Kernel Embeddings for Meta Learning

Dino Sejdinovic

Department of Statistics
University of Oxford

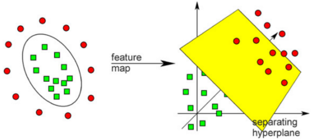
NLDL 2020, Tromso
21/01/2020

Outline

1. Hyperparameter Learning via Distributional Transfer
2. Meta Learning for Conditional Density Estimation

Kernel Trick and Kernel Mean Trick

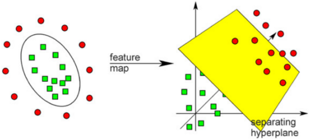
- implicit feature map $x \mapsto k(\cdot, x) \in \mathcal{H}_k$
  replaces $x \mapsto [\phi_1(x), \ldots, \phi_s(x)] \in \mathbb{R}^s$
- $\langle k(\cdot, x), k(\cdot, y)\rangle_{\mathcal{H}_k} = k(x, y)$
  inner products readily available
  - nonlinear decision boundaries, nonlinear regression functions, learning on non-Euclidean/structured data



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

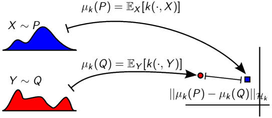
Kernel Trick and Kernel Mean Trick

- 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



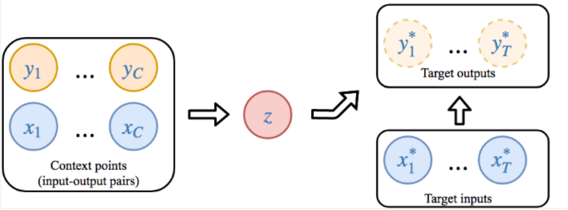
[Cortes & Vapnik, 1995; Schölkopf & Smola, 2001]

- **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



[Gretton et al, 2005; Gretton et al, 2006; Fukumizu et al, 2007; DS et al, 2013; Muandet et al, 2012; Szabo et al, 2015]

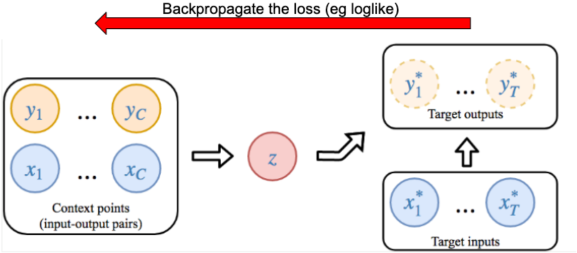
Probabilistic Meta Learning Framework [Garnelo et al 18]



- Let $\mathcal{T} = \{T_1, \ldots, T_L\}$ be the set of L tasks, each divided into context $\mathcal{D}_c^l = \{(x_i^{l,c}, y_i^{l,c})\}$ and target data $\mathcal{D}_t^l = \{(x_i^{l,t}, y_i^{l,t})\}$
- Context set is to extract the meta information, encoded as the "task embedding"
- Target set is to test how well the information was extracted by compute the loss on the target set.
- During testing time we only have context set and are asked to predict on any new x^*

[Thrun and Pratt, 1998; Ravi and Larochelle, 2016; Santoro et al., 2016]

Probabilistic Meta Learning Framework [Garnelo et al 18]



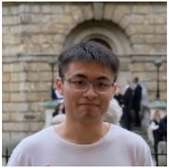
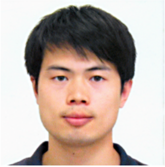
- Let $\mathcal{T} = \{T_1, \ldots, T_L\}$ be the set of L tasks, each divided into context $\mathcal{D}_c^l = \{(x_i^{l,c}, y_i^{l,c})\}$ and target data $\mathcal{D}_t^l = \{(x_i^{l,t}, y_i^{l,t})\}$
- Context set is to extract the meta information, encoded as the "task embedding"
- Target set is to test how well the information was extracted by compute the loss on the target set.
- During testing time we only have context set and are asked to predict on any new x^*

[Thrun and Pratt, 1998; Ravi and Larochelle, 2016; Santoro et al., 2016]

Kernel Embeddings for Meta Learning

- Ho Chung Leon Law, Peilin Zhao, Lucian Chan, Junzhou Huang, and DS. **Hyperparameter Learning via Distributional Transfer**. NeurIPS 2019.
- Jean-Francois Ton, Lucian Chan, Yee Whye Teh, and DS. **Noise Contrastive Meta Learning for Conditional Density Estimation using Kernel Mean Embeddings**. *ArXiv e-prints:1906.02236*, appearing in NeurIPS Meta Learning Workshop 2019.



