CHAPTER 4

Covariate Shift in Machine Learning

SANTOSH CHAPANERI and DEEPAK JAYASWAL

Department of Electronics and Telecommunication Engineering, St. Francis Institute of Technology, University of Mumbai, Mumbai, Maharashtra, India, E-mail: santoshchapaneri@sfit.ac.in (S. Chapaneri)

ABSTRACT

Covariate shift occurs in machine learning when the input train and test probability distributions are different even though the conditional distribution of output given the train and test inputs remain unchanged. Most existing supervised machine learning techniques make an assumption that the train and test data samples follow the same probability distribution, but this is violated in practice for many real-world applications. In this work, the covariate shift is corrected by learning the importance weights for re-weighting the train data such that the training samples closer to the test data samples get more importance during modeling. To estimate the importance weights, a computationally efficient Frank-Wolfe optimization algorithm is used. Structured prediction is required to model the dependency that can exist in the multi-dimensional target variables, for example, in the application of human pose estimation. Twin Gaussian process (TGP) structured regression is used to model this dependency for improving the prediction performance relative to learning the multiple target dimensions separately. A computationally efficient method of TGP is presented for covariate shift correction and the performance is evaluated on the benchmark HumanEva dataset resulting in significantly reduced regression error.

4.1 INTRODUCTION

A fundamental assumption in most supervised machine learning (ML) methods is that the train and test data points are sampled from the *same*

underlying probability distribution. Given the joint training distribution $p_{\text{tr}}(\mathbf{x}, \mathbf{y})$ and the joint test distribution $p_{\text{te}}(\mathbf{x}, \mathbf{y})$ where $\mathbf{x} \in \boldsymbol{\chi}$ is the feature vector and $\mathbf{y} \in \boldsymbol{\gamma}$ is the target, the assumption is that the joint distributions remain the same, that is $p_{\text{tr}}(\mathbf{x}, \mathbf{y}) = p_{\text{te}}(\mathbf{x}, \mathbf{y})$. However, this assumption is generally not valid for real-life test data. For example, consider the problem of price prediction of an insurance policy based on age, income, employment type, etc. With the typical ML pipeline of preprocessing, data cleaning, feature selection, and model training, the model performance can be evaluated on the test data. The test performance can degrade if the test distribution of age is different from the train distribution of age, i.e., if the model was learned on the age group of 15 to 45, but the test data also includes customers belonging to the age group of 50 and above. Due to this potential mismatch between the distributions, the test performance is always lower than the training performance.

Broadly, there are three types of dataset shift studied in the literature:

1. **Covariate Shift:** which occurs due to a shift in the distribution of independent variables for $\chi \rightarrow \gamma$ problems. In this case, the conditional distributions remain the same but the marginal distributions of inputs differ, i.e., $p_{\text{tr}}(\text{x,y}) = p_{\text{tr}}(\text{y|x}) p_{\text{tr}}(\text{x})$ and $p_{\text{te}}(\text{x,y}) = p_{\text{te}}(\text{y|x}) p_{\text{te}}(\text{x})$, so $p_{\text{tr}}(\text{y|x}) = p_{\text{te}}(\text{y|x})$ and $p_{\text{tr}}(\text{x}) \neq p_{\text{te}}(\text{x})$.
2. **Prior Probability Shift:** which occurs due to a shift in the distribution of dependent variables. This dataset shift scenario is applicable to $\gamma \rightarrow \chi$ problems where the target variable determines the covariate values. An example is the field of medical diagnosis where the disease label determines the symptoms. In this case, we have $p_{tr}(x|y) = p_{te}(x|y)$ and $p_{tr}(y) \neq p_{te}(y)$.
3. **Concept Shift:** Also referred to as concept drift, can occur in both $\chi \rightarrow \gamma$ and $\gamma \rightarrow \chi$ problems due to a changing relationship between the independent and dependent variables. In this case, we have $p_{tr}(y|x) \neq p_{te}(y|x)$ and $p_{tr}(x) = p_{te}(x)$ for $\chi \rightarrow \gamma$ problems and $p_{tr}(x|y) \neq p_{te}(x|y)$ and $p_{tr}(y) \neq p_{te}(y)$ for $\gamma \rightarrow \chi$ problems.

A detailed overview of dataset shift is presented in [1, 2], illustrating its various types, causes, and several applications. We focus on covariate shift since it is widely applicable to many existing supervised ML methods. The covariate shift can be corrected by estimating the importance weight $w(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{tr}(\mathbf{x})}$ from the probability distributions of train and test data. But this is non-trivial to solve due to the problem of curse of dimensionality and can

also be not reliable for input data in high dimensions [3]. A feasible solution is to thus learn the importance weight directly from the given data without the need to estimate the train and test probability densities.

For correcting the covariate shift, various techniques are proposed in the literature, e.g., Kullback-Leibler importance estimation procedure (KLIEP) [3], least-squares importance fitting (LSIF) [4], relative unconstrained least-squares importance fitting (RuLSIF) [5], log-linear KLIEP (LL-KLIEP) [6], KLIEP based on Gaussian mixture models (GM-KLIEP) [7], KLIEP based on mixture of probabilistic principal component analyzers (PM-KLIEP) [8], etc. These important weight estimation techniques are unsupervised since only the features are required for estimating the importance weight without requiring the target variable. The LSIF and RuLSIF have a closed-form analytical solution for the importance weights; however, their resulting estimates can be biased [5]. KLIEP is used in this work for estimating the importance weights since a) it has a convex optimization problem with a unique global solution, b) its weight estimate is unbiased and c) the Frank-Wolfe (FW) algorithm can be used to solve its optimization problem efficiently.

In this work, the correction of covariate shift is done following the unsupervised principle (i.e., without the target variables) using the *projection-free* FW optimization algorithm [9]. The FW optimization algorithm can iteratively solve constrained convex optimization problems efficiently using the linearization principle and can obtain sparser solutions [9, 10]. In Ref. [11], a covariate shift based on the standard FW method was proposed to compute the importance weights of KLIEP. Instead of using the standard FW algorithm, a more efficient FW covariate shift algorithm using the pairwise steps is used in this work resulting in a more sparser solution with significantly better computational efficiency.

Structured regression is an increasingly studied concept in ML when the multi-dimensional target variables are dependent on each other across dimensions. The task is to learn the functional mapping $f : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$ from input $\mathbf{x} \in \mathbb{R}^{d_x}$ to target $\mathbf{y} \in \mathbb{R}^{d_y}$ (both multivariate) by exploiting any correlation that may exist between multivariate data dimensions. Twin Gaussian process (TGP) regression [12] is used in this work for modeling the dependency to reduce the regression error. The TGP regression model is updated to correct the covariate shift during the training phase. Experimental results on the benchmark HumanEva dataset show that the PFWCS algorithm used with the importance weighted TGP method outperforms the non-weighted TGP method, thus reducing the effect of covariate shift.

4.2 ESTIMATION OF THE IMPORTANCE WEIGHTS

4.2.1 KULLBACK LEIBLER IMPORTANCE ESTIMATION PROCEDURE

KLIEP was proposed in [3] to compute the importance weight $w(\mathbf{x}) = \hat{p}_{\text{te}}(\mathbf{x}) / p_{\text{tr}}(\mathbf{x})$. This is achieved by reducing the Kullback-Leibler (KL) divergence from the actual test probability density $p_{\text{te}}(\mathbf{x})$ to its estimate $\hat{p}_{\text{te}}(\mathbf{x})$ without the need to compute the train and test probability densities explicitly. Based on mixtures of Gaussians, the importance weights are modeled as Eqn. (1), where $\alpha = \left[\alpha_1, \ldots, \alpha_{n_{\text{te}}}\right]^{\text{T}}$ denote the mixing coefficients, $\kappa(\cdot)$ is a kernel function satisfying Mercer's conditions, n_{tr} is the number of train samples and n_{te} is the number of test samples. The importance weights can be intuitively understood as follows: if the training sample is closer to the test distribution, then this sample should be given more importance; likewise, if the training sample is far from the test distribution, then such a sample should be assigned a less weight. This implies that the training samples should be re-weighted by $w(\mathrm{x}^{\text{tr}})$.

$$w\left(\mathbf{x}^{\text{tr}}\right) = \sum_{l=1}^{n_{\text{te}}} \alpha_l \kappa_l\left(\mathbf{x}^{\text{tr}}\right) = \sum_{l=1}^{n_{\text{te}}} \alpha_l \exp\left(\frac{-\mathbf{x}^{\text{tr}} - \mathbf{x}_l^{\text{te}2}}{2\sigma^2}\right) \tag{1}$$

Using the importance weights, the test probability density can be estimated as $\hat{p}_{\text{te}}(\mathbf{x}) = w\left(\mathbf{x}^{\text{tr}}\right) p_{\text{tr}}(\mathbf{x})$. Accordingly, the KL divergence measure from the true test probability density to its estimated version is given by Eqn. (2).

$$\begin{aligned} KL\left[p_{\text{te}}(\mathbf{x}) \hat{p}_{\text{te}}(\mathbf{x})\right] &= \int p_{\text{te}}(\mathbf{x}) \log \frac{p_{\text{te}}(\mathbf{x})}{\hat{p}_{\text{te}}(\mathbf{x})} d\mathbf{x}^{\text{te}} = \int p_{\text{te}}(\mathbf{x}) \log \frac{p_{\text{te}}(\mathbf{x})}{w(\mathbf{x}) p_{\text{tr}}(\mathbf{x})} d\mathbf{x}^{\text{te}} \\ &= \int p_{\text{te}}(\mathbf{x}) \log \frac{p_{\text{te}}(\mathbf{x})}{p_{\text{tr}}(\mathbf{x})} d\mathbf{x}^{\text{te}} - \int p_{\text{te}}(\mathbf{x}) \log w(\mathbf{x}) d\mathbf{x}^{\text{te}} \end{aligned} \tag{2}$$

Since the first term of the KL divergence does not involve the parameter α, it can be ignored for the optimization process. The objective function can be empirically approximated by Eqn. (3), which is to be maximized, since maximizing J_{KLIP} is equivalent to minimizing the KL divergence. Further, the integral of $\hat{p}_{\text{te}}(\mathbf{x})$ should be one, since it is a probability density, resulting in the constraint given by Eqn. (4).

$$J_{\text{KLIEP}} = \int p_{\text{te}}(\mathbf{x}) \log w(\mathbf{x}) dx^{\text{te}} \approx \frac{1}{n_{\text{te}}} \sum_{j=1}^{n_{\text{te}}} \log w\left(\mathbf{x}_j^{\text{te}}\right) = \frac{1}{n_{\text{te}}} \sum_{j=1}^{n_{\text{te}}} \log\left(\sum_{l=1}^{n_{\text{te}}} \alpha_l \kappa_l\left(\mathbf{x}_j^{\text{te}}\right)\right) \tag{3}$$

$$\int \hat{p}_{\text{te}}(\mathbf{x})\, d\mathbf{x}^{\text{te}} = 1 = \int w(\mathbf{x})\, p_{\text{tr}}(\mathbf{x})\, d\mathbf{x}^{\text{tr}} \approx \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} w\left(\mathbf{x}_i^{\text{tr}}\right) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \sum_{j=1}^{n_{\text{te}}} \alpha_j \kappa_j\left(\mathbf{x}_j^{\text{tr}}\right) \tag{4}$$

The constrained convex optimization problem of KLIEP given by Eqn. (5) can be solved using the *projected gradient method* [3] as shown in Figure 4.1.

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} F(\boldsymbol{\alpha}) = \sum_{j=1}^{n_{\text{te}}} \log\left(\sum_{l=1}^{n_{\text{te}}} \alpha_l \kappa_l\left(\mathbf{x}_j^{\text{te}}\right)\right) \\ &s.t. \sum_{i=1}^{n_{\text{tr}}} \sum_{j=1}^{n_{\text{te}}} \alpha_j \kappa_j\left(\mathbf{x}_i^{\text{tr}}\right) = n_{\text{tr}}; \alpha_1, \alpha_2, \ldots, \alpha_{n_{\text{te}}} \geq 0 \end{aligned} \tag{5}$$

Algorithm KLIEP

Input: $\{\mathbf{x}_i^{\text{tr}}\}_{i=1}^{n_{\text{tr}}}, \{\mathbf{x}_j^{\text{te}}\}_{j=1}^{n_{\text{te}}}$, **Output:** $\mathbf{w} = \{w(\mathbf{x}_i^{\text{tr}})\}_{i=1}^{n_{\text{tr}}}$

1: $(\mathbf{A})_{j,l} = \kappa_l(\mathbf{x}_j^{\text{te}}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_j^{\text{te}} - \mathbf{x}_l^{\text{te}}\|^2\right)$ for $j,\ l = \{1, \ldots, n_{\text{te}}\}$

2: $b_l = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \kappa_l(\mathbf{x}_i^{\text{tr}}) = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i^{\text{tr}} - \mathbf{x}_l^{\text{te}}\|^2\right)$

3: Initialize $\boldsymbol{\alpha}\ (> \mathbf{0})$ and $\epsilon\ (0 < \epsilon \ll 1)$

4: **repeat** ▷ projected gradient ascent

5: $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \epsilon \mathbf{A}^{\top}(\mathbf{1}./\mathbf{A}\boldsymbol{\alpha})$ ▷ element-wise division

6: $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \frac{(1 - \mathbf{b}^{\top}\boldsymbol{\alpha})\mathbf{b}}{(\mathbf{b}^{\top}\mathbf{b})}$

7: $\quad \boldsymbol{\alpha} \leftarrow \max(\boldsymbol{\alpha}, \mathbf{0})$ ▷ element-wise maximum

8: $\quad \boldsymbol{\alpha} \leftarrow \frac{\boldsymbol{\alpha}}{(\mathbf{b}^{\top}\boldsymbol{\alpha})}$ ▷ constraint satisfaction

9: **until** convergence

10: Estimate the importance weights $w(\mathbf{x}_i^{\text{tr}}) = \sum_{l=1}^{n_{\text{te}}} \alpha_l \kappa_l(\mathbf{x}_i^{\text{tr}}) \quad \forall i = 1, \ldots, n_{\text{tr}}$

FIGURE 4.1 Algorithm for Kullback-Leibler importance estimation procedure.

