Learning Representations for Hyperparameter Transfer Learning

Cédric Archambeau
cedrica@amazon.com

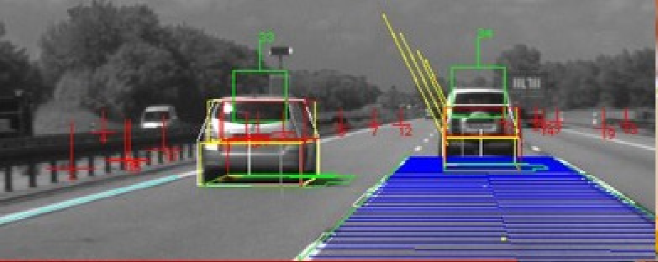
DALI 2018
Lanzarote, Spain

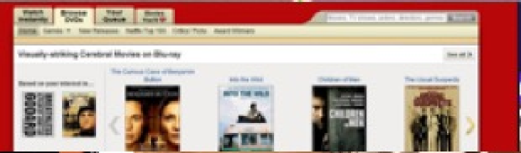
My co-authors

Rodolphe Jenatton

Valerio Perrone

Matthias Seeger




Visually-striking Cerebral Movies on Blu-ray

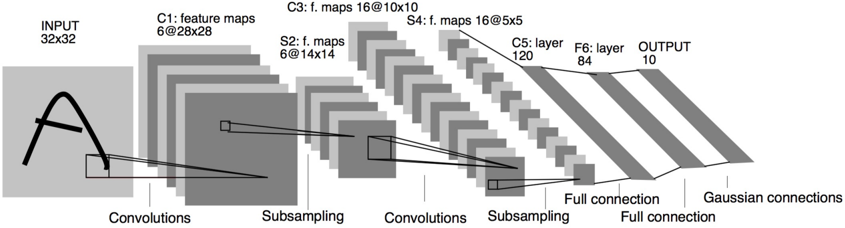

Search


WALK OUT


amazon go
BARS & NUTS


GOLD
SERVICE
SALES
SERVICE

Tuning deep neural nets for optimal performance

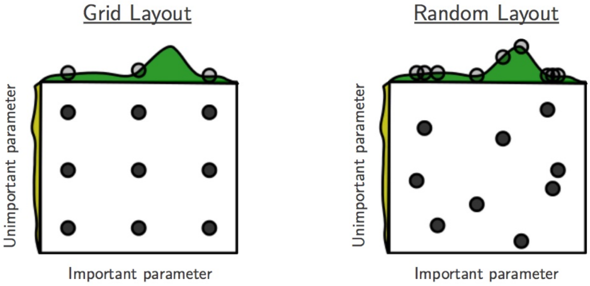


LeNet5 [LBBH98]

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



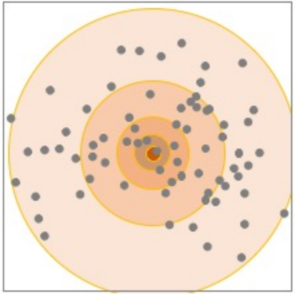
(Figure by Bergstra and Bengio, 2012)

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

Can we do better?