Outline

Towards End-to-End Learning

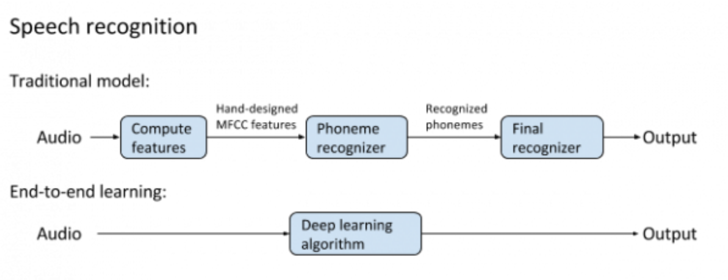
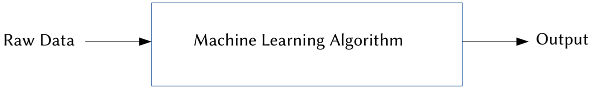
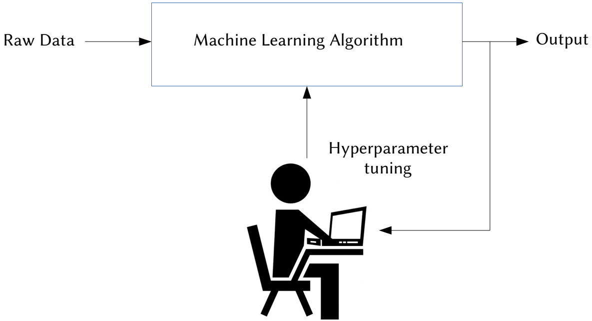


figure from https://blog.easysol.net/building-ai-applications/

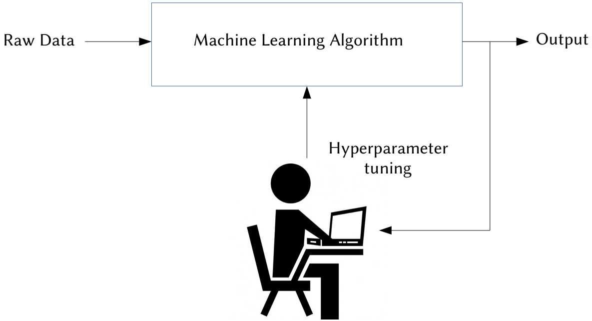
Towards End-to-End Learning



Towards End-to-End Learning



Towards End-to-End Learning



Grid search, random search, trial-and-error, graduate student descent,...

Optimizing "black-box" functions

We are interested in optimizing a 'well behaved' function $f : \Theta \to \mathbb{R}$ over some bounded domain of hyperparameters $\Theta \subset \mathbb{R}^d$, i.e. in solving

$$\theta_\star = \operatorname{argmin}_{\theta \in \Theta} f(\theta).$$

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 θ such that $f(\theta) \approx f(\theta_\star)$ while minimizing the number of evaluations of f.

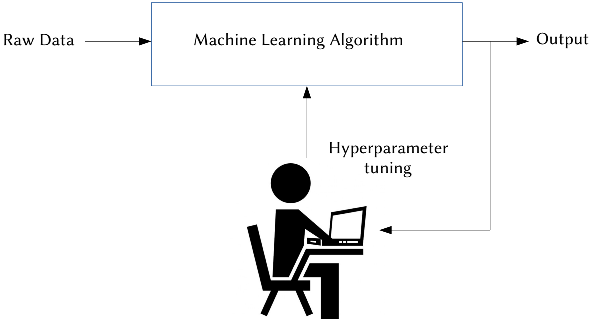
Probabilistic model for the objective f

Assuming that f is well behaved, we build a surrogate probabilistic model for it (typically a Gaussian Process Details).

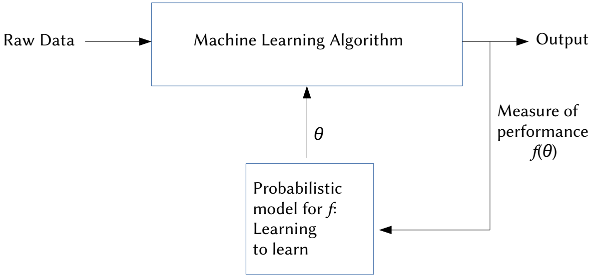
1. Compute the posterior predictive distribution of f using all evaluations so far.
2. Optimize a cheap proxy / acquisiton function instead of f which takes into account predicted values of f at new points as well as the *uncertainty in those predictions*: this proxy is typically much cheaper to evaluate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

The proxy / acquisiton function should balance exploration against exploitation.

Towards End-to-End Learning

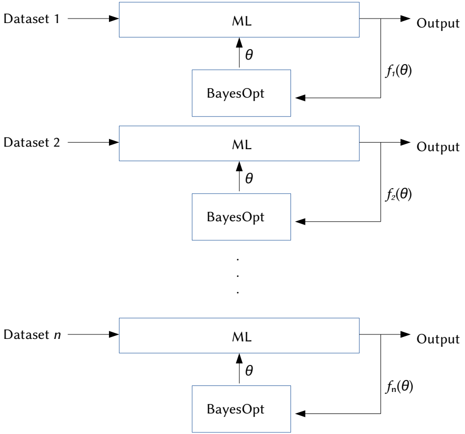


Towards End-to-End Learning



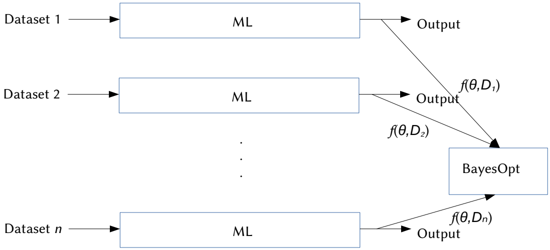
Transfer Hyperparameter Learning

- 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.