The tuning parameter σ needs to be obtained with the likelihood cross-validation method of model selection; however, the resulting value can be biased due to the covariate shift [3]. Thus, the unbiased importance weighted cross-validation (IWCV) [13] procedure is utilized for determining the optimum σ of the KLIEP kernel function as shown in Figure 4.2. In IWCV, the test data set is split into T disjoint subsets $\{\mathcal{X}_t^{\text{te}}\}_{t=1}^{T}$ in line 1. In lines 2 to 8, the importance weights are obtained for each possible σ_m candidate $(m \in \mathcal{M})$

for each split t and the score is computed as $J(m)$. The optimum σ is obtained as the one that maximizes the score in line 9. The importance weight is then obtained in line 10 using the optimum σ value.

Algorithm Frank-Wolfe

Input: $g_0 \in \mathcal{G}$, **Output:** Optimal point g

1: **for** iteration $t = 0, 1, 2, \ldots$ **do**

2: **if** $\nabla F(g_t) = 0$, **return** g_t ▷ convergence

3: Find $s_t \leftarrow \underset{s \in \mathcal{G}}{\operatorname{argmin}} \langle s, \nabla F(g_t) \rangle$ ▷ linear minimization principle

4: Update: $g_{t+1} = (1 - \rho_t) g_t + \rho_t s_t$, for $\rho_t \in [0, 1]$ ▷ ρ_t obtained with line-search

$= g_t + \rho_t (s_t - g_t)$

5: **end for**

FIGURE 4.2 Algorithm for importance weighted cross-validation.

4.2.2 VARIANTS OF KLIEP

Some variants of KLIEP model are proposed in the literature with respect to the parameterization of the importance weights. The log-linear KLIEP method [6] uses a non-linear parameterization given by Eqn. (6) with the denominator representing the normalization constant. The non-linear model with the log function can take only non-negative values and thus the *unconstrained* optimization problem of LL-KLIEP is given by Eqn. (7).

$$w\left(\mathrm{x}^{\mathrm{tr}}\right) = \frac{\exp\left(\sum_{l=1}^{n_{\mathrm{te}}} \alpha_l \kappa_l \left(\mathrm{x}^{\mathrm{tr}}\right)\right)}{\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \exp\left(\sum_{l=1}^{n_{\mathrm{te}}} \alpha_l \kappa_l \left(\mathrm{x}_i^{\mathrm{tr}}\right)\right)} \tag{6}$$

$$J_{\mathrm{LL-KLIEP}} = \frac{1}{n_{\mathrm{te}}} \sum_{l=1}^{n_{\mathrm{te}}} \alpha_l \kappa_l \left(\mathrm{x}^{\mathrm{tr}}\right) - \log \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \exp\left(\sum_{l=1}^{n_{\mathrm{te}}} \alpha_l \kappa_l \left(\mathrm{x}_i^{\mathrm{tr}}\right)\right) \tag{7}$$

Since the implementation of KLIEP uses a spherical Gaussian kernel $\kappa(\cdot)$, it cannot capture any correlation existing in the data. To solve this problem, KLIEP based on Gaussian mixture model (GM-KLIEP) method was proposed in [7] to learn the covariance matrix directly from the data by using a mixture of Gaussians given by Eqn. (8).

$$w\left(x^{tr}\right) = \sum_{l=1}^{K} \pi_l N\left(x^{tr} \middle| \mu_l, \Sigma_l\right) \tag{8}$$

Here, K is the number of Gaussian mixtures, π_l are the mixing coefficients and $N\left(\mathrm{x}^{\mathrm{tr}}|\mu_l,\Sigma_l\right)\propto\exp\left(-\frac{1}{2}\left(\mathrm{x}^{\mathrm{tr}}-\mu_l\right)^{\mathrm{T}}\Sigma_l^{-1}\left(\mathrm{x}^{\mathrm{tr}}-\mu_l\right)\right)$ is the multivariate Gaussian probability density having mean vector μ_l and covariance matrix $\sum_l$ of the l^{th} Gaussian. The constrained optimization problem of GM-KLIEP given by Eqn. (9) can be solved using the expectation-maximization (EM) algorithm [14].

$$\begin{aligned}&\max_{\{\pi_l,\mu_l,\Sigma_l\}_{l=1}^{K}}\left[\sum_{j=1}^{n_{\mathrm{te}}}\log\left(\sum_{l=1}^{K}\pi_l N(\mathrm{x}_j^{\mathrm{te}}\mid\mu_l,\Sigma_l)\right)\right]\\&s.t.\sum_{i=1}^{n_{\mathrm{tr}}}\sum_{l=1}^{K}\pi_l N(\mathrm{x}_i^{\mathrm{tr}}\mid\mu_l,\Sigma_l)=n_{\mathrm{tr}};\pi_1,\pi_2,\ldots,\pi_K\geq 0\end{aligned}\tag{9}$$

The GM-KLIEP method can handle the correlation in the data, but its training involves computing the inverse of covariance matrices, which can be unstable numerically when the data has a rank-deficient covariance matrix. This problem of coping with rank deficient data is solved in [8] using the dimensionality reduction method of principal component analysis (PCA). The importance weight is thus modeled by Eqn. (10) using the mixture of probabilistic principal component analyzers (PPCA) [15] resulting in the PPCA-mixture KLIEP (PM-KLIEP) method, where K refers to the number of PPCA mixtures, π_l are the mixing coefficients, and $\left\{p_l\left(\mathrm{x}^{\mathrm{tr}}\right)\right\}_{l=1}^{K}$ are the probabilistic principal component analyzers.

$$w\left(\mathrm{x}^{\mathrm{tr}}\right)=\sum_{l=1}^{K}\pi_l p_l\left(\mathrm{x}^{\mathrm{tr}}\right)=\sum_{l=1}^{K}\pi_l\frac{1}{\left(2\pi\sigma_l^2\right)^{d/2}\left|\mathrm{C}_l\right|^{1/2}}\exp\left(-\frac{1}{2}\left(\mathrm{x}^{\mathrm{tr}}-\mu_l\right)^{\mathrm{T}}\mathrm{C}_l^{-1}\left(\mathrm{x}^{\mathrm{tr}}-\mu_l\right)\right)\tag{10}$$

Here, $\mathrm{C}_l=\mathrm{W}_l\mathrm{W}_l^{\mathrm{T}}+\sigma_l^2\mathrm{I}_d$, d is the data dimensionality, I_d is the d-dimensional identity matrix, μ_l is the mean vector of the l^{th} Gaussian and $\mathrm{W}\in\mathbb{R}^{d\times m}$ is the projection matrix onto a m-dimensional ($m\leq d$) reduced space. The constrained optimization problem of PM-KLIEP given by Eqn. (11) can also be solved using the expectation-maximization (EM) algorithm similar to that of GM-KLIEP.

$$\begin{aligned}&\max_{\{\pi_l,\mu_l,\mathbf{W}_l,\sigma_l\}_{l=1}^{K}}\left[\sum_{j=1}^{n_{\mathrm{te}}}\log\left(\sum_{l=1}^{K}\pi_l p_l\left(\mathbf{x}_j^{\mathrm{te}}\right)\right)\right]\\&s.t.\sum_{i=1}^{n_{\mathrm{tr}}}\sum_{l=1}^{K}\pi_l p_l\left(\mathbf{x}_i^{\mathrm{tr}}\right)=n_{\mathrm{tr}};\pi_1,\pi_2,\ldots,\pi_K\geq 0\end{aligned}\tag{11}$$

When the reduced feature space dimensionality is the same as that of the data dimensionality, i.e., $m = d$, the PM-KLIEP method becomes equivalent to the GM-KLIEP method. Both GM-KLIEP and PM-KLIEP are non-convex

optimization problems and thus the optimal solution found is only local but not global.

4.3 FRANK-WOLFE COVARIATE SHIFT

4.3.1 FRANK-WOLFE OPTIMIZATION CONCEPT

The FW optimization algorithm is used to solve problems of the form $\min_{g\in\mathcal{G}} F(g)$ for any function $F:\mathcal{G}\to\mathbb{R}$ that is continuously differentiable and convex with $\mathcal{G}$ as the convex set having the inner product $\mathbf{u},\mathbf{v}=\sum u_j v_j$. The working of the standard FW algorithm for minimization problems is shown in Figure 4.3, where for the initial point $g_0\in\mathcal{G}$, the goal is to determine the optimal point g. Iteratively, the algorithm determines the point $s_t\in\mathcal{G}$ such that the inner product of gradient of F at the current point g_t and the current candidate s is minimized, thus utilizing the linearization principle instead of the costly projected gradient. The next iterate g_t+1 is updated by moving toward the direction of s_t using a convex combination of g_t and s_t with ρ_t as the step-size, which can be obtained using line-search. The standard FW algorithm converges at the rate $\mathcal{O}(1/T)$ with T iterations required to achieve convergence [16].

Algorithm IWCV for KLIEP

Input: $\{\mathbf{x}_i^{\text{tr}}\}_{i=1}^{n_{\text{tr}}}, \{\mathbf{x}_j^{\text{te}}\}_{j=1}^{n_{\text{te}}}$, **Output:** $\mathbf{w} = \{w(\mathbf{x}_i^{\text{tr}})\}_{i=1}^{n_{\text{tr}}}$

Split the test data $\{\mathbf{x}_j^{\text{te}}\}_{j=1}^{n_{\text{te}}}$ into T disjoint subsets $\{\mathcal{X}_t^{\text{te}}\}_{t=1}^{T}$
for each candidate σ_m **do**
  for each test split $t = 1,\ldots,T$ **do**
    $w_t(\mathbf{x}) \leftarrow KLIEP\left(\{\mathbf{x}_i^{\text{tr}}\}_{i=1}^{n_{\text{tr}}}, \{\mathcal{X}_j^{\text{te}}\}_{j\neq t}\right)$ using σ_m
    $J_t(m) \leftarrow \frac{1}{|\mathcal{X}_t^{\text{te}}|}\sum_{\mathbf{x}\in\mathcal{X}_t^{\text{te}}} \log w_t(\mathbf{x})$
  end for
  $J(m) \leftarrow \frac{1}{T}\sum_{t=1}^{T} J_t(m)$
end for
Optimum $m' \leftarrow \underset{m}{\text{argmax}}\, J(m)$, $\sigma_{opt} \leftarrow \sigma_{m'}$
Estimate the importance weights $\mathbf{w} \leftarrow KLIEP\left(\{\mathbf{x}_i^{\text{tr}}\}_{i=1}^{n_{\text{tr}}}, \{\mathbf{x}_j^{\text{te}}\}_{j=1}^{n_{\text{te}}}\right)$ using σ_{opt}

FIGURE 4.3 Algorithm for standard Frank-Wolfe optimization.

The FW optimization concept has been applied in the literature for various applications: (i) matrix factorization where the convergence was shown using the approximation quality obtained with the FW duality gap [9], (ii) image and video co-localization formulated using FW leading to an improved computational efficiency [10], (iii) structured support vector machine (SSVM) with the SVM duality gap equivalent to the FW duality gap [11], etc. To remove the influence of "bad" visited vertices, the away-steps FW algorithm [18] was shown to converge at a linear convergence rate in [19].

The iterates g_t obtained by the FW algorithm converges to an optimal value at a linear convergence rate only when this optimal solution belongs to the interior of a polytope. If this is not the case, then the convergence rate can be sublinear because of the zig-zagging effects as demonstrated in [20]. To obtain a linear convergence rate, the FW modifications, namely Away-steps FW and Pairwise FW are suggested in the literature [18, 20–22]. These variants of FW algorithms were studied in detail in [20] and a global linear convergence rate was derived for these modifications.

Figure 4.4 shows the directions of the standard FW algorithm as well as the away-steps and pairwise FW from the current solution α_t with α^* as the optimal solution for maximization problems. The toward direction $\mathrm{d}_t^{\mathrm{FW}}$ is chosen during each iteration by determining the location l_t^{FW} in the standard FW algorithm since this is the direction that maximizes the gradient. But, if the optimal solution lies closer to the boundary of the polytope, the iterates of the standard FW algorithm would zig-zag and thus the algorithm need more iterations to converge [20]. Thus, the away-step direction $\mathrm{d}_t^{\mathrm{AFW}}$ is preferred by moving away from the location l_t^{AFW} that minimizes the gradient. In the pairwise FW variant, the iterates move pairwise from the away direction to the toward direction to obtain the pairwise direction $\mathrm{d}_t^{\mathrm{PFW}} = \mathrm{d}_t^{\mathrm{FW}} + \mathrm{d}_t^{\mathrm{AFW}}$. Since the pairwise FW variant is efficient in arriving at the optimum solution quickly [20], we present the pairwise FW covariate shift algorithm for computing the importance weights [23].

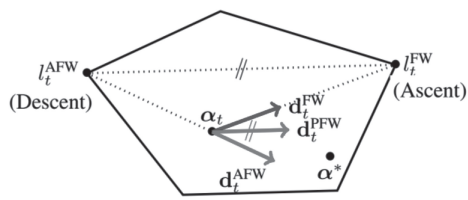


FIGURE 4.4 Toward, away, and pairwise Frank-Wolfe directions for maximization problems [23].

