Scalable Hyperparameter Transfer learning

Valerio Perrone[†],
Rodolphe Jenatton[†], Cédric Archambeau[*], Matthias Seeger[*]

AWS AI[†]/Amazon Research[*], Berlin

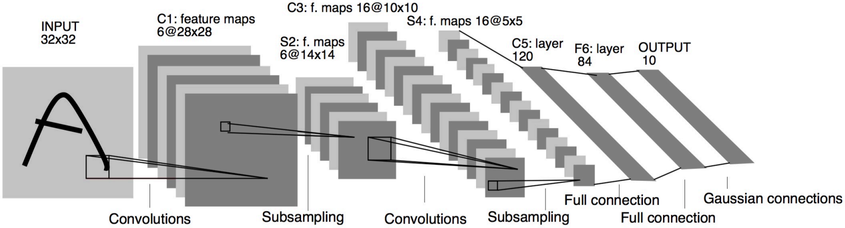
Co-authors

R. Jenatton C. Archambeau M. Seeger

Most of the material from

V. Perrone, R. Jenatton, M. Seeger, C. Archambeau
Scalable Hyperparameter Transfer learning. NeurIPS 2018

Tuning deep neural nets for optimal performance

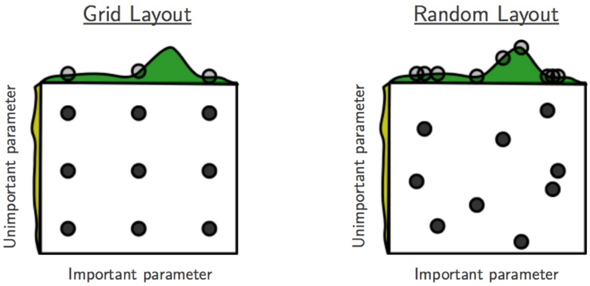


LeNet5 [LBBH98]

The search space $\mathcal{X}$ is large and diverse:

- Architecture: # hidden layers, activation functions, ...
- Model complexity: regularization, dropout, ...
- Optimisation parameters: learning rates, momentum, batch size, ...

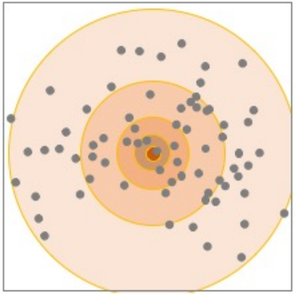
Two straightforward approaches



(Figure by Bergstra and Bengio, 2012)

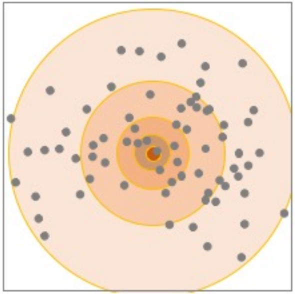
- Exhaustive search on a regular or random grid
- Complexity is exponential in p
- Wasteful of resources, but easy to parallelise
- Memoryless

