Kernel Embeddings, Meta Learning and Distributional Transfer

Dino Sejdinovic

Department of Statistics
University of Oxford

DeepMind
07/06/2019
Outline

1. Preliminaries on Kernel Embeddings
2. Hyperparameter Learning for Distributional Transfer
3. Meta Learning for Conditional Density Estimation
Outline

1 Preliminaries on Kernel Embeddings

2 Hyperparameter Learning for Distributional Transfer

3 Meta Learning for Conditional Density Estimation
Kernels and Reproducing Kernel Hilbert Spaces

- **Kernel function** is as an *inner product of features*: any function \( k : X \times X \to \mathbb{R} \) for which there exists a Hilbert space \( \mathcal{H} \) and a map \( \varphi : X \to \mathcal{H} \) s.t. \( k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} \) for all \( x, x' \in X \).

- **Kernel method** is any method that endows a generic abstract domain \( X \) with an inner product structure induced by some feature transformation.

- Feature map \( \varphi \) and feature space \( \mathcal{H} \) are not unique, but the inner product structure (kernel) is.
Kernels and Reproducing Kernel Hilbert Spaces

- **Kernel function** is as an *inner product of features*: any function \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) for which there exists a Hilbert space \( \mathcal{H} \) and a map \( \varphi : \mathcal{X} \rightarrow \mathcal{H} \) s.t. \( k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} \) for all \( x, x' \in \mathcal{X} \).

- **Kernel method** is any method that endows a generic abstract domain \( \mathcal{X} \) with an inner product structure induced by some feature transformation.

- Feature map \( \varphi \) and feature space \( \mathcal{H} \) are not unique, but the inner product structure (kernel) is.

---

**Definition ([Aronszajn, 1950; Berlinet & Thomas-Agnan, 2004])**

Let \( \mathcal{X} \) be a non-empty set and \( \mathcal{H} \) be a Hilbert space of real-valued functions defined on \( \mathcal{X} \). A function \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is called a *reproducing kernel* of \( \mathcal{H} \) if:

1. \( \forall x \in \mathcal{X}, \ k(\cdot, x) \in \mathcal{H}, \) and
2. \( \forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \ \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x). \)

If \( \mathcal{H}_k \) has a reproducing kernel, it is said to be a *reproducing kernel Hilbert space*.

In particular, for any \( x, y \in \mathcal{X} \), \( k(x, y) = \langle k(\cdot, y), k(\cdot, x) \rangle_{\mathcal{H}} = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}} \).

Thus, RKHS serves as a *canonical feature space* with *canonical feature map* \( x \mapsto k(\cdot, x) \).
Kernels and Reproducing Kernel Hilbert Spaces

- Equivalent definition: Hilbert space of functions where evaluations \( f \mapsto f(x) \) at any given point \( x \) are continuous maps (norm convergence implies pointwise convergence).

- Moore-Aronszajn Theorem: every positive semidefinite \( k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is a reproducing kernel and has a unique RKHS \( \mathcal{H}_k \).

- Example: Gaussian RBF kernel \( k(x, x') = \exp \left( -\frac{1}{2\gamma^2} \| x - x' \|^2 \right) \) has an infinite-dimensional \( \mathcal{H} \) with elements \( h(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x) \) and their limits which give completion with respect to the inner product

\[
\left\langle \sum_{i=1}^{n} \alpha_i k(x_i, \cdot), \sum_{j=1}^{m} \beta_j k(y_j, \cdot) \right\rangle = 
\sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, y_j).
\]
Kernel Trick and Kernel Mean Trick

- implicit feature map \( x \mapsto k(\cdot, x) \in \mathcal{H}_k \)
  replaces \( x \mapsto [\phi_1(x), \ldots, \phi_s(x)] \in \mathbb{R}^s \)

\[ \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}_k} = k(x, y) \]

inner products readily available

- nonlinear decision boundaries, nonlinear regression functions, learning on non-Euclidean/structured data

[Cortes & Vapnik, 1995; Schölkopf & Smola, 2001]
Kernel Trick and Kernel Mean Trick

- implicit feature map \( x \mapsto k(\cdot, x) \in \mathcal{H}_k \)
  replaces \( x \mapsto [\phi_1(x), \ldots, \phi_s(x)] \in \mathbb{R}^s \)

\[ \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}_k} = k(x, y) \]
inner products readily available

- nonlinear decision boundaries, nonlinear regression functions, learning on non-Euclidean/structured data

**RKHS embedding**: implicit feature mean

[Smola et al, 2007; Sriperumbudur et al, 2010; Muandet et al, 2017]

\( P \mapsto \mu_k(P) = \mathbb{E}_{X \sim P} k(\cdot, X) \in \mathcal{H}_k \)
replaces \( P \mapsto [\mathbb{E}\phi_1(X), \ldots, \mathbb{E}\phi_s(X)] \in \mathbb{R}^s \)

\[ \langle \mu_k(P), \mu_k(Q) \rangle_{\mathcal{H}_k} = \mathbb{E}_{X \sim P, Y \sim Q} k(X, Y) \]
inner products easy to estimate

- nonparametric two-sample, independence, conditional independence, interaction testing, learning on distributions

[Cortes & Vapnik, 1995; Schölkopf & Smola, 2001]

[Smola et al, 2007; Sriperumbudur et al, 2010; Muandet et al, 2017]

\( P \mapsto \mu_k(P) = \mathbb{E}_X k(\cdot, X) \)

\( Y \sim Q \)

\[ \|\mu_k(P) - \mu_k(Q)\|_{\mathcal{H}_k} \]