4.3.2 PAIRWISE FRANK-WOLFE COVARIATE SHIFT ALGORITHM

To use the pairwise FW variant for computing the importance weights of KLIEP, the gradient $\mathrm{g} = \left[g_1, \ldots, g_{n_{\mathrm{te}}}\right]^{\mathrm{T}}$ of the objective function and the upper bound $\beta = \left[\beta_1, \ldots, \beta_{n_{\mathrm{te}}}\right]^{\mathrm{T}}$ for the constraint on α is given by Eqn. (12) with $0 \le \alpha_l \le \beta_l$, for $l = 1, \ldots, n_{\mathrm{te}}$.

$$g_l = \frac{\partial F(\alpha)}{\partial \alpha_l} = \sum_{j=1}^{n_{\mathrm{te}}} \frac{\kappa_l\left(\mathbf{x}_j^{\mathrm{te}}\right)}{\sum_{l'=1}^{n_{\mathrm{te}}} \alpha_{l'} \kappa_{l'}\left(\mathbf{x}_j^{\mathrm{te}}\right)}; \beta_l = \frac{n_{\mathrm{tr}}}{\sum_{i=1}^{n_{\mathrm{tr}}} \kappa_l\left(\mathbf{x}_i^{\mathrm{tr}}\right)} \tag{12}$$

Figure 4.5 shows the working of pairwise FW covariate shift (PFWCS) method with the inputs as the train and test data and the output as the importance weight for each input data sample. The iterate α_t can be written as an atomic decomposition given by Eqn. (13) with $\beta^{(l)} = \beta \odot \mathrm{e}^{(l)}$ where the basis vector $e^{(l)}$ has 0 everywhere except 1 at the l index, and $\odot$ refers to the Hadamard product. $\mathcal{S}_t = \{l : \mu_t(l) \neq 0\}$ (the active set) maintains the locations visited by the algorithm up to the t^{th} iteration such that μ_t is non-zero at those locations. In line 1, the upper bound β is computed and t (iteration counter) is set to 0. In line 2, the following quantities are initialized: α (the mixing coefficients), μ (the atoms) and S (the active set).

$$\alpha_t = \sum_{l=1}^{n_{\mathrm{te}}} \mu_t(l) \beta^{(l)}; \text{constraints}: \sum_{l=1}^{n_{\mathrm{te}}} \mu_t(l) = 1, \mu_t(l) \ge 0 \tag{13}$$

Lines 3–15 are repeated till the convergence of F to obtain the optimum α. The gradient g_t of F is calculated with Eqn. (12) using which the toward location l_t^{FW} is determined as the index of the largest element-wise product of g_t and β. Using the FW linear maximization principle, the toward (ascent) direction is obtained as $\mathrm{d}_t^{\mathrm{FW}} = \beta^{\left(l_t^{\mathrm{FW}}\right)} - \alpha_t$. The away direction is also found in line 6 using the location l_t^{AFW}, which corresponds to the index of the smallest element-wise product of g_t and β. A smaller active set S_t is used to find the location l_t^{AFW} making it fundamentally easier to find relative to l_t^{FW}. The away direction is given by $\mathrm{d}_t^{\mathrm{AFW}} = \alpha_t - \beta^{\left(l_t^{\mathrm{AFW}}\right)}$ since we move away from the location of descent. The algorithm proceeds in the pairwise direction given by Eqn. (14) at each iteration t in line 7 instead of either the toward or away direction, i.e., the solution gets closer to the location l_t^{FW} and also moves away from the location l_t^{AFW} in the *same* iteration.

Algorithm Pairwise Frank-Wolfe Covariate Shift (PFWCS)

Input: $\{\mathbf{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}, \{\mathbf{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ **, Output:** $\mathbf{w} = \{w(\mathbf{x}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$

1: Compute upper bound $\boldsymbol{\beta}$ using Eq. (12) and set $t \leftarrow 0$

2: Initialize: $\boldsymbol{\alpha}_0 \leftarrow \mathbf{0}$, $i_\beta \leftarrow \underset{i}{\mathrm{argmin}}(\boldsymbol{\beta})_i$, $\boldsymbol{\alpha}_0(i_\beta) \leftarrow \boldsymbol{\beta}(i_\beta)$, $\boldsymbol{\mu}_0 \leftarrow \mathbf{0}$, $\boldsymbol{\mu}_0(i_\beta) \leftarrow 1$, $\mathcal{S}_0 \leftarrow \{i_\beta\}$

3: **repeat**

4: Compute gradient $\mathbf{g}_t$ using Eq. (12) ▷ using previous $\boldsymbol{\alpha}_t$

5: Find $l_t^{\mathrm{FW}} \leftarrow \underset{l}{\mathrm{argmax}}\,(\mathbf{g}_t \odot \boldsymbol{\beta})_l$;
Compute $\mathbf{d}_t^{\mathrm{FW}} \leftarrow \boldsymbol{\beta}^{(l_t^{\mathrm{FW}})} - \boldsymbol{\alpha}_t$ ▷ toward direction

6: Find $l_t^{\mathrm{AFW}} \leftarrow \underset{l \in \mathcal{S}_t}{\mathrm{argmin}}\,(\mathbf{g}_t \odot \boldsymbol{\beta})_l$;
Compute $\mathbf{d}_t^{\mathrm{AFW}} \leftarrow \boldsymbol{\alpha}_t - \boldsymbol{\beta}^{(l_t^{\mathrm{AFW}})}$ ▷ away direction

7: Compute $\mathbf{d}_t^{\mathrm{PFW}} \leftarrow \mathbf{d}_t^{\mathrm{FW}} + \mathbf{d}_t^{\mathrm{AFW}}$ ▷ pairwise direction

8: Compute $\rho_{\max} \leftarrow \boldsymbol{\mu}_t(l_t^{\mathrm{AFW}})$ ▷ maximum step of pairwise direction

9: Find $\rho_t \leftarrow \underset{\rho \in [0, \rho_{\max}]}{\mathrm{argmax}}\; F(\boldsymbol{\alpha}_t + \rho \mathbf{d}_t^{\mathrm{PFW}})$ ▷ found using line search

10: Update: $\boldsymbol{\alpha}_{t+1} \leftarrow \boldsymbol{\alpha}_t + \rho_t \mathbf{d}_t^{\mathrm{PFW}}$

11: Update: $\boldsymbol{\mu}_{t+1} \leftarrow \boldsymbol{\mu}_t$;
$\boldsymbol{\mu}_{t+1}(l_t^{\mathrm{FW}}) \leftarrow \boldsymbol{\mu}_t(l_t^{\mathrm{FW}}) + \rho_t$;
$\boldsymbol{\mu}_{t+1}(l_t^{\mathrm{AFW}}) \leftarrow \boldsymbol{\mu}_t(l_t^{\mathrm{AFW}}) - \rho_t$

12: Update: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{l : \boldsymbol{\mu}_{t+1}(l) \neq 0\}$

13: **if** $(\rho_t == \rho_{\max})$ **do** ▷ modify the active set
if $(l_t^{\mathrm{FW}} \in \mathcal{S}_t)$ **do**
$\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{t+1} \setminus \{l_t^{\mathrm{AFW}}\}$, ▷ drop step
$\boldsymbol{\mu}_{t+1}(l_t^{\mathrm{AFW}}) \leftarrow 0$
else
$\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{t+1} \cup \{l_t^{\mathrm{FW}}\} \setminus \{l_t^{\mathrm{AFW}}\}$ ▷ swap step

14: $t \leftarrow t + 1$

15: **until** the convergence of objective function F in Eq. (5)

16: Compute weights $\mathbf{w}$ with $\boldsymbol{\alpha}_t$ using Eq. (1)

FIGURE 4.5 Algorithm for pairwise Frank-Wolfe covariate shift.

$$\mathrm{d}_t^{\mathrm{PFW}} = \underbrace{\boldsymbol{\beta}^{\left(l_t^{\mathrm{FW}}\right)} - \boldsymbol{\alpha}_t}_{\blacklozenge} + \underbrace{\boldsymbol{\alpha}_t - \boldsymbol{\beta}^{\left(l_t^{\mathrm{AFW}}\right)}}_{\blacklozenge} = \boldsymbol{\beta}^{\left(l_t^{\mathrm{FW}}\right)} - \boldsymbol{\beta}^{\left(l_t^{\mathrm{AFW}}\right)} = \mathrm{d}_t^{\mathrm{FW}} + \mathrm{d}_t^{\mathrm{AFW}} \tag{14}$$

In lines 8 and 9, the maximum step-size $\rho_{\max}$ guarantees the current solution feasibility with a line search method shown in Figure 4.6, which iterates till the Armijo condition [24] given by Eqn. (15) is satisfied to guarantee an increase in the value of F, where C is the Lipschitz constant.

$$F(\alpha_t + \rho \mathrm{d}_t) \geq F(\alpha_t) + \tau \rho \mathrm{g}_t, \mathrm{d}_t \; \forall \rho \in \left[0, \rho^*\right], \text{where } \rho^* = \frac{2(\tau - 1)\mathrm{g}_t, \mathrm{d}_t}{C\mathrm{d}_{t2}^2} \tag{15}$$

Algorithm Armijo Line Search

Input: $\alpha_t, \mathbf{d}_t, F, \mathbf{g}_t, \rho_{\max}, \tau \in (0, 1), \xi \in (0, 1)$, **Output:** ρ_t

1: Set $\rho \leftarrow \rho_{\max}$
2: **while** $F(\alpha_t + \rho \mathbf{d}_t) < F(\alpha_t) + \tau \rho \langle \mathbf{g}_t, \mathbf{d}_t \rangle$ **do**
3: $\quad \rho \leftarrow \xi \rho$
4: **end while**
5: $\rho_t \leftarrow \rho$

FIGURE 4.6 Algorithm for Armijo line search.

For any ρ_t of the PFWCS algorithm, we have:

$$\begin{aligned}
&\alpha_t + \rho_t \mathrm{d}_t^{\mathrm{PFW}} = \alpha_t + \rho_t \left(\beta^{\left(l_t^{\mathrm{FW}}\right)} - \beta^{\left(l_t^{\mathrm{AFW}}\right)} \right) = \sum_{l=1}^{n_{\mathrm{te}}} \mu_t(l) \beta^{(l)} + \rho_t \left(\beta^{\left(l_t^{\mathrm{FW}}\right)} - \beta^{\left(l_t^{\mathrm{AFW}}\right)} \right) \\
&= \sum_{l \neq l_t^{\mathrm{FW}}, l_t^{\mathrm{AFW}}} \mu_t(l) \beta^{(l)} + \mu_t\left(l_t^{\mathrm{FW}}\right) \beta^{\left(l_t^{\mathrm{FW}}\right)} + \mu_t\left(l_t^{\mathrm{AFW}}\right) \beta^{\left(l_t^{\mathrm{AFW}}\right)} + \rho_t \left(\beta^{\left(l_t^{\mathrm{FW}}\right)} - \beta^{\left(l_t^{\mathrm{AFW}}\right)} \right) \\
&= \sum_{l \neq l_t^{\mathrm{FW}}, l_t^{\mathrm{AFW}}} \mu_t(l) \beta^{(l)} + \left\{ \mu_t\left(l_t^{\mathrm{FW}}\right) + \rho_t \right\} \beta^{\left(l_t^{\mathrm{FW}}\right)} + \left\{ \mu_t\left(l_t^{\mathrm{AFW}}\right) - \rho_t \right\} \beta^{\left(l_t^{\mathrm{AFW}}\right)} \\
&= \sum_{l=1}^{n_{\mathrm{te}}} \hat{\mu}_t^{\mathrm{PFW}}(l) \beta^{(l)}
\end{aligned} \tag{16}$$

To guarantee a feasible solution, it is required that $\sum_{l=1}^{n_{\mathrm{te}}} \hat{\mu}_t^{\mathrm{PFW}}(l) = 1$ and $\hat{\mu}_t^{\mathrm{PFW}}(l) \geq 0$. From Eqn. (16), it follows that:

$$\begin{aligned}
\sum_{l=1}^{n_{\mathrm{te}}} \hat{\mu}_t^{\mathrm{PFW}}(l) &= \sum_{l \neq l_t^{\mathrm{FW}}, l_t^{\mathrm{AFW}}} \mu_t(l) + \left\{ \mu_t\left(l_t^{\mathrm{FW}}\right) + \rho_t \right\} + \left\{ \mu_t\left(l_t^{\mathrm{AFW}}\right) - \rho_t \right\} \\
&= \sum_{l \neq l_t^{\mathrm{FW}}, l_t^{\mathrm{AFW}}} \mu_t(l) + \rho_t - \rho_t = 1, since \sum_{l=1}^{n_{\mathrm{te}}} \mu_t(l) = 1
\end{aligned}$$

Since $\hat{\mu}_t^{\mathrm{PFW}}(l) \geq 0 \forall l$, the maximum value of the step-size $\rho_{\max}$ is given by Eqn. (17).

$$\mu_t\left(l_t^{\mathrm{AFW}}\right)-\rho_{\max}=0\therefore\rho_{\max}=\mu_t\left(l_t^{\mathrm{AFW}}\right) \tag{17}$$

After obtaining the optimum step size using the line search algorithm, α, μ and S are updated using Eqns. (18)–(20). Due to the use of the pairwise direction instead of the toward or away directions, only the atoms related to l_t^{FW} and l_t^{AFW} are updated in line 11 without changing the other atoms, resulting in higher computational efficiency. In contrast, the standard FW and its away-steps variant would require to update *all* atoms in each iteration.

