Hyperparameter Learning via Distributional Transfer

Dino Sejdinovic

Department of Statistics
University of Oxford

Workshop on Functional Inference and Machine Intelligence
The Institute of Statistical Mathematics, Tokyo
28/03/2019

Reference

- Ho Chung Leon Law, Peilin Zhao, Junzhou Huang, and DS, Hyperparameter Learning via Distributional Transfer, *ArXiv e-prints:1810.06305*, 2018.



Towards End-to-End Learning

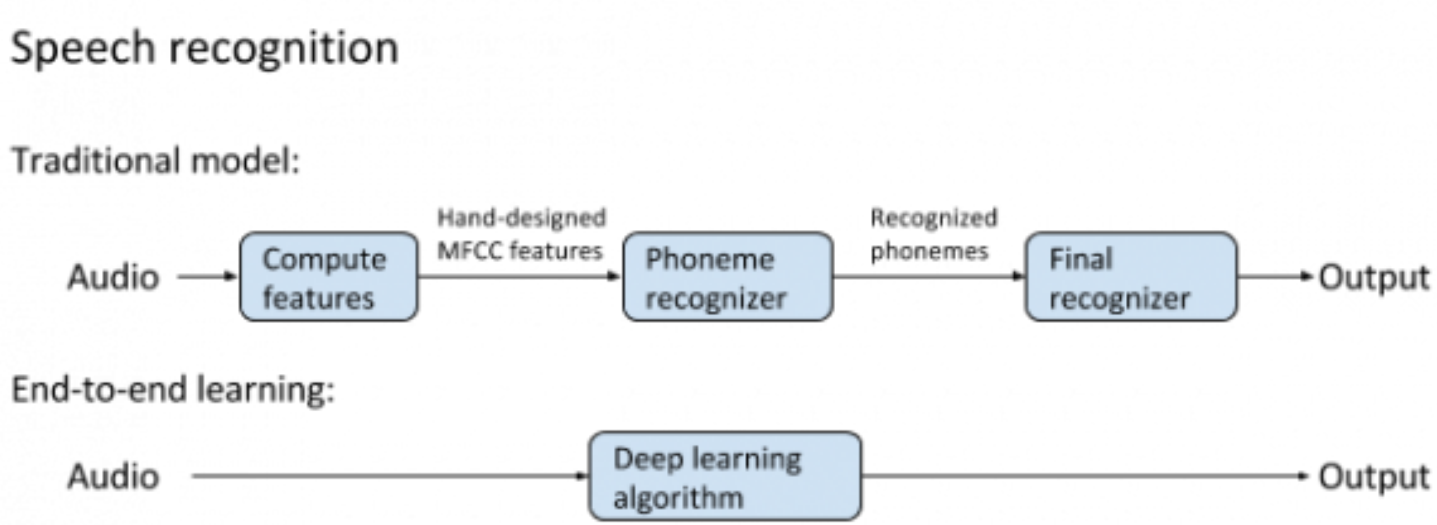
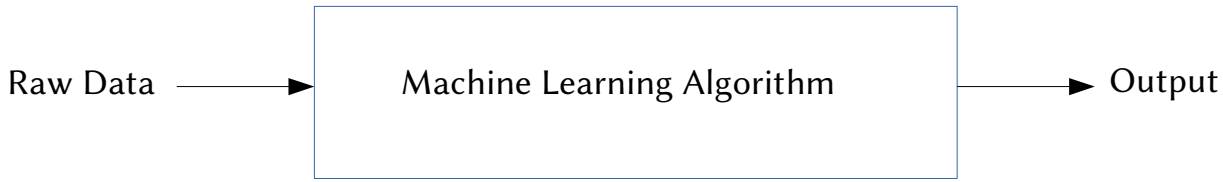
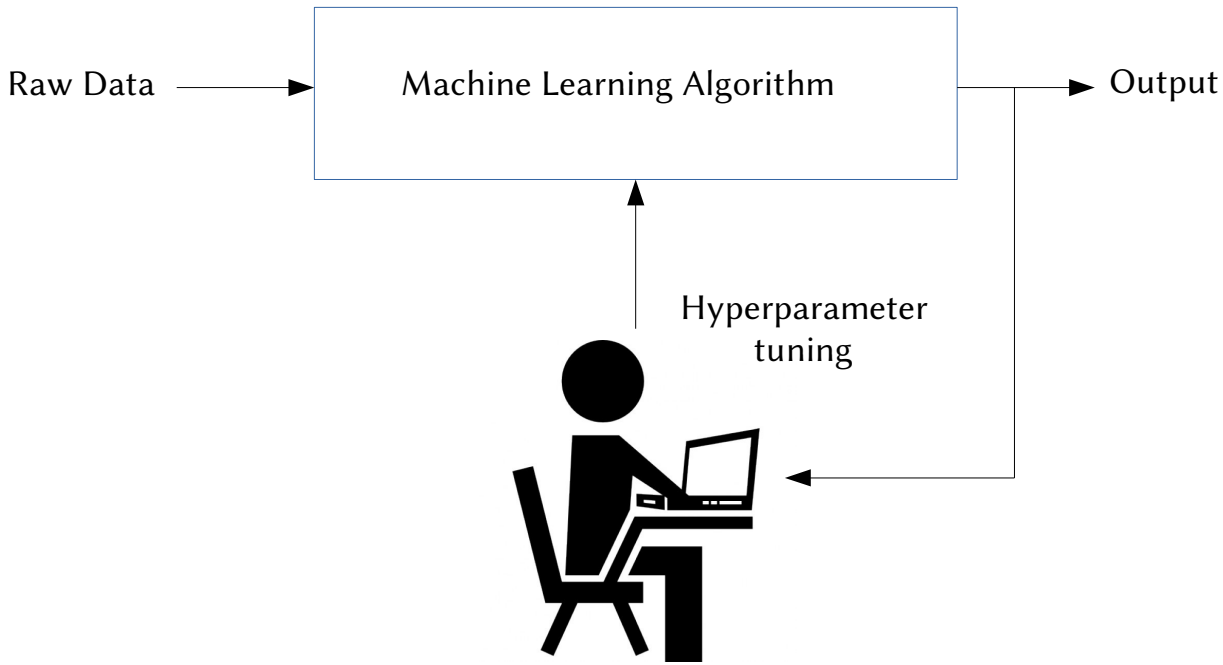


