UNIVERSITY OF OXFORD DEPARTMENT OF STATISTICS

Hyperparameter Learning via Distributional Transfer

Ho Chung Leon Law[1], Peilin Zhao[2], Lucian Chan[1], Junzhou Huang[2] and Dino Sejdinovic[1]

[1]University of Oxford and [2]Tencent AI Lab

Tencent AI Lab

SUMMARY

Goal: Optimise f^{target} (target objective) w.r.t θ, i.e.

$$\theta^*_{target} = \text{argmax}_{\theta\in\Theta} f^{\text{target}}(\theta)$$

Scenario: There are n (potentially) related source tasks f^i, $i = 1, \dots n$. For each source task, we assume we have $\{\theta^i_k, f^i(\theta^i_k)\}_{k=1}^{N_i}$ from past runs and N_i denotes the number of evaluations of f^i from task i.

Method: Transfer information across tasks using kernel mean embeddings of distributions of training datasets used in those tasks.

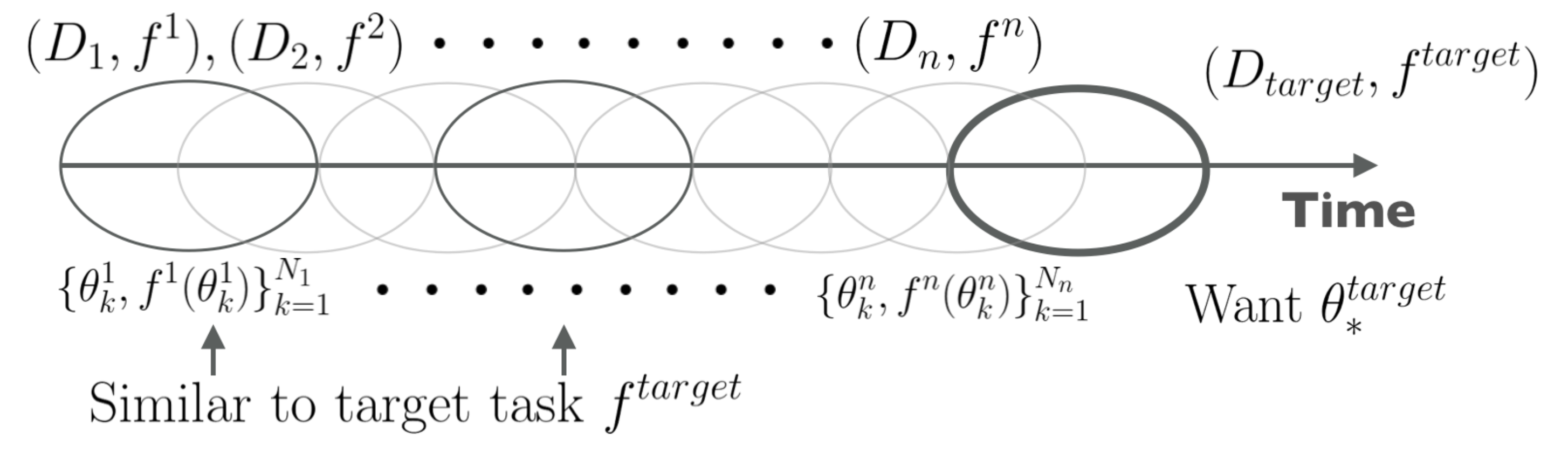
Here f^i is the accuracy of a trained machine learning model with training data $D_i = \{\mathbf{x}^i_\ell, y^i_\ell\}_{\ell=1}^{s_i}$. i.e. we have per source task:

$$(f^i, D_i = \{\mathbf{x}^i_\ell, y^i_\ell\}_{\ell=1}^{s_i}, \{\theta^i_k, f^i(\theta^i_k)\}_{k=1}^{N_i}) \qquad i = 1, \dots n$$

Our strategy now is to measure the similarity between datasets (as a representation of the task itself), in order to find θ^*_{target}.