Hyperparameter transfer learning

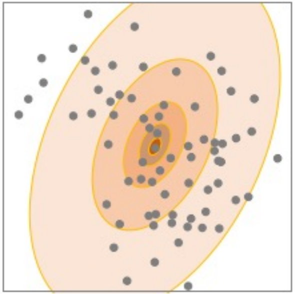


HPO Job 1

Hyperparameter transfer learning

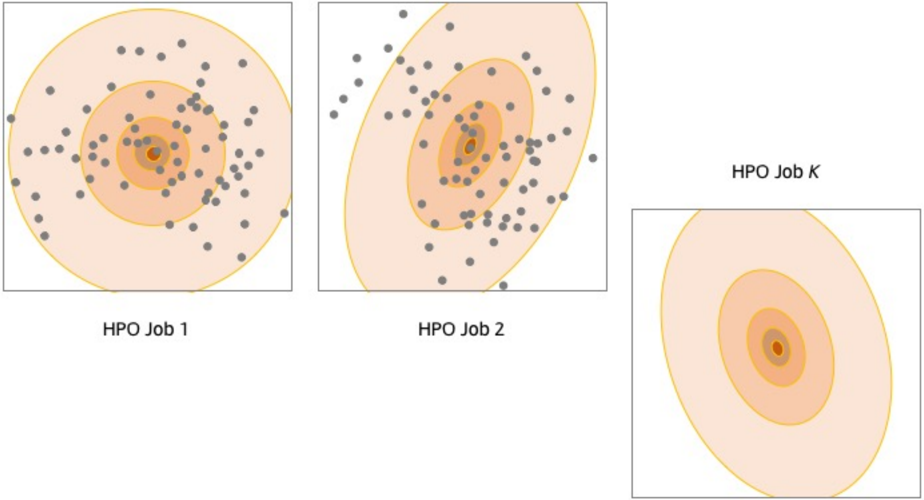


HPO Job 1

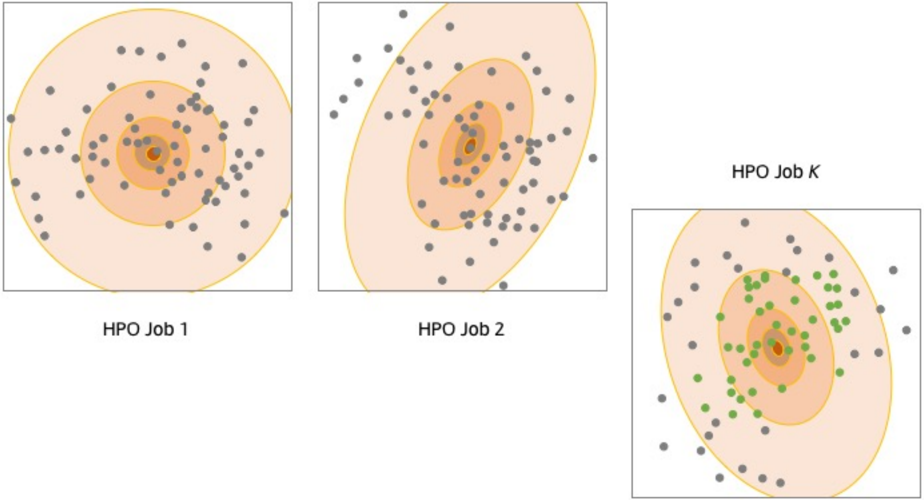


HPO Job 2

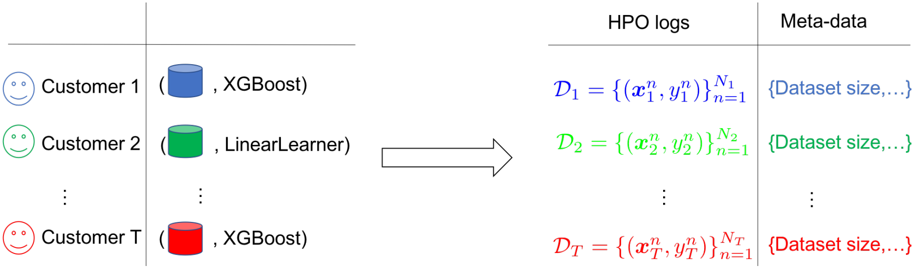
Hyperparameter transfer learning



Hyperparameter transfer learning

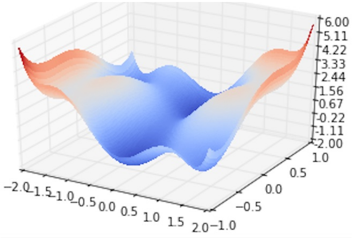


Motivation



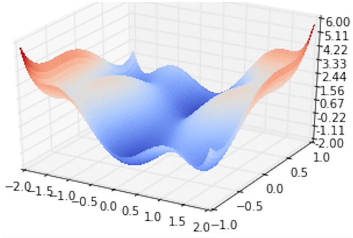
- **Transfer learning**: Exploit evaluations of related past tasks
  - A given ML algorithm tuned over different datasets
  - Can we do it in absence of meta-data?
- **Scalability**: Both with respect to
  - #evaluations: $\sum_{t=1}^{T} N_t$
  - #tasks: T

Black-box global optimisation



- The function f to optimise can be non-convex.
- The number of hyperparameters p is moderate (typically < 20).

Black-box global optimisation



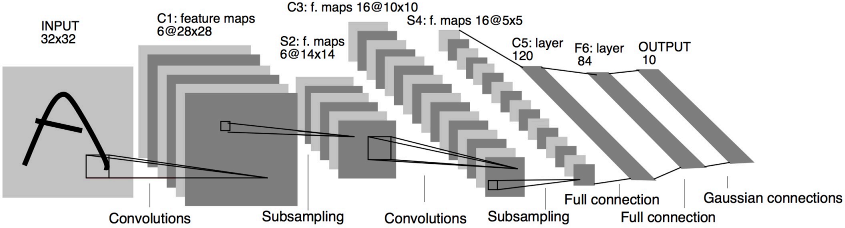
- The function f to optimise can be non-convex.
- The number of hyperparameters p is moderate (typically < 20).

Our goal is to solve the following optimisation problem:

$$\boldsymbol{x}_\star = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmin}} \; f(\boldsymbol{x}).$$

- Evaluating $f(\boldsymbol{x})$ is expensive.
- No analytical form or gradient.
- Evaluations may be noisy.

Example: tuning deep neural nets [SLA12, SRS+15, KFB+16]



LeNet5 [LBBH98]

- $f(\mathbf{x})$ is the validation loss of the neural net as a function of its hyperparameters $\mathbf{x}$.
- Evaluating $f(\mathbf{x})$ is very **costly** $\approx$ up to weeks!