amazon

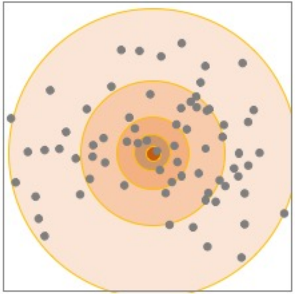
Hyperparameter transfer learning



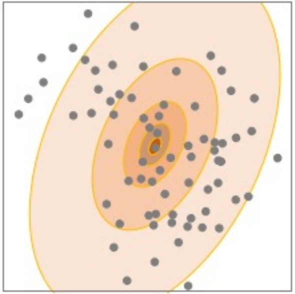
HPO Job 1

amazon

Hyperparameter transfer learning

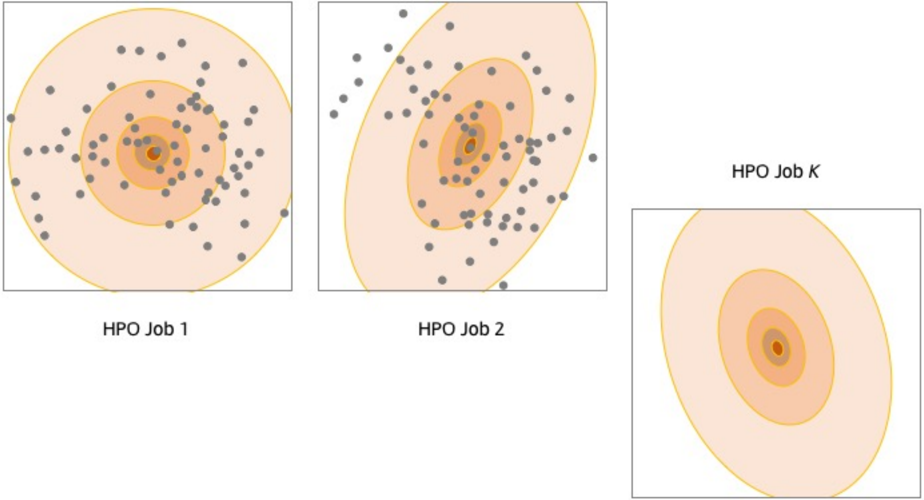


HPO Job 1

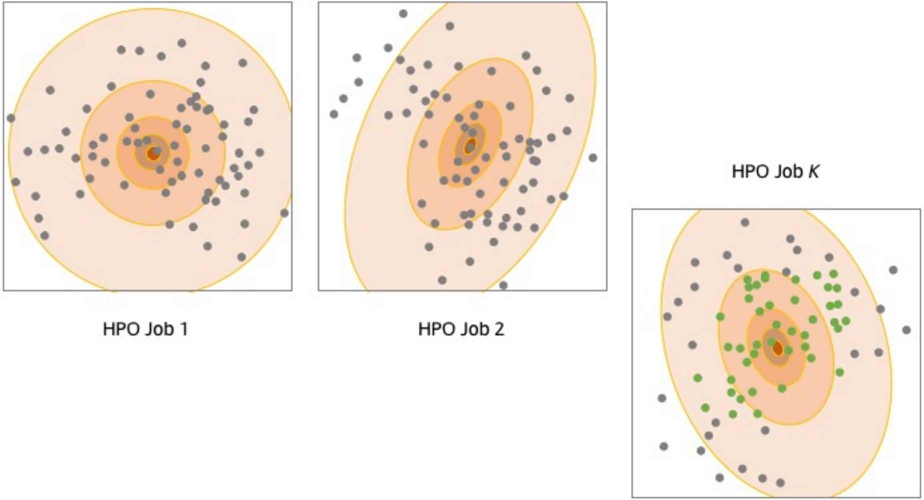


HPO Job 2

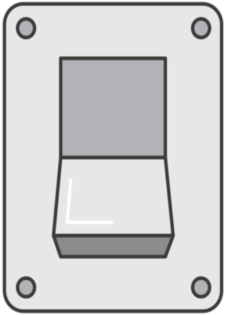
Hyperparameter transfer learning



Hyperparameter transfer learning



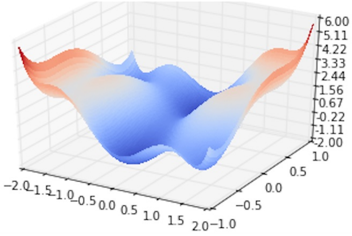
Democratising machine learning



- Abstract away training algorithms
- Abstract away representation (feature engineering)

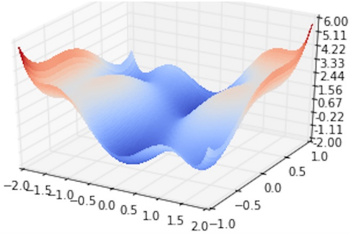
- Abstract away computing infrastructure
- Abstract away memory constraints
- Abstract away network architecture

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



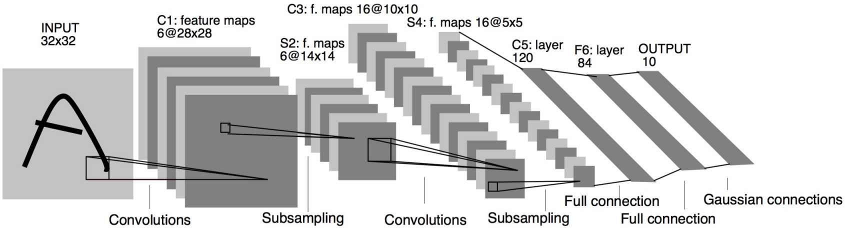
- The function f to optimise can be non-convex.
- The number of hyperparameters p is moderate (typically < 20).