MOTIVATION

Consider a scenario, where we have to constantly retrain our model in an online setting. For example, suppose that D is some traffic data, and that f is the accuracy of a machine learning model attempting to optimise taxi allocations.


$(D_1, f^1), (D_2, f^2) \cdots\cdots (D_n, f^n)$
(D_{target}, f^{target})
Time
$\{\theta^1_k, f^1(\theta^1_k)\}_{k=1}^{N_1} \cdots\cdots \{\theta^n_k, f^n(\theta^n_k)\}_{k=1}^{N_n}$
Want θ^{target}_*
Similar to target task f^{target}

Now as the data distribution changes (e.g. weekend vs weekday), we might expect $\theta^\star$ to also change.

RELATED WORK

The transfer of information from different tasks has been studied in the context of multi-task BO (multiBO) and meta-learning (manualBO) :

1. Learning task similarities through evaluations of f
2. Hand crafted meta-features for task similarities

Case 1: This would imply we need to observe sufficient evaluations from the target task in order to learn these task correlations.

Case 2: Using hand crafted meta-features can have an adverse effect on exploration if they are defined poorly.

Our methodology can be seen as a combination of these two frameworks, using learnt embeddings of the joint distribution of the data, while capturing correlation across tasks.

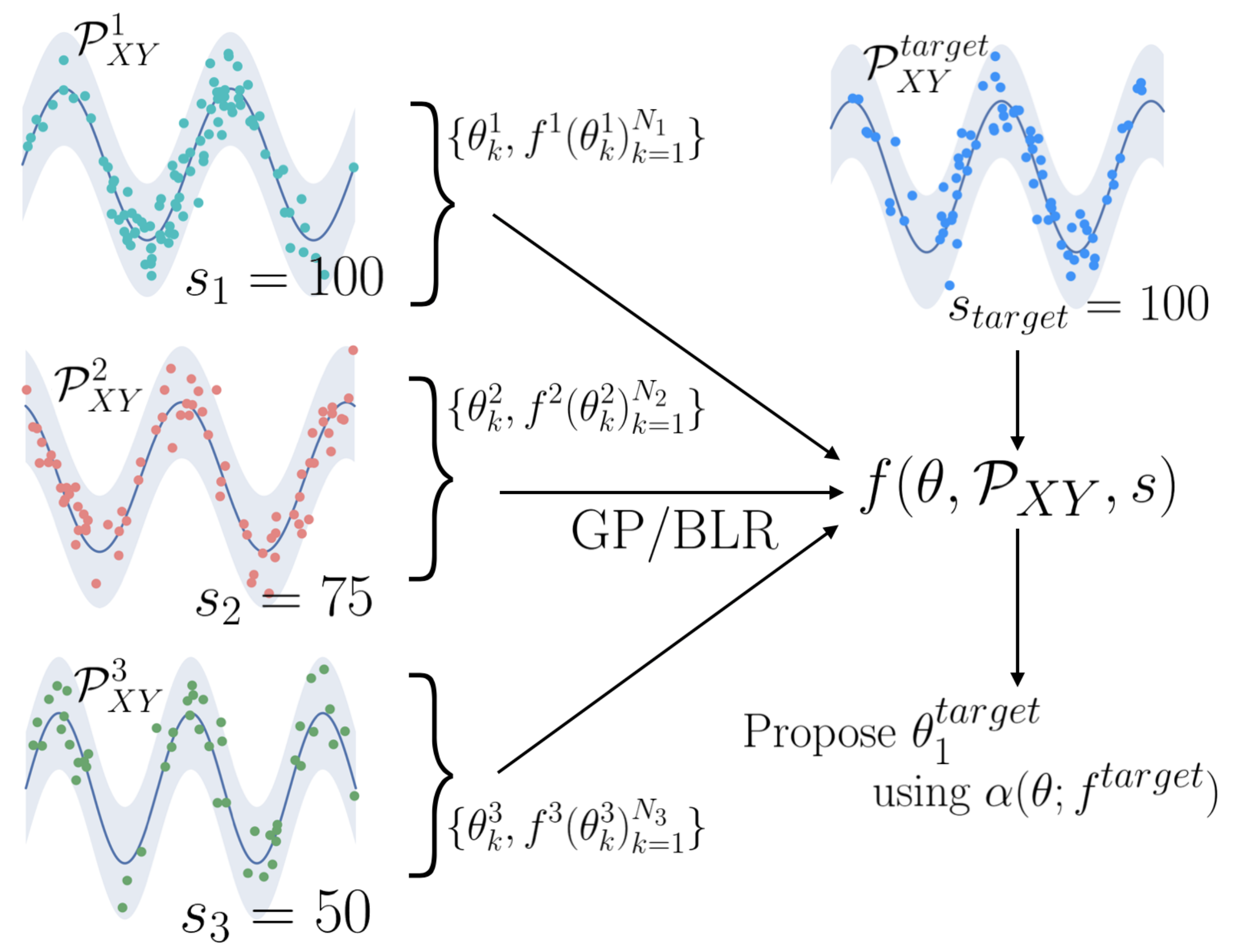
ASSUMPTIONS

1. $\mathbf{x}^i_\ell \in \mathcal{X}$ and $y^i_\ell \in \mathcal{Y}$ for all i, ℓ, supervised learning model M is the same.
2. $D_i = \{\mathbf{x}^i_\ell, y^i_\ell\}_{\ell=1}^{s_i} \sim \mathcal{P}^i_{XY}$, where $\mathcal{P}^i_{XY}$ is the joint distribution of the data for source task i.