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$$

Canonical algorithm:

- Surrogate model $\mathcal{M}$ of f `#cheaper to evaluate`
- Set of evaluated candidates $\mathcal{C} = \{\}$

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

Canonical algorithm:

- Surrogate model $\mathcal{M}$ of f `#cheaper to evaluate`
- Set of evaluated candidates $\mathcal{C} = \{\}$
- While some `BUDGET` available:
  - Select candidate $\mathbf{x}_{\text{new}} \in \mathcal{X}$ using $\mathcal{M}$ and $\mathcal{C}$ `#exploration/exploitation`

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

Canonical algorithm:

- Surrogate model $\mathcal{M}$ of f `#cheaper to evaluate`
- Set of evaluated candidates $\mathcal{C} = \{\}$
- While some `BUDGET` available:
  - Select candidate $\mathbf{x}_{\text{new}} \in \mathcal{X}$ using $\mathcal{M}$ and $\mathcal{C}$ `#exploration/exploitation`
  - Collect evaluation y_{new} of f at $\mathbf{x}_{\text{new}}$ `#time-consuming`

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmin}} \; f(\boldsymbol{x})$$

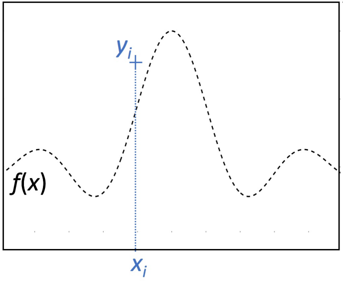
Canonical algorithm:

- Surrogate model $\mathcal{M}$ of f `#cheaper to evaluate`
- Set of evaluated candidates $\mathcal{C} = \{\}$
- While some `BUDGET` available:
  - Select candidate $\mathbf{x}_{\text{new}} \in \mathcal{X}$ using $\mathcal{M}$ and $\mathcal{C}$ `#exploration/exploitation`
  - Collect evaluation y_{new} of f at $\mathbf{x}_{\text{new}}$ `#time-consuming`
  - Update $\mathcal{C} = \mathcal{C} \cup \{(\mathbf{x}_{\text{new}}, y_{\text{new}})\}$
  - Update $\mathcal{M}$ with $\mathcal{C}$ `#Update surrogate model`
  - Update `BUDGET`

Bayesian (black-box) optimisation with Gaussian processes

1. Learn a probabilistic model of f, which is cheap to evaluate:

$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

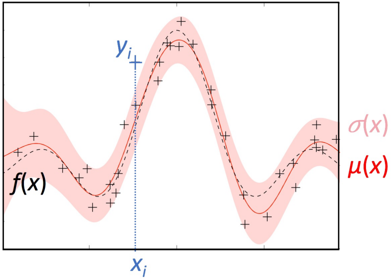


Bayesian (black-box) optimisation with Gaussian processes

1. Learn a probabilistic model of f, which is cheap to evaluate:

$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

2. Given the observations $\boldsymbol{y} = (y_1, \ldots, y_n)$, compute the predictive mean and the predictive standard deviation:

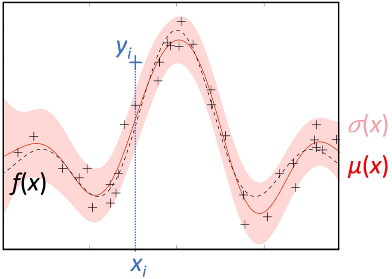


Bayesian (black-box) optimisation with Gaussian processes

1. Learn a probabilistic model of f, which is cheap to evaluate:

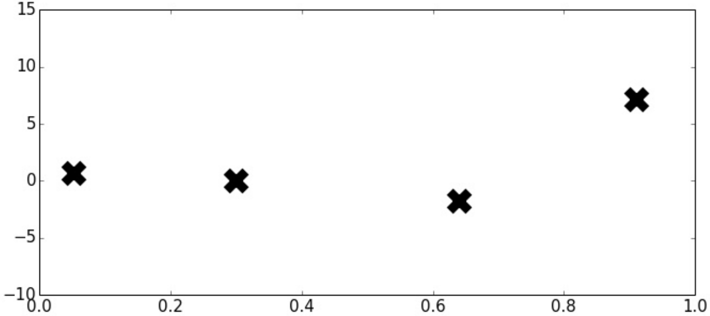
$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

2. Given the observations $\boldsymbol{y} = (y_1, \ldots, y_n)$, compute the predictive mean and the predictive standard deviation:

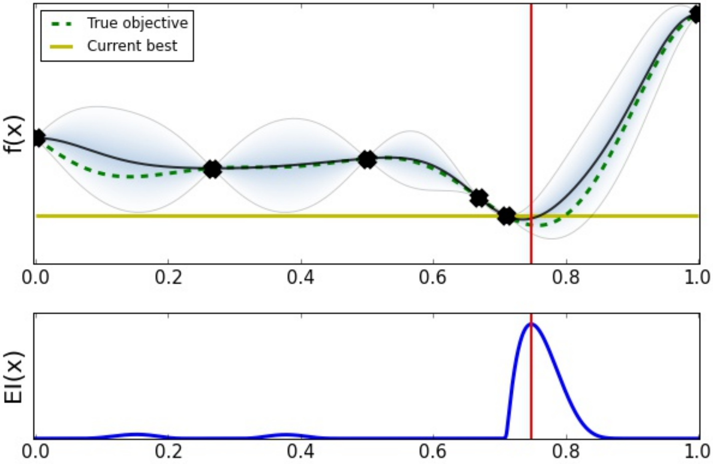


3. Repeatedly query f by balancing **exploitation** against **exploration**

Where is the minimum of $f(x)$?