Transfer Hyperparameter Learning

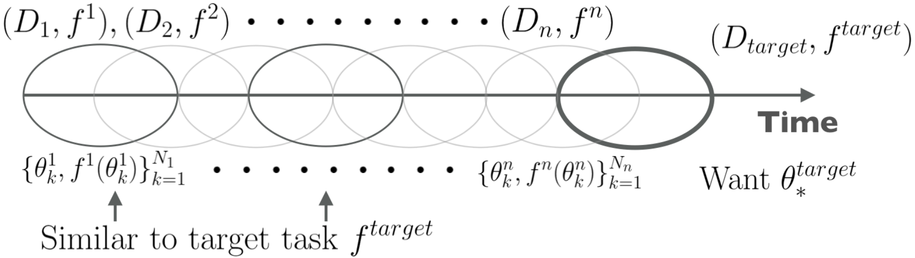
- Multiple hyperparameter learning tasks which share the same model: variability in f across tasks is due to changing datasets.
- Is performance measure f really a black-box function of hyperparameters? Highly structured problem corresponding to training a specific model on a specific dataset.



Motivating Example

Example from [Poloczek et al, 2016] to motivate warm-starting Bayesian optimization: live stream of data arriving in time:

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



AutoML: representing datasets using metafeatures

Warmstart target hyperparameters to the optimal values from source datasets with closest metafeatures.

[Michie et al, 1994; Pfahringer et al, 2000; Bardenet et al, 2013; Feurer et al, 2014; Hutter et al, 2019]

- 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 $\{y_\ell^i\}_{\ell=1}^{s_i}$.
  - *Regression landmarkers*: accuracy on a held out dataset of 1-nn, linear and decision tree regression.

Dataset (task) representation for hyperparameter learning

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

$$f(\theta, D) = \frac{1}{s} \sum_{\ell=1}^{s} L\left(h_\theta(\mathbf{x}_\ell), y_\ell\right),$$

where L is the loss function and h_θ is the model's predictor.

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

- Hyperparameters θ
- Joint (empirical) measure $\mathcal{P}_{XY}$ of the dataset
- Sample size s

Dataset (task) representation for hyperparameter learning

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

$$f(\theta, D) = \frac{1}{s}\sum_{\ell=1}^{s} L\left(h_\theta(\mathbf{x}_\ell), y_\ell\right),$$

where L is the loss function and h_θ is the model's predictor.

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

- Hyperparameters θ
- Joint (empirical) measure $\mathcal{P}_{XY}$ of the dataset
- Sample size s

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

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

Need to learn representation $\psi(\mathcal{P}_{XY})$ *useful for hyperparameter learning*.

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) $\phi_x(\mathbf{x})$, $\phi_y(y)$ and $\phi_{xy}([\mathbf{x}, y])$ which we will learn (treated as GP kernel parameters).
- Marginal Distribution $\mathcal{P}_X$: $\hat{\mu}_{P_X} = \frac{1}{s}\sum_{\ell=1}^{s}\phi_x(\mathbf{x}_\ell)$ (*e.g. noisier covariates require less complex models*).
- Conditional Distribution $\mathcal{P}_{Y|X}$:

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

  where $\Phi_x = [\phi_x(\mathbf{x}_1), \ldots, \phi_x(\mathbf{x}_s)]^T$, $\Phi_y = [\phi_y(y_1), \ldots, \phi_y(y_s)]^T$ and λ is a parameter that we learn. (*e.g. captures smoothess of the regression functions*).
- Joint Distribution $\mathcal{P}_{XY}$:

$$\hat{\mathcal{C}}_{XY} = \frac{1}{s}\sum_{\ell=1}^{s}\phi_x(\mathbf{x}_\ell)\otimes\phi_y(y_\ell) = \frac{1}{s}\Phi_x^\top\Phi_y$$

  Alternatively, learn a joint feature map ϕ_{xy} and compute $\hat{\mu}_{P_{XY}} = \frac{1}{s}\sum_{\ell=1}^{s}\phi_{xy}([\mathbf{x}_\ell, y_\ell])$.

DistBO

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:

$$\mathcal{E} = \{\{(\theta_r^i, \mathcal{P}_{XY}^i, s_i), f^i(\theta_r^i)\}_{r=1}^{N_i}\}_{i=1}^n,$$

   fitting any GP kernel parameters (e.g. those of feature maps ϕ_x, ϕ_y) by maximising the marginal likelihood of the GP.
2. Let $f^{target}(\theta) = f(\theta, \mathcal{P}_{XY}^{target}, s_{target})$. Maximise the acquisition function at the target $\alpha(\theta; f^{target})$ to select next θ_{new}
3. Evaluate $f^{target}(\theta_{new})$, add $\{(\theta_{new}, \mathcal{P}_{XY}^{target}, s_{target}), f^{target}(\theta_{new})\}$ to $\mathcal{E}$ and go to 1.