Maximum Mean Discrepancy

- Maximum Mean Discrepancy (MMD) \[ \text{[Borgwardt et al, 2006; Gretton et al, 2007]} \]
  between \( P \) and \( Q \):

\[
\text{MMD}_k(P, Q) = \| \mu_k(P) - \mu_k(Q) \|_{\mathcal{H}_k} = \sup_{f \in \mathcal{H}_k: \|f\|_{\mathcal{H}_k} \leq 1} |\mathbb{E}f(X) - \mathbb{E}f(Y)|
\]

- Characteristic kernels: \( \text{MMD}_k(P, Q) = 0 \) iff \( P = Q \) (also metrizes weak* \[\text{[Sriperumbudur, 2010]}\]).
  - Gaussian RBF \( \exp(-\frac{1}{2\sigma^2} \|x - x'\|_2^2) \), Matérn family, inverse multiquadrics.
  - Can encode structural properties in the data: kernels on non-Euclidean domains, networks, images, text...

Dino Sejdinovic (Oxford)
Some uses of MMD

MMD has been applied to:

- two-sample tests and independence tests (on graphs, text, audio...) [Gretton et al, 2009, Gretton et al, 2012]
- model criticism and interpretability [Lloyd & Ghahramani, 2015; Kim, Khanna & Koyejo, 2016]
- analysis of Bayesian quadrature [Briol et al, 2018]
- ABC summary statistics [Park, Jitkrittum & DS, 2015; Mitrovic, DS & Teh, 2016]
- summarising streaming data [Paige, DS & Wood, 2016]
- traversal of manifolds learned by convolutional nets [Gardner et al, 2015]
- MMD-GAN: training deep generative models [Dziugaite, Roy & Ghahramani, 2015; Sutherland et al, 2017; Li et al, 2017]

MMD\(^2_k\)\((P, Q) = \mathbb{E}_{X, X' \sim d_P} k(X, X') + \mathbb{E}_{Y, Y' \sim d_Q} k(Y, Y') - 2\mathbb{E}_{X \sim P, Y \sim Q} k(X, Y).\)
Some uses of MMD

**within-sample average similarity**

- between-sample average similarity

MMD has been applied to:

- two-sample tests and independence tests (on graphs, text, audio...) [Gretton et al, 2009, Gretton et al, 2012]

- model criticism and interpretability [Lloyd & Ghahramani, 2015; Kim, Khanna & Koyejo, 2016]

- analysis of Bayesian quadrature [Briol et al, 2018]

- ABC summary statistics [Park, Jitkrittum & DS, 2015; Mitrovic, DS & Teh, 2016]

- summarising streaming data [Paige, DS & Wood, 2016]

- traversal of manifolds learned by convolutional nets [Gardner et al, 2015]

- MMD-GAN: training deep generative models [Dziugaite, Roy & Ghahramani, 2015; Sutherland et al, 2017; Li et al, 2017]

\[
\text{\text{MMD}_k}^2(P, Q) = \frac{1}{n_x(n_x - 1)} \sum_{i \neq j} k(X_i, X_j) + \frac{1}{n_y(n_y - 1)} \sum_{i \neq j} k(Y_i, Y_j) - \frac{2}{n_x n_y} \sum_{i, j} k(X_i, Y_j). 
\]
Labels \( y_i = f(P_i) \) but observe only \( \{x_i^j\}_{j=1}^{N_i} \sim P_i \).

The goal: build a predictive model \( \hat{y}_* = f(\{x_*^j\}_{j=1}^{N_*}) \) for a new sample \( \{x_*^j\}_{j=1}^{N_*} \sim P_* \).

Represent each sample with the empirical mean embedding
\[
\hat{\mu}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} k(\cdot, x_i^j) \in \mathcal{H}_k.
\]

Now can use the induced inner product structure on empirical measures to build a regression model:
- Linear kernel on the RKHS:
  \[
  K (\hat{\mu}_i, \hat{\mu}_j) = \langle \hat{\mu}_i, \hat{\mu}_j \rangle_{\mathcal{H}_k} = \frac{1}{N_i N_j} \sum_{r,s} k(x_i^r, x_j^s)
  \]
- Gaussian kernel on the RKHS:
  \[
  K (\hat{\mu}_i, \hat{\mu}_j) = \exp(\gamma\|\hat{\mu}_i - \hat{\mu}_j\|_{\mathcal{H}_k}^2) = \exp\left(\gamma\overline{\text{MMD}_{k}^2}(P_i, P_j)\right)
  \]
Kernel Embeddings for Learning on Distribution Inputs

-0.856

0.562

1.39

- Labels \( y_i = f(P_i) \) but observe only \( \{x_i^j\}_{j=1}^{N_i} \sim P_i \).

- The goal: build a predictive model \( \hat{y}_* = f(\{x_*^j\}_{j=1}^{N_*}) \) for a new sample \( \{x_*^j\}_{j=1}^{N_*} \sim P_* \).