Our goal is to solve the following optimisation problem:

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

- 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]

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

amazon

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 $\mathcal{C} = \{\}$

amazon

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

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

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`

amazon

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

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

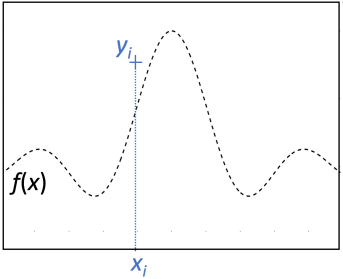
Canonical algorithm:

- 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 [JSW98]

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

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

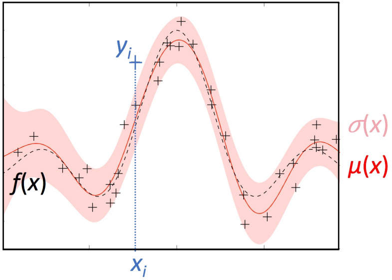


Bayesian (black-box) optimisation with Gaussian processes [JSW98]

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:

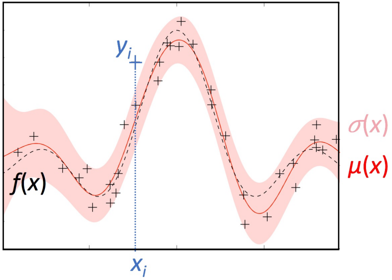


Bayesian (black-box) optimisation with Gaussian processes [JSW98]

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

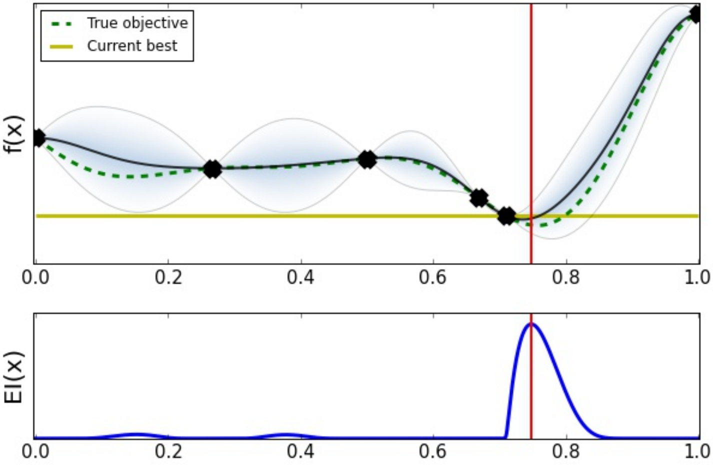
$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

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**!

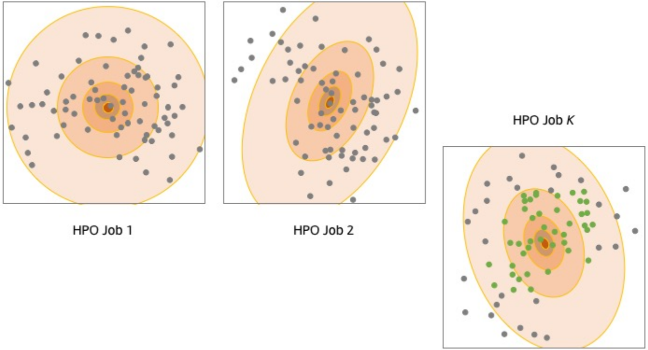
Bayesian optimisation in practice



(Image credit: Javier González)

amazon

What is wrong with the Gaussian process surrogate?



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

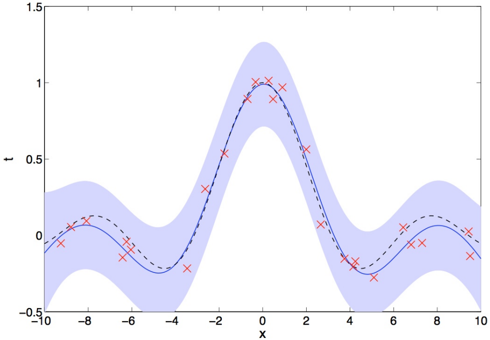
Adaptive Bayesian linear regression (ABLR) [Bis06]

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

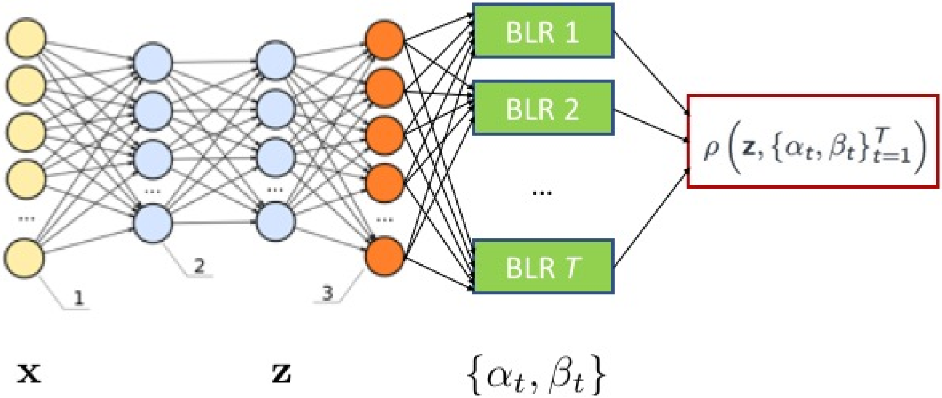
1. Multi-task extension of the model:

