	(0.23 +17.3) sec	0.52 sec	52.8	(0.49 + 46.06) sec	64.13 sec
	ODC ($p = 0.0, t = 1, K' = 1$)-RPC = [?]	7.46	(0.15 + 1.47) sec	0.27 sec	54.6	(0.261 + 4.58) sec	43.52 sec
	FITC [?]	7.63 (+/- 0.4)	(- + 20.63)	0.3106	68.36(+/- 0.84)	-	101.5442 (+/- 1.36) sec

HumanEva Dataset

The parameters $2\rho_x^2$, $2\rho_y^2$, λ_X , and λ_Y were assigned to 5, 500000, 10^{-3} , and 10^{-3} , respectively.

Human 3.6 Dataset

The parameters $2\rho_x^2$, $2\rho_y^2$, λ_X , and λ_Y were assigned to 5, 500000, 10^{-3} , and 10^{-3} , respectively.

Equal Size Kmeans (EKmeans): More Details

Iterative Minimum-Distance Assignments (IMDA) k-means EKmeans

We tried another variant for Ekmeans that we call Iterative Minimum-Distance Assignments (IMDA) k-means. Tis algorithm works as follows. We initialize a pool of unassigned points $\tilde{X} = X$ and initialize all clusters as empty. Given the means computed from the previous update steps, we compute the distances $d(\mathbf{x}_i, \mu_j)$ for all points/center pairs. We iteratively pick the minimum distance pair

$$(\mathbf{x}_p, \mu_l) : d(\mathbf{x}_p, \mu_l) \leq d(\mathbf{x}_i, \mu_j) \forall \mathbf{x}_i \in \tilde{X} \text{ and } |C_l| < N/K$$

and assign point \mathbf{x}_p to cluster l . The point is then removed from the pool of unassigned points. if $|C_l| = N/K$, then it is marked as balanced and no longer considered. The process is repeated until the pool is empty.

Comparison between IMDA and Assign and Balance (AB) EKmeans(presented in the paper)

Table 2 presents the average *intra* measure over 10 runs of IMDA k-means and AB kmeans algorithms, initialized at different centers, that were selected at random. As illustrated in table 2, the AB kmeans outperforms IMDA k-means in these experiments, which motivated us to utilize AB Ekmeans, which is presented in the paper, against IMDA k-means under our ODC prediction framework. Our interpretation for these results is because AB Ekmeans initializes the assignment with an assignment that minimizes $J(C) = \min \sum_{j=1}^K \sum_{\mathbf{x}_i \in C_j} d(\mathbf{x}_i, \mu_j)$ given the cluster centers and then balance the clusters

Table 2: $J(C)$ of AB-kmeans and IMDA-kmeans on a dataset of 10,000 random 2D points, averaged over 10 runs

	K=5	K = 10	K=50
AB-kmeans	1077.3	540.241	105.505
IMDA-kmeans	1290.6	657.446	122.006
Error Reduction	16.53%	17.83%	13.52%

More figures on AB Ekmeans

Figure 2 shows the clustering performance on 300000 random 2D point (K=5, 57). Figure 3 shows the clustering output of our algorithm visualized on using the first three principal components of Human Eva training hog features. The figures shows that the cluster are spatially cohesive but not necessarily circular. This makes the elliptic

distribution of the data captured by Mode 3 gives more accuracy membership measure me to the subdomains.