- Represent each sample with the empirical mean embedding
  \[
  \hat{\mu}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} k(\cdot, x_i^j) \in \mathcal{H}_k.
  \]

- Now can use the induced inner product structure on empirical measures to build a regression model:

  - Linear kernel on the RKHS:
    \[
    K(\hat{\mu}_i, \hat{\mu}_j) = \langle \hat{\mu}_i, \hat{\mu}_j \rangle_{\mathcal{H}_k} = \frac{1}{N_i N_j} \sum_{r,s} k(x_i^r, x_j^s)
    \]
  
  - Gaussian kernel on the RKHS:
    \[
    K(\hat{\mu}_i, \hat{\mu}_j) = \exp(-\gamma \|\hat{\mu}_i - \hat{\mu}_j\|_{\mathcal{H}_k}^2) = \exp\left(-\gamma \widehat{\text{MMD}}_k^2(P_i, P_j)\right)
    \]
Conditional Embeddings

Consider a joint distribution $P_{XY}$ over the random variables $(X, Y)$ taking values in $\mathcal{X} \times \mathcal{Y}$. The conditional mean embedding (CME) of $Y|X = x$ is defined as:

$$\mu_{Y|X=x} := \mathbb{E}_{Y|X=x}[k_y(\cdot, Y)] = \int_{\mathcal{Y}} k_y(\cdot, y) dP(y|x) \in \mathcal{H}_{k_y}$$

To model conditional embeddings as functions of $x$, we associate them with a linear operator $C_{Y|X} : \mathcal{H}_{k_x} \rightarrow \mathcal{H}_{k_y}$, which satisfies

$$\mu_{Y|X=x} = C_{Y|X} k_x(\cdot, x).$$
Estimation of Conditional Embeddings

Considering finite-dimensional feature maps \( \phi_x \) and \( \phi_y \), the finite sample estimator of \( C_{Y|X} \) based on dataset \( \{(x_i, y_i)\}_{i=1}^n \) is given by feature-to-feature regression coefficients:

\[
\hat{C}_{Y|X} = \Phi_y (K_{xx} + \lambda I)^{-1} \Phi_x^T,
\]

where \( \Phi_y := (\phi_y(y_1), \ldots, \phi_y(y_n)) \) and \( \Phi_x := (\phi_x(x_1), \ldots, \phi_x(x_n)) \) are the feature matrices, \( K_{xx} := \Phi_x \Phi_x^T \) is the kernel matrix with entries \( [K_{xx}]_{i,j} = k_x(x_i, x_j) := \langle \phi_x(x_i), \phi_x(x_j) \rangle \), and \( \lambda > 0 \) is a regularization parameter of feature-to-feature regression.

\[
\hat{\mu}_{Y|x} = \hat{C}_{Y|x} \phi(x) = \hat{C}_{YX} (\hat{C}_{XX} + \lambda I)^{-1} \phi(x) = \sum_{i}^{m} \beta_i(x) \phi(y_i)
\]
Kernel Embeddings for Meta Learning


Outline

1. Preliminaries on Kernel Embeddings

2. Hyperparameter Learning for Distributional Transfer

3. Meta Learning for Conditional Density Estimation
Towards End-to-End Learning

figure from https://blog.easysol.net/building-ai-applications/
Towards End-to-End Learning

Raw Data → Machine Learning Algorithm → Output
Towards End-to-End Learning

Raw Data → Machine Learning Algorithm → Output

Hyperparameter tuning

Dino Sejdinovic (Oxford)
Towards End-to-End Learning

Raw Data → Machine Learning Algorithm

Hyperparameter tuning

Output

Grid search, random search, trial-and-error, graduate student descent,...
Most machine learning models have hyperparameters to be tuned:

- **deep neural networks**: number of layers, regularization parameters, dropout parameters, layer size, batch size, learning rate, momentum, ...
- **kernel methods**: kernel lengthscale parameters, regularization parameters, number and type of random features, ...
- **variational methods**: prior parameters, variational family, choice of divergence, type of the variational bound, batch size, learning rate, ...

An objective function: a measure of generalization performance for a given set of hyperparameters obtained using held-out dataset or cross-validation.
We are interested in optimizing a 'well behaved' function $f : \Theta \to \mathbb{R}$ over some bounded domain $\Theta \subset \mathbb{R}^d$, i.e. in solving

$$\theta_* = \arg\min_{\theta \in \Theta} f(\theta).$$

However, $f$ is not known explicitly, i.e. it is a black-box function and we can only ever obtain noisy and expensive evaluations of $f$.

**Goal:** Find $\theta$ such that $f(\theta) \approx f(\theta_*)$ while minimizing the number of evaluations of $f$. 
Assuming that $f$ is well behaved, we build a surrogate probabilistic model for it (Gaussian Process).

1. Compute the posterior predictive distribution of $f$ using all evaluations so far.
2. Optimize a cheap proxy / acquisition function instead of $f$ which takes into account predicted values of $f$ at new points as well as the uncertainty in those predictions: this proxy is typically much cheaper to evaluate than the actual objective $f$.
3. Evaluate the objective $f$ at the optimum of the proxy and go to 1.

The proxy / acquisition function should balance exploration against exploitation.
Surrogate Gaussian Process model

Assume that the noise in the evaluations of the black-box function is i.i.d. $\mathcal{N}(0, \tau^2)$. Having evaluated the objective at locations $\theta = \{\theta_i\}_{i=1}^m$, we denote the observed values by $y = [y_1, \ldots, y_m]^\top$ and the true function values by $f = [f(\theta_1), \ldots, f(\theta_m)]^\top$. Then

$$f \sim \mathcal{N}(0, K),$$
$$y | f \sim \mathcal{N}(f, \tau^2 I).$$

GP model gives the posterior predictive mean $\mu(\theta)$ and the posterior predictive variance $\sigma^2(\theta) = \kappa(\theta, \theta)$ at any new location $\theta$, i.e.

$$f(\theta) | y \sim \mathcal{N}(\mu(\theta), \kappa(\theta, \theta)),$$

where

$$\mu(\theta) = k_{\theta\theta}(K + \tau^2 I)^{-1}y,$$
$$\kappa(\theta, \theta) = k(\theta, \theta) - k_{\theta\theta}(K + \tau^2 I)^{-1}k_{\theta\theta}$$

- **Exploitation**: seeking locations with low posterior mean $\mu(\theta)$,
- **Exploration**: seeking locations with high posterior variance $\kappa(\theta, \theta)$. 