Bayesian optimisation in practice



(Image credit: Javier González)

Bayesian optimization with transfer learning

Problem statement:

- T functions $\{f_t(\mathbf{x})\}_{t=1}^{T}$ with observations $\mathcal{D}_t = \{(\mathbf{x}_t^n, y_t^n)\}_{n=1}^{N_t}$
- May/may not have meta-data (or contextual features) for $\{f_t(\mathbf{x})\}_{t=1}^{T}$
- **Goal:** Optimize some fixed $f_{t_0}(\mathbf{x})$ while exploiting $\{\mathcal{D}_t\}_{t=1}^{T}$
- (this is not multi-objective!)

Bayesian optimization with transfer learning

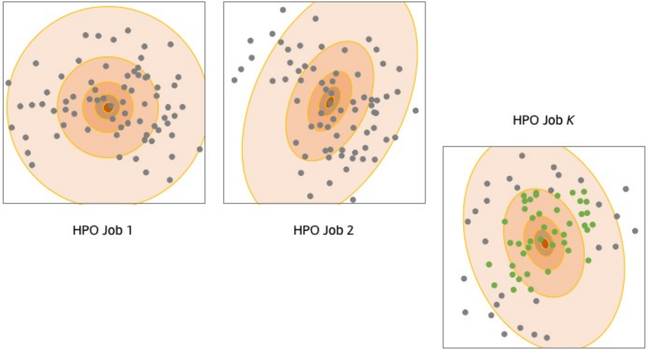
Problem statement:

- T functions $\{f_t(\mathbf{x})\}_{t=1}^{T}$ with observations $\mathcal{D}_t = \{(\mathbf{x}_t^n, y_t^n)\}_{n=1}^{N_t}$
- May/may not have meta-data (or contextual features) for $\{f_t(\mathbf{x})\}_{t=1}^{T}$
- **Goal:** Optimize some fixed $f_{t_0}(\mathbf{x})$ while exploiting $\{\mathcal{D}_t\}_{t=1}^{T}$
- (this is not multi-objective!)

Previous work:

- Multitask GP (Swersky et al. 2013, Poloczek et al. 2016)
- GP + filter evaluations by task similarity (Feurer et al. 2015)
- Various ensemble-based approaches
  - GPs (Feurer et al. 2018)
  - Feedforward NNs (Schilling et al. 2015)

What is wrong with the Gaussian process surrogate?



Scaling is $\mathcal{O}\left(N^3\right)$

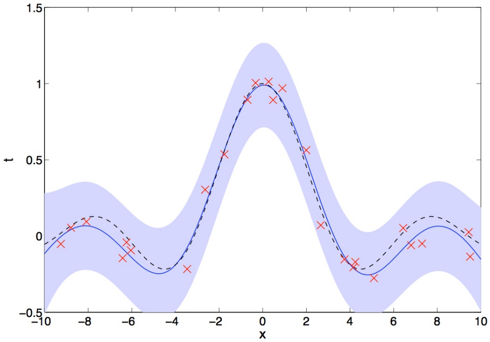
Adaptive Bayesian linear regression (ABLR) [Bis06]

The model:

$$P(\mathbf{y}|\mathbf{w}, \mathbf{z}, \beta) = \prod_n \mathcal{N}(\phi_{\mathbf{z}}(\mathbf{x}_n)\mathbf{w}, \beta^{-1}),$$
$$P(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I}_D).$$

The predictive distribution:

$$P(y^*|\mathbf{x}^*, \mathcal{D}) = \int P(y^*|\mathbf{x}^*, \mathbf{w})P(\mathbf{w}|\mathcal{D})d\mathbf{w}$$
$$= \mathcal{N}(\mu_t(\mathbf{x}^*), \sigma_t^2(\mathbf{x}^*))$$



Multi-task ABLR for transfer learning

1. Multi-task extension of the model:

$$P(\mathbf{y}_t|\mathbf{w}_t, \mathbf{z}, \beta_t) = \prod_{n_t} \mathcal{N}(\phi_{\mathbf{z}}(\mathbf{x}_{n_t})\mathbf{w}_t, \beta_t^{-1}), \qquad P(\mathbf{w}_t|\alpha_t) = \mathcal{N}(\mathbf{0}, \alpha_t^{-1}\mathbf{I}_D).$$