Implication

- The source of differences of f^i across i and f^{target} is due to the distribution P_i and P_{target}.
- Sample size is related to hyperparameter choice, which we also encode.

MODEL AND EMBEDDING (DistBO)


$\mathcal{P}^1_{XY}$
$s_1 = 100$
$\{\theta^1_k, f^1(\theta^1_k)_{k=1}^{N_1}\}$
$\mathcal{P}^2_{XY}$
$s_2 = 75$
$\{\theta^2_k, f^2(\theta^2_k)_{k=1}^{N_2}\}$
$\mathcal{P}^3_{XY}$
$s_3 = 50$
$\{\theta^3_k, f^3(\theta^3_k)_{k=1}^{N_3}\}$
GP/BLR
$\mathcal{P}^{target}_{XY}$
$s_{target} = 100$
$f(\theta, \mathcal{P}_{XY}, s)$
Propose θ^{target}_1 using $\alpha(\theta; f^{target})$

Under this setting, we will consider $f(\theta, \mathcal{P}_{XY}, s)$, where f is a function of hyperparameters θ, joint distribution of the underlying data $\mathcal{P}_{XY}$ and sample size s. E.g. f could be the negative empirical risk, i.e.

$$f(\theta, \mathcal{P}_{XY}, s) = -\frac{1}{s}\sum_{l=1}^{s} L(h_\theta(\mathbf{x}_l), y_l))$$

where L is the loss function and h_θ is the model's predictor. In particular, we have that: $f^i(\theta) = f(\theta, \mathcal{P}^i_{XY}, s_i)$ and $f^{\text{target}}(\theta) = f(\theta, \mathcal{P}^{\text{target}}_{XY}, s_{target})$.

Using a Gaussian process for f, we define a a covariance function C:

$$C(\{\theta_1, \mathcal{P}^1_{XY}, s_1\}, \{\theta_2, \mathcal{P}^2_{XY}, s_2\}) = \nu k_\theta(\theta_1, \theta_2) k_p([\psi(D_1), s_1], [\psi(D_2), s_2])$$

where ν is a constant, k_θ and k_p are the standard Matérn-3/2 kernel.