Dino Sejdinovic (Oxford)
Acquisition functions

- **GP-LCB.** “optimism in the phase of uncertainty”; minimize the lower \((1 - \alpha)\)-credible bound of the posterior of the unknown function values \(f(\theta)\), i.e.

\[
\alpha_{LCB}(\theta) = \mu(\theta) - z_{1-\alpha} \sigma(\theta),
\]

where \(z_{1-\alpha} = \Phi^{-1}(1 - \alpha)\) is the desired quantile of the standard normal distribution.

- **PI (probability of improvement).** \(\tilde{\theta}\): the optimal location so far, \(\tilde{y}\): the observed minimum. Let \(u(\theta) = 1 \{f(\theta) < \tilde{y}\}\),

\[
\alpha_{PI}(\theta) = \mathbb{E}[u(\theta)|\mathcal{D}] = \Phi(\gamma(\theta)), \quad \gamma(\theta) = \frac{\tilde{y} - \mu(\theta)}{\sigma(\theta)}
\]

- **EI (expected improvement).** Let \(u(\theta) = \max(0, \tilde{y} - f(\theta))\)

\[
\alpha_{EI}(\theta) = \mathbb{E}[u(\theta)|\mathcal{D}] = \sigma(\theta) (\gamma(\theta) \Phi(\gamma(\theta)) + \phi(\gamma(\theta)))
\]
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
Illustrating Bayesian Optimization

figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
Illustrating Bayesian Optimization

figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams
Towards End-to-End Learning

![Diagram showing the process of raw data being input into a machine learning algorithm, with hyperparameter tuning as a feedback loop to improve the algorithm output.]
Towards End-to-End Learning

Machine Learning Algorithm

Raw Data

Output

Measure of performance \( f(\theta) \)

Probabilistic model for \( f \):
Learning to learn

\( \theta \)
Multiple hyperparameter learning tasks which share the same model: variability in $f$ across tasks is due to changing datasets.

Is performance measure $f$ really a black-box function of hyperparameters? Highly structured problem corresponding to training a specific model on a specific dataset.
Transfer Hyperparameter Learning

- **Multiple hyperparameter learning tasks** which share the same model: variability in $f$ across tasks is due to changing datasets.
- Is performance measure $f$ really a black-box function of hyperparameters? Highly structured problem corresponding to training a specific model on a specific dataset.

![Diagram showing multiple datasets and a BayesOpt algorithm](image)
Consider a standard supervised learning setting: $f(\theta, D)$ is a performance measure of a trained ML model with hyperparameters $\theta$ and data $D = \{x_l, y_l\}_{l=1}^{s}$, $x_l \in \mathcal{X}$ covariates and $y_l \in \mathcal{Y}$ labels. Assume the same domains $\mathcal{X}$ and $\mathcal{Y}$ for all tasks.

Assume that we have already solved $n$ source tasks by computing $N_i$ evaluations of the objective, i.e. we have $\{\theta^i_r, f(\theta^i_r, D_i)\}_{r=1}^{N_i}$, with source datasets

$$D_i = \{x^i_l, y^i_l\}_{l=1}^{s_i}, i = 1, \ldots, n.$$ 

The goal is to utilise information from source tasks to help us model $f_{\text{target}}(\theta) = f(\theta, D_{\text{target}})$ and speed up BayesOpt on an unseen target dataset

$$D_{\text{target}} = \{x^\text{target}_l, y^\text{target}_l\}_{l=1}^{s_{\text{target}}},$$

i.e.

$$\theta^\ast_{\text{target}} = \arg\min_{\theta \in \Theta} f_{\text{target}}(\theta)$$
Motivating Example

Example from [Poloczek et al, 2016] to motivate warm-starting Bayesian optimization.

- Model that assigns drivers to passengers (e.g. Uber or Lyft)
- Have to tune hyperparameters $\theta$, with objective $f$
- Live stream of data arriving in time

Problem:

- Re-train model every 12 hours, on the last 24 hours of data, and deploy asap.
- Optimal hyperparameters $\theta$ shift as data distribution changes e.g. weekend vs weekday or holiday vs no holiday
- Not all previous tasks are equally useful.

\[
(D_1, f^1), (D_2, f^2) \cdots \cdots \cdots \cdots (D_n, f^n)
\]

\[
\{\theta^1_k, f^1(\theta^1_k)\}_{k=1}^{N_1} \cdots \cdots \cdots \cdots \{\theta^n_k, f^n(\theta^n_k)\}_{k=1}^{N_n}
\]

Want $\theta^*_{target}$

Similar to target task $f_{target}$

(D_{target}, f_{target})
Dataset representation for hyperparameter learning

Assume $D = \{x_l, y_l\}_{l=1}^s \overset{i.i.d.}{\sim} P_{XY}$ and that $f$ is the empirical risk, i.e.

$$f(\theta, D) = \frac{1}{s} \sum_{\ell=1}^s L(h_\theta(x_\ell), y_\ell),$$

where $L$ is the loss function and $h_\theta$ is the model’s predictor.