- In practice, joint GP modelling comes at a higher computational cost, but we can resort to Bayesian linear regression (BLR) on learned feature maps (with time and storage linear in the number of evaluations). Details
- Conceptually similar to [Perrone et al, 2018] which fits BLR per task sharing representation of hyperparameters. Our joint model allows one-shot proposal of hyperparameters without seeing any evaluations on the target task.

Experiments

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*: `https://github.com/hcllaw/distBO`. GP/BLR marginal likelihood optimized using ADAM. To obtain source task evaluations, we use standard BayesOpt.

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 $\mathbf{x}_\ell^i \in \mathbb{R}^{166}$ 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).

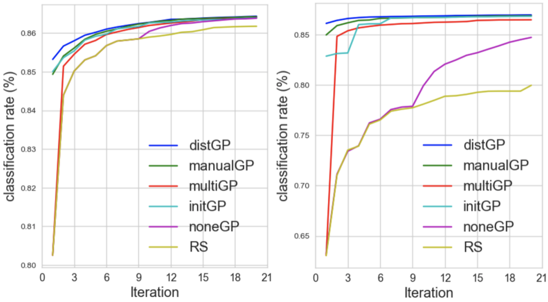
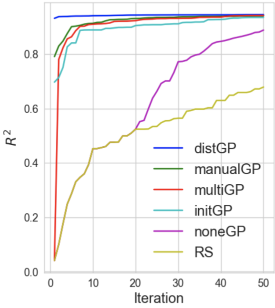
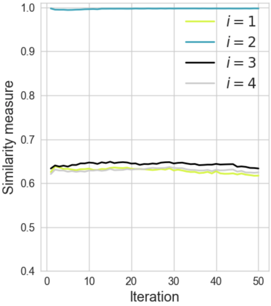


Figure: **Left:** Jaccard kernel C-SVM. **Right:** Random forest

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 $\mathbf{x}_\ell^i \in \mathbb{R}^6$ and $y_\ell^i \in \mathbb{R}$:

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

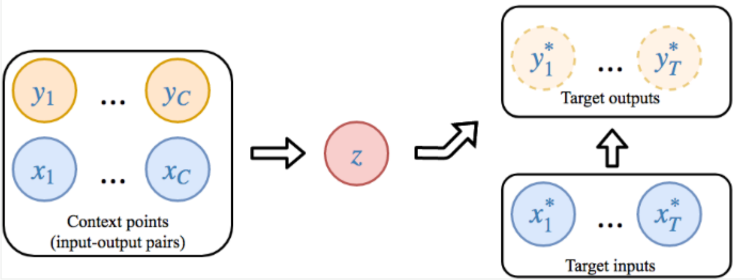
i, ℓ, j denote task, sample and dimension, respectively; sample size is $s_i = 5000$.

Conclusion

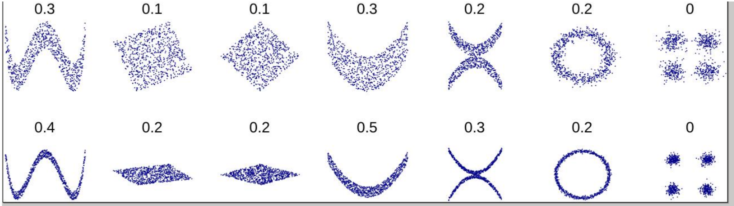
- Method to borrow strength between multiple hyperparameter learning tasks by making use of the similarity between training datasets.
- Allows few-shot hyperparameter learning especially if similar prior tasks are present.
- Towards opening the black box function of hyperparameter learning: consider model performance as a function of all its sources of variability.

Outline

Meta Learning Setup



Beyond Functional Relationships

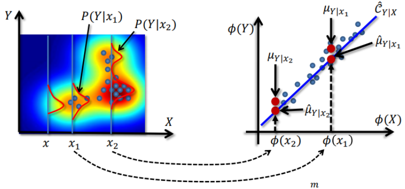


- In supervised learning, we often focus on functional relationships, e.g. conditional expectations $\mathbb{E}[y|x]$ in regression.
- More expressive representation may be needed due to e.g. multimodality or heteroscedasticity: y cannot be meaningfully represented using a single function $f(x)$ of the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

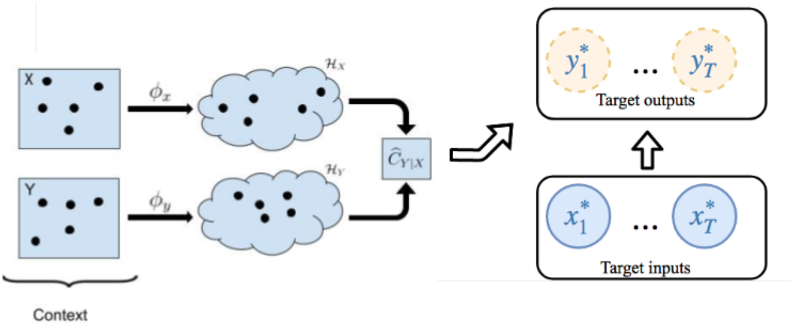
- "Augment" the representation of y by using a feature map $\phi_y(y)$ and consider CME $\mathbb{E}[\phi_y(y)|x]$
- We require an expressive feature map ϕ_y so that CME $\mathbb{E}[\phi_y(y)|x]$ captures the relevant information about the relationship between y and x.
- However, CMEs do not give a way to estimate *conditional densities*.
- Idea: use the conditional mean embedding operator as a task embedding of a given conditional density estimation task – turn estimation into classification using noise contrastive approach [Gutmann and Hyvärinen, 2010]. Details