To specify $\psi(D)$ we consider feature maps $\phi_x(\mathbf{x}) \in \mathbb{R}^p$ and $\phi_y(y) \in \mathbb{R}^q$. To embed a joint distribution $\mathcal{P}^i_{XY}$, we use the cross covariance operator $\mathcal{C}^i_{XY}$ [1], estimated by D_i with:

$$\hat{\mathcal{C}}^i_{XY} = \frac{1}{s_i}\sum_{\ell=1}^{s_i} \phi_x(\mathbf{x}^i_\ell) \otimes \phi_y(y^i_\ell) = \frac{1}{s_i}\Phi^i_x(\mathbf{x})\Phi^i_y(y)^\top \in \mathbb{R}^{p\times q}$$

where $\Phi^i_x(\mathbf{x}) = [\phi_x(\mathbf{x}^i_1), \dots, \phi_x(\mathbf{x}^i_{s_i})] \in \mathbb{R}^{p\times s_i}$, $\Phi^i_2(y) = [\phi_y(y^i_1), \dots, \phi_y(y^i_{s_i})] \in \mathbb{R}^{q\times s_i}$. Flattening $\hat{\mathcal{C}}^i_{XY}$, we obtain $\psi(D_i) \in \mathbb{R}^{pq}$.

IMPLEMENTATION

For ϕ_x, ϕ_y, we will consider a flexible representation, specifically in the form of neural networks (1 hidden and 1 output layer in this case).

Explanation: Suppose we have two task i, j and that $P^i_{XY} \approx P^j_{XY}$, this will imply that $f^i \approx f^j$, and hence $\theta^\star_i \approx \theta^\star_j$. However, the converse does not hold in general, i.e. $f^i \approx f^j$ does not imply $P^i_{XY} \approx P^j_{XY}$!

Example: Ridge regression with regularisation λ are likely to be robust to rotations of the covariates, which leads to a different P_X.

Optimisation: Maximise the marginal likelihood to optimise neural network and model parameters in an end-to-end fashion.

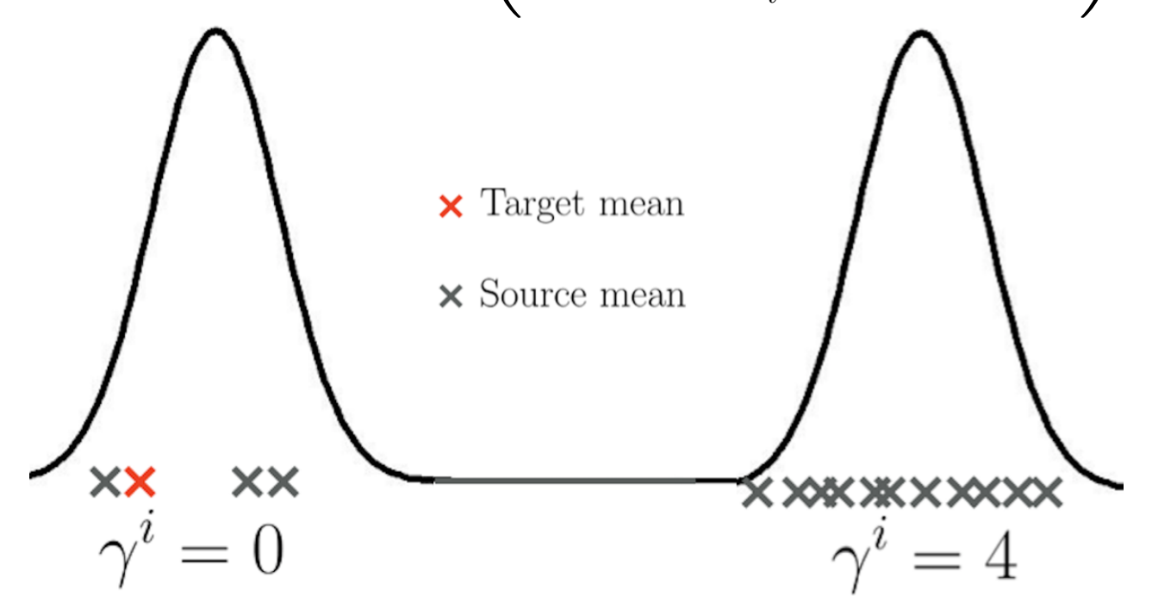
Scalability: Replace GP by Bayesian linear regression [2], which scales linearly with the number of observations $N = \sum_{i=1}^N$ (distBLR). As $S = \sum_{i=1}^N s_i$ is likely to be large, we can employ different random subsample of batch-size b for each step of optimisation.