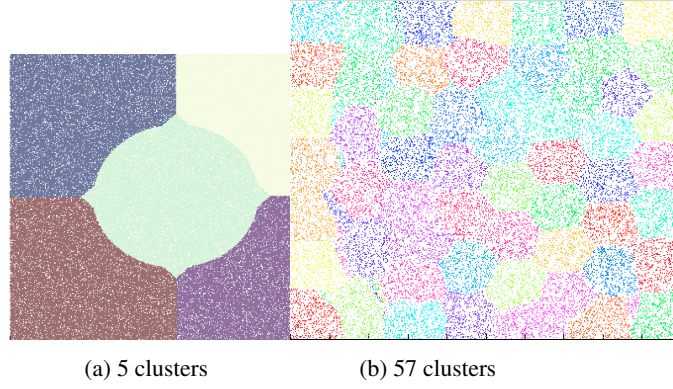


Figure 2: Applying our Assign and Balance variant of Kmeans on 300,000 random 2D points

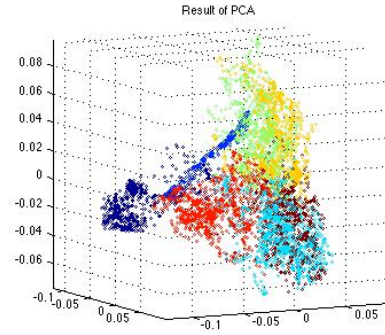


Figure 3: Human Eva clustering first three Principal Components

Equal Size Assignment Algorithms Pseudocode

As presented in the paper, our k-means variant algorithms modify only the assignment step of the standard k-means algorithm. Algorithm 1 and 2 show the pseudo-code of the assignment steps on IMDA-k-means and AB-k-means algorithms respectively. We attach the MATLAB implementation of both algorithms in "Ekmeans-assign" folder. We plan to release the whole implementation of our paper as well as soon as we well-document of the code.

It is also important to note that, we initialize both algorithms by the cluster centers computed by running the regular k-means.

Input: $\mathbf{X}(N \times d_x), \{\boldsymbol{\mu}_i\}_{i=1}^K$

Output: labels

- 1- Create a matrix $\mathbf{D} \in R^{N \times K}$, where $\mathbf{D}[i, j]$ is the distance between the i^{th} point to the j^{th} cluster center.
- 2- Get the coordinate (i_*, j_*) that maps the smallest distance in \mathbf{D} .
- 3- Remove the i_*^{th} row from matrix \mathbf{D} and mark it as assigned to the j^{th} cluster
- 4- If the size of the cluster j achieves the ideal size (i.e. n/K), then remove the j^{th} column from matrix \mathbf{D} .
- 5- Go to step 2 if there is still unassigned points

Algorithm 1: Iterative Minimum-Distance Assignments (IMDA) k-means: Assignment Step

Input: $\mathbf{X}(N \times d_x), \{\boldsymbol{\mu}_i\}_{i=1}^K$

Output: labels

- 1- Assign the points initially to its closest center; this will put the clusters into 3 groups (1) balanced clusters (2) overflowed clusters (3) under-flowed clusters.
- 2- Create a matrix $\mathbf{D} \in R^{N \times K}$, where $\mathbf{D}[i, j]$ is the distance between the i^{th} point to the j^{th} cluster center; rows are restricted points belongs only to the overflowed clusters; columns are restricted to underflowed cluster centers
- 3- Get the coordinate (i_*, j_*) that maps the smallest distance in \mathbf{D} .
- 4- Remove the i_*^{th} row from matrix \mathbf{D} and mark it as assigned to the j^{th} cluster
- 5- If the size of the cluster j achieves the ideal size (i.e. n/K), then remove the j^{th} column from matrix \mathbf{D} .
- 6- Go to step 3 if there is still unassigned points

Algorithm 2: Assign and Balance (AB) k-means: Assignment Step

Overlapping Domain Cover(ODC) Generation-Algorithm

Algorithm 3 shows how the overlapping sub-domains are generated from the equal size clusters from the closest r clusters.