For a fixed ML model, there are three sources of variability to the performance measure $f$:

- Hyperparameters $\theta$
- Joint (empirical) measure $P_{XY}$ of the dataset
- Sample size $s$

Thus we will model $f(\theta, P_{XY}, s)$, assuming that $f$ varies smoothly not only as a function of $\theta$, but also as a function of $P_{XY}$ and $s$ ([Klein et al, 2016] considers $f$ varying in $s$ to speed up BayesOpt on a single large dataset).
To model a joint GP in $(\theta, \mathcal{P}_{XY}, s)$, we construct a product covariance function:

$$K(\{\theta_1, \mathcal{P}^1_{XY}, s_1\}, \{\theta_2, \mathcal{P}^2_{XY}, s_2\}) = k_\theta(\theta_1, \theta_2)k_p(\psi(\mathcal{P}^1_{XY}), \psi(\mathcal{P}^2_{XY}))k_s(s_1, s_2)$$

Common choices might include $k_\theta$ are $k_p$ as Matérn-3/2, and $k_s$ as the sample size kernel from [Klein et al, 2016]

Need to learn representation $\psi(\mathcal{P}_{XY})$ useful for hyperparameter learning, i.e. the one which can yield representations invariant to variations in the training data irrelevant for hyperparameter choice.
AutoML: representing datasets using metafeatures

No joint GP model, but warmstart target hyperparameters to the optimal values from source datasets with closest metafeatures.


- **General:**
  - *Skewness, kurtosis of each input dimension:* extract the minimum, maximum, mean and standard deviation across the dimensions.
  - *Correlation, covariance of each pair of input dimensions:* extract the minimum, maximum, mean and standard deviation across the pairs.
  - *PCA skewness, kurtosis:* run PCA, project onto the first principal component and compute skewness and kurtosis.
  - *Intrinsic dimensionality:* number of principal components to explain 95% of variance.

- **Classification specific:**
  - *Label summaries:* empirical class distribution and its entropy.
  - *Classification landmarkers:* accuracy on a held out dataset of 1-nn classifier, linear discriminant analysis, naive Bayes and decision tree classifier.

- **Regression specific:**
  - *Label summaries:* Mean, stdev, skewness, kurtosis of the labels \( \{ y^i \}_{i=1}^{s} \).
  - *Regression landmarkers:* accuracy on a held out dataset of 1-nn, linear and decision tree regression.
Learning kernel embeddings

Need to learn a representation of empirical joint distributions for comparison across tasks.

- Start with parametrized feature maps (e.g. neural networks) \( \phi_x(x), \phi_y(y) \) and \( \phi_{xy}([x, y]) \) which we will learn (treated as GP kernel parameters).

- Marginal Distribution \( \mathcal{P}_X \): \( \hat{\mu}_P = \frac{1}{s} \sum_{\ell=1}^{s} \phi_x(x_\ell) \) (e.g. noisier covariates require less complex models).

- Conditional Distribution \( \mathcal{P}_{Y|X} \):

\[
\hat{C}_{Y|X} = \Phi_y^\top (\Phi_x \Phi_x^\top + \lambda I)^{-1} \Phi_x
\]

where \( \Phi_x = [\phi_x(x_1), \ldots, \phi_x(x_s)]^T \), \( \Phi_y = [\phi_y(y_1), \ldots, \phi_y(y_s)]^T \) and \( \lambda \) is a parameter that we learn. (e.g. captures smoothness of the regression functions).

- Joint Distribution \( \mathcal{P}_{XY} \):

\[
\hat{C}_{XY} = \frac{1}{s} \sum_{\ell=1}^{s} \phi_x(x_\ell) \otimes \phi_y(y_\ell) = \frac{1}{s} \Phi_x^\top \Phi_y
\]

Alternatively, learn a joint feature map \( \phi_{xy} \) and compute

\[
\hat{\mu}_{P_{XY}} = \frac{1}{s} \sum_{\ell=1}^{s} \phi_{xy}([x_\ell, y_\ell]).
\]
DistBO Algorithm

With a joint GP model on inputs \((\theta, \mathcal{P}_{XY}, s)\), we can now

1. Fit the GP on all performance evaluations so far:

\[
\mathcal{E} = \left\{ \left\{ (\theta^i_r, \mathcal{P}^i_{XY}, s^i), f^i(\theta^i_r) \right\}_{r=1}^{N_i} \right\}_{i=1}^{n},
\]

fitting any GP kernel parameters (e.g. those of feature maps \(\phi_x, \phi_y\)) by maximising the marginal likelihood of the GP.

2. Let \(f^{\text{target}}(\theta) = f(\theta, \mathcal{P}^{\text{target}}_{XY}, s^{\text{target}})\). Maximise the acquisition function at the target \(\alpha(\theta; f^{\text{target}})\) to select next \(\theta_{\text{new}}\).

3. Evaluate \(f^{\text{target}}(\theta_{\text{new}})\), add \{\((\theta_{\text{new}}, \mathcal{P}^{\text{target}}_{XY}, s^{\text{target}}), f^{\text{target}}(\theta_{\text{new}})\)\} to \(\mathcal{E}\) and go to 1.
Adaptive Bayesian Linear Regression: DistBLR

- Joint GP modelling comes at a high computational cost: $O(N^3)$ time and $O(N^2)$ storage, where $N$ is the total number of observations: $N = \sum_{i=1}^{n} N_i$
- GP cost can outweigh the cost of computing $f$ in the first place.
- Since we are learning dataset representation inside the kernel anyway – can instead simply adopt Bayesian linear regression ($O(N)$ time and storage)

\[
\begin{align*}
    z | \beta & \sim \mathcal{N}(\Upsilon \beta, \sigma^2 I) \quad \beta \sim \mathcal{N}(0, \alpha I) \\
    \Upsilon &= [\nu([\theta_1^1, \Psi_1]), \ldots, \nu([\theta_1^{N_1}, \Psi_1]), \ldots, \\
                \nu([\theta_n^1, \Psi_n]), \ldots, \nu([\theta_n^{N_n}, \Psi_n])]^\top \in \mathbb{R}^{N \times d}
\end{align*}
\]

where $\alpha > 0$ denotes the prior regularisation. Here $\nu$ denotes a feature map of dimension $d$ on concatenated hyperparameters $\theta$, data embedding $\psi(D)$ and sample size $s$.