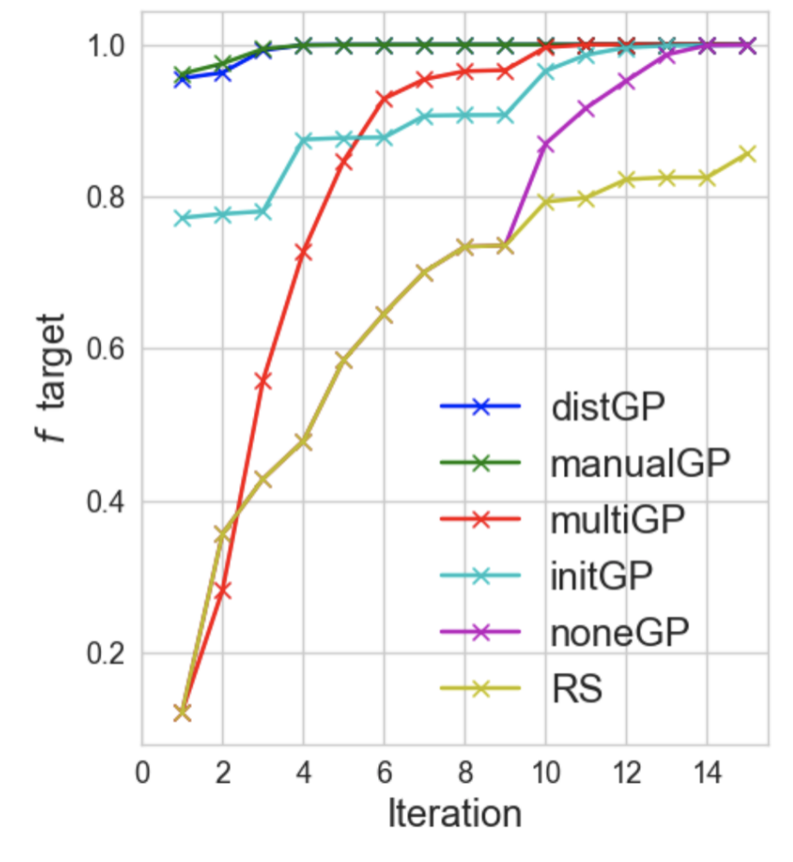
EXPERIMENTS

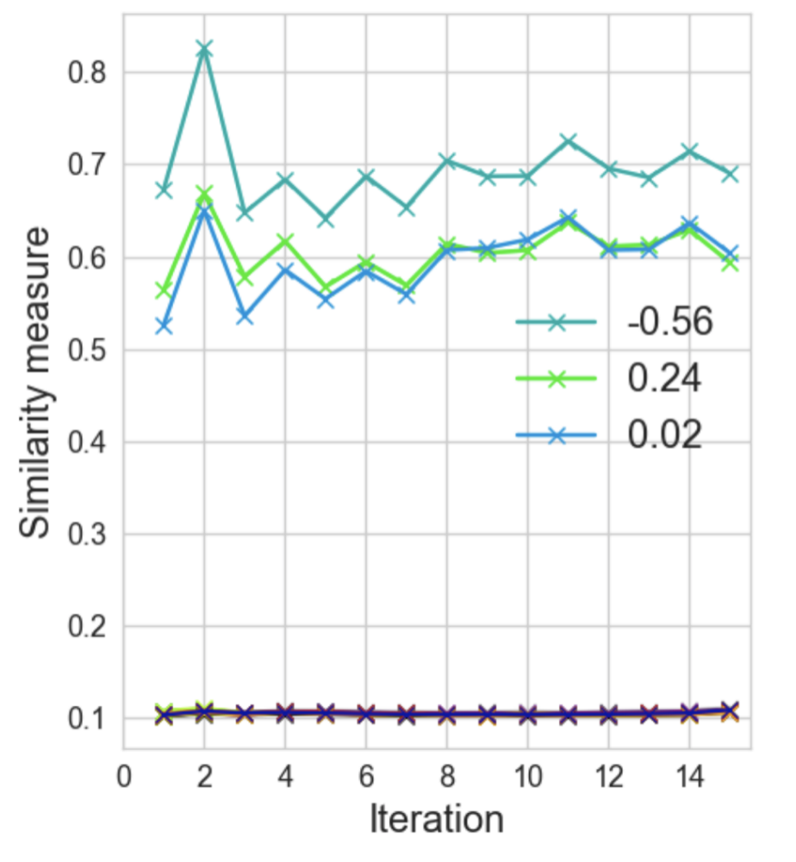
Toy example.