IWTGP integration under ODC-Framwork

Yamada et al [?] proposed the importance-weighted variant of twin Gaussian processes [?] called IWTGP. The weights are calculated using RuLSIF [?] (relative unconstrained least-squares importance fitting). The weights were modeled as $w_\alpha(\mathbf{x}, \boldsymbol{\theta}) = \sum_{l=1}^{n_{te}} \theta_l k(\mathbf{x}, \mathbf{x}_l)$ to minimize $E_{p_{te}(\mathbf{x})}[(w_\alpha(\mathbf{x}, \boldsymbol{\theta}) - w_\alpha(\mathbf{x}))^2]$, where $k(\mathbf{x}, \mathbf{x}_l) = \exp(-\frac{\|\mathbf{x} - \mathbf{x}_l\|^2}{2\tau^2})$, $w_\alpha(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{(1-\alpha)p_{te}(\mathbf{x}) + \alpha p_{tr}(\mathbf{x})}$, $0 \leq \alpha \leq 1$. To cope with this instability issue, setting α to $0 \leq \alpha \leq 1$ is practically useful for stabilizing the covariate shift adaptation, even though it cannot give an unbiased model under covariate shift [?]. According [?] the optimal $\hat{\boldsymbol{\theta}}$ vector is computed in a closed form solution as follows.

$$\hat{\boldsymbol{\theta}} = (\hat{\mathbf{H}} + \nu \mathbf{I})^{-1} \hat{\mathbf{h}} \quad (6)$$

where $\hat{\mathbf{H}}_{l,l'} = \frac{1-\alpha}{n_{te}} \sum_{i=1}^{n_{te}} k(\mathbf{x}_i^{te}, \mathbf{x}_l^{te}) k(\mathbf{x}_i^{te}, \mathbf{x}_{l'}^{te}) + \frac{\alpha}{n_{tr}} \sum_{j=1}^{n_{tr}} k(\mathbf{x}_j^{tr}, \mathbf{x}_l^{te}) k(\mathbf{x}_j^{tr}, \mathbf{x}_{l'}^{te})$, $\hat{\mathbf{h}}$ is n_{te} dimensional vector with the l^{th} element $\hat{h}_l = \frac{1}{n_{te}} \sum_{i=1}^{n_{te}} k(\mathbf{x}_i^{te}, \mathbf{x}_l^{te})$, \mathbf{I} is $n_{te} \times n_{te}$ -dimensional identity matrix. where n_{te} and n_{tr} and the number of testing and training points respectively. Model selection of RuLSIF is based on cross-validation

Input: Clusters $\{C_k\}_{k=1}^K$ **Output:** Overlapping subdomains $\{D_k\}_{k=1}^K$

foreach Cluster C_k **do**

- Compute the closest r clusters $\{C'_i\}_{i=1}^r$ based on $DK_i = \|\mu_k - \mu_i\|, i \neq k$
- Let $LK_i = 1/DK_i, WK_i = \frac{LK_i}{\sum_{l=1}^r LK_l} i = 1 : r$
- Let $NPK_i = \text{floor}(WK_i * OPC), i = 1 : r$
- Let $ExKPts = (1 - p)M - \sum_{l=1}^r NPK_l$
- Let $NPK_i = NPK_i + 1, i = 1 : ExKPts$
- $D_k = C_k$
- Let $overflow = 0$
 - ▷ The following for loop goes over the r clusters on an increasing order of DK_i
- for** $i=1 : r$ **do**
 - if** $NPK_i < |C_i|$ **then**
 - $overflow = overflow + NPK_i - |C_i|$
 - $NPK_i = |C_i|$
 - if** $NPK_i \geq |C_i|$ **then**
 - $G_i = \min(overflow, |C_i| - NPK_i)$
 - $NPK_i = NPK_i + G_i$
 - $overflow = overflow - G_i$
 - $Ps_i = KNN(OVC_{K_j}, NPK_i)$
 - $D_k = D_k \cup Ps_i$
- for** $i=1 : r$ **do**
 - $Ps_i = KNN(OVC_{K_j}, NPK_i)$
 - $D_k = D_k \cup Ps_i$
 - ▷ where KNN is the K-nearest neighbors algorithms. For high performance calculation of KNN , we use FLANN [?] to calculate KNN .