figure from https://blog.easysol.net/building-ai-applications/

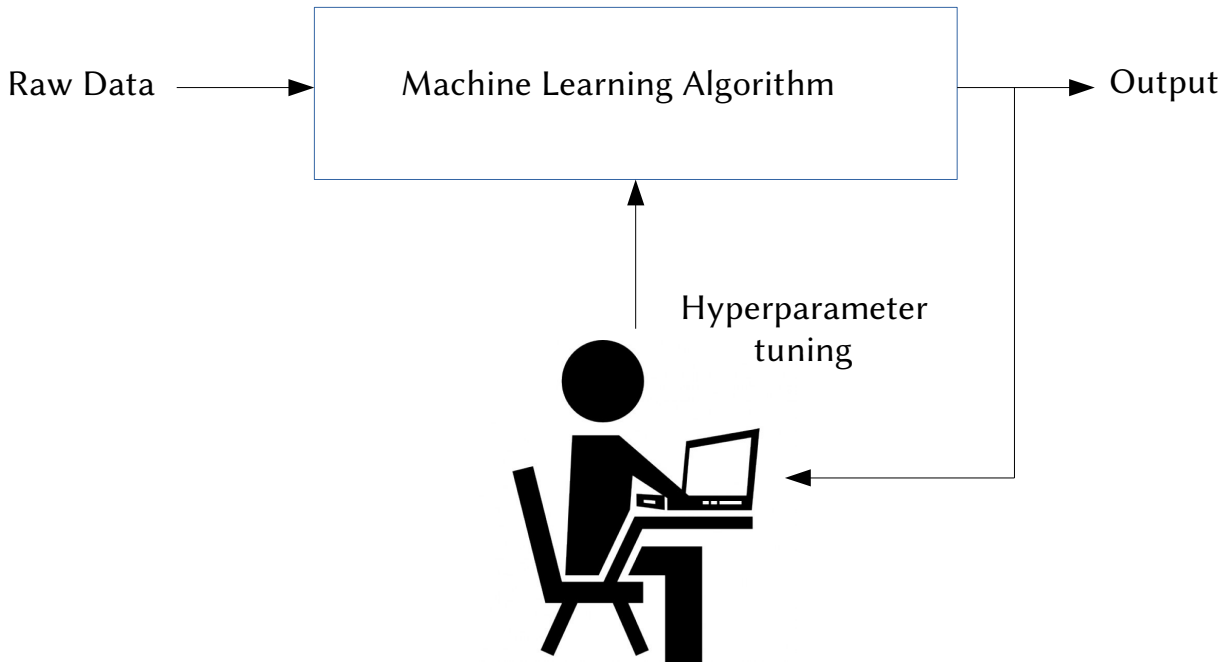
Towards End-to-End Learning



Towards End-to-End Learning



Towards End-to-End Learning



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

Optimizing "black-box" functions

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.

Optimizing "black-box" functions

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

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

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 (Gaussian Process).

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.

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

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

where

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

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

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.

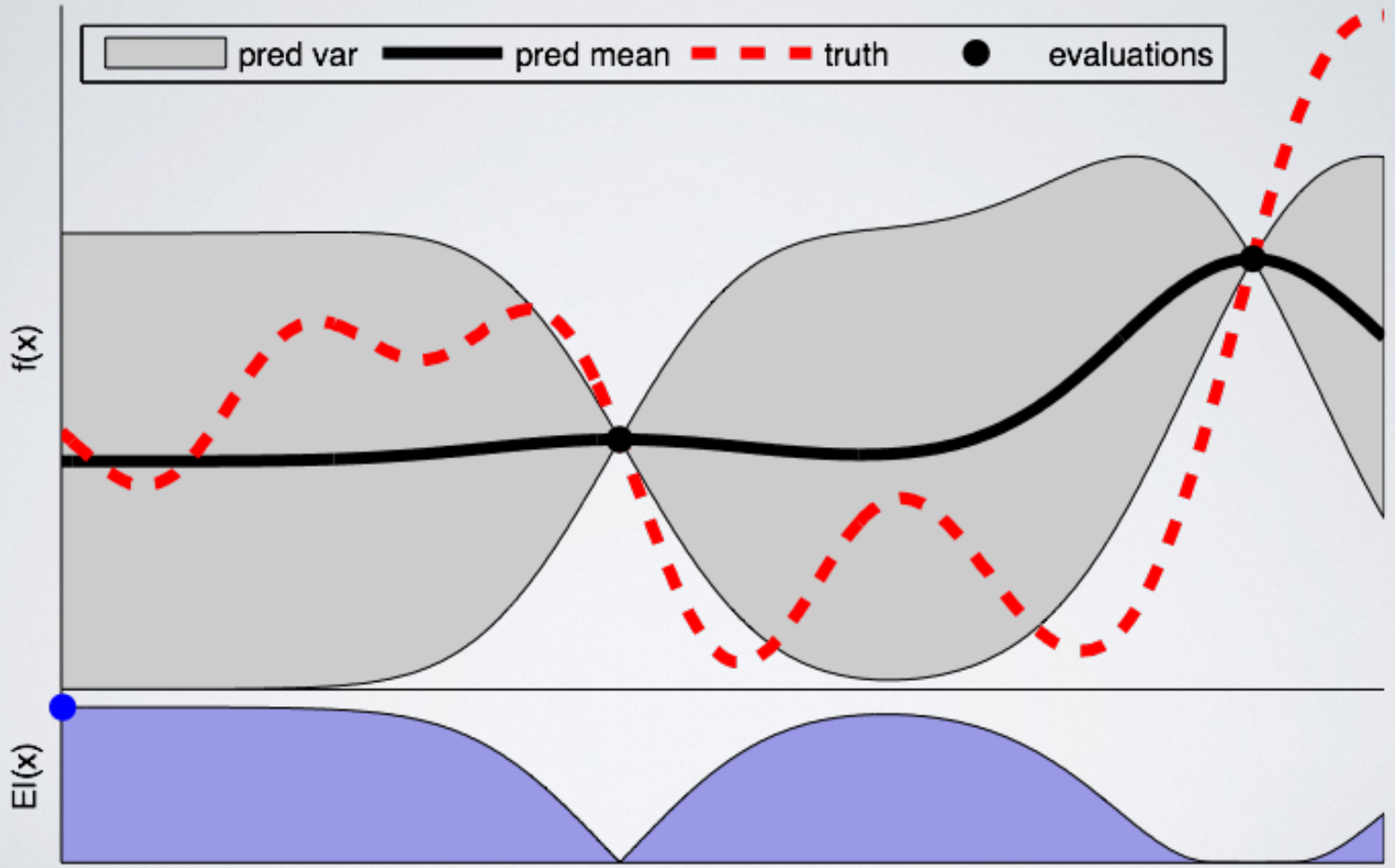
- PI (probability of improvement). $\tilde{\theta}$: the optimal location so far, $\tilde{y}$: the observed minimum. Let $u(\theta) = \mathbf{1}\{f(\theta) < \tilde{y}\}$,

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

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

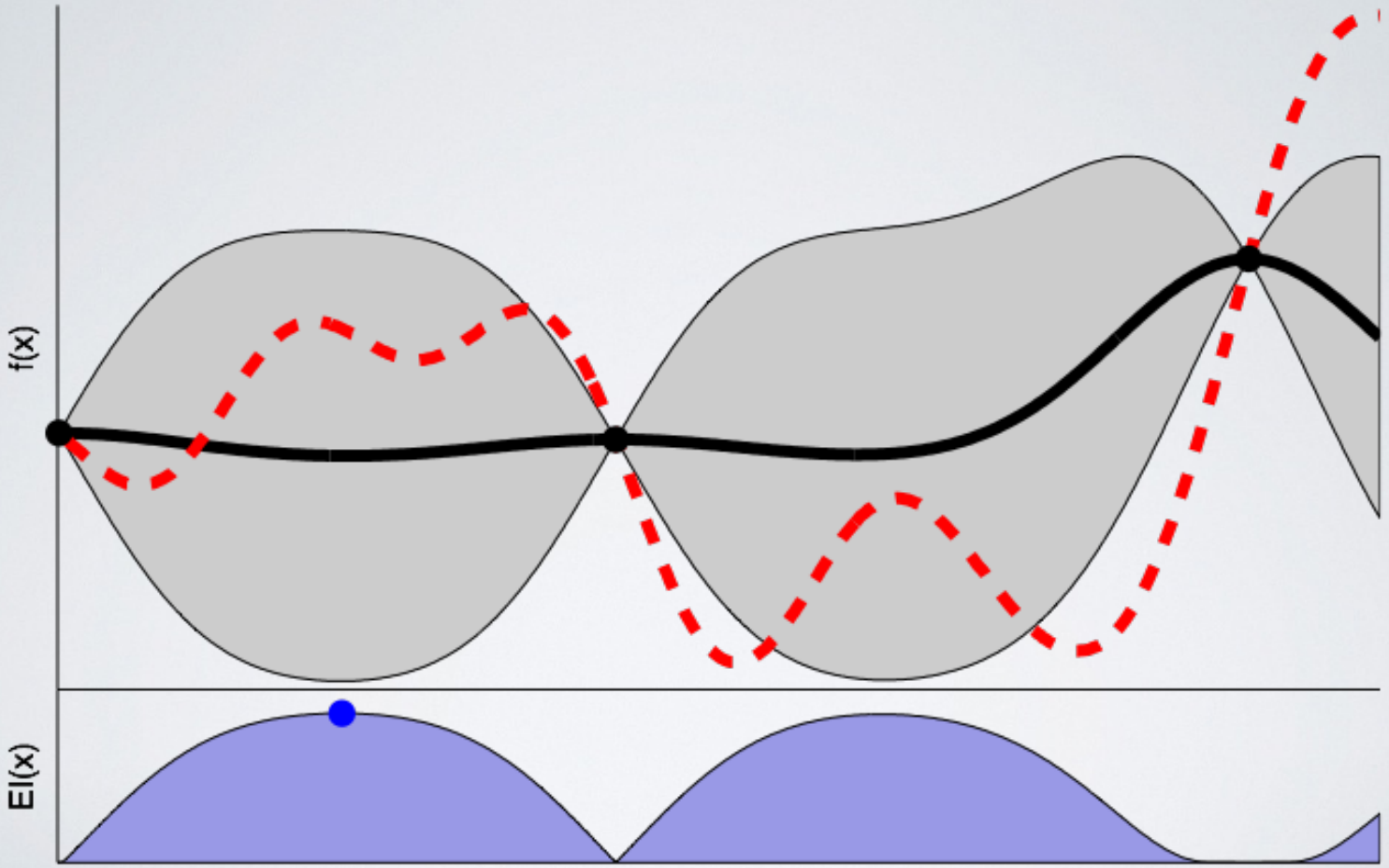
$$\alpha_{EI}(\theta) = \mathbb{E}[u(\theta)|\mathcal{D}] = \sigma(\theta)(\gamma(\theta)\Phi(\gamma(\theta)) + \phi(\gamma(\theta))).$$