$X^i|\mu^i \sim \mathcal{N}(\mu^i, 1)$

$f(\theta; \mu^i) = \exp\left(-(\theta - \frac{1}{s_i}\sum x^i_\ell)^2/2\right)$

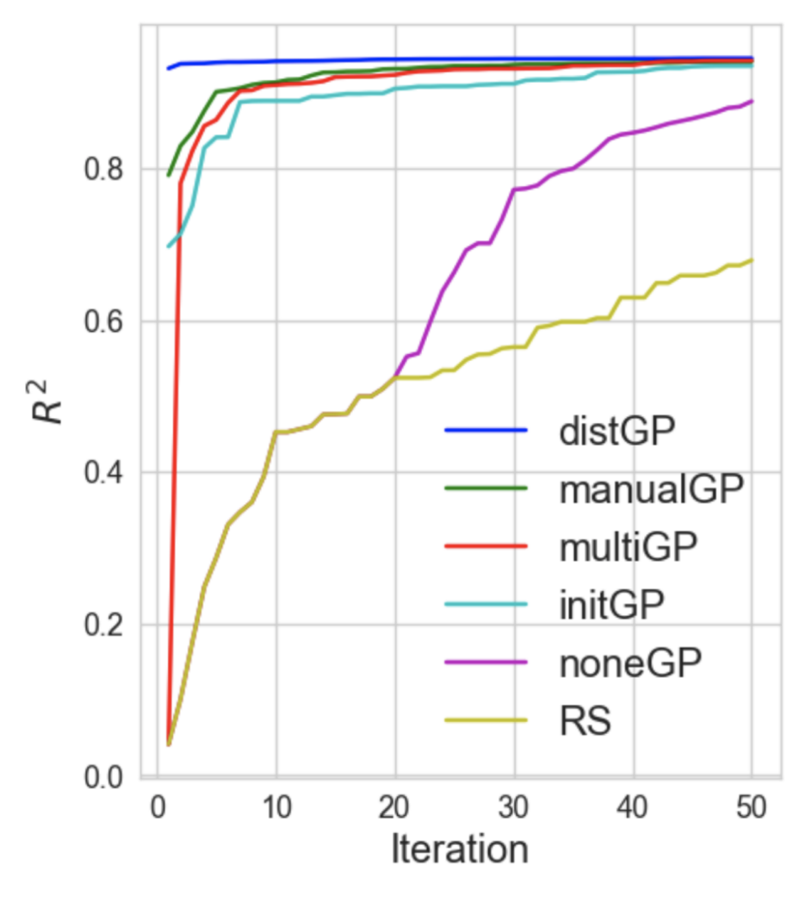


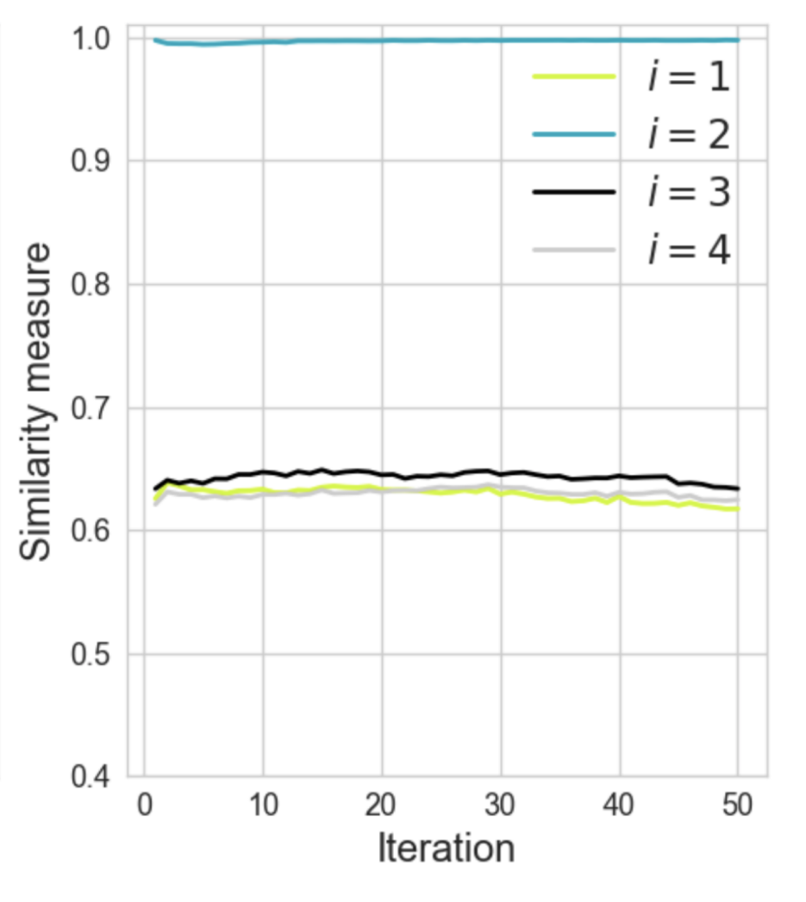