$$\alpha_{t+1}=\alpha_t+\rho_t\mathrm{d}_t^{\mathrm{PFW}} \tag{18}$$

$$\mu_{t+1}(l)=\begin{cases}\mu_t(l)\,if\ l\neq l_t^{\mathrm{FW}}\ and\ l\neq l_t^{\mathrm{AFW}},\\ \mu_t(l)+\rho_t\ if\ l=l_t^{\mathrm{FW}},\\ \mu_t(l)-\rho_t\ if\ l=l_t^{\mathrm{AFW}}\end{cases} \tag{19}$$

$$\mathcal{S}_{t+1}=\begin{cases}\mathcal{S}_t\setminus\left\{l_t^{\mathrm{AFW}}\right\}\ if\ \rho_t=\rho_{\max}\ and\ l_t^{\mathrm{FW}}\in\mathcal{S}_t\ \text{(drop step)},\\ \mathcal{S}_t\cup\left\{l_t^{\mathrm{FW}}\right\}\setminus\left\{l_t^{\mathrm{AFW}}\right\}\ if\ \rho_t=\rho_{\max}\ and\ l_t^{\mathrm{FW}}\notin\mathcal{S}_t\ \text{(swap step)},\\ \mathcal{S}_t\ if\ \rho_t<\rho_{\max}\ and\ l_t^{\mathrm{FW}}\in\mathcal{S}_t,\\ \mathcal{S}_t\cup\left\{l_t^{\mathrm{FW}}\right\}\ if\ \rho_t<\rho_{\max}\ and\ l_t^{\mathrm{FW}}\notin\mathcal{S}_t\end{cases} \tag{20}$$

The PFWCS algorithm can take several steps as follows:

- *Good step*, if rmax = 1, or rmax < 1 and rt < rmax;
- *Drop step*, if rt = rmax < 1 and $l_t^{\mathrm{FW}}\in\mathcal{S}_t$; in this case, $\mu_t\left(l_t^{\mathrm{AFW}}\right)=0$ as well as the location l_t^{AFW} is removed from the active set St;
- *Swap step*, if rt = rmax < 1 and $l_t^{FW}\notin S_t$; in this case, the locations l_t^{AFW} and l_t^{FW} are swapped in St.

For the PFWCS algorithm, with the set $\mathcal{A}\subseteq\mathbb{R}^d$ consisting of a finite number of atoms, the number of good steps, drop steps and swap steps till the t^{th} iteration are bounded by $t/\left(3|\mathcal{A}|!+1\right)$, $t/2$ and $3|\mathcal{A}|!$, respectively [20]. The computational complexity of the PFWCS algorithm is $\mathcal{O}(1/t)$ since the error function $h_t=F(\alpha_t)-\mathrm{F}(\alpha^*)$ for an optimal solution α^* follows a linear convergence rate $h_t<h_0\exp(-vs(t))$ where v depends on the curvature constant of the objective function F.

To determine the importance weights $\mathrm{w} = \left\{ w\left(\mathrm{x}_i^{\mathrm{tr}}\right) \right\}_{i=1}^{n_{\mathrm{tr}}}$, we can use the PFWCS algorithm instead of the original KLIEP algorithm as illustrated in the next subsection and the optimum tuning σ parameter can be obtained using the importance-weighted cross-validation as before using PFWCS algorithm in the place of KLIEP. The KLIEP algorithm uses only a random subset of the test data for the Gaussian centers to reduce the computational overhead; however, this may not be an optimal approach due to the randomness involved. An advantage of the PFWCS algorithm is that the entire test data set can be used for determining the optimum mixing coefficients, since during each iteration, only one Gaussian center is activated due to the FW linear maximization and minimization principle.

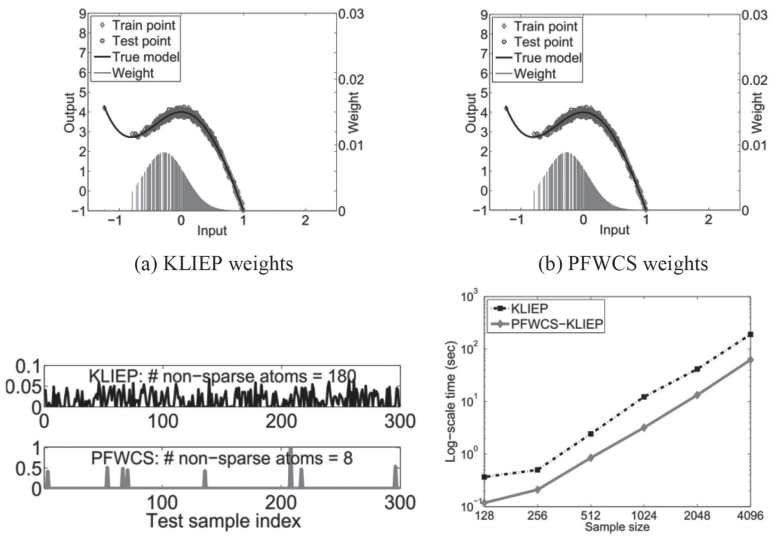
4.3.3 ILLUSTRATIVE RESULTS OF KLIEP AND PFWCS

To evaluate the performance of PFWCS algorithm relative to the original KLIEP algorithm, synthetic data is generated using $y = 1 - 2x^3 + 3\ \mathrm{sinc}\ (x) + \in$ with the noise $\in \sim \mathcal{N}\left(0, 0.1^2\right)$. For simulating the covariate shift, 500 training data points are sampled from $\mathcal{N}\left(0.5, 0.5^2\right)$ and 300 testing data points are sampled from $\mathcal{N}\left(0, 0.3^2\right)$. Figures 4.7(a) and 4.7(b) shows the importance weights of KLIEP as well as PFWCS algorithms on this synthetic data, where it is observed that both algorithms obtain identical importance weights; however, as shown in Figure 4.7(c), the PFWCS algorithm results in significantly sparser mixing coefficients α. Figure 4.7(d) illustrates the log-scale run-time with changing sample size of the synthetic data ($n_{\mathrm{tr}} = n_{\mathrm{te}}$). It is observed that the PFWCS algorithm is consistently faster relative to the KLIEP algorithm for estimating the importance weights. Thus, the PFWCS algorithm is computationally efficient and obtains sparse solutions.

4.4 IMPORTANCE WEIGHTED STRUCTURED REGRESSION

The regression theory aims at predicting a real value from the observed variables. Specifically, structured regression is applicable when the output target variable is multi-dimensional and these target dimensions may be dependent on each other. Some examples are i) human pose estimation is a structured regression problem in computer vision since the target pose is multi-dimensional and ii) music mood estimation is a structured regression

problem in music information retrieval since the music mood represented as a numerical vector value on the two-dimensional valence arousal space is correlated to a certain degree between valence and arousal [25]. Instead of estimating inaccurately each target dimension separately, structured regression aims to consider the dependency between the target dimensions to improve the prediction performance. We discuss TGP structured regression concept proposed in [12] to address this aspect and apply the PFWCS importance weights to correct the covariate shift during the learning of structured regression.



(a) KLIEP weights (b) PFWCS weights

(c) Sparsity of $\boldsymbol{\alpha}$ (d) Run-time to determine $\mathbf{w}$

FIGURE 4.7 Comparison of KLIEP and PFWCS algorithms.

4.4.1 GAUSSIAN PROCESS REGRESSION

Gaussian process regression (GPR) is used to describe distributions over functions and can be interpreted as a nonparametric Bayesian version of support vector regression (SVR) [26]. Formally, the gaussian process (GP) is defined as "a collection of random variables any finite number of which has a joint Gaussian distribution" [26]. Consider the feature matrix $\mathrm{X} \in \mathbb{R}^{N \times D}$ of N input instances with D dimensions. Any finite number of function values $f(\mathrm{x})$ has a joint multivariate normal (MVN) distribution defined by the mean

function $\mu:\mathbb{R}^D \to \mathbb{R}$ and the covariance function $\Sigma:\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ with the elements $\Sigma_{ij} = \kappa(\mathrm{x}_i, \mathrm{x}_j)$ where κ is any kernel function satisfying Mercer's conditions. The GP defines a prior over functions given by Eqn. (21) using which the posterior predictive density can be obtained using the observed training data.

$$f(\mathrm{x}) \sim GP(m(\mathrm{x}), \kappa(\mathrm{x}, \mathrm{x}'))$$
$$m(\mathrm{x}) = E[f(\mathrm{x})]; \kappa(\mathrm{x}_i, \mathrm{x}_j) = E\left[(f(\mathrm{x}_i) - m(\mathrm{x}))(f(\mathrm{x}_j) - m(\mathrm{x}))^{\mathrm{T}}\right] \tag{21}$$

Consider the noisy observations with training data $\mathcal{D} = \{(\mathrm{x}_i, \mathrm{y}_i = f(\mathrm{x}_i) + \in)\}_{i=1}^N$ where $\in \sim \mathcal{N}(0, \sigma_y^2)$. For the test data $\mathrm{X}_* \in \mathbb{R}^{N_* \times D}$, the task is to predict the corresponding outputs y_*. Using the GP definition, the joint distribution is given by Eqn. (22) assuming zero-mean, where $\mathrm{K} = \kappa(\mathrm{X}, \mathrm{X}) \in \mathbb{R}^{N \times N}$ is the Gram matrix, $\mathrm{K}_* = \kappa(\mathrm{X}, \mathrm{X}_*) \in \mathbb{R}^{N \times N_*}$, $\mathrm{K}_{**} = \kappa(\mathrm{X}_*, \mathrm{X}_*) \in \mathbb{R}^{N_* \times N_*}$ and I_N is the identity matrix.

$$\begin{pmatrix} \mathrm{y} \\ \mathrm{y}_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathrm{K} + \sigma_y^2 \mathrm{I}_N & \mathrm{K}_* \\ \mathrm{K}_*^{\mathrm{T}} & \mathrm{K}_{**} \end{bmatrix}\right) \tag{22}$$

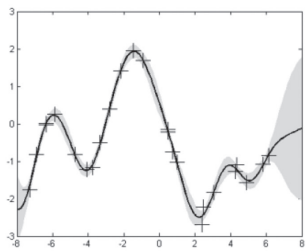
Using the Bayes' rule, the posterior predictive density is given by Eqn. (23) and for a single test sample, the posterior is given by Eqn. (24) where $\mathrm{k}_* = [\kappa(\mathrm{x}_*, \mathrm{x}_1), \ldots, \kappa(\mathrm{x}_*, \mathrm{x}_N)] \in \mathbb{R}^N$ and $k_{**} = \kappa(\mathrm{x}_*, \mathrm{x}_*) \in \mathbb{R}$. The log marginal likelihood of GP is given by Eqn. (25), where the first term is the only term that involves the output data and controls the quality of model fit, the second (regularization) term penalizes the complexity of the GP model and the third term is a constant.

$$p(\mathrm{y}_* \mid \mathrm{X}_*, \mathrm{X}, \mathrm{y}) = N(\mathrm{y}_* \mid \mu_*, \Sigma_*),$$
$$\mu_* = \mathrm{K}_*^{\mathrm{T}} \left(\mathrm{K} + \sigma_y^2 \mathrm{I}_N\right)^{-1} y,\ \Sigma_* = \mathrm{K}_{**} - \mathrm{K}_*^{\mathrm{T}} \left(\mathrm{K} + \sigma_y^2 \mathrm{I}_N\right)^{-1} \mathrm{K}_* \tag{23}$$

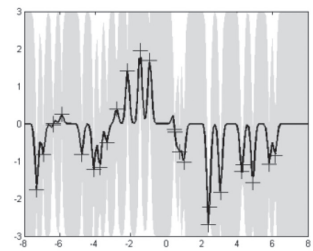
$$p(y_* \mid \mathrm{x}_*, \mathrm{X}, \mathrm{y}) = \mathcal{N}\left(y_* \mid \mathrm{k}_*^{\mathrm{T}} \left(\mathrm{K} + \sigma_y^2 \mathrm{I}_N\right)^{-1} \mathrm{y}, k_{**} - \mathrm{k}_*^{\mathrm{T}} \left(\mathrm{K} + \sigma_y^2 \mathrm{I}_N\right)^{-1} \mathrm{k}_*\right) \tag{24}$$

$$\log p(\mathrm{y} \mid \mathrm{X}) = -\frac{1}{2} \mathrm{y}^{\mathrm{T}} \left(\mathrm{K} + \sigma_y^2 \mathrm{I}_N\right)^{-1} \mathrm{y} - \frac{1}{2} \log \left|\mathrm{K} + \sigma_y^2 \mathrm{I}_N\right| - \frac{N}{2} \log 2\pi \tag{25}$$

The performance of GPR depend on the kernel choice as well as its hyper-parameters. For the squared exponential kernel $\kappa(\mathrm{x}_i, \mathrm{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\mathrm{x}_i - \mathrm{x}_j^{\,2}\right)$, l represents the function's horizontal scale, σ_f^2 and σ_y^2 refers to the function's vertical scale and the noise variance, respectively. Overfitting is a less significant problem in GP since a relatively small number of parameters needs to be estimated. An illustration of the GP prediction with various hyper-parameter settings is shown in Figure 4.8, where the result varies from a smooth fit to a wiggly fit.



(a) $(l, \sigma_f, \sigma_y) = (1, 1, 0.1)$



(b) $(l, \sigma_f, \sigma_y) = (0.1, 1.5, 0.0002)$

FIGURE 4.8 1D GPs fitted to 25 noisy observations using a squared exponential 1 kernel.