2. Shared features $\phi_{\mathbf{z}}(\mathbf{x})$:
   - Explicit features set (e.g., RBF)
   - Random kitchen sinks [RR+07]
   - Learned by feedforward neural net

3. Multi-task objective:

$$\rho\left(\mathbf{z}, \{\alpha_t, \beta_t\}_{t=1}^{T}\right) = -\sum_{t=1}^{T} \log P(\mathbf{y}_t|\mathbf{z}, \alpha_t, \beta_t)$$

Examples of $\phi_{\mathbf{z}}$

Feedforward neural networks:

$$\phi_{\mathbf{z}}(\mathbf{x}) = a_L\left(\mathbf{Z}_L a_{L-1}\left(\ldots \mathbf{Z}_2 a_1\left(\mathbf{Z}_1 \mathbf{x}\right)\ldots\right)\right).$$

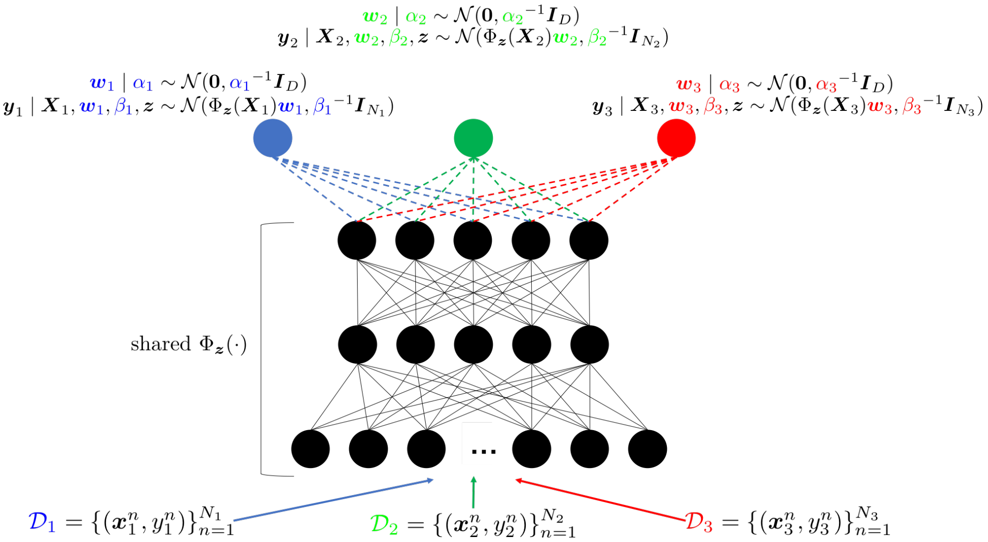
- $\mathbf{z}$ consists of all $\{\mathbf{Z}_l\}_{l=1}^{L}$

Random Fourier features:

$$\phi_{\mathbf{z}}(\mathbf{x}) = \sqrt{2/D}\cos\left\{\frac{1}{\sigma}\mathbf{U}\mathbf{x} + \mathbf{b}\right\}, \quad \text{with} \quad \mathbf{U} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \text{and} \quad \mathbf{b} \sim \mathcal{U}([0, 2\pi]).$$

- $\mathbf{z}$ only consists of $1/\sigma$

Pictorial summary of ABLR



Posterior inference

Hyperparameters:

- $\{\alpha_t, \beta_t\}_{t=1}^{T}$ for each task t
- $\mathbf{z}$ for the shared basis function

Empirical Bayesian approach:

- Marginalize out the Bayesian linear regression parameters $\{\mathbf{w}_t\}_{t=1}^{T}$
- *Jointly* learn the hyper-parameters of the model $\{\alpha_t, \beta_t\}_{t=1}^{T}$ and $\mathbf{z}$

Minimize

$$\rho\left(\mathbf{z}, \{\alpha_t, \beta_t\}_{t=1}^{T}\right) = -\sum_{t=1}^{T} \log\{\mathbb{P}(\mathbf{y}_t \mid \mathbf{X}_t, \alpha_t, \beta_t, \mathbf{z})\}$$

Posterior inference (cont'd)

We have closed-forms for posterior mean and variance:

$$\boldsymbol{\mu}_t(\mathbf{x}_t^*; \mathcal{D}_t, \alpha_t, \beta_t, \mathbf{z}) = \frac{\beta_t}{\alpha_t} \phi_{\mathbf{z}}(\mathbf{x}_t^*)^\top \mathbf{K}_t^{-1} \boldsymbol{\Phi}_t^\top \mathbf{y}_t$$
$$\sigma_t^2(\mathbf{x}_t^*; \mathcal{D}_t, \alpha_t, \beta_t, \mathbf{z}) = \frac{1}{\alpha_t} \phi_{\mathbf{z}}(\mathbf{x}_t^*)^\top \mathbf{K}_t^{-1} \phi_{\mathbf{z}}(\mathbf{x}_t^*) + \frac{1}{\beta_t}$$

and marginal likelihood:

$$\rho\left(\mathbf{z}, \{\alpha_t, \beta_t\}_{t=1}^T\right) = -\sum_{t=1}^T \left[\frac{N_t}{2} \log \beta_t - \frac{\beta}{2}\left(||\mathbf{y}_t||^2 - \frac{\beta_t}{\alpha_t}||\mathbf{c}_t||^2\right) - \sum_{i=1}^D \log([\mathbf{L}_t]_{ii})\right]$$