Handcrafted features.

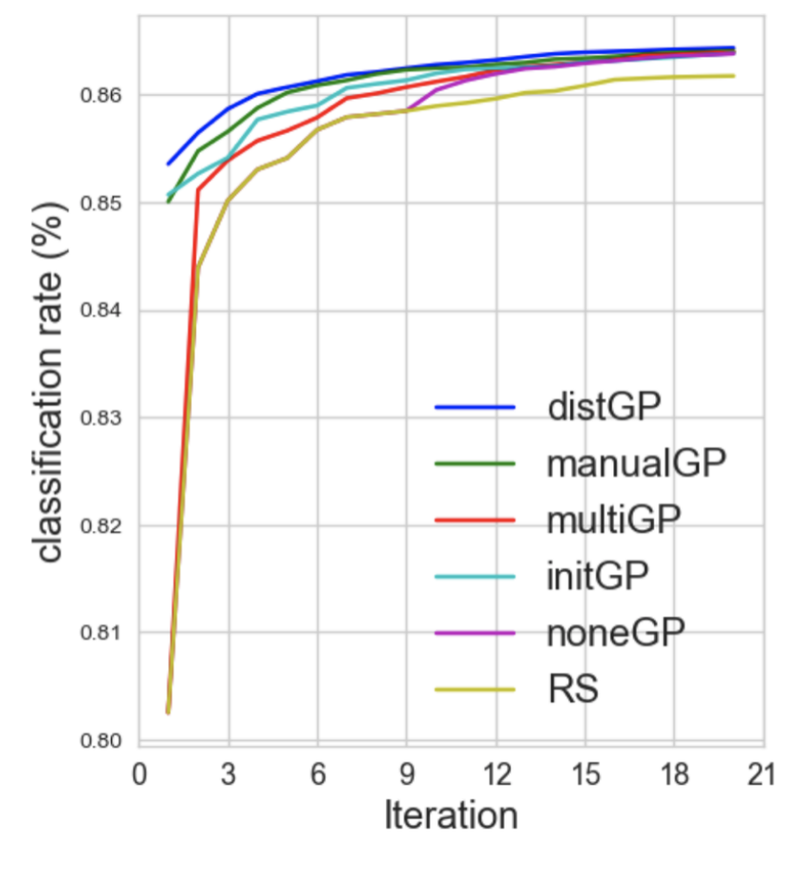
- Regression process invariant to manual meta-features
- 4 source tasks (1 similar)
- Ridge regression (ARD)
- 6 hyperparameters