An algorithm to compute the mean, variance, and the log marginal likelihood of GPR is shown in Figure 4.9, where Cholesky decomposition is used for $\mathrm{K}_y = \mathrm{K} + \sigma_y^2\mathrm{I} = \mathrm{LL}^{\mathrm{T}}$ to determine the inverse of K_y instead of directly finding the inverse to avoid numerical instability. The computational complexity is $\mathcal{O}(N^3)$ for the Cholesky decomposition and $\mathcal{O}(N^2)$ for $\zeta = \mathrm{K}_y^{-1}\mathrm{y}$.

Algorithm Gaussian Process Regression

Input: $\mathbf{X}$ (features), $\mathbf{y}$ (target), κ (kernel), σ_y^2 (noise variance), $\mathbf{x}_*$ (test data)

Output: Mean, variance and log marginal likelihood

1: $\mathbf{L} = \mathrm{chol}(\mathbf{K} + \sigma_y^2\mathbf{I})$ ▷ Cholesky decomposition

2: $\boldsymbol{\zeta} = \mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{y} = \mathbf{L}^{\top} \setminus (\mathbf{L} \setminus \mathbf{y})$

3: Predictive mean: $\mathbb{E}[\mathbf{x}_*] = \mathbf{k}_*^{\top}\boldsymbol{\zeta}$

4: Predictive variance: $\mathbb{V}[\mathbf{x}_*] = k_{**} - \mathbf{k}_*^{\top}\mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{k}_*$

5: Log marginal likelihood: $\log p(\mathbf{y} \mid \mathbf{X}) = -\frac{1}{2}\mathbf{y}^{\top}\boldsymbol{\zeta} - \sum_i \log L_{ii} - \frac{N}{2}\log(2\pi)$

FIGURE 4.9 Algorithm for gaussian process regression.

To maximize the log marginal likelihood, the gradient with respect to the hyper-parameters of the kernel denoted by θ is computed by Eqn. (26). The value for $\frac{\partial \mathbf{K}_y}{\partial \theta_j}$ depends on the kernel used in GP and any standard gradient-based optimizer can be used to estimate the hyper-parameters of GP.

$$\begin{aligned}\frac{\partial}{\partial \theta_j}\log p(\mathbf{y}\mid \mathbf{X}) &= \frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_j}\mathbf{K}_y^{-1}y - \frac{1}{2}tr\left(\mathbf{K}_y^{-1}\frac{\partial \mathbf{K}_y}{\partial \theta_j}\right)\\ &= \frac{1}{2}tr\left(\left(\zeta\alpha^{\mathrm{T}} - \mathbf{K}_y^{-1}\right)\frac{\partial \mathbf{K}_y}{\partial \theta_j}\right)\end{aligned} \tag{26}$$

4.4.2 TWIN GAUSSIAN PROCESS REGRESSION

The conventional GPR technique does not model any dependency existing in the structured (multi-dimensional) output, and thus TGP can be used that models this dependency between the target dimensions. The TGP structured regression based on Kullback-Leibler divergence [12] uses the input joint probability distribution given by $p(\mathrm{X},\mathrm{x}) = \mathcal{N}_{\mathrm{X}}\left(0,\mathrm{K}_{\mathrm{X}\cup x}\right)$ and the output joint probability distribution given by $p(\mathrm{Y},\mathrm{y}) = \mathcal{N}_{\mathrm{Y}}\left(0,\mathrm{K}_{\mathrm{Y}\cup y}\right)$. The joint kernels are defined by Eqn. (27) where x is the test data corresponding to the unknown structured output y. The input and output Gaussian similarity kernels are given by Eqn. (28), where λ_{x} and λ_{Y} are the regularization parameters, ρ_{x} and ρ_{Y} correspond to the kernel bandwidths, and δ is the Kronecker delta function. Here, K_{X} is a $N \times N$ kernel matrix, $\mathrm{k}_{\mathrm{X}}^{\mathrm{x}}$ is a vector with elements given by $\left(\mathrm{k}_{\mathrm{X}}^{\mathrm{x}}\right)_i = \mathrm{K}_{\mathrm{X}}\left(\mathrm{x}_i,\mathrm{x}\right)$, $k_{\mathrm{x}}(\mathrm{x},\mathrm{x})$ is a scalar, and thus $\mathrm{K}_{\mathrm{X}\cup x} \in \mathbb{R}^{(N+1)\times(N+1)}$. The output kernels $\mathrm{K}_{\mathrm{YUy}}$, K_{Y}, $\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}$ and $k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})$ can be similarly defined.

$$\mathrm{K}_{\mathrm{X}\cup x} = \begin{bmatrix}\mathrm{K}_{\mathrm{X}} & \mathrm{k}_{\mathrm{X}}^{\mathrm{x}}\\ \mathrm{k}_{\mathrm{X}}^{\mathrm{xT}} & k_{\mathrm{X}}\left(\mathrm{x},\mathrm{x}\right)\end{bmatrix}, \mathrm{K}_{\mathrm{Y}\cup y} = \begin{bmatrix}\mathrm{K}_{\mathrm{Y}} & \mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\\ \mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}} & k_{\mathrm{Y}}\left(\mathrm{y},\mathrm{y}\right)\end{bmatrix} \tag{27}$$

$$\left(\mathrm{K}_{\mathrm{X}}\right)_{ij} = \exp\left(-\frac{\mathrm{x}_i - \mathrm{x}_j{}^2}{2\rho_{\mathrm{X}}^2}\right) + \lambda_{\mathrm{X}}\delta_{ij}; \left(\mathrm{K}_{\mathrm{Y}}\right)_{ij} = \exp\left(-\frac{\mathrm{y}_i - \mathrm{y}_j{}^2}{2\rho_{\mathrm{Y}}^2}\right) + \lambda_{\mathrm{Y}}\delta_{ij} \tag{28}$$

The goal of KLTGP is to ensure that (i) the training data samples distant from the test data should be given less importance towards prediction, (ii) the training data samples close to the test data but with corresponding training outputs distant from the test output should also be given less importance towards prediction, and (iii) the similar training and test inputs and outputs should contribute significantly for the prediction [12]. KLTGP minimizes

the KL divergence between the marginal input and output GPes and the predicted multivariate output is obtained using the optimization problem given by Eqn. (29), where $\mathrm{u}_{\mathrm{x}} = \mathrm{K}_{\mathrm{X}}^{-1}\mathrm{k}_{\mathrm{X}}^{\mathrm{x}}$ and $\eta_{\mathrm{x}} = k_{\mathrm{X}}(\mathrm{x},\mathrm{x}) - \mathrm{k}_{\mathrm{X}}^{\mathrm{xT}}\mathrm{u}_{\mathrm{x}}$. The gradient with respect to the r^{th} multivariate output dimension is given by Eqn. (30). For Gaussian kernels, $\frac{\partial k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})}{\partial y^{(r)}} = 0$ and the remaining partial derivatives are given by Eqn. (31).

$$\hat{\mathrm{y}} = \mathrm{argmin}_{\mathrm{y}}\left[L_{KL}(\mathrm{x},\mathrm{y}) = k_{\mathrm{Y}}(\mathrm{y},\mathrm{y}) - 2\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{u}_{\mathrm{x}} - \eta_{\mathrm{x}}\log\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y}) - \mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)\right] \tag{29}$$

$$\frac{\partial L_{KL}(\mathrm{x},\mathrm{y})}{\partial y^{(r)}} = \frac{\partial k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})}{\partial y^{(r)}} - 2\mathrm{u}_{\mathrm{x}}^{\mathrm{T}}\frac{\partial \mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}}{\partial y^{(r)}} - \eta_{\mathrm{x}}\frac{\log\left[\frac{\partial k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})}{\partial y^{(r)}} - 2\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\frac{\partial \mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}}{\partial y^{(r)}}\right]}{\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y}) - \mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)} \tag{30}$$

$$\frac{\partial \mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}}{\partial y^{(r)}} = \begin{bmatrix} -\frac{1}{\rho_{\mathrm{Y}}^2}\left(y_1^{(r)} - y^r\right)k_{\mathrm{Y}}(\mathrm{y}_1,\mathrm{y}) \\ -\frac{1}{\rho_{\mathrm{Y}}^2}\left(y_2^{(r)} - y^r\right)k_{\mathrm{Y}}(\mathrm{y}_2,\mathrm{y}) \\ \cdots \\ -\frac{1}{\rho_{\mathrm{Y}}^2}\left(y_N^{(r)} - y^r\right)k_{\mathrm{Y}}(\mathrm{y}_N,y) \\ \\ \end{bmatrix} \tag{31}$$

Another TGP model based on the generalized Sharma-Mittal divergence (SMTGP) was proposed in [27] with two parameters (τ,ψ) and was found to perform better than KLTGP on several datasets. The Sharma-Mittal (SM) divergence between two Gaussian distributions $\mathcal{N}_p = \mathcal{N}(\mu_p,\Sigma_p)$ and $\mathcal{N}_q = \mathcal{N}(\mu_q,\Sigma_q)$ is given by Eqn. (32) where $\Delta\mu = \mu_p - \mu_q$, $|\cdot|$denotes the matrix determinant and $\tau\Sigma_p^{-1} + (1-\tau)\Sigma_q^{-1}$ is a positive definite matrix.

$$L_{\tau,\psi}(\mathcal{N}_p,\mathcal{N}_q)\& = \frac{1}{\psi-1}\left[\left(\frac{|\Sigma_p|^{\tau}|\Sigma_q|^{1-\tau}}{\left|\left(\tau\Sigma_p^{-1} + (1-\tau)\Sigma_q^{-1}\right)^{-1}\right|}\right)^{\frac{1-\psi}{2(1-\tau)}} \times \exp\left(-\frac{\tau(1-\psi)}{2}\Delta\mu^{\mathrm{T}}\left(\tau\Sigma_p^{-1} + (1-\tau)\Sigma_q^{-1}\right)^{-1}\Delta\mu\right) - 1\right] \tag{32}$$

Applying the closed-form expression of SM divergence given in Eqn. (32) between the input and output joint probability distributions with $\Delta\mu = 0$ results in

$$L_{\tau,\psi}\left(p(\mathrm{X},\mathrm{x}),p(\mathrm{Y},\mathrm{y})\right)=\frac{1}{\psi-1}\left[\left(\frac{\left|\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}\right|^{\tau}\left|\mathrm{K}_{\mathrm{Y}\cup y}\right|^{1-\tau}}{\left|\left(\tau\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}^{-1}+(1-\tau)\mathrm{K}_{\mathrm{Y}\cup y}^{-1}\right)^{-1}\right|}\right)^{-\frac{1-\psi}{2(1-\tau)}}-1\right] \tag{33}$$

Using the concept from matrix algebra that $\left|\mathrm{A}^{-1}\right|=1/|\mathrm{A}|=|\mathrm{A}|^{-1}$ and $|\mathrm{AB}|=|\mathrm{A}||\mathrm{B}|$, the SM divergence under the TGP setting is simplified as Eqn. (34) and applying this to Eqn. (33) results in Eqn. (35).

$$L_{\tau,\psi}\left(\mathcal{N}_p,\mathcal{N}_q\right)=\frac{1}{\psi-1}\left[\left(\frac{\left|\Sigma_p\right|^{1-\tau}\left|\Sigma_q\right|^{\tau}}{\left|\tau\Sigma_q+(1-\tau)\Sigma_p\right|}\right)^{\frac{1-\psi}{2(1-\tau)}}-1\right] \tag{34}$$

$$L_{\tau,\psi}\left(p(\mathrm{X},\mathrm{x}),p(\mathrm{Y},\mathrm{y})\right)=\frac{1}{\psi-1}\left[\left(\frac{\left|\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}\right|^{1-\tau}\left|\mathrm{K}_{\mathrm{Y}\cup y}\right|^{\tau}}{\left|(1-\tau)\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}+\tau\mathrm{K}_{\mathrm{Y}\cup y}\right|}\right)^{\frac{1-\psi}{2(1-\tau)}}-1\right] \tag{35}$$

Since the joint kernel matrix $\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}$ is square and non-singular, it can be decomposed as Eqn. (36) using the Aitken block-diagonalization formula [28]. Here, $\eta_{\mathrm{x}}=k_{\mathrm{X}}(\mathrm{x},\mathrm{x})-\mathrm{k}_{\mathrm{X}}^{\mathrm{xT}}\mathrm{K}_{\mathrm{X}}^{-1}\mathrm{k}_{\mathrm{X}}^{\mathrm{x}}$ refers to the Schur complement of $\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}$ whose determinant is $\left|\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}\right|=|\mathrm{K}_{\mathrm{X}}|\times\eta_{\mathrm{x}}$. Similarly, $\eta_{\mathrm{y}}=k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})-\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}$ refers to the Schur complement of $\mathrm{K}_{\mathrm{Y}\cup\mathrm{y}}$ whose determinant is $\left|\mathrm{K}_{\mathrm{Y}\cup y}\right|=|\mathrm{K}_{\mathrm{Y}}|\times\eta_{\mathrm{y}}$.

$$\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}=\begin{bmatrix}\mathrm{I} & 0\\ \mathrm{k}_{\mathrm{X}}^{\mathrm{xT}}\mathrm{K}_{\mathrm{X}}^{-1} & \mathrm{I}\end{bmatrix}\begin{bmatrix}\mathrm{K}_{\mathrm{X}} & 0\\ 0 & \eta_{\mathrm{x}}\end{bmatrix}\begin{bmatrix}\mathrm{I} & \mathrm{K}_{\mathrm{X}}^{-1}\mathrm{k}_{\mathrm{X}}^{\mathrm{x}}\\ 0 & \mathrm{I}\end{bmatrix} \tag{36}$$