Illustrating Bayesian Optimization



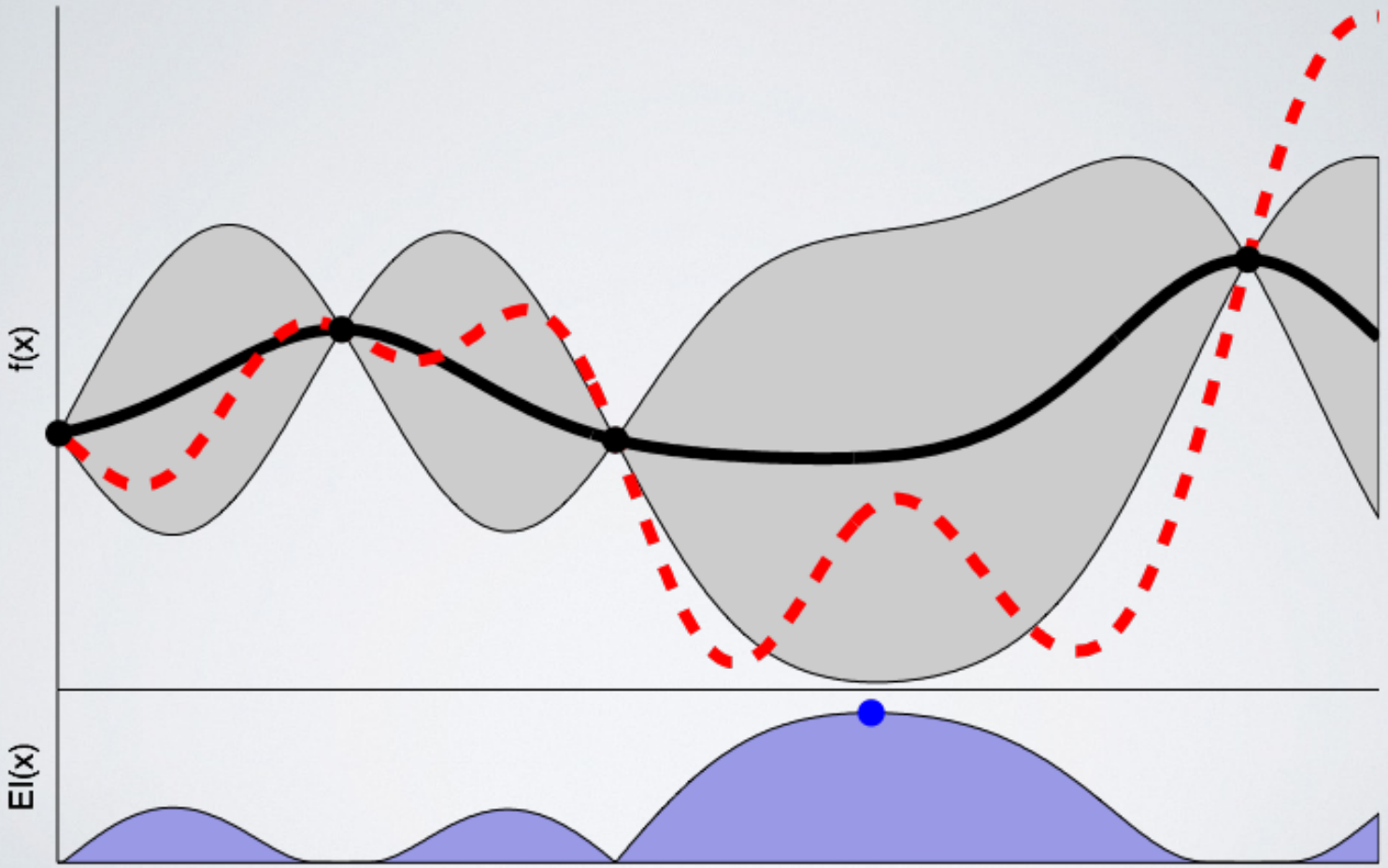
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