Algorithm 3: Subdomains Generation (Note: All $\{D_k\}_{k=1}^K$ are stored as indices to X).

with respect to the squared-error criterion J in [?]. Having computed $\hat{\theta}$, each input and output examples are simply re-weighted by $w_\alpha^{\frac{1}{2}}$ [?]. Therefore, the output of the importance weighted TGP (IWTGP) is given by

$$\hat{y} = \underset{y}{\operatorname{argmin}} [K_Y(\mathbf{y}, \mathbf{y}) - 2k_y(\mathbf{y})^T \mathbf{u}_w - \eta_w \log(K_Y(\mathbf{y}, \mathbf{y}) - k_y(\mathbf{y})^T \mathbf{W}^{\frac{1}{2}} (\mathbf{W}^{\frac{1}{2}} \mathbf{K}_Y \mathbf{W}^{\frac{1}{2}} + \lambda_y I)^{-1} \mathbf{W}^{\frac{1}{2}} k_y(\mathbf{y}))] \quad (7)$$

where $\mathbf{u}_w = \mathbf{W}^{\frac{1}{2}} (\mathbf{W}^{\frac{1}{2}} \mathbf{K}_X \mathbf{W}^{\frac{1}{2}} + \lambda_x I)^{-1} \mathbf{W}^{\frac{1}{2}} k_x(\mathbf{x})$, $\eta_w = k_X(\mathbf{x}, \mathbf{x}) - k_x(\mathbf{x})^T \mathbf{u}_w$. IWTGP can also be solved using a second order, BFGS quasi-Newton optimizer with cubic polynomial line search for optimal step size selection.

Training

It is not obvious how to factor out computations that does not depend on the test data in the case of IWTGP, since the computational extensive factor (i.e. $(\mathbf{W}^{i\frac{1}{2}} \mathbf{K}_X^i \mathbf{W}^{i\frac{1}{2}} + \lambda_x^i \mathbf{I})^{-1}$, $(\mathbf{W}^{i\frac{1}{2}} \mathbf{K}_Y^i \mathbf{W}^{i\frac{1}{2}} + \lambda_y^i \mathbf{I})^{-1}$) does depend on the test set since \mathbf{W}^i is computed on test time. However, we utilized the following theorem from linear algebra to help us factor out the computation; proof is attached in the supplementary materials.

$$(\mathbf{D} \mathbf{A} \mathbf{D} + \lambda \mathbf{I})^{-1} = \mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1} - \frac{\lambda \mathbf{D}^{-2} \mathbf{A}^{-2} \mathbf{D}^{-2}}{1 + \lambda \cdot \operatorname{tr}(\mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})} \quad (8)$$

, where \mathbf{D} is a diagonal matrix, I is the identity matrix, and $\operatorname{tr}(\mathbf{B})$ is the trace of matrix \mathbf{B} . Mapping \mathbf{D} to $\mathbf{W}^{i\frac{1}{2}}$, \mathbf{A} to either of \mathbf{K}_X^i or \mathbf{K}_Y^i , we can compute $\mathcal{M}^i = \{\mathbf{K}_X^{i-1}, \mathbf{K}_Y^{i-1}\}$. Having computed \mathbf{W}^i on test time, $(\mathbf{W}^{i\frac{1}{2}} \mathbf{K}_X^i \mathbf{W}^{i\frac{1}{2}} + \lambda_x \mathbf{I})^{-1}$, $(\mathbf{W}^{i\frac{1}{2}} \mathbf{K}_Y^i \mathbf{W}^{i\frac{1}{2}} + \lambda_y \mathbf{I})^{-1}$ could be computed in quadratic time given \mathcal{M}^i following equation 8, since the inverse and the power of $\mathbf{W}^{i\frac{1}{2}}$ has linear computational complexity.