The Schur complements η_x and η_y can be interpreted as the change in the variance of GP relative to the test data x. Both η_x and η_y have an upper bound of $k_{\mathrm{X}}(\mathrm{x},\mathrm{x})$ and $k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})$ and they decrease when the test data x becomes close to X. The terms of Eqn. (35) can be written as:

$$\left|\mathrm{K}_{\mathrm{Y}\cup y}\right|^{\frac{\tau(1-\psi)}{2(1-\tau)}}=|\mathrm{K}_{\mathrm{Y}}|^{\frac{\tau(1-\psi)}{2(1-\tau)}}\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})-\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)^{\frac{\tau(1-\psi)}{2(1-\tau)}}$$

and

$$\left|(1-\tau)\mathrm{K}_{\mathrm{X}\cup\mathrm{x}}+\tau\mathrm{K}_{\mathrm{Y}\cup y}\right|^{-\frac{(1-\psi)}{2(1-\tau)}}=\left|(1-\tau)\mathrm{K}_{\mathrm{X}}+\tau\mathrm{K}_{\mathrm{Y}}\right|^{-\frac{(1-\psi)}{2(1-\tau)}}$$
$$\times\left(k_{\mathrm{XY}}^{\tau}-\mathrm{k}_{\mathrm{XY}}^{\mathrm{xyT}}\left((1-\tau)\mathrm{K}_{\mathrm{X}}+\tau\mathrm{K}_{\mathrm{Y}}\right)^{-1}\mathrm{k}_{\mathrm{XY}}^{\mathrm{xy}}\right)^{-\frac{(1-\psi)}{2(1-\tau)}}$$

where $k_{\mathrm{XY}}^{\tau}=(1-\tau)k_{\mathrm{X}}(\mathrm{x},\mathrm{x})+\tau k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})\in\mathbb{R}$, $\mathrm{k}_{\mathrm{XY}}^{\mathrm{xy}}=(1-\tau)\mathrm{k}_{\mathrm{X}}^{\mathrm{x}}+\tau\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\in\mathbb{R}^{N}$ and $\mathrm{K}_{\mathrm{XY}}=(1-\tau)\mathrm{K}_{\mathrm{X}}+\tau\mathrm{K}_{\mathrm{Y}}\in\mathbb{R}^{N\times N}$. Ignoring the constants $\left|\mathrm{K}_{\mathrm{X}\cup\mathrm{X}}\right|^{\frac{1-\psi}{2}}$, $\left|\mathrm{K}_{\mathrm{Y}}\right|^{\frac{\tau(1-\psi)}{2(1-\tau)}}$, $\left|(1-\tau)\mathrm{K}_{\mathrm{X}}+\tau\mathrm{K}_{\mathrm{Y}}\right|^{-\frac{(1-\psi)}{2(1-\tau)}}$ and (–1) that do not depend on the unknown y, the SMTGP optimization function to determine the predicted output is given by Eqn. (37).

$$\hat{\mathrm{y}}=\mathrm{argmin}_{\mathrm{y}}\frac{1}{\psi-1}\left[\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})-\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)^{\frac{\tau(1-\psi)}{2(1-\tau)}}\times\left(k_{\mathrm{XY}}^{\tau}-\mathrm{k}_{\mathrm{XY}}^{\mathrm{xyT}}\mathrm{K}_{\mathrm{XY}}^{-1}\mathrm{k}_{\mathrm{XY}}^{\mathrm{xy}}\right)^{-\frac{(1-\psi)}{2(1-\tau)}}\right] \tag{37}$$

The gradient of the SMTGP optimization function with respect to the r^{th} structured output dimension is given by Eqn. (38) obtained using chain-rule and rearrangement.

$$\begin{aligned}\frac{\partial L_{\tau,\psi}}{\partial y^{(r)}}&=\frac{-\tau}{2(1-\tau)}\&\left[\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})-\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)^{\frac{\tau(1-\psi)}{2(1-\tau)}-1}\right.\\&\left.\times\left(-2\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\frac{\partial\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}}{\partial y^{(r)}}\right)\times\left(k_{\mathrm{XY}}^{\tau}-\mathrm{k}_{\mathrm{XY}}^{\mathrm{xyT}}\mathrm{K}_{\mathrm{XY}}^{-1}\mathrm{k}_{\mathrm{XY}}^{\mathrm{xy}}\right)^{-\frac{(1-\psi)}{2(1-\tau)}}\right]\\&+\frac{1}{2(1-\tau)}\left[\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})-\mathrm{k}_{\mathrm{Y}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)^{\frac{\tau(1-\psi)}{2(1-\tau)}}\right.\\&\left.\times\left(k_{\mathrm{XY}}^{\tau}-\mathrm{k}_{\mathrm{XY}}^{\mathrm{xyT}}\mathrm{K}_{\mathrm{XY}}^{-1}\mathrm{k}_{\mathrm{XY}}^{\mathrm{xy}}\right)^{-\frac{(1-\psi)}{2(1-\tau)}-1}\times\left(-2\mathrm{k}_{\mathrm{XY}}^{\mathrm{xyT}}\mathrm{K}_{\mathrm{XY}}^{-1}\tau\frac{\partial\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}}{\partial y^{(r)}}\right)\right]\end{aligned} \tag{38}$$

4.4.3 DIRECT TGP

An efficient version of KLTGP known as direct TGP (DTGP) was proposed in [29] using simple algebraic manipulations of the KLTGP optimization problem. Instead of directly optimizing for y, DTGP optimizes the output kernel function $\mathrm{t}_{\mathrm{y}}=\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}$ having elements $\left(\mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}\right)_i=\mathrm{K}_{\mathrm{Y}}(\mathrm{y}_i,\mathrm{y})$. After estimating t_{y}, the top P nearest outputs are obtained by ranking based on t_{y} and then the prediction is obtained as the weighted sum over these P nearest neighbors (NN). The optimization function of DTGP is given by Eqn. (39) where $k_{\mathrm{Y}}(\mathrm{y},\mathrm{y})=1+\lambda_{\mathrm{Y}}$ due to Gaussian kernels.

$$\min_{\mathrm{t_y}} \left[1+\lambda_{\mathrm{Y}}-2\mathrm{t}_{\mathrm{Y}}^{\mathrm{T}}\mathrm{u}_{\mathrm{x}}-\eta_{\mathrm{x}}\log\left(1+\lambda_{\mathrm{Y}}-\mathrm{t}_{\mathrm{Y}}^{\mathrm{T}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{t}_{\mathrm{y}}\right)\right] \quad s.t. 0 \le t_{\mathrm{y},i} \le 1+\lambda_{\mathrm{Y}}, i=1,2,\ldots,N \tag{39}$$

Differentiating Eqn. (39) with respect to $\mathrm{t_y}$ without considering the constraints and equating to zero results in Eqn. (40), where μ is a scalar given by Eqn. (41).

$$\mathrm{t}_{\mathrm{y}} = \mu \mathrm{K}_{\mathrm{Y}} \mathrm{u}_{\mathrm{x}} \tag{40}$$

$$\mu = \frac{1+\lambda_{\mathrm{Y}}-\mathrm{t}_{\mathrm{Y}}^{\mathrm{T}}\mathrm{K}_{\mathrm{Y}}^{-1}\mathrm{t}_{\mathrm{y}}}{\eta_{\mathrm{x}}} \tag{41}$$

Plugging Eqn. (40) into Eqn. (39), the optimization problem can now be re-written as:

$$\min_{\mu}\left[1+\lambda_{\mathbf{Y}}-2\mu a-\eta_{\mathbf{x}}\log\left(1+\lambda_{\mathbf{Y}}-\mu^2 a\right)\right] \tag{42}$$

where; $a = \mathrm{u}_{\mathrm{x}}^{\mathrm{T}}\mathrm{K}_{\mathrm{Y}}\mathrm{u}_{\mathrm{x}}$ is a constant. This is a much easier problem to solve since only a scalar μ has to be optimized instead of optimizing for $\mathrm{t_y}$ or y. Differentiating Eqn. (42) with respect to μ and equating to zero, the closed-form solution is given by:

$$\hat{\mu} = \frac{-\eta_{\mathbf{x}} + \sqrt{\eta_{\mathbf{x}}^2 + 4a\left(1+\lambda_{\mathbf{Y}}\right)}}{2a} \tag{43}$$

The final optimal solution is given by Eqn. (44) with the computational complexity of $O(N)$. Here, the min and max operations for vectors are applied element-wise.

$$\hat{\mathrm{t}}_{\mathrm{y}} = \min\left(\left(1+\lambda_{\mathrm{Y}}\right)1,\ \max\left(0, \hat{\mu}\mathrm{K}_{\mathrm{Y}}\mathrm{u}_{\mathrm{x}}\right)\right) \tag{44}$$

To simplify the computations of DTGP, P nearest neighbors (NN) $\left\{\mathbf{y}_p, \hat{t}_{\mathbf{y},p}\right\}_{p=1}^{P}$ are found from $\{\mathbf{y}_i\}_{i=1}^{N}$ for each test sample x depending on the weights γ_p. The structured output $\hat{\mathbf{y}}$ is predicted using the weighted sum of P nearest training data samples given by Eqn. (45). Since $\hat{t}_{\mathbf{y},p}$ is a Gaussian kernel, i.e., $\hat{t}_{\mathbf{y},p} = \exp\left(-\frac{1}{2\rho_{\mathbf{Y}}^2}\mathbf{y}-\mathbf{y}_p{}^2\right)$, it can also be re-written as $\log\left(\hat{t}_{\mathbf{y},p}\right) \propto -\mathbf{y}-\mathbf{y}_p{}^2$. The weight γ_p captures the input-output and between-output dependency for structured regression [29].

$$\hat{\mathbf{y}} = \sum_{p=1}^{P} \gamma_p \mathbf{y}_p ; \gamma_p = \frac{\hat{t}_{\mathbf{y},p}}{\sum_{p=1}^{P} \hat{t}_{\mathbf{y},p}} \tag{45}$$

The DTGP method is easy to implement and can estimate the predicted output $\hat{\mathbf{y}}$ without the need for computationally expensive non-linear optimizers that are required for KLTGP and SMTGP structured regression methods.

4.4.4 IMPORTANCE WEIGHTED TGPS

The likelihood of the GP regression model for the r^{th} dimension under the covariate shift is given by Eqn. (46) [30, 31], where $\hat{\mathbf{y}}$ is the predicted regression value.

$$\prod_{i=1}^{n_{\text{tr}}} p\left(y_i^{(r)\text{tr}} \mid \mathrm{x}_i^{\text{tr}}\right)^{w\left(\mathrm{x}_i^{\text{tr}}\right)} \propto \prod_{i=1}^{n_{\text{tr}}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2} w^{\frac{1}{2}}\left(\mathrm{x}_i^{\text{tr}}\right) y_i^{(r)\text{tr}} - w^{\frac{1}{2}}\left(\mathrm{x}_i^{\text{tr}}\right) \hat{y}_i^{(r)\text{tr}2}\right] \tag{46}$$

Thus, the GPR model can therefore be updated by re-weighting each data point by $w^{\frac{1}{2}}\left(\mathrm{x}_i^{\text{tr}}\right)$. Accordingly, the KLTGP, SMTGP, and DTGP structured regression methods can be modified with the importance weights to correct the covariate shift. The importance weighted joint and marginal kernels of IW-KLTGP, IW-SMTGP, and IW-DTGP are defined in Eqn. (47), where $\mathrm{W} = \text{diag}\left\{w\left(\mathrm{x}_1\right), \ldots, w\left(\mathrm{x}_{n_{\text{tr}}}\right)\right\}$ is a diagonal matrix consisting of the importance weights w.

$$\begin{aligned}
&\left(\mathrm{K}_{\mathrm{X_w}}\right)_{ij} = w^{\frac{1}{2}}\left(\mathrm{x}_i\right) w^{\frac{1}{2}}\left(\mathrm{x}_j\right) \exp\left(-\frac{\mathrm{x}_i - \mathrm{x}_j^{\,2}}{2\rho_{\mathrm{X}}^2}\right) + \lambda_{\mathrm{X}} \delta_{ij} \Rightarrow \mathrm{K}_{\mathrm{X_w}} = \mathrm{W}^{\frac{1}{2}} \mathrm{K}_{\mathrm{X}} \mathrm{W}^{\frac{1}{2}}, \\
&\left(\mathrm{K}_{\mathrm{Y_w}}\right)_{ij} = w^{\frac{1}{2}}\left(\mathrm{x}_i\right) w^{\frac{1}{2}}\left(\mathrm{x}_j\right) \exp\left(-\frac{\mathrm{y}_i - \mathrm{y}_j^{\,2}}{2\rho_{\mathrm{Y}}^2}\right) + \lambda_{\mathrm{Y}} \delta_{ij} \Rightarrow \mathrm{K}_{\mathrm{Y_w}} = \mathrm{W}^{\frac{1}{2}} \mathrm{K}_{\mathrm{Y}} \mathrm{W}^{\frac{1}{2}}, \\
&\left(\mathrm{k}_{\mathrm{X_w}}^{\mathrm{x}}\right)_i = \mathrm{K}_{\mathrm{X_w}}\left(\mathrm{x}_i, \mathrm{x}\right) = w^{\frac{1}{2}}\left(\mathrm{x}_i\right) \exp\left(-\frac{\mathrm{x}_i - \mathrm{x}^2}{2\rho_{\mathrm{X}}^2}\right) \Rightarrow \mathrm{k}_{\mathrm{X_w}}^{\mathrm{x}} = \mathrm{W}^{\frac{1}{2}} \mathrm{k}_{\mathrm{X}}^{\mathrm{x}}, \\
&\left(\mathrm{k}_{\mathrm{Y_w}}^{\mathrm{y}}\right)_i = \mathrm{K}_{\mathrm{Y_w}}\left(\mathrm{y}_i, \mathrm{y}\right) = w^{\frac{1}{2}}\left(\mathrm{x}_i\right) \exp\left(-\frac{\mathrm{y}_i - \mathrm{y}^2}{2\rho_{\mathrm{Y}}^2}\right) \Rightarrow \mathrm{k}_{\mathrm{Y_w}}^{\mathrm{y}} = \mathrm{W}^{\frac{1}{2}} \mathrm{k}_{\mathrm{Y}}^{\mathrm{y}}, \\
&\mathrm{k}_{\mathrm{XY_w}}^{\mathrm{xy}} = (1-\tau)\mathrm{k}_{\mathrm{X_w}}^{\mathrm{x}} + \tau \mathrm{k}_{\mathrm{Y_w}}^{\mathrm{y}} = \mathrm{W}^{\frac{1}{2}} \mathrm{k}_{\mathrm{XY}}^{\mathrm{xy}}, \mathrm{K}_{\mathrm{XY_w}} = (1-\tau)\mathrm{K}_{\mathrm{X_w}} + \tau \mathrm{K}_{\mathrm{Y_w}} = \mathrm{W}^{\frac{1}{2}} \mathrm{K}_{\mathrm{XY}} \mathrm{W}^{\frac{1}{2}}
\end{aligned} \tag{47}$$