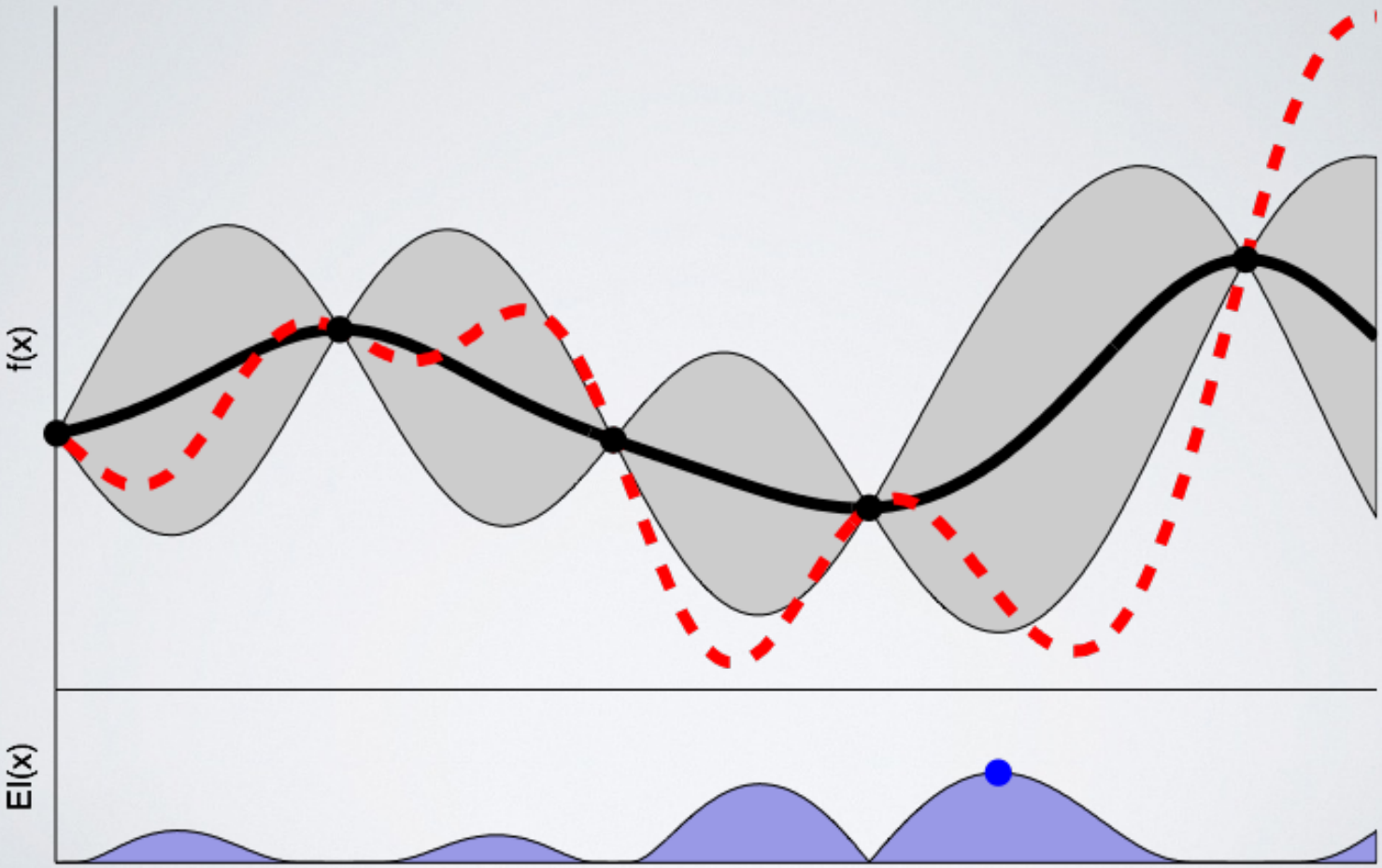
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



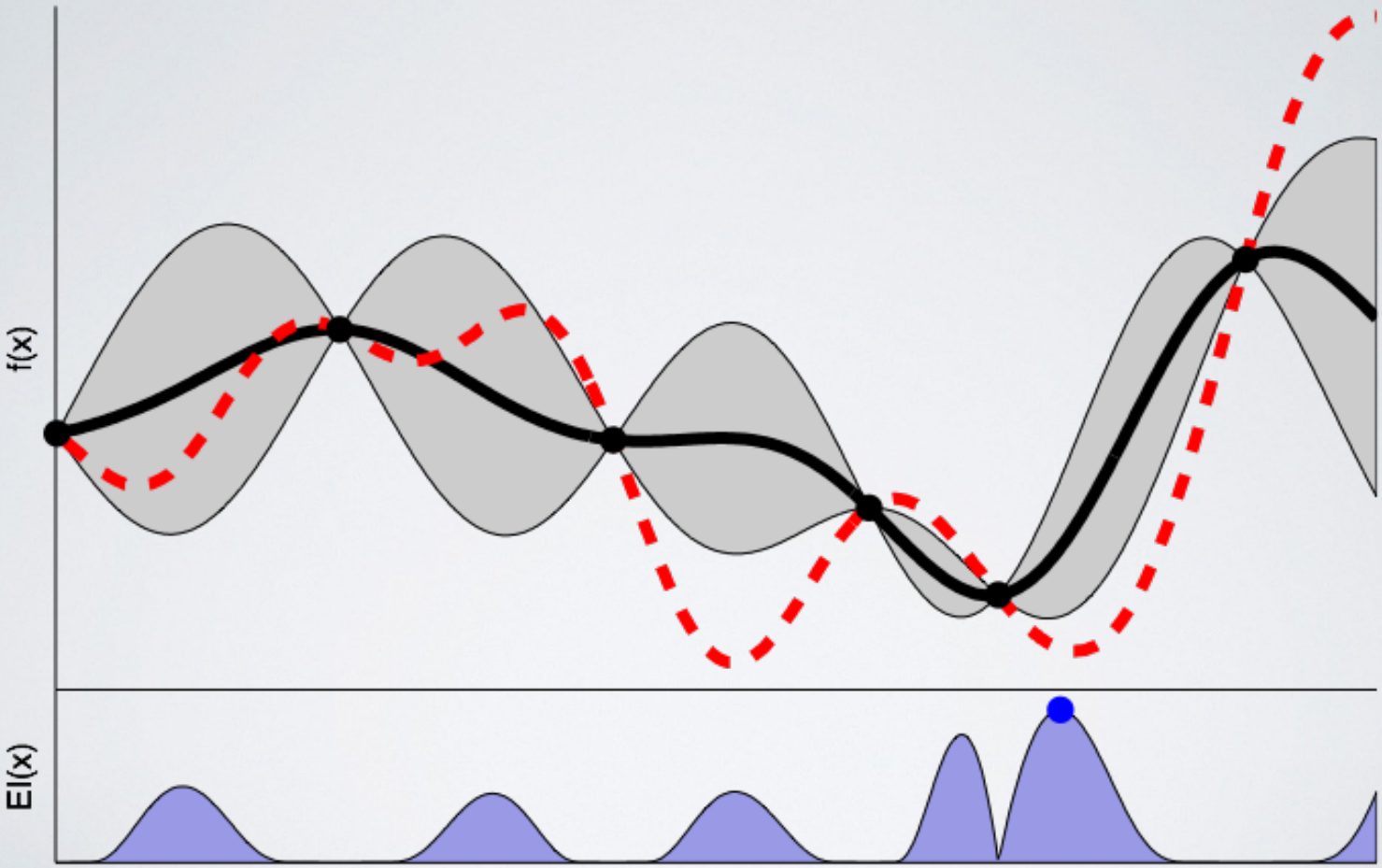
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