Proof that $(\mathbf{D} \mathbf{A} \mathbf{D} + \lambda \mathbf{I})^{-1} = \mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1} - \frac{\lambda (\mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})^2}{1 + \lambda^{-1} \cdot \operatorname{tr}(\mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})}$ Kenneth Miller [?] proposed the following Lemma on Matrix Inverse.

$$(G + H)^{-1} = G^{-1} - \frac{1}{1 + \operatorname{tr}(GH^{-1})} G^{-1} H G^{-1} \quad (9)$$

Applying Miller's lemma, where $G = \mathbf{D} \mathbf{A} \mathbf{D}$ and $H = \lambda \mathbf{I}$, leads to $(\mathbf{D} \mathbf{A} \mathbf{D} + \lambda \mathbf{I})^{-1} = \mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1} - \frac{\lambda (\mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})^2}{1 + \lambda^{-1} \cdot \operatorname{tr}(\mathbf{D}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})}$.

Prediction

From the above discussion, the prediction for each subdomain is computed as follows

$$\begin{aligned} \hat{Y}_{x_*}^i = \underset{Y_{x_*}^i}{\operatorname{argmin}} [& k_Y(\mathbf{Y}_{x_*}^i, \mathbf{Y}_{x_*}^i) - 2k_y(\mathbf{Y}_{x_*}^i)^T \mathbf{u}_w - \\ & \eta_w \log(K_Y(\mathbf{Y}_{x_*}^i, \mathbf{Y}_{x_*}^i) - \\ & k_y(\mathbf{Y}_{x_*}^i)^T \mathbf{W}^{i\frac{1}{2}} (\mathbf{W}^{i\frac{1}{2}} \mathbf{K}_Y \mathbf{W}^{i\frac{1}{2}} + \lambda_y I)^{-1} \\ & \mathbf{W}^{i\frac{1}{2}} k_y(\mathbf{Y}_{x_*}^i))] \end{aligned} \quad (10)$$

¹ \mathbf{W} is a diagonal matrix

where $\mathbf{u}_w = \mathbf{W}^{\frac{1}{2}}(\mathbf{W}^{\frac{1}{2}}\mathbf{K}_X\mathbf{W}^{\frac{1}{2}} + \lambda_x\mathbf{I})^{-1}\mathbf{W}^{\frac{1}{2}}k_x(\mathbf{x})$, $\eta_w = k_X(\mathbf{x}, \mathbf{x}) - k_x(\mathbf{x})^T\mathbf{u}_w$, $(\mathbf{W}^{i\frac{1}{2}}\mathbf{K}_X^i\mathbf{W}^{i\frac{1}{2}} + \lambda_x\mathbf{I})^{-1}$, $(\mathbf{W}^{i\frac{1}{2}}\mathbf{K}_X\mathbf{W}^{i\frac{1}{2}} + \lambda_x\mathbf{I})^{-1}$ could be computed in quadratic time given \mathcal{M}^i and W . Hence, the $\hat{Y}_{x_*j}^i$ has $O(\textit{iters} \cdot M)^2$ complexity, where *iters* is the number of iterations.

IWTGP-ODC Experiments

Tables 3 and 4 details the results of IWTGP-ODC experiments on Poser and HumanEva datasets in terms of error and speedup in prediction time.

	IWTTGP ($M = 800, M_{tst} = 418$)	IWTGP-ODC ($M = 800, M_{tst} = 418$)
error (deg)	6.1	5.32
err reduction (deg)	-	0.783
err reduction %	-	12.836%
Prediction Time (sec)	360.0	26.61
speedUp	-	13.5

Table 3: POSER dataset IWTGP-NN vs IWTGP-ODC

	IWTGP ($M = M_{tst} = 800$)	IWTTGP-ODC ($M = M_{tst} = 800$)
error (mm)	39.1	39.3
err reduction (mm)	-	-0.2
err reduction %	-	-0.512%
Prediction Time (sec)	7938.15	569.66
speedUp	-	13.92

Table 4: Humen Eva dataset: IWTGPKNN vs IWTGP-ODC