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



$$\hat{\mu}_{Y|x} = \hat{\mathcal{C}}_{Y|X}\phi(x) = \hat{\mathcal{C}}_{YX}(\hat{\mathcal{C}}_{XX} + \lambda I)^{-1}\phi(x) = \sum_i^m \beta_i(x)\phi(y_i)$$

Proposed Method



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 $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and $b_\theta(x)$ models the normalizing constant.

Use scoring function:

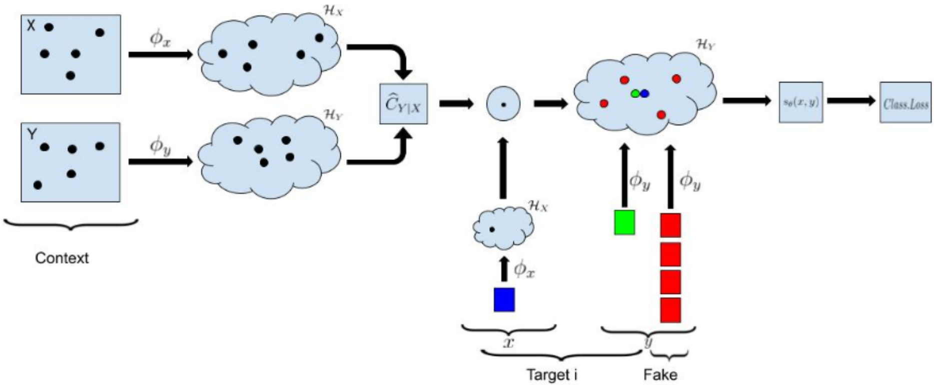
$$s_\theta(x',y') = \hat{\mu}_{Y|X=x'}(y') = \langle \widehat{\mathcal{C}}_{Y|X}\phi_x(x'), \phi_y(y')\rangle_{\mathcal{H}_Y}.$$

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:

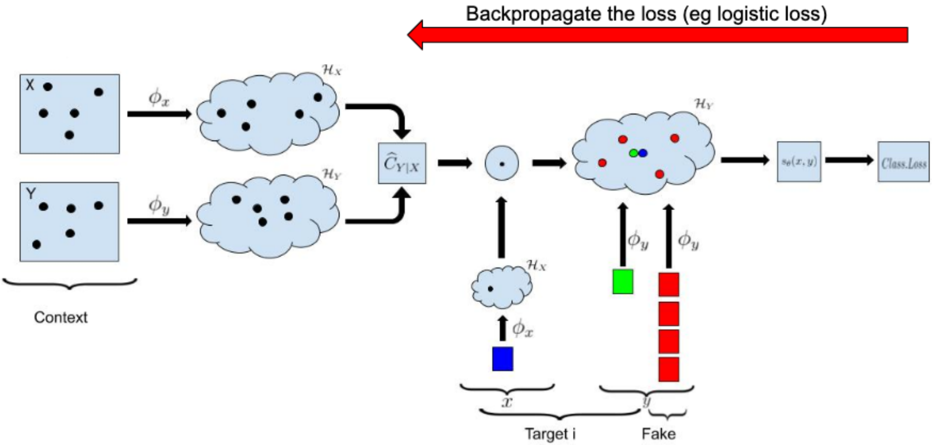
$$\mu_{Y|X=x'}(y') = \mathbb{E}\left[k_y(y',Y)|X=x'\right] = \int k_y(y',y)p(y|x')dy.$$

NCE: [Gutmann and Hyvärinen, 2010] train a classifier discriminating between true and artificial (fake) samples.

Proposed Method



Proposed Method



MetaCDE

- Three neural networks $\phi_x(x)$, $\phi_y(y)$, $b_\theta(x)$ (all parameters collated into θ) – these will be shared across tasks.
- Let $\mathcal{T} = \{T_1, \ldots, T_L\}$ be the set of L conditional density estimation tasks –each divided into context $\mathcal{D}_c^l = \{(x_i^{l,c}, y_i^{l,c})\}$ 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 κ fake responses $y_{i,j}^{l,f}$ from $p_f(y)$.
- Learn θ by training a True/Fake classifier on the True/Fake labels by minimizing the logistic loss across target data for all tasks jointly (SGD).