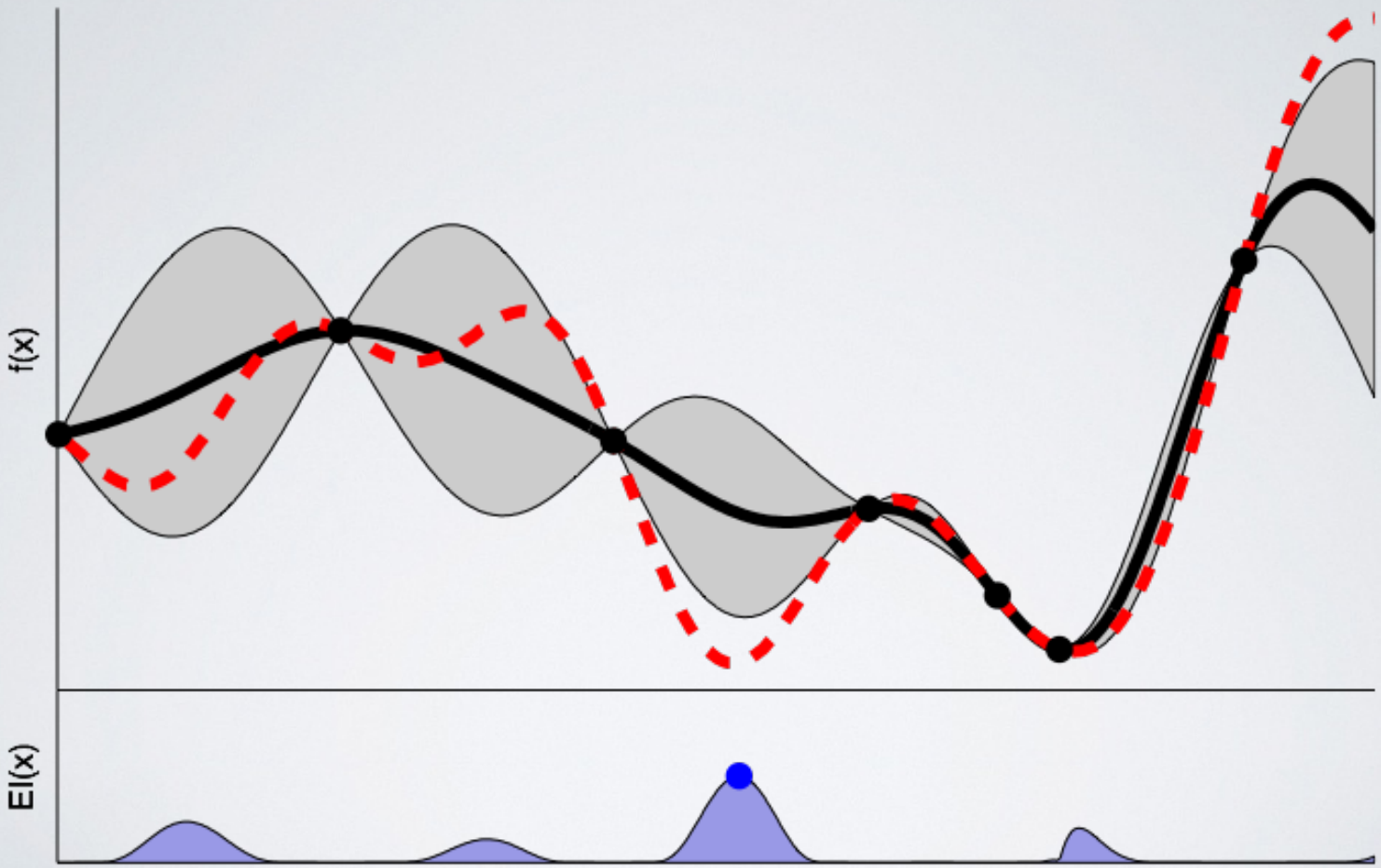
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



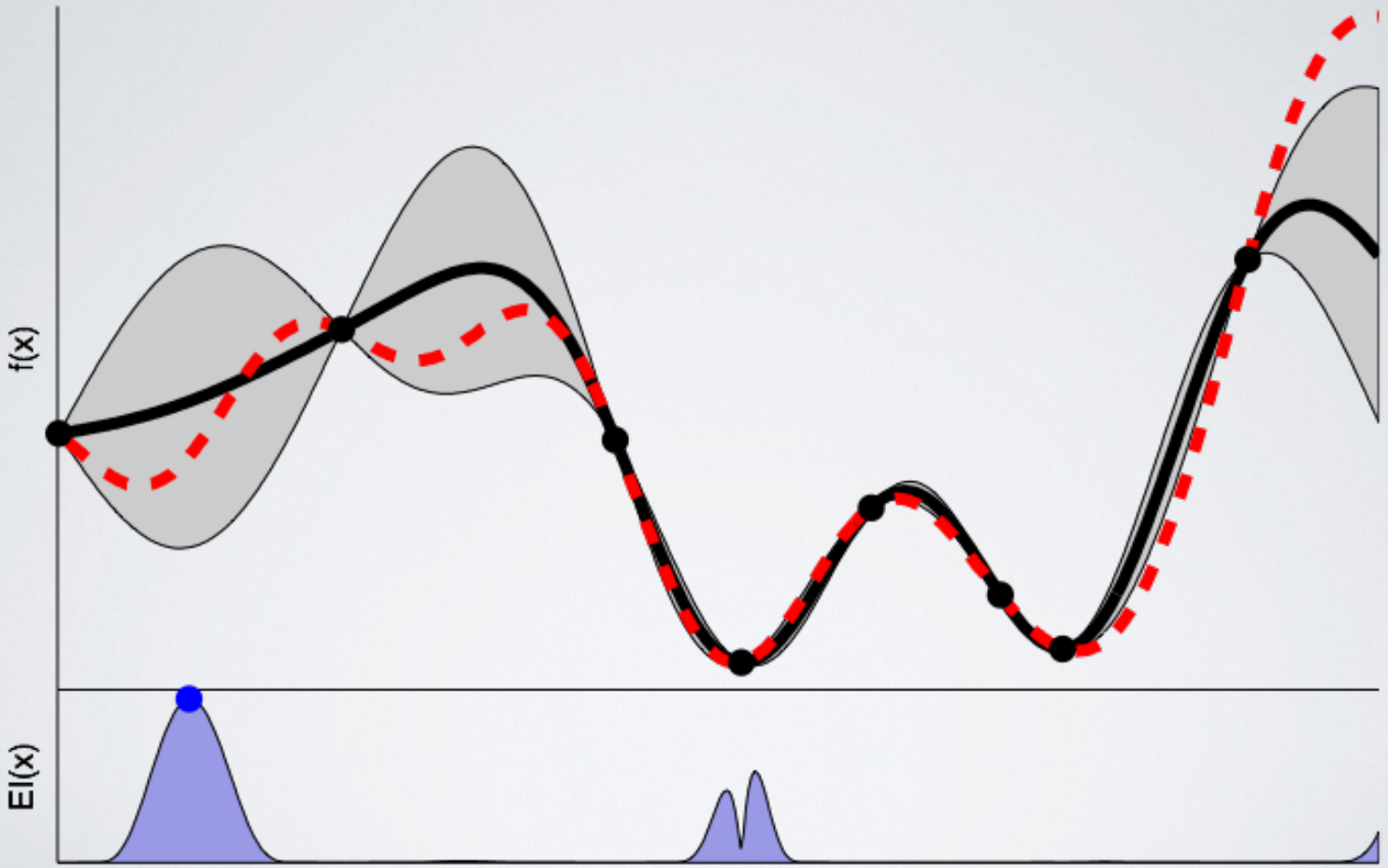
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization



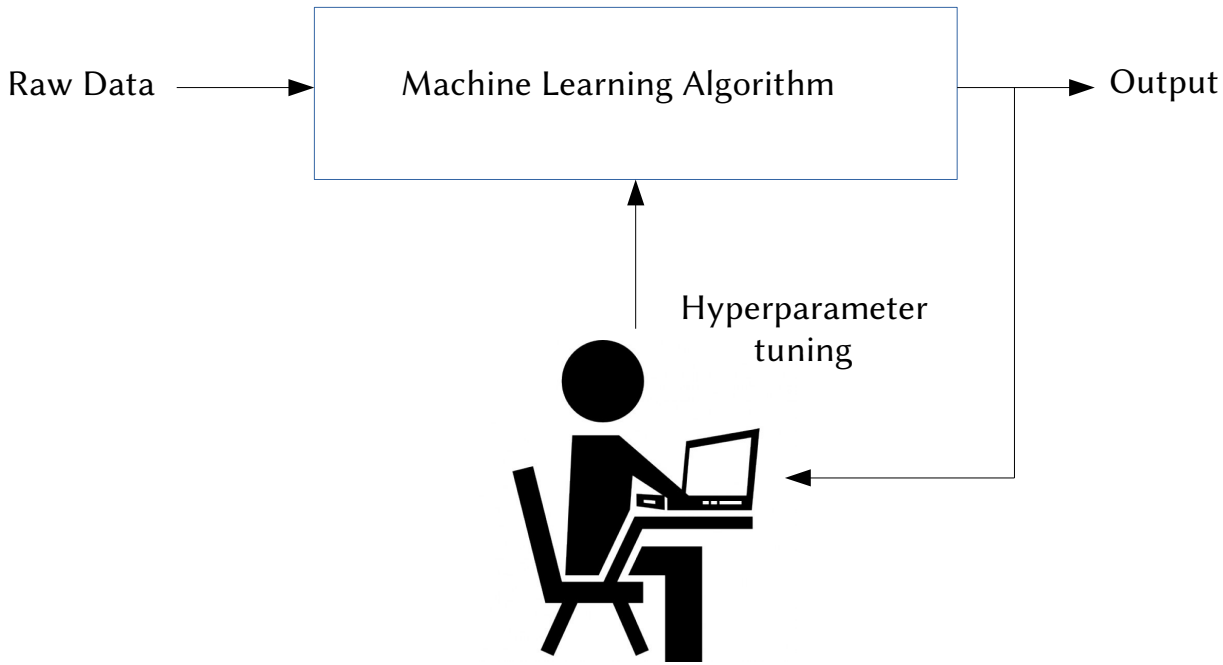
figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Illustrating Bayesian Optimization