- Cholesky for $\mathbf{K}_t = \frac{\beta_t}{\alpha_t} \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \mathbf{I}_D = \mathbf{L}_t {\mathbf{L}_t}^\top$
- $\mathbf{c}_t = {\mathbf{L}_t}^{-1} \boldsymbol{\Phi}_t^\top \mathbf{y}_t$

Leveraging `MXNet`

In Bayesian optimization, derivatives needed for

- Posterior inference: $(\mathbf{z}, \{\alpha_t, \beta_t\}_{t=1}^{T}) \mapsto \rho(\mathbf{z}, \{\alpha_t, \beta_t\}_{t=1}^{T})$
- Acquisition functions $\mathcal{A}$, typically of the form (e.g., EI, PI, UCB,...):

$$\mathbf{x}^* \mapsto \mathcal{A}(\mu_t(\mathbf{x}^*; \mathcal{D}_t, \alpha_t, \beta_t, \mathbf{z}), \sigma_t^2(\mathbf{x}^*; \mathcal{D}_t, \alpha_t, \beta_t, \mathbf{z}))$$

Leverage `MXNet` (Seeger et al. 2017):

- Auto-differentiation
- Backward operator for Cholesky
- Can use any $\phi_{\mathbf{z}}$

Optimization of the marginal likelihood

Optimization properties:

- Number of tasks: $T \approx$ few tens
- Number of points per task: $N_t \gg 1$
- Not standard SGD regime
- We apply `L-BFGS` *jointly* over all parameters $\mathbf{z}$ and $\{\alpha_t, \beta_t\}_{t=1}^{T}$
- Warm-start parameters: Re-convergence in a very few steps

Surrogate models used in Bayesian optimization

Various types of models used:

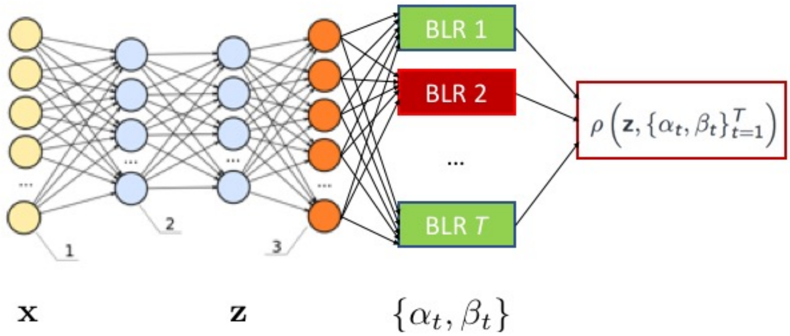
- Gaussian processes (Jones et al. 1998, Snoek et al. 2012,. . .)
- Sparse gaussian processes (McIntire et al. 2016)
- Variants (DKL/KISS-GP) of Gaussian processes (Pleiss et al. 2018)
- Random forests (Hutter et al. 2011)
- (Bayesian) NNs (Snoek et al. 2015, Springenberg et al. 2016)

ABLR

Contributions:

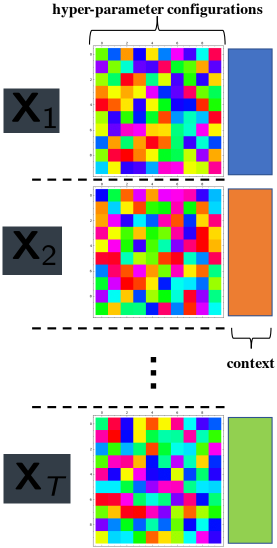
- Simplicity
- Scalability
- Transfer learning in absence of meta-data
- Extend DNGO (Snoek et al. 2015) with:
  - Joint inference
  - Transfer learning and handling of heterogenous tasks

Warm-start procedure for hyperparameter optimisation (HPO)



Leave-one-task out.