Dihedral angles in molecules

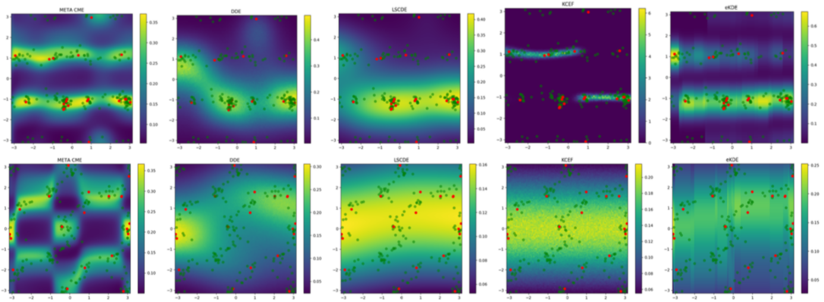


Figure: Left to right: MetaCDE (ours), DDE, LSCDE, KCEF, ϵ-KDE. The red dots are the context/training points and the green dots are points from the true density.

- 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.

Results

		Synthetic	Chemistry	NYC Taxi
MetaCDE (Ours)	Loglike	**197.84 $\pm$ 22.4**	**-305.49 $\pm$ 46.9**	**-1685.52$\pm$ 608.35**
MetaNN (Ours)	Loglike	132.776$\pm$130.87	-317.91$\pm$51.3	-2276.55 $\pm$ 608.9
	p-value	4.781e-06	1e-03	3.89e-10
Neural Process	Loglike	-81.11$\pm$18.5	-426.75$\pm$ 47.3	-3050.2 $\pm$ 822.8
	p-value	<2.2e-16	<2.2e-16	3.89e-10
DDE	Loglike	162.98 $\pm$ 69.0	-399.68 $\pm$ 41.3	-2236.07 $\pm$ 565.9
	p-value	8.14e-07	1.65e-15	3.89e-10
KCEF	Loglike	-388.30 $\pm$ 703.1	-724.40 $\pm$ 891.6	-1695.89$\pm$435.4
	p-value	<2.2e-16	9.72e-14	0.025
LSCDE	Loglike	44.95 $\pm$ 74.3	-407.32 $\pm$ 80.1	-2748.01 $\pm$ 549.2
	p-value	<2.2e-16	2.57e-14	3.89e-10
ϵ-KDE	Loglike	116.31 $\pm$ 236.9	-485.10 $\pm$ 303.4	-2337.90 $\pm$ 501.1
	p-value	2.38e-07	2.94e-14	4.13e-10

Table: Average held out log-likelihood on 100 different tasks. 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.
- The approach builds on the probabilistic approaches to meta learning, i.e. neural processes: MetaCDE also learns a task embedding based on the context set, but this embedding takes a specific form of the conditonal embedding operator and it is the feature maps that are learned.
- Combining the feature map networks using kernel mean embedding formalism gives better performance than learning the task embedding directly.

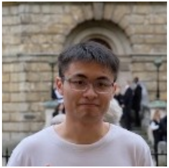
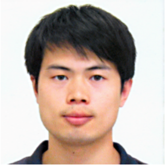
Summary

- Statistical modelling can be brought to bear in tandem with deep learning.
- 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

- Ho Chung Leon Law, Peilin Zhao, Lucian Chan, Junzhou Huang, and DS. **Hyperparameter Learning via Distributional Transfer**. NeurIPS 2019.
- Jean-Francois Ton, Lucian Chan, Yee Whye Teh, and DS. **Noise Contrastive Meta Learning for Conditional Density Estimation using Kernel Mean Embeddings**. *ArXiv e-prints:1906.02236*, appearing in NeurIPS Meta Learning Workshop 2019.



Surrogate Gaussian Process model

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

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

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

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

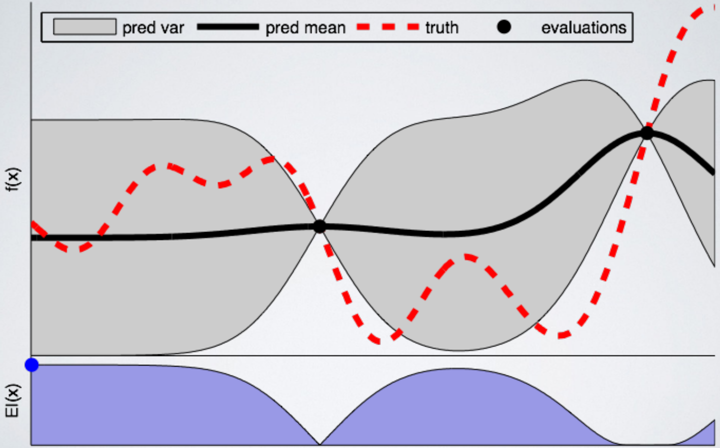
where

$$\begin{aligned}\mu\left(\theta\right) &= \mathbf{k}_{\theta\boldsymbol{\theta}}(\mathbf{K}+\tau^2 I)^{-1}\mathbf{y},\\ \kappa\left(\theta,\theta\right) &= k(\theta,\theta)-\mathbf{k}_{\theta\boldsymbol{\theta}}(\mathbf{K}+\tau^2 I)^{-1}\mathbf{k}_{\boldsymbol{\theta}\theta}\end{aligned}$$