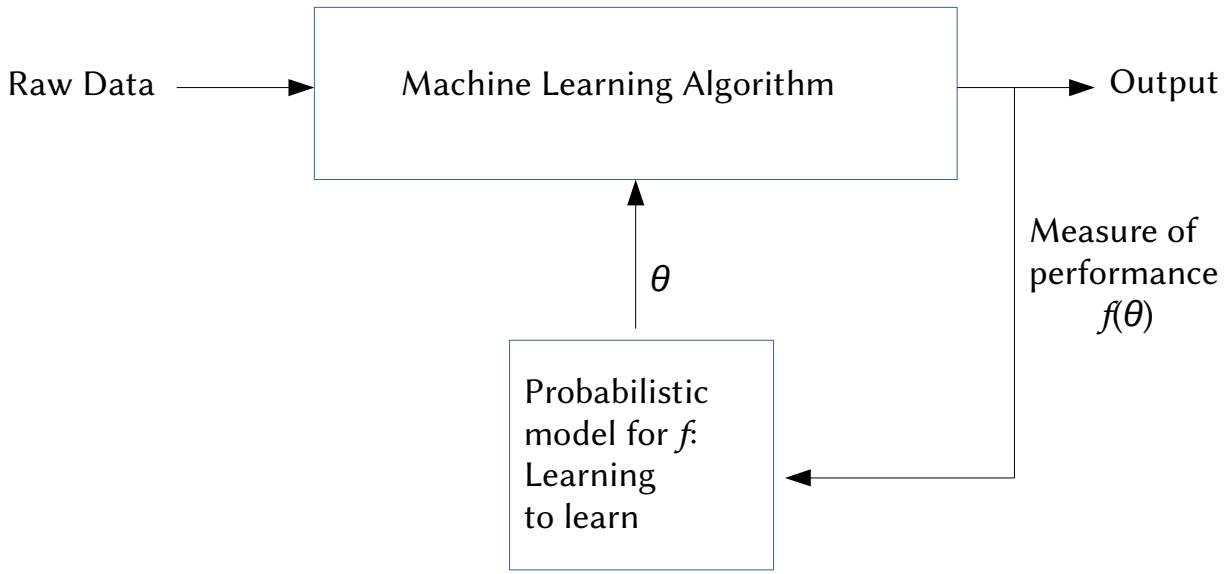


figures from *A Tutorial on Bayesian Optimization for Machine Learning* by Ryan Adams

Towards End-to-End Learning

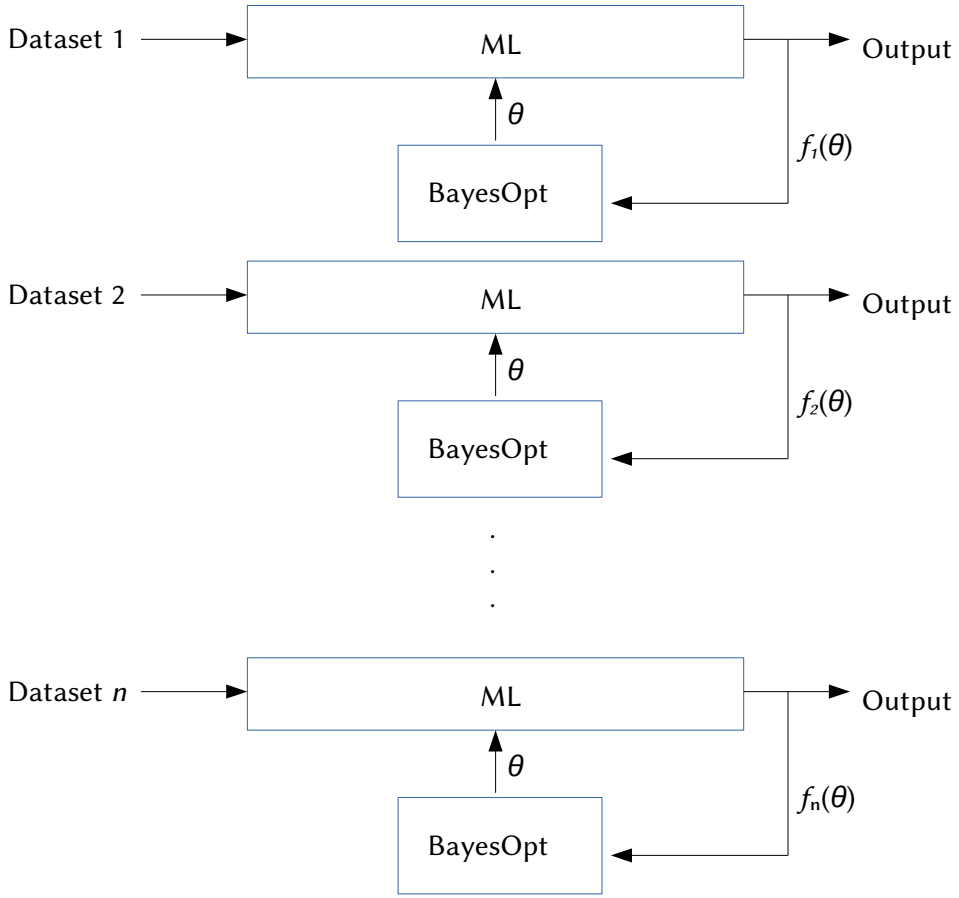


Towards End-to-End Learning



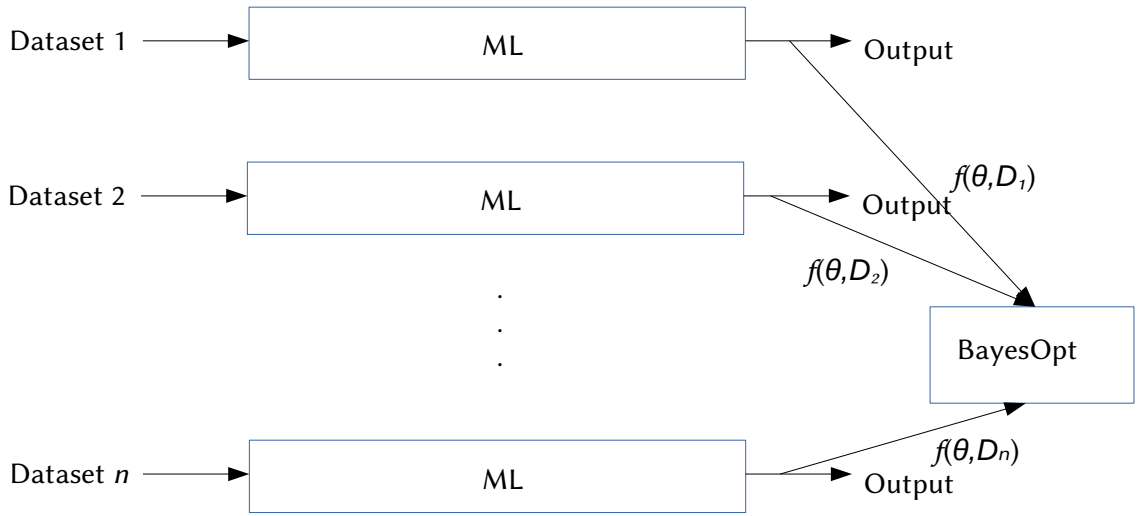
Transfer Hyperparameter Learning

- 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.