Pictorial view of different transfer learning approaches



1. Single marg. likelihood, stack across tasks

$$\begin{bmatrix} \mathbf{X}_1 & \text{context}_1 \\ \vdots & \vdots \\ \mathbf{X}_T & \text{context}_T \end{bmatrix} \in \mathbb{R}^{\sum_{t=1}^{T} N_t \times (P + |\text{context}|)}$$

2. One marg. likelihood per $\mathbf{X}_t$ (no context!)
3. One marg. likelihood per $[\mathbf{X}_t, \text{context}_t]$

Small-scale synthetic example: Transfer learning across quadratic functions

3-dimensional parameterized quadratic functions:

$$f_t(\mathbf{x}) = \frac{1}{2} a_t \|\mathbf{x}\|_2^2 + b_t \mathbf{1}^\top \mathbf{x} + c_t,$$

- One task $=$ one function f_t
- $(a_t, b_t, c_t) \in [0.1, 10]^3$, contextual information
- $T = 30$ tasks
- "Leave-one-task-out"

Experimental protocol

Comparisons with:

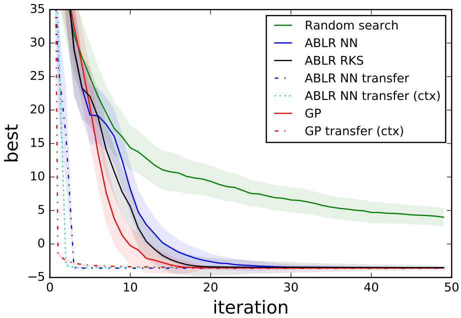
- Random search (Bergstra et al. 2012)
- Gaussian process (based on `GPyOpt` implementation)
- Gaussian process + "L_1 heuristic" (Feurer et al. 2015)
- DNGO[1] (Snoek et al. 2015)
- BOHAMIANN[1] (Springenberg et al. 2016)

Other considerations:

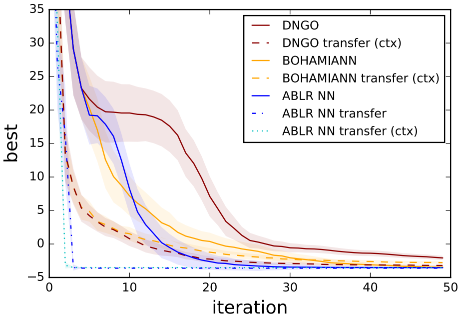
- Results aggregated over 30 replicates.
- Expected improvement used for all model-based approaches.
- Architecture of ABLR is $(50, 50, 50)$ (following Snoek et al. 2015).

[1] Implementation from `https://github.com/automl/RoBO`

Transfer learning across quadratic functions

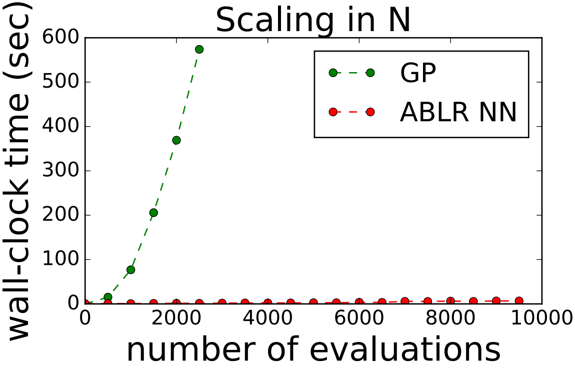


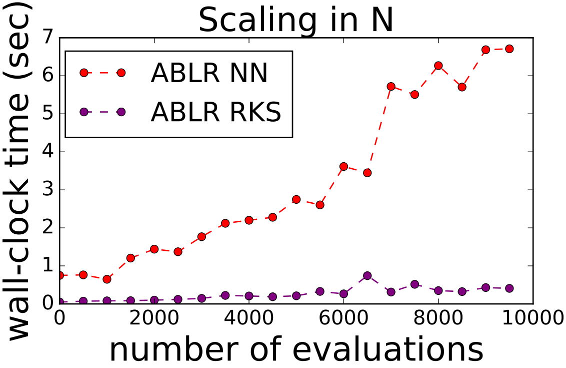
Transfer learning with baselines [KO11].



Transfer learning with neural nets [SRS+15, SKFH16].

Scalability: GP vs ABLR

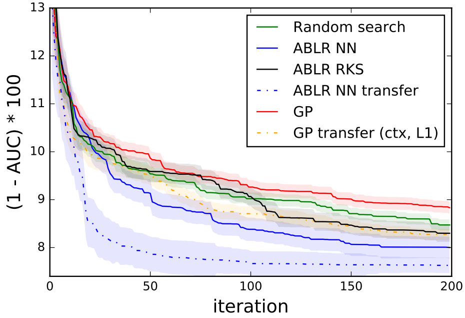




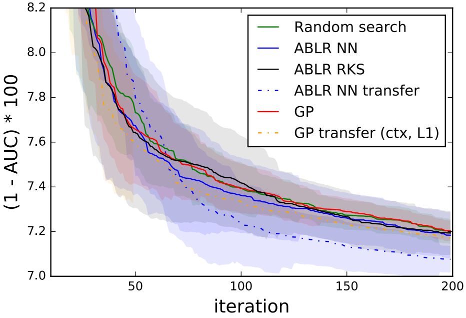
Transfer learning - OpenML data (Vanschoren et al. 2014)

- One task = one dataset
- Collect $\{(\mathbf{X}_t, \mathbf{y}_t)\}_{t=1}^{T}$ from OpenML (Vanschoren et al. 2014)
- `SVM`: 4 HPs, `XGBoost`: 10 HPs
- Take T=30 datasets (`flow_ids`)
  - $\sum_t N_t$ up to 7.5×10^5 evaluations

Transfer learning across OpenML data sets



Transfer learning in SVM.