Conceptually similar setting to [Perrone et al, 2018] who fit a single BLR per task.
We will compare **DistBO** with the following baselines:

- **manualBO**: joint GP with $\psi(D)$ as the selection of 13 AutoML meta-features,
- **multiBO**: i.e. multiGP [Swersky et al, 2013] and multiBLR [Perrone et al, 2018] which uses no meta-information, i.e. each task is encoded by its index, but the representation of hyperparameters is shared across tasks,
- **initBO**: plain BayesOpt warm-started with the top 3 hyperparameters from the three most similar source tasks in terms of AutoML meta-features,
- **noneBO**: plain BayesOpt,
- **RS**: random search.

Implementation in *TensorFlow*, with GP/BLR marginal likelihood optimized using ADAM. To obtain source task evaluations, we use standard BayesOpt.
Toy Example

$D_i$ is obtained for some fixed $\gamma^i$ as $\mu^i \sim \mathcal{N}(\gamma^i, 1)$, $\{x^i_\ell\}_{\ell=1}^{s_i} | \mu^i \overset{i.i.d.}{\sim} \mathcal{N}(\mu^i, 1)$ and the objective to maximize is

$$f(\theta; D_i) = \exp \left( - \frac{1}{s_i} \sum_{\ell=1}^{s_i} x^i_\ell \right)^2$$

where $\theta$ plays the role of a “hyperparameter”.

15 source tasks, 3 with $\gamma_i = 0$ and 12 with $\gamma_i = 4$. Target has $\gamma_i = 0$. 

$\times$ Target mean

$\times$ Source mean
Toy Example

- feature representation learned to place high similarity on the three source datasets sharing the same $\gamma^i$ and hence having similar values of $\mu^i$, while placing low similarity on the other source datasets
- manualBO also few-shots the optimum as it encodes the mean feature
- initBO and multiBO converge more slowly without any meta-information

$D_i$ is obtained for some fixed $\gamma^i$ as $\mu^i \sim \mathcal{N}(\gamma^i, 1)$, $\{x^i_\ell \}_{\ell=1}^{s_i} | \mu^i \overset{i.i.d.}{\sim} \mathcal{N}(\mu^i, 1)$ and the objective to maximize is

$$f(\theta; D_i) = \exp \left( - \frac{(\theta - \frac{1}{s_i} \sum_{\ell=1}^{s_i} x^i_\ell)^2}{2} \right),$$

where $\theta$ plays the role of a “hyperparameter”.

15 source tasks, 3 with $\gamma_i = 0$ and 12 with $\gamma_i = 4$. Target has $\gamma_i = 0$. 
Switching feature relevance

- handcrafted meta-features do not capture any information about the optimal hyperparameters
- three-variable interaction: the difference between tasks is invisible by considering marginal distributions of covariates and their pairwise relationships.

Dataset $i$ with $x_{i\ell}^i \in \mathbb{R}^6$ and $y_{i\ell}^i \in \mathbb{R}$:

\[
\begin{align*}
\left[ x_{i\ell}^i \right]_j & \overset{i.i.d.}{\sim} \mathcal{N}(0, 2^2), \quad j = 1, \ldots, 6, \\
\left[ x_{i\ell}^i \right]_{i+2} &= \text{sign}(\left[ x_{i\ell}^i \right]_1 \left[ x_{i\ell}^i \right]_2) \left| \left[ x_{i\ell}^i \right]_{i+2} \right|, \\
\left[ x_{i\ell}^i \right]_{i+2} &= \text{sign}(\left[ x_{i\ell}^i \right]_1 \left[ x_{i\ell}^i \right]_2) \left| \left[ x_{i\ell}^i \right]_{i+2} \right|, \\
y_{i\ell}^i &= \log \left( 1 + \left( \prod_{j \in \{1,2,i+2\}} [x_{i\ell}^i]_j \right)^3 \right) + \mathcal{N}(0, 0.5^2).
\end{align*}
\]

$i, \ell, j$ denote task, sample and dimension, respectively; sample size is $s_i = 5000$. 
Protein data classification

- Datasets on 7 proteins extracted from ChEMBL database [Gaulton et al, 2016]. Each protein corresponds to a task, containing 1037 – 4434 molecules with binary features $x^i_\ell \in \mathbb{R}^{166}$ computed using chemical fingerprinting. The binary label per molecule is whether it can bind to the protein target.

- Two classifiers: Jaccard kernel C-SVM (hyperparameter $C$), and random forest (hyperparameters n_trees, max_depth, min_samples_split, min_samples_leaf).

- Designate each protein as the target task, while using remaining 6 as source tasks. Results reported obtained by averaging over target tasks (20 runs per task).

![Graphs showing classification rates for different methods over iterations for Jaccard kernel C-SVM and random forest.]