Transfer Hyperparameter Learning

- Consider a standard supervised learning setting: $f(\theta, D)$ is a performance measure of a trained ML model with hyperparameters θ and data $D = \{\mathbf{x}_l, y_l\}_{l=1}^{s}$, $\mathbf{x}_l \in \mathcal{X}$ 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 $\{\theta_r^i, f(\theta_r^i, D_i)\}_{r=1}^{N_i}$, with source datasets

$$D_i = \{\mathbf{x}_l^i, y_l^i\}_{l=1}^{s_i}, i = 1, \ldots, n.$$

- 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

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

i.e.

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

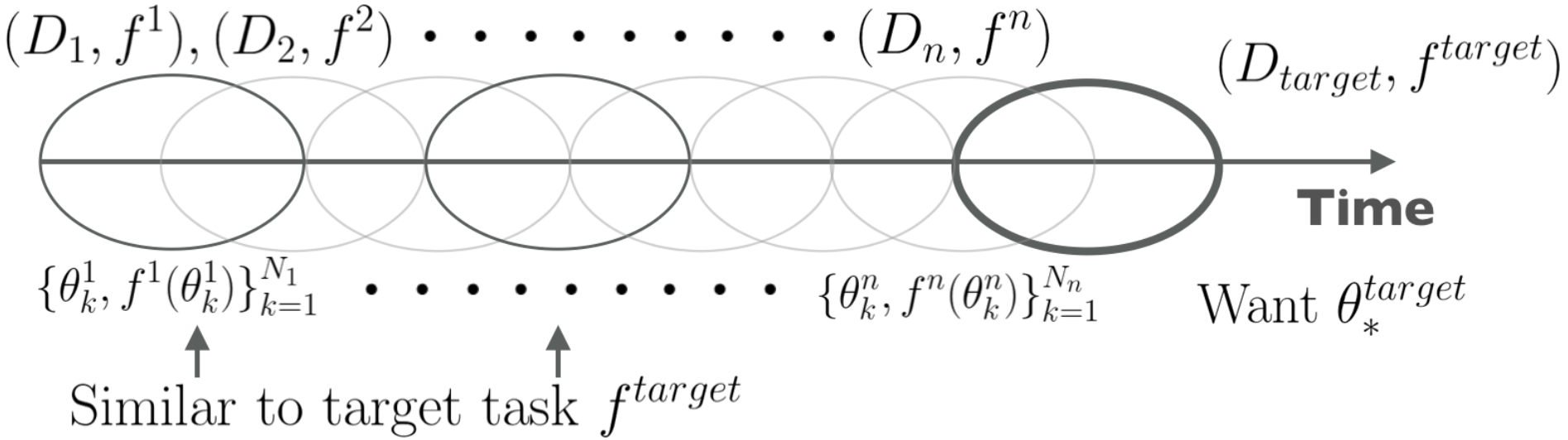
Motivating Example

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 θ, 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 θ shift as data distribution changes e.g. weekend vs weekday or holiday vs no holiday
- Not all previous tasks are equally useful.



Dataset 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).

Dataset representation for hyperparameter learning

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_θ 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

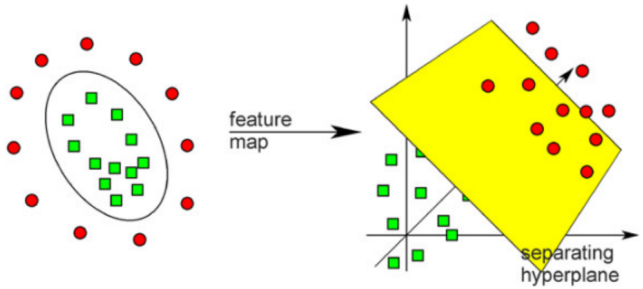
No joint GP model, but warmstart target hyperparameters to the optimal values from source datasets with closest metafeatures.

[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.

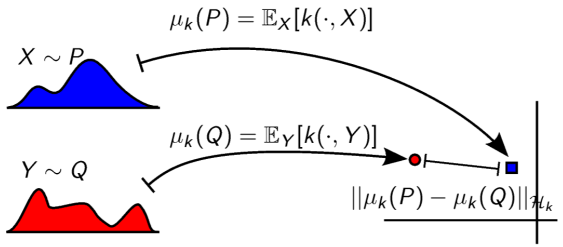
Kernel Mean Embeddings

- implicit feature map $x \mapsto k(\cdot, x) \in \mathcal{H}_k$
  replaces $x \mapsto [\phi_1(x), \dots, \phi_s(x)] \in \mathbb{R}^s$
- $\langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}_k} = k(x, y)$
  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 $P \mapsto [\mathbb{E}\phi_1(X), \dots, \mathbb{E}\phi_s(X)] \in \mathbb{R}^s$
- $\langle \mu_k(P), \mu_k(Q) \rangle_{\mathcal{H}_k} = \mathbb{E}_{X \sim P, Y \sim Q} k(X, Y)$
  inner products easy to estimate
  - nonparametric two-sample, independence, conditional independence, interaction testing, learning on distribution inputs



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

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 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:

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

   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.

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)

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

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

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.