Now can construct acquisition functions Details which balance

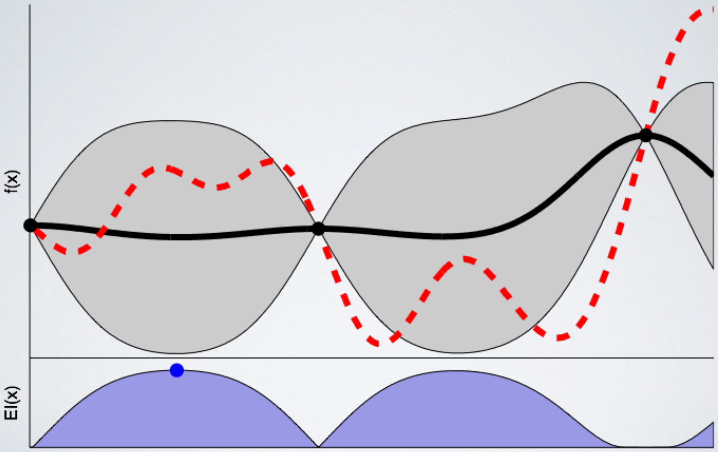
- Exploitation: seeking locations with low posterior mean $\mu\left(\theta\right)$,
- Exploration: seeking locations with high posterior variance $\kappa\left(\theta,\theta\right)$.

Illustrating Bayesian Optimization



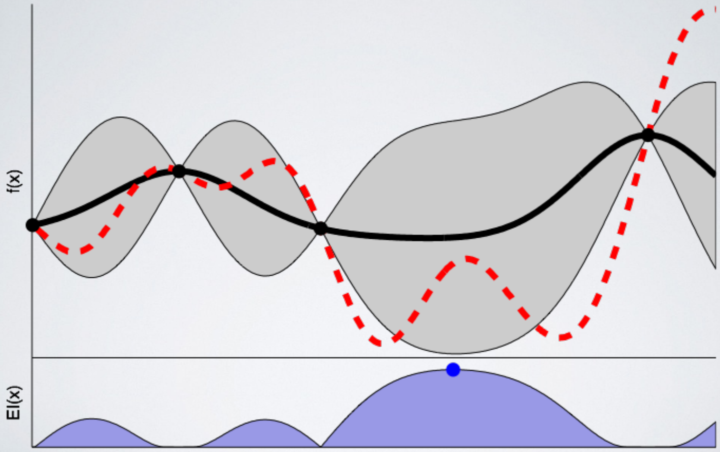
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization

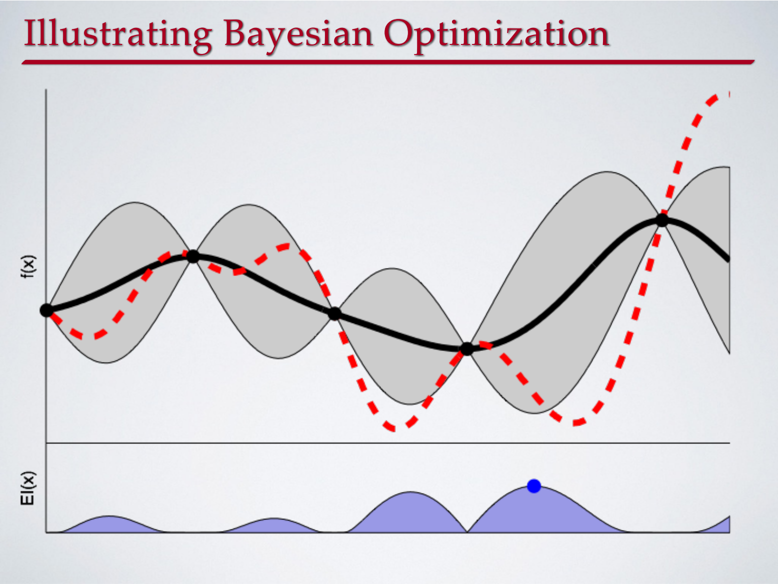


figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization

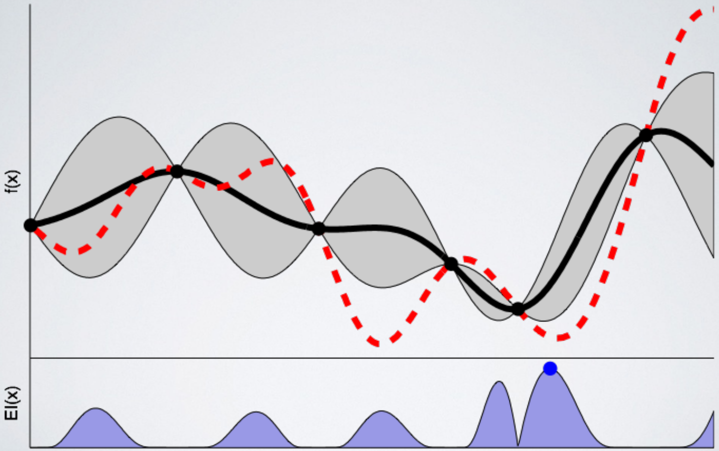


figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams



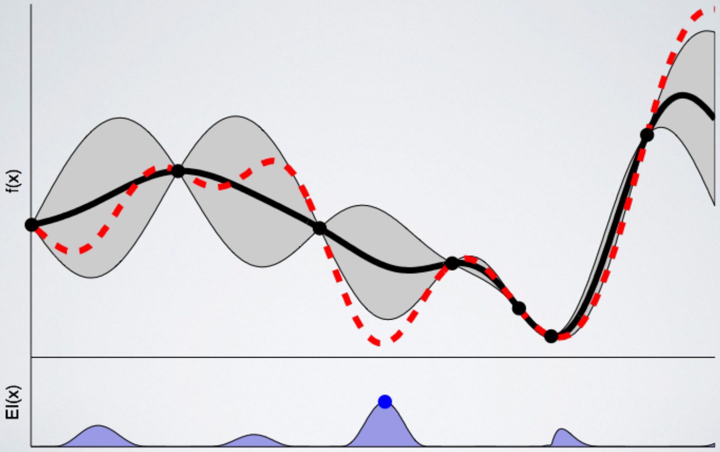
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



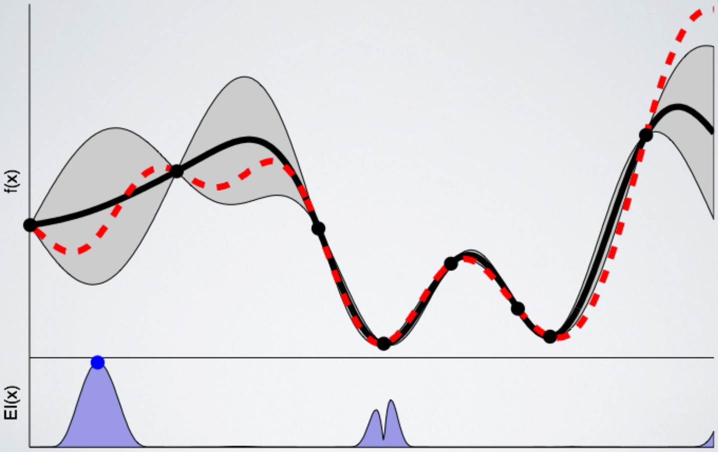
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



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

Illustrating Bayesian Optimization



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

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 distibution.

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

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

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

$$\alpha_{EI}(\theta) = \mathbb{E}[u(\theta)|\mathcal{D}] = \sigma(\theta)(\gamma(\theta)\Phi(\gamma(\theta)) + \phi(\gamma(\theta))).$$

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)

$$z|\beta \sim \mathcal{N}(\Upsilon\beta, \sigma^2 I) \qquad \beta \sim \mathcal{N}(0, \alpha I)$$

$$\Upsilon = [\upsilon([\theta_1^1, \Psi_1]), \dots, \upsilon([\theta_{N_1}^1, \Psi_1]), \dots, \\ \upsilon([\theta_1^n, \Psi_n]), \dots, \upsilon([\theta_{N_n}^n, \Psi_n])]^\top \in \mathbb{R}^{N \times d}$$

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

Conceptually similar setting to [Perrone et al, 2018] who fit a single BLR per task.

Noise Contrastive Estimation

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

$$\begin{aligned} P_\theta(\text{True}|y,x) &= \frac{\exp(s_\theta(x,y) + b_\theta(x))}{\exp(s_\theta(x,y) + b_\theta(x)) + \kappa p_f(y)} \\ &= \sigma\left(s_\theta(x,y) + b_\theta(x) - \log(\kappa p_f(y))\right). \end{aligned}$$

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

Defining s_θ

- Map x_i and y_i using feature maps (neural networks) $\phi_x : \mathcal{X} \to \mathcal{H}_X$ and $\phi_y : \mathcal{Y} \to \mathcal{H}_Y$ with all parameters collated into θ
- Estimate the conditional mean embedding operator $\widehat{\mathcal{C}}_{Y|X} = \Phi_y(K_{xx} + \lambda I)^{-1}\Phi_x^T$
- Given $\widehat{\mathcal{C}}_{Y|X}$, we can estimate the conditional mean embedding for any new x' using

$$\hat{\mu}_{Y|X=x'} = \widehat{\mathcal{C}}_{Y|X}\phi_x(x')$$

- We can then evaluate the conditional mean embedding at any new y' using

$$\hat{\mu}_{Y|X=x'}(y') = \langle \hat{\mu}_{Y|X=x'}, \phi_y(y') \rangle_{\mathcal{H}_Y} = \langle \widehat{\mathcal{C}}_{Y|X}\phi_x(x'), \phi_y(y') \rangle_{\mathcal{H}_Y}$$

Scoring function:

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

Defining s_θ

- 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:
$$\mu_{Y|X=x'}(y') = \mathbb{E}\left[k_y(y', Y)|X = x'\right] = \int k_y(y', y)p(y|x')dy,$$
where $k_y(y, y') := \langle \phi_y(y), \phi_y(y') \rangle_{\mathcal{H}_y}$.

- Recall that
$$P_\theta(\text{True}|y, x) = \sigma\left(s_\theta(x, y) + b_\theta(x) - \log(\kappa p_f(y))\right).$$