**Figure:** **Left:** Jaccard kernel C-SVM. **Right:** Random forest
Conclusion

- Method to borrow strength between multiple hyperparameter learning tasks by making use of the similarity between training datasets.
- Allows few-shot hyperparameter learning especially if similar prior tasks are present.
- Towards opening the black box function of hyperparameter learning: consider model performance as a function of all its sources of variability.
- Future work: straightforward to consider the setting where we solve multiple tasks jointly, due to the presence of the joint GP model. Acquisition function?
Outline

1 Preliminaries on Kernel Embeddings

2 Hyperparameter Learning for Distributional Transfer

3 Meta Learning for Conditional Density Estimation
In supervised learning, we often focus on functional relationships, e.g. conditional expectations $\mathbb{E}[y|x]$ in regression. More expressive representation may be needed due to e.g. multimodality or heteroscedasticity: $y$ cannot be meaningfully represented using a single function $f(x)$ of the features $x$, such as $\mathbb{E}[y|x]$.

Example: $D = \{(x_i, y_i)\}_{i=1}^n$ sampled uniformly from an annulus $r^2 \leq x^2 + y^2 \leq R^2$ – any regression model would fail to capture the dependence between $y$ and $x$ because clearly $\mathbb{E}[y|x] = 0$.

Goal: conditional density estimation $p(y|x)$ based on paired samples $\{(x_i, y_i)\}_{i=1}^n$.

Use a flexible nonparametric model of the full conditional density in the meta learning setting.
Conditional Embeddings

- “Augment” the representation of $y$ by using a feature map $\phi_y(y)$ — e.g. $\phi_y(y) = [y, y^2]$ renders relationship trivial in the annulus example.
- In general, we require an expressive feature map $\phi_y$ so that CME $\mathbb{E}[\phi_y(y)|x]$ captures the relevant information about the relationship between $y$ and $x$.
- However, CMEs do not give a way to estimate conditional densities.

\[
\hat{C}_{Y|X} = \Phi_y (K_{xx} + \lambda I)^{-1} \Phi_x^T, \quad \hat{\mu}_{Y|X=x} = \mathbb{E}[\phi_y(y)|x] = \hat{C}_{Y|X} \phi_x(x).
\]
Conditional Embeddings

- "Augment" the representation of $y$ by using a feature map $\phi_y(y)$ — e.g. $\phi_y(y) = [y, y^2]$ renders relationship trivial in the annulus example.
- In general, we require an expressive feature map $\phi_y$ so that CME $\mathbb{E}[\phi_y(y)|x]$ captures the relevant information about the relationship between $y$ and $x$.
- However, CMEs do not give a way to estimate conditional densities.
- Idea: use the conditional mean embedding operator as a task embedding of a given conditional density estimation task.

$$\hat{C}_{Y|X} = \Phi_y(K_{xx} + \lambda I)^{-1} \Phi_x^T, \quad \hat{\mu}_{Y|X=x} = \mathbb{E}[\phi_y(y)|x] = \hat{C}_{Y|X} \phi_x(x).$$
Noise contrastive estimation [Gutmann and Hyvärinen, 2010] is an approach to the model parameter estimation based on classifiers discriminating between true and artificial (fake) samples. In our case, $y_i | x_i \sim p_\theta(y | x)$, and those from $\{y^f_{i,j}\}_{j=1}^\kappa \sim p_f(y)$, for a given $p_f(y)$. Giving weights proportional to $(1, \kappa)$, probability that the sample came from the true model is:

$$P_\theta(\text{True} | y, x) = \frac{p_\theta(y | x)}{p_\theta(y | x) + \kappa p_f(y)}.$$

Assuming that the learned classifier is Bayes optimal:

$$p_\theta(y | x) = \frac{\kappa p_f(y) P_\theta(\text{True} | y, x)}{1 - P_\theta(\text{True} | y, x)}.$$
Density model

Consider the density model given by

$$p_\theta(y|x) = \frac{\exp(s_\theta(x, y))}{\int \exp(s_\theta(x, y'))dy'} = \exp(s_\theta(x, y) + b_\theta(x))$$

for some scoring function $s_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ and $b_\theta(x)$ models the normalizing constant. Hence

$$P_\theta(\text{True}|y, x) = \frac{\exp(s_\theta(x, y) + b_\theta(x))}{\exp(s_\theta(x, y) + b_\theta(x)) + \kappa p_f(y)}$$

$$= \sigma \left( s_\theta(x, y) + b_\theta(x) - \log(\kappa p_f(y)) \right).$$

where $\sigma(t) = 1/(1 + e^{-t})$ is the logistic function.
Defining $s_\theta$

- Map $x_i$ and $y_i$ using feature maps (neural networks) $\phi_x : \mathcal{X} \rightarrow \mathcal{H}_X$ and $\phi_y : \mathcal{Y} \rightarrow \mathcal{H}_Y$ with all parameters collated into $\theta$

- Estimate the conditional mean embedding operator
  $$\hat{C}_{Y|X} = \Phi_y(K_{xx} + \lambda I)^{-1}\Phi^T_x$$

- Given $\hat{C}_{Y|X}$, we can estimate the conditional mean embedding for any new $x'$ using
  $$\hat{\mu}_{Y|X=x'} = \hat{C}_{Y|X} \phi(x')$$

- We can then evaluate the conditional mean embedding at any new $y'$ using
  $$\hat{\mu}_{Y|X=x'}(y') = \langle \hat{\mu}_{Y|X=x'}, \phi_y(y') \rangle_{\mathcal{H}_Y} = \langle \hat{C}_{Y|X} \phi(x'), \phi_y(y') \rangle_{\mathcal{H}_Y}$$

Scoring function:

$$s_\theta(x', y') = \hat{\mu}_{Y|X=x'}(y')$$
Defining $s_\theta$

- Scoring function:
  \[
  s_\theta(x', y') = \hat{\mu}_{Y|X=x'}(y')
  \]

- We expect this value to be high when $y'$ is drawn from the true conditional distribution $Y|X = x'$ and low in cases where $y'$ falls in a region where the true conditional density $p(y|x')$ is low:
  \[
  \mu_{Y|X=x'}(y') = \mathbb{E}[k_y(y', Y)|X = x'] = \int k_y(y', y)p(y|x')dy,
  \]
  where $k_y(y, y') := \langle \phi_y(y), \phi_y(y') \rangle_{\mathcal{H}_y}$.