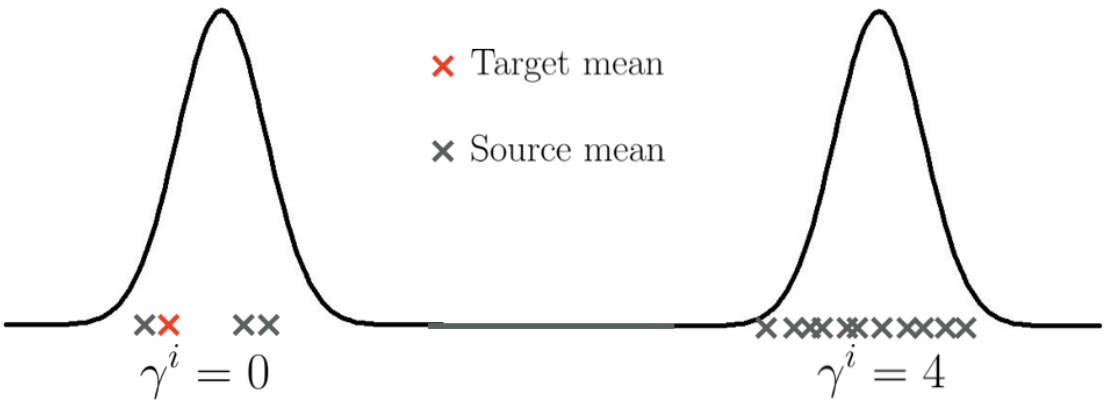
Experiments

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

Toy Example

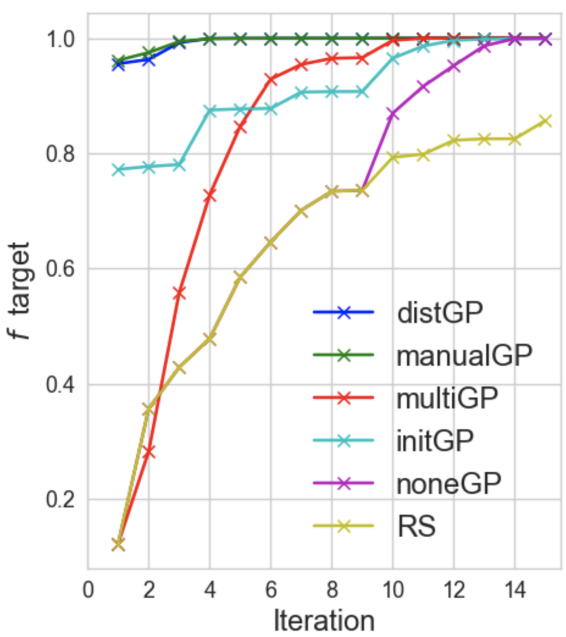


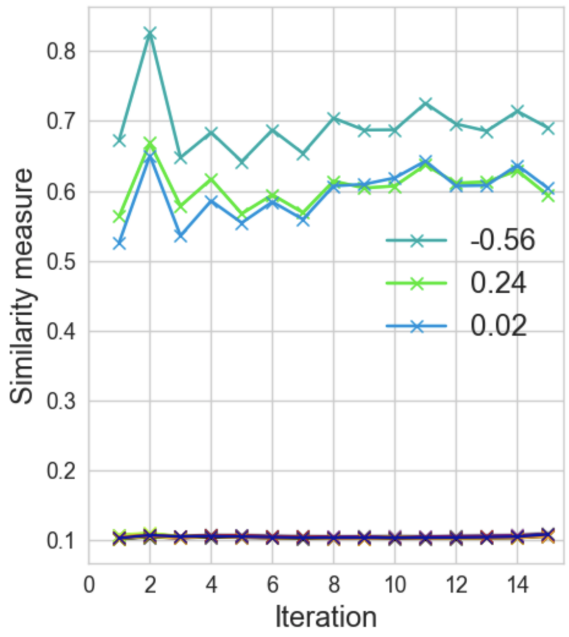
D_i is obtained for some fixed γ^i as $\mu^i \sim \mathcal{N}(\gamma^i, 1)$, $\quad \{x_\ell^i\}_{\ell=1}^{s_i} | \mu^i \overset{i.i.d.}{\sim} \mathcal{N}(\mu^i, 1)$ and the objective to maximize is

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

where θ plays the role of a "hyperparameter".
15 source tasks, 3 with $\gamma_i = 0$ and 12 with $\gamma_i = 4$. Target has $\gamma_i = 0$.

Toy Example





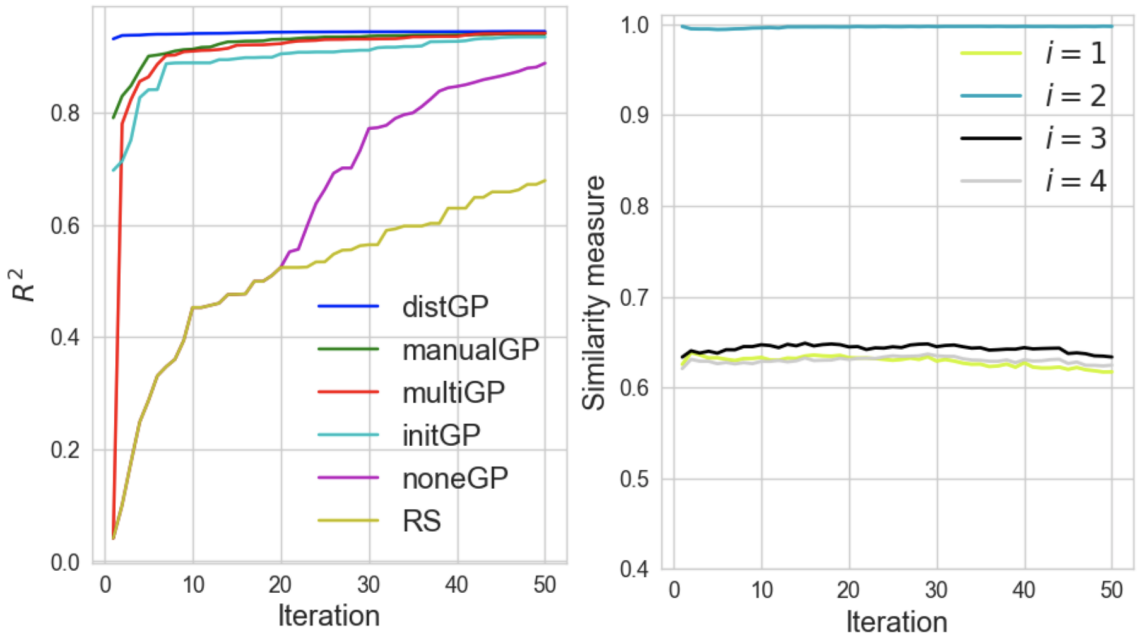
- feature representation learned to place high similarity on the three source datasets sharing the same γ^i and hence having similar values of μ^i, while placing low similarity on the other source datasets
- manualBO also few-shots the optimum as it encodes the mean feature
- initBO and multiBO converge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

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

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

where θ 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.

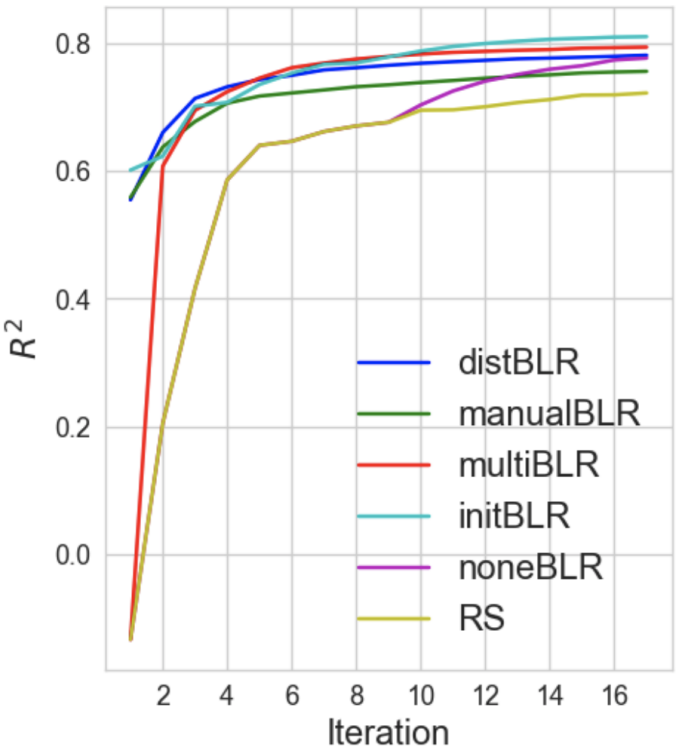
Dataset i with $\mathbf{x}_\ell^i \in \mathbb{R}^6$ and $y_\ell^i \in \mathbb{R}$:

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

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

Parkinson's telemonitoring

- The Parkinson's telemonitoring dataset: voice measurements using a telemonitoring device for 42 patients with Parkinson's disease. The label is the clinician's symptom score for *each recording*.
- Following [Blanchard et al, 2017], we treat each patient as a separate regression task, using R^2 as the performance measure.
- We designate each patient as the target and all others as sources, averaging results. Full GP is prohibitive, so use BLR.



- Varying results across different patients, but on average all transfer methods are able to leverage the source task information and for many patients few-shot the optimum.
- Task similarity can be exploited in the context of hyperparameter learning.

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?

Reference

- Ho Chung Leon Law, Peilin Zhao, Junzhou Huang, and DS, Hyperparameter Learning via Distributional Transfer, *ArXiv e-prints:1810.06305*, 2018.



Thank you!