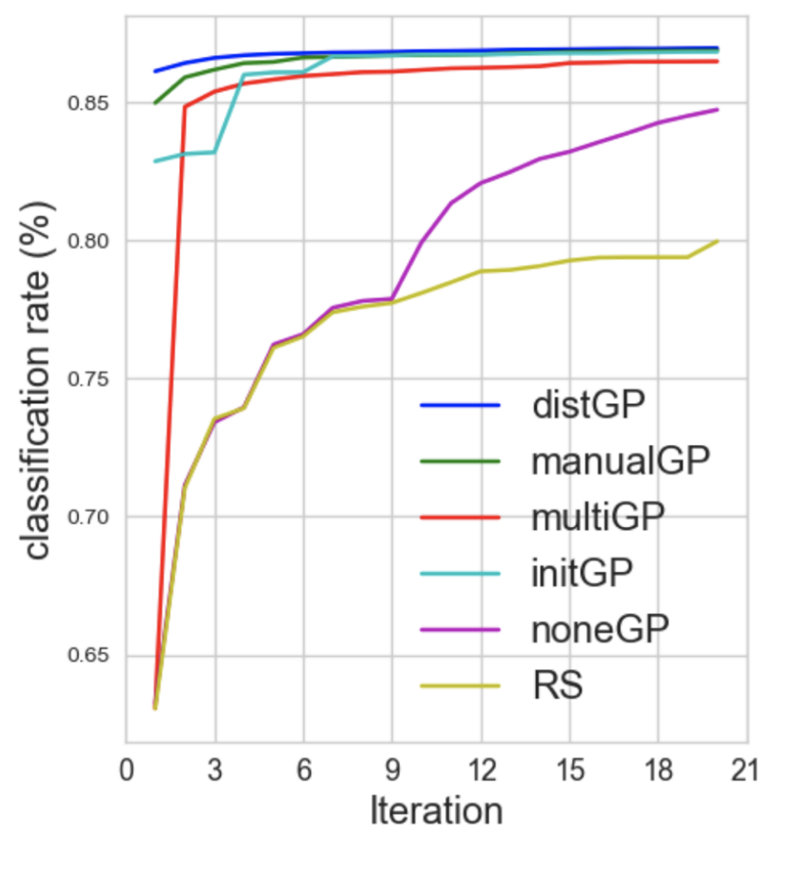




Protein dataset.

- Protein-ligand binding classification problem in the area of drug design
- 7 different proteins
- Jaccard kernel C-SVM (left)
- Random forest (right)





REFERENCES

[1] *Muandet, et al. 'Kernel Mean Embedding of Distributions: A Review and Beyond', 2017*

[2] *Perrone, et al. 'Scalable hyperparameter transfer learning', NeurIPS, 2018*