Transfer learning in XGBoost.

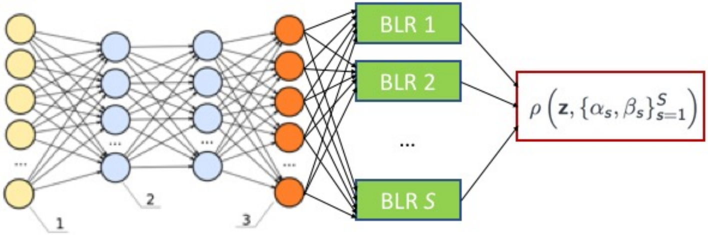
Transfer learning vs. exploiting side signals

	transfer learning	**side signals**
# active task(s)	1	T
# optimized task	1	1
N_t	non-active N_t fixed	growing $N_t = N$
marg. likelihood	a tuning experiment	a signal

Typical use cases

- **Transfer learning**: Reuse data of previous tuning experiments
- **Side signals**: The training of ML models generate multiple signals

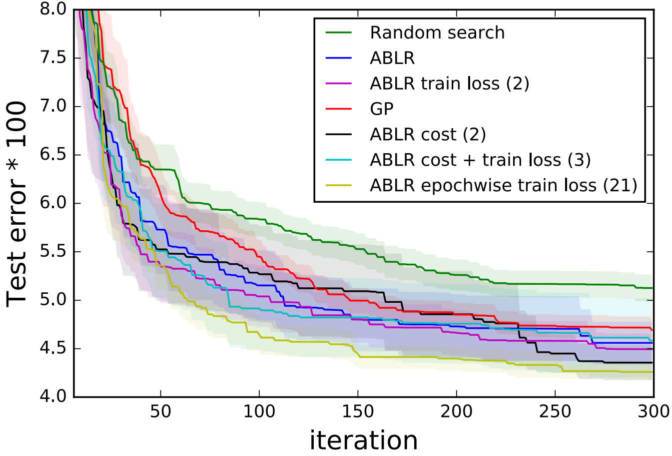
Leveraging multiple signals



Goal: Tune feedforward NNs for binary classification

- Main signal: Validation accuracy
- Side signals: Training accuracy and CPU time ("come for free")
- **Idea**: Side signals can help learn $\phi_{\mathbf{z}}$

Leveraging multiple signals



Transfer learning across LIBSVM data sets.

Conclusion

Bayesian optimisation is a model-based approach that automates machine learning:

- Algorithm tuning
- Model tuning

ABLR [PJSA17]:

- Scalable
- Fully leverages `MXNet`
- Transfers knowledge across tasks and signals

Thank you!

References I

James Bergstra and Yoshua Bengio.
Random search for hyper-parameter optimization.
Journal of Machine Learning Research, 13:281–305, 2012.

C. M. Bishop.
Pattern Recognition and Machine Learning.
Springer New York, 2006.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter.
Fast Bayesian optimization of machine learning hyperparameters on large datasets.
Technical report, preprint arXiv:1605.07079, 2016.

Andreas Krause and Cheng S Ong.
Contextual gaussian process bandit optimization.
In *Advances in Neural Information Processing Systems (NIPS)*, pages 2447–2455, 2011.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.

References II

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas.
The application of Bayesian methods for seeking the extremum.
Towards Global Optimization, 2(117-129):2, 1978.

V. Perrone, R. Jenatton, M. Seeger, and C. Archambeau.
Multiple Adaptive Bayesian Linear Regression for Scalable Bayesian Optimization with Warm Start.
ArXiv e-prints, December 2017.

Ali Rahimi, Benjamin Recht, et al.
Random features for large-scale kernel machines.
In *Advances in Neural Information Processing Systems (NIPS) 20*, pages 1177–1184, 2007.

Matthias Seeger, Asmus Hetzel, Zhenwen Dai, and Neil D Lawrence.
Auto-differentiating linear algebra.
Technical report, preprint arXiv:1710.08717, 2017.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter.
Bayesian optimization with robust Bayesian neural networks.
In *Advances in Neural Information Processing Systems (NIPS)*, pages 4134–4142, 2016.

References III

Jasper Snoek, Hugo Larochelle, and Ryan P Adams.
Practical Bayesian optimization of machine learning algorithms.
In *Advances in Neural Information Processing Systems (NIPS)*, pages 2960–2968, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams.
Scalable Bayesian optimization using deep neural networks.
In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2171–2180, 2015.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas.
Taking the human out of the loop: A review of Bayesian optimization.
Proceedings of the IEEE, 104(1):148–175, 2016.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo.
OpenML: networked science in machine learning.
ACM SIGKDD Explorations Newsletter, 15(2):49–60, 2014.