The optimization problems of IW-KLTGP, IW-SMTGP, and IW-DTGP are thus given by Eqns. (48), (49) and (50), respectively, using the weighted kernels, where $\mathrm{u}_{\mathrm{X_w}} = \mathrm{K}_{\mathrm{X_w}}^{-1}\mathrm{k}_{\mathrm{X_w}}^{\mathrm{x}}$ and $\eta_{\mathrm{X_w}} = k_{\mathrm{X}}(\mathrm{x},\mathrm{x}) - \mathrm{k}_{\mathrm{X_w}}^{\mathrm{xT}}\mathrm{u}_{\mathrm{X_w}}$. Note that $k_{\mathrm{Y}}(y,y) = 1 + \lambda_{\mathrm{Y}}$ when using Gaussian kernels and thus is not affected by the importance weights. The corresponding gradients can be accordingly updated with the importance weighted kernels.

$$\hat{\mathrm{y}} = \mathrm{argmin}_{\mathrm{y}}\left[k_{\mathrm{Y}}(\mathrm{y},\mathrm{y}) - 2\mathrm{k}_{\mathrm{Y_w}}^{\mathrm{yT}}\mathrm{u}_{\mathrm{X_w}} - \eta_{\mathrm{X_w}}\log\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y}) - \mathrm{k}_{\mathrm{Y_w}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y_w}}^{-1}\mathrm{k}_{\mathrm{Y_w}}^{\mathrm{y}}\right)\right] \tag{48}$$

$$\hat{\mathrm{y}} = \mathrm{argmin}_{\mathrm{y}}\frac{1}{\psi - 1}\left[\left(k_{\mathrm{Y}}(\mathrm{y},\mathrm{y}) - \mathrm{k}_{\mathrm{Y_w}}^{\mathrm{yT}}\mathrm{K}_{\mathrm{Y_w}}^{-1}\mathrm{k}_{\mathrm{Y_w}}^{\mathrm{y}}\right)^{\frac{\tau(1-\psi)}{2(1-\tau)}} \times \left(k_{\mathrm{XY}}^{\tau} - \mathrm{k}_{\mathrm{XY_w}}^{\mathrm{xyT}}\mathrm{K}_{\mathrm{XY_w}}^{-1}\mathrm{k}_{\mathrm{XY_w}}^{\mathrm{xy}}\right)^{-\frac{(1-\psi)}{2(1-\tau)}}\right] \tag{49}$$

$$\begin{aligned} &\min_{\mathrm{t_y}}\left[1 + \lambda_{\mathrm{Y}} - 2\mathrm{t}_{\mathrm{Y}}^{\mathrm{T}}\mathrm{u}_{\mathrm{x_w}} - \eta_{\mathrm{x_w}}\log\left(1 + \lambda_{\mathrm{Y}} - \mathrm{t}_{\mathrm{Y}}^{\mathrm{T}}\mathrm{K}_{\mathrm{Y_w}}^{-1}\mathrm{t}_{\mathrm{y}}\right)\right] \\ &s.t. 0 \le t_{\mathrm{y},i} \le 1 + \lambda_{\mathrm{Y}}, i = 1,2,\ldots,N \end{aligned} \tag{50}$$

Using the procedure outlined for DTGP, the analytical solution of IW-DTGP is given by Eqn. (51), where the scalars are obtained using Eqn. (52). The predicted value of IW-DTGP can be obtained using Eqn. (45) by substituting Eqn. (51).

$$\hat{\mathrm{t}}_{\mathrm{y}} = \min\left((1 + \lambda_{\mathrm{Y}})1,\ \max\left(0, \hat{\mu}_{\mathrm{W}}\mathrm{K}_{\mathrm{Y_w}}\mathrm{u}_{\mathrm{x_w}}\right)\right) \tag{51}$$

$$\hat{\mu}_{\mathrm{W}} = \frac{-\eta_{\mathrm{x_w}} + \sqrt{\eta_{\mathrm{x_w}}^2 + 4a_{\mathrm{W}}(1 + \lambda_{\mathrm{Y}})}}{2a_{\mathrm{W}}}; a_{\mathrm{W}} = \mathrm{u}_{\mathrm{x_w}}^{\mathrm{T}}\mathrm{K}_{\mathrm{Y_w}}\mathrm{u}_{\mathrm{x_w}} \tag{52}$$

4.5 APPLICATION: HUMAN POSE ESTIMATION

Human pose estimation is a challenging research topic in the computer vision domain with several applications such as sports performance analysis, gaming, human-robot interaction, video surveillance, etc. The estimation of structured pose from a single image frame is an ill-posed problem, since multiple joints of body limbs can obtain the same pose projection. Also, the prediction model should be robust to several aspects such as skin color, background scenes, clothing texture and shape, lighting, etc. Covariate shift

can be observed in these applications due to changing conditions of pose changes, background, location, etc. The standard benchmark HumanEva dataset [32] is used in this work to evaluate the performance of KLIEP and PFWCS for computing the importance weights as well as the importance weighted TGP structured regression methods.

The HumanEva dataset [32] comprises of multiple view video and motion data with the extracted histogram of oriented Gradients features $\left(\mathbf{x} \in \mathbb{R}^{270}\right)$ of 3 subjects (S_1,S_2,S_3) performing the following motions: gesturing, jogging, boxing, walking, and throwing/catching. Both motion and video captures were synchronized by the software. The associated ground truth given by the output structured pose $\left(\mathbf{y} \in \mathbb{R}^{60}\right)$ is represented using 20 joint markers via torsoDistal, captured with three cameras (C_1,C_2,C_3) resulting in 9,630 image-poses for each camera. Figure 4.10 illustrates sample examples of the HumanEva dataset. Five possible covariate shift scenarios can be observed in the HumanEva dataset for the pose estimation:

1. **Selection Bias ($\mathbf{C_1}$):** All subjects (S_1,S_2,S_3) are considered in the training phase and one subject, either S_1 or S_2 or S_3, is considered in the testing phase. Only the camera C_1 data is used in this scenario.
2. **Selection Bias ($\mathbf{C_{1-3}}$):** The subjects are selected as above but all three camera data C_1,C_2,C_3 is used.
3. **Subject Transfer ($\mathbf{C_1}$):** The subjects in the training and testing phases are different, for example, the subjects S_1 and S_2 are included in the training phase and the subject S_3 is included in the testing phase.
4. **Motion Transfer ($\mathbf{C_{1-3}}$):** Different motions are used in the training and testing phase. The motions used in the training phase are gesturing, jogging, and boxing, and the motions used in the testing phase are walking and throwing/catching.
5. **Camera Transfer:** The camera data used in the training and testing phases is different. C_1 data is used in the training phase and C_2 data is used in the testing phase.

4.5 APPLICATION: HUMAN POSE ESTIMATION

Human pose estimation is a challenging research topic in the computer vision domain with several applications such as sports performance analysis, gaming, human-robot interaction, video surveillance, etc. The estimation of structured pose from a single image frame is an ill-posed problem, since

multiple joints of body limbs can obtain the same pose projection. Also, the prediction model should be robust to several aspects such as skin color, background scenes, clothing texture and shape, lighting, etc. Covariate shift can be observed in these applications due to changing conditions of pose changes, background, location, etc. The standard benchmark HumanEva dataset [32] is used in this work to evaluate the performance of KLIEP and PFWCS for computing the importance weights as well as the importance weighted TGP structured regression methods.

The HumanEva dataset [32] comprises of multiple view video and motion data with the extracted histogram of oriented Gradients features $\left(\mathbf{x} \in \mathbb{R}^{270}\right)$ of 3 subjects (S_1,S_2,S_3) performing the following motions: gesturing, jogging, boxing, walking, and throwing/catching. Both motion and video captures were synchronized by the software. The associated ground truth given by the output structured pose $\left(\mathbf{y} \in \mathbb{R}^{60}\right)$ is represented using 20 joint markers via torsoDistal, captured with three cameras (C_1,C_2,C_3) resulting in 9,630 image-poses for each camera. Figure 4.10 illustrates sample examples of the HumanEva dataset. Five possible covariate shift scenarios can be observed in the HumanEva dataset for the pose estimation:

1. **Selection Bias ($\mathbf{C_1}$):** All subjects (S_1,S_2,S_3) are considered in the training phase and one subject, either S_1 or S_2 or S_3, is considered in the testing phase. Only the camera C_1 data is used in this scenario.
2. **Selection Bias ($\mathbf{C_{1-3}}$):** The subjects are selected as above but all three camera data (C_1,C_2,C_3) is used.
3. **Subject Transfer ($\mathbf{C_1}$):** The subjects in the training and testing phases are different, for example, the subjects S_1 and S_2 are included in the training phase and the subject S_3 is included in the testing phase.
4. **Motion Transfer ($\mathbf{C_{1-3}}$):** Different motions are used in the training and testing phase. The motions used in the training phase are gesturing, jogging, and boxing, and the motions used in the testing phase are walking and throwing/catching.
5. **Camera Transfer:** The camera data used in the training and testing phases is different. C_1 data is used in the training phase and C_2 data is used in the testing phase.

All of the above scenarios obey the definition of covariate shift due to different train and test data distributions.

The comparison of PFWCS and KLIEP algorithms for computing the importance weights **w** of the HumanEva dataset is shown in Table 4.1 for four sample covariate shift scenarios. To determine the optimal σ parameter

of Eqn. (1) for both KLIEP and PFWCS algorithms, the IWCV algorithm is used. The PFWCS algorithm obtains a lower number of non-sparse atoms of α relative to the number of test samples in all covariate shift scenarios. Each experiment is run for 50 trials and the average training run time (seconds) along with the standard deviation is shown. The PFWCS algorithm outperforms the KLIEP algorithm due to its capability of providing sparser solutions with higher computational efficiency. The PFWCS algorithm is thus used for further experiments to determine the importance weights in all covariate shift scenarios.

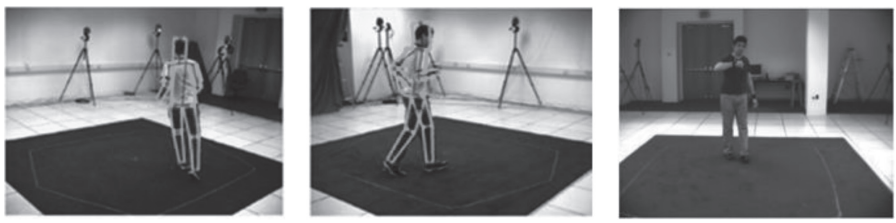


FIGURE 4.10 Samples of the HumanEva dataset.

TABLE 4.1 Comparison of KLIEP and PFWCS for the HumanEva Dataset

Covariate Shift	Method	# Non-Sparse Atoms of a	Time (s) to Find w
Subject transfer (C_1)	KLIEP	2019 (64%)	27.1 ± 0.24
Train: S_2, S_3; Test: S_1 (# samples: 3135)	PFWCS	471 (15%)	5.4 ± 0.21
Motion transfer (C_{1-3})	KLIEP	2608 (72%)	29.3 ± 0.19
Train: S_1, S_2, S_3; Test: S_2 (# samples: 3622)	PFWCS	689 (19%)	6.2 ± 0.27
Selection bias (C_1)	KLIEP	1792 (62%)	26.7 ± 0.23
Train: S_1, S_2, S_3; Test: S_3 (# samples: 2873)	PFWCS	364 (12%)	4.8 ± 0.41
Camera transfer (C_1)	KLIEP	2341 (74%)	28.5 ± 0.47
Train: S_1, S_2, S_3; S_1, C_2 Test: (# samples: 3135)			

The computational complexity of the various TGP structured regression methods is shown in Table 4.2. The inverse W^{-1} can be trivially computed as $\mathrm{W}^{-1} = \mathrm{diag}\left\{w\left(\mathrm{x}_1\right)^{-1},\ldots,w\left(\mathrm{x}_{n_{\mathrm{tr}}}\right)^{-1}\right\}$ since W is a diagonal matrix. Thus, the complexity of IW-{KL/SM/D}TGP methods is the same as that of the respective {KL/SM/D}TGP methods.