- Recall that
  \[
  P_\theta(\text{True}|y, x) = \sigma \left( s_\theta(x, y) + b_\theta(x) - \log(\kappa p_f(y)) \right).
  \]
Three neural networks $\phi_x(x), \phi_y(y), b_\theta(x)$ (all parameters collated into $\theta$).

Let $\mathcal{T} = \{T_1, \ldots, T_L\}$ be the set of $L$ conditional density estimation tasks with each $T_l$ divided into context data $\mathcal{D}_c^l = \{(x_i^{l,c}, y_i^{l,c})\}$ and target data $\mathcal{D}_t^l = \{(x_i^{l,t}, y_i^{l,t})\}$.

For every target input $x_i^{l,t}$, we generate $\kappa$ fake responses $y_i^{l,f}$ from $p_f(y)$.

Now train the True/Fake classifier by maximizing conditional log-likelihood of the True/Fake labels, i.e. minimizing the logistic loss across all tasks jointly (SGD):

$$\min_{\theta} \sum_l \sum_i \left\{ \log \left( 1 + \frac{\kappa p_f(y_i^{l,t})}{\exp(s_\theta(x_i^{l,t}, y_i^{l,t}) + b_\theta(x_i^{l,t}))} \right) \right. $$

$$\left. + \sum_{j=1}^{\kappa} \log \left( 1 + \frac{\exp(s_\theta(x_i^{l,t}, y_{i,j}^{l,f}) + b_\theta(x_i^{l,t}))}{\kappa p_f(y_{i,j}^{l,f})} \right) \right\}$$

with

$$s_\theta(x, y) = \langle \hat{C}^l_{Y|X} \phi_x(x), \phi_y(y) \rangle_{\mathcal{H}_Y},$$

and $\hat{C}^l_{Y|X}$ computed on the context set $\mathcal{D}_c^l$. 

Dino Sejdinovic (Oxford)
Synthetic data experiment

Figure: Left to right: MetaCDE (ours), DDE, LSCDE, KCEF, $\epsilon$-KDE. The red dots are the context/training points and the green dots are points from the true density.

- $y_i \sim U(0, 1)$, $x_i | y_i = \cos(a y_i + b) + \mathcal{N}(0, \sigma^2)$, where $a, b, \sigma^2$ vary between tasks.
- Note that in this case $x$ can be written as a simple function of $y$ with added noise, but not vice versa, leading to the multimodality of $p(y|x)$. 
Synthetic data experiment

Figure: Left to right: MetaCDE (ours), DDE, LSCDE, KCEF, $\epsilon$-KDE. The red dots are the context/training points and the green dots are points from the true density.

Table: Average held out log-likelihood on 100 different synthetic $\cos$ tasks. Also reporting the p-values for the one sided signed Wilcoxon test wrt to MetaCDE.
Dihedral angles in molecules

Figure: Left to right: MetaCDE (ours), DDE, LSCDE, KCEF, $\epsilon$-KDE. The red dots are the context/training points and the green dots are points from the true density.

- Interested in understanding possible conformations of molecular structures, i.e. energetically allowed regions of dihedral angles in bonds. The data extracted from crystallography database COD [Gražulis et al, 2011].
- The multimodality of the dataset arises from the molecular symmetries such as reflection and rotational symmetry.
Dihedral angles in molecules

Figure: Left to right: MetaCDE (ours), DDE, LSCDE, KCEF, $\epsilon$-KDE. The red dots are the context/training points and the green dots are points from the true density.

<table>
<thead>
<tr>
<th></th>
<th>MetaCDE</th>
<th>DDE</th>
<th>LSCDE</th>
<th>KCEF</th>
<th>$\epsilon$-KDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean over 100 Log-likelihoods</td>
<td><strong>-297.58 ± 67.63</strong></td>
<td>-315.49 ± 204.82</td>
<td>-335.85 ± 192.09</td>
<td>-596.95 ± 871.97</td>
<td>-422.99 ± 346.46</td>
</tr>
<tr>
<td>P-value for Wilcoxon test</td>
<td>NA</td>
<td>4.932e-05</td>
<td>2.692e-05</td>
<td>1.397e-05</td>
<td>9.49e-07</td>
</tr>
</tbody>
</table>

Table: Average held out log-likelihood on 100 different molecules. Also reporting the p-values for the one sided signed Wilcoxon test wrt to MetaCDE.
Conclusions

- Learn a data representation informative for the conditional density estimation tasks, by borrowing strength across tasks.

- Reminiscent to neural processes [Garnelo et al, 2018]: MetaCDE learns a task embedding, but the task embedding based on context set, but this embedding takes a specific form of the conditional embedding operator and it is the feature maps that are learned.

- Future work: choices of fake distributions, including those depending on the conditioning variable.
Summary

- Statistical modelling can be brought to bear in tandem with performant machine learning models.
- Increasing confluence between statistics and ML: making use of the well engineered ML infrastructure, with bespoke statistical models for the problem at hand.
- Flexibility of the RKHS framework as a common ground between machine learning and statistical inference.
References