TABLE 4.2 Computational Complexity of TGP Structured Regression Methods

Method	Complexity
KLTGP & IW-KLTGP	$O(N^3)$
SMTGP & IW-SMTGP	$O(N^3)$
DTGP & IW-DTGP	$O(N)$

Both IW-KLTGP and IW-SMTGP has $O(N^3)$ complexity, where N is the number of training data samples, due to the matrix inversion required for solving the optimization function. However, the complexity of IW-DTGP is only $O(N)$ due to the use of simple algebra solution given by Eqn. (51) instead of solving the optimization problem of Eqn. (50), thus avoiding the matrix inversion. Thus, for subsequent experiments, the DTGP and IW-DTGP methods are used for the performance evaluation of covariate shift correction. Even though IW-SMTGP performs relatively better than IW-KLTGP [23], the IW-DTGP method is preferred due to its significantly lower computational complexity.

The evaluation metric for human pose estimation is the joint pose prediction error measured using the Euclidean distance given by Eqn. (53). Here, the true pose is $\mathbf{y}^*$, the estimated pose is $\hat{\mathbf{y}}$ and i ranges from 1 to 20 due to the use of 20 3D joint markers.

$$Error_{pose}\left(\mathrm{y}^*, \hat{\mathrm{y}}\right) = \frac{1}{20}\sum_{i=1}^{20} \mathrm{y}^{*(i)} - \hat{\mathrm{y}}^{(i)} \tag{53}$$

For DTGP, the parameter setting of [12] is used: $\lambda_{\mathbf{X}} = 10^{-3}, \lambda_{\mathbf{Y}} = 10^{-3}, 2\rho_{\mathbf{X}}^2 = 5$ and $2\rho_{\mathbf{Y}}^2 = 5\times 10^5$. To predict the pose $\hat{\mathbf{y}}$, the number (P) of the nearest training instances for (IW-)DTGP is empirically set to 30. The performance of (IW-)DTGP is relatively insensitive to the changes in the value of P since the distance between the samples of both input and output are taken into account using Gaussian kernels [29].

The performance of the joint prediction error for DTGP and IW-DTGP structured regression methods is shown in Table 4.3 for five covariate shift scenarios with different train and test data distributions, averaged across all 5 motions. 50% non-overlapping training and testing sets are chosen randomly for the performance evaluation. This procedure is repeated for 10 trials to avoid the bias due to random sampling, and the resulting average joint errors are reported. The parameter settings are shared by both DTGP and IW-DTGP, but the importance weights $\mathbf{w}$ from the PFWCS algorithm

leads to an improved pose estimation thus reducing the regression error of IW-DTGP for all scenarios of covariate shift correction.

TABLE 4.3 Performance Evaluation of Covariate Shift Correction for the HumanEva Dataset

Covariate Shift	Train	Test	DTGP	IW-DTGP
Selection	–	S_1	53.18	52.24
Bias	S_1,S_2,S_3	S_2	52.43	51.65
(C_1)	–	S_3	57.93	56.08
Selection	–	S_1	84.78	82.31
Bias	S_1,S_2,S_3	S_2	85.61	82.76
(C_{1-3})	–	S_3	87.28	85.39
Subject	S_2,S_3	S_1	122.77	120.41
Transfer	S_1,S_3	S_2	106.82	105.18
(C_1)	S_1,S_2	S_3	149.21	135.69
Motion	–	S_1	146.92	143.14
Transfer	S_1,S_2,S_3	S_2	155.28	152.76
(C_{1-3})	–	S_3	149.31	147.83
Camera	–	S_1,C_2	148.66	146.38
Transfer	S_1,S_2,S_3	S_2,C_2	149.29	147.12
(C_1)	–	S_3,C_2	146.71	144.19

Figure 4.11 illustrates the performance of the baseline GPR as well as the DTGP and IW-DTGP structured regression methods for 15 possible covariate shift combinations. For each case, the average joint pose error is shown with respect to the number of training data samples averaged over 10 trials and all 5 motions. If the number of training samples is less, the samples n_{tr} are randomly sampled with replacement from the full training set. These graphs show that the IW-DTGP structured regression method improves the performance significantly relative to the baseline GPR and DTGP for all scenarios. Also, the results of IW-DTGP are significantly better than GPR and DTGP statistically based on a 5% significance level paired t-test.

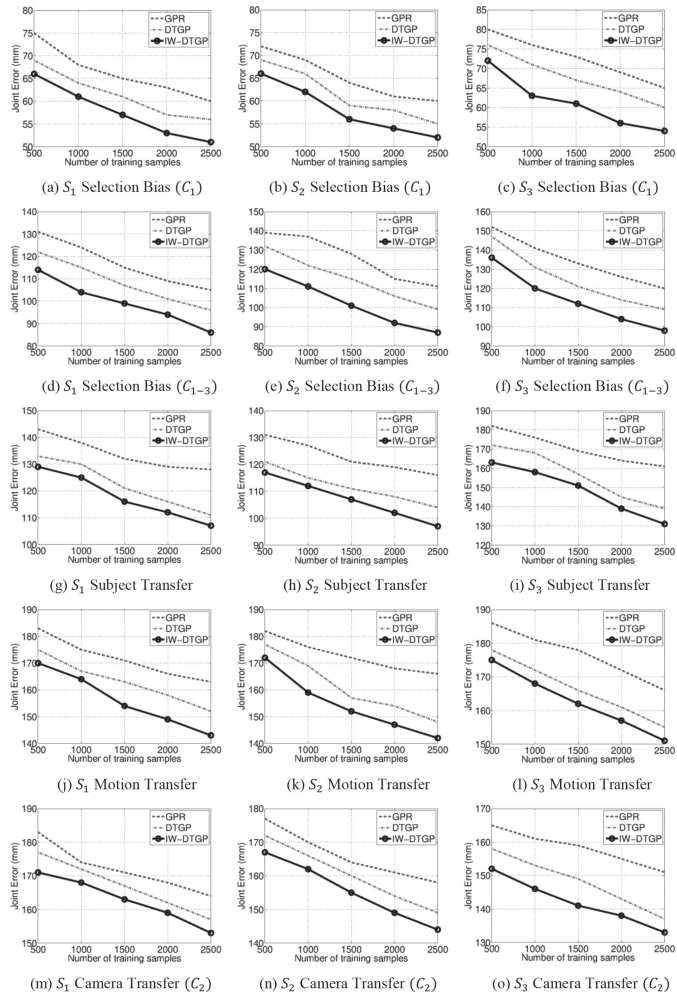


FIGURE 4.11 Performance evaluation of GPR, DTGP, and IW-DTGP for covariate shift correction of the HumanEva dataset.

4.6 CONCLUSION

In many real-world supervised ML applications (such as language processing, intrusion detection systems, etc.), the phenomenon of covariate shift is evident due to the mismatch between the train and test data probability distributions. This problem is solved by computing the importance weights directly from the data without performing the hard task of density estimation in high dimensions.

The training data samples are re-weighted using the importance weights to give more significance to the training data samples closer to the test data. For computing the importance weights, the KLIEP method is used since it has a constrained convex optimization problem which can be efficiently solved. The pairwise FW covariate shift optimization algorithm is presented as an alternative to the KLIEP method resulting in a sparser solution. The PFWCS algorithm is based on the linear maximization/minimization principle of the FW optimization concept, and thus the number of updates required to reach the optimal solution is less leading to higher computational efficiency.

For structured regression problems such as human pose estimation, it is necessary to model the dependency between the multiple target dimensions. Twin GPR methods based on Kullback-Leibler or Sharma-Mittal divergence measures can be used for these problems; however, they require a non-linear optimizer during the training procedure. The direct TGP method has an analytical solution obtained with simple algebra and is thus preferred over KLTGP and SMTGP methods since DTGP can be trained with $O(N)$ cost. These TGP methods are modified for covariate shift correction during the learning process by incorporating the importance weights. The performance is evaluated on the HumanEva dataset for estimating human poses and the covariate shift correction in the learning process results in a reduced regression error. The importance weighted DTGP outperforms DTGP due to the PFWCS importance weights. Following this line of research, correcting the covariate shift for supervised, structured classification problems can be a promising direction to pursue.

KEYWORDS

- **multivariate normal**
- **principal component analysis**
- **structured support vector machine**
- **Twin Gaussian process**
- **Pairwise Frank-Wolfe covariate shift**

REFERENCES

1. Moreno-Torres, J., Raeder, T., Alaiz-Rodriguez, R., Chawla, N., & Herrera, F., (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530.

2. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N., (2009). *Dataset Shift in Machine Learning*. The MIT Press.
3. Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., Von, B. P., & Kawanabe, M., (2008). Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics, 60*(4), 699–746.
4. Kanamori, T., Hido, S., & Sugiyama, M., (2009). A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, *10*, 1391–1445.
5. Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., & Sugiyama, M., (2013). Relative density-ratio estimation for robust distribution comparison. *Neural Computation*, *25*(5), 1324–1370.
6. Tsuboi, Y., Kashima, H., Bickel, S., & Sugiyama, M., (2009). Direct density ratio estimation for large-scale covariate shift adaptation. *Journal of Information Processing, 17*, 138–155.
7. Yamada, M., & Sugiyama, M., (2009). Direct importance estimation with Gaussian mixture models. *IEICE Transactions on Information and Systems*, *92*(10), 2159–2162.
8. Yamada, M., Sugiyama, M., Wichern, G., & Simm, J., (2010). Direct importance estimation with a mixture of probabilistic principal component analyzers. *IEICE Transactions on Information and Systems, 93*(10), 2846–2849.
9. Jaggi, M., (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Proc. of the 30th Intl. Conf. on Machine Learning (PMLR), 28*(1), 427–435.
10. Joulin, A., Tang, K., & Fei-Fei, L., (2014). Efficient image and video co-localization with Frank-Wolfe algorithm. *Proc. European Conf. on Computer Vision, 8694*, 253–268.
11. Wen, J., Greiner, R., & Schuurmans, D., (2015). Correcting covariate shift with the Frank-Wolfe algorithm. *Proc. of the 24th Intl. Joint Conf. on Artificial Intelligence*, 1010–1016.
12. Bo, L., & Sminchisescu, C., (2010). Twin Gaussian processes for structured prediction. *Intl. Journal of Computer Vision, 87*(28), 1–25.
13. Sugiyama, M., Krauledat, M., & Muller, K., (2007). Covariate shift adaptation by importance weighted cross-validation. *Journal of Machine Learning Research, 8*, 985–1005.
14. Bishop, C., (2006). *Pattern Recognition and Machine Learning*; Springer-Verlag.
15. Tipping, M., & Bishop, C., (1999). Mixtures of probabilistic principal component analyzers. *Neural Computation, 11*(2), 443–482.
16. Garber, D., & Hazan, E., (2015). Faster rates for the Frank-Wolfe method over strongly-convex sets. *Proc. of the 32nd Intl. Conf. on Machine Learning,* 541–549.
17. Osokin, A., Alayrac, J., Lukasewitz, I., Dokania, P., & Lacoste-Julien, S., (2016). Minding the gaps for block Frank-Wolfe optimization of structured SVMs. *Proc. of the 33rd Intl. Conf. on Machine Learning, 48*, 593–602.
18. GueLat, J., & Marcotte, P., (1986). Some comments on Wolfe's away step. *Mathematical Programming, 35*(1), 110–119.
19. Beck, A., & Shtern, S., (2017). Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming, 164*(1, 2), 1–27.
20. Lacoste-Julien, S., & Jaggi, M., (2015). On the global linear convergence of Frank-Wolfe optimization variants. *Proc. Conf. Advances in Neural Information Processing Systems*.

21. Allende, H., Frandi, E., Nanculef, R., & Sartori, C., (2013). Pairwise away steps for the Frank-Wolfe algorithm. *Proc. Conf. Advances in Neural Information Processing Systems*.
22. Nanculef, R., Frandi, E., Sartori, C., & Allende, H., (2014). A novel Frank-Wolfe algorithm: Analysis and applications to large-scale SVM training. *Information Sciences, 285*, 66–99.
23. Chapaneri, S., & Jayaswal, D., (2019). Covariate shift adaptation for structured regression with Frank-Wolfe algorithms. *IEEE Access, 7*, 73804–73818.
24. Sun, W., & Ya-Xiang, Y., (2006). *Optimization Theory and Methods: Non-linear Programming*. Springer-Verlag.
25. Chapaneri, S., & Jayaswal, D., (2017). Structured prediction of music mood with twin gaussian processes. *Proc. Pattern Recognition and Machine Intelligence (PReMI), 10597*, 647–654.
26. Rasmussen, C., & Williams, C., (2006). *Gaussian Processes for Machine Learning*. The MIT Press.
27. Elhoseiny, M., & Elgammal, A., (2015). Generalized twin Gaussian processes using Sharma-Mittal divergence. *Journal of Machine Learning, 100*(2), 399–424.
28. Tian, Y., & Takane, Y., (2009). More on generalized inverses of partitioned matrices with Banachiewicz-Schur forms. *Linear Algebra and its Applications, 430*(5, 6), 1641–1655.
29. Yamada, M., Sigal, L., & Chang, Y., (2014). Domain adaptation for structured regression. *Intl. Journal of Computer Vision, 109*(2), 126–145.
30. Yamada, M., Sigal, L., & Raptis, M., (2014). Covariate shift adaptation for discriminative 3D pose estimation. *IEEE Tran. Pattern Analysis and Machine Intelligence, 36*(2), 235–247.
31. Shimodaira, H., (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference, 90*(2), 227–244.
32. Sigal, L., Balan, A., & Black, M., (2010). HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Intl. Journal of Computer Vision, 87*(1, 2), 4–27.