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

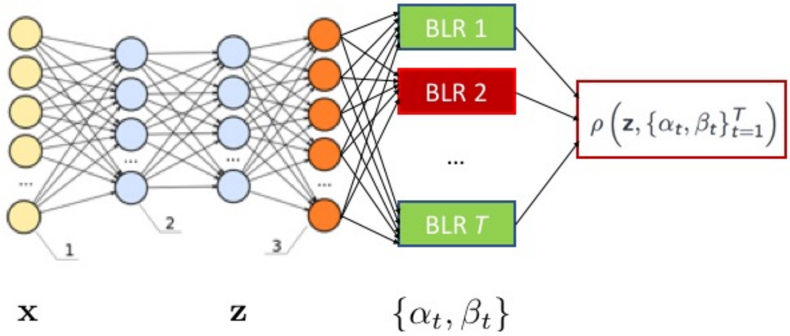
2. Shared features $\phi_{\mathbf{z}}(\mathbf{x})$:
   - Explicit features set (e.g., RBF)
   - Random kitchen sinks [RR+07]
   - Learned by feedforward neural net

3. Multi-task objective:

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

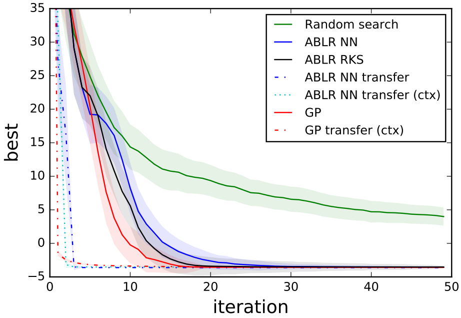

BLR 1
BLR 2
...
BLR T
$\rho\left(\mathbf{z}, \{\alpha_t, \beta_t\}_{t=1}^{T}\right)$
1
2
3
$\mathbf{x}$
$\mathbf{z}$
$\{\alpha_t, \beta_t\}$

Warm-start procedure for hyperparameter optimisation (HPO)

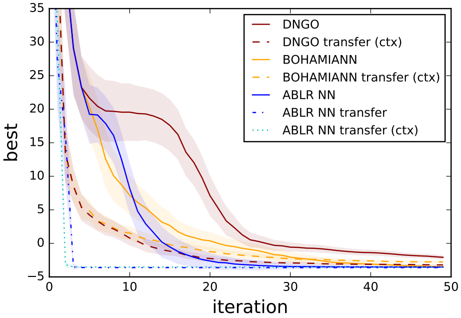


Leave-one-task out.

A representation to optimise parametrised quadratic functions



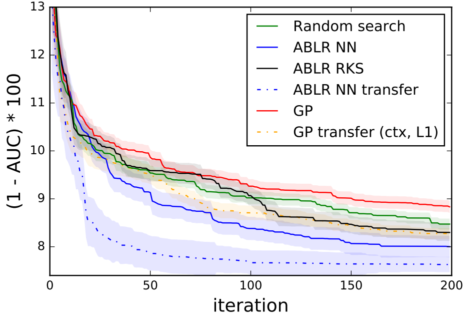
Transfer learning with baselines [KO11].



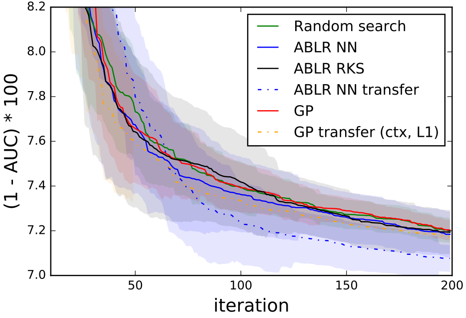
Transfer learning with neural nets [SRS+15, SKFH16].

$$f_t(\boldsymbol{x}) = \frac{a_{2,t}}{2}\|\boldsymbol{x}\|^2 + a_{1,t}\mathbf{1}^\top \boldsymbol{x} + a_{0,t}$$

A representation to warm-start HPO across OpenML data sets

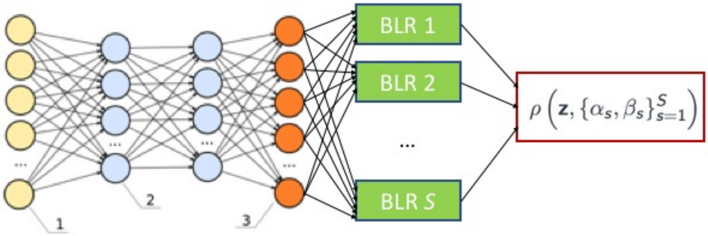


Transfer learning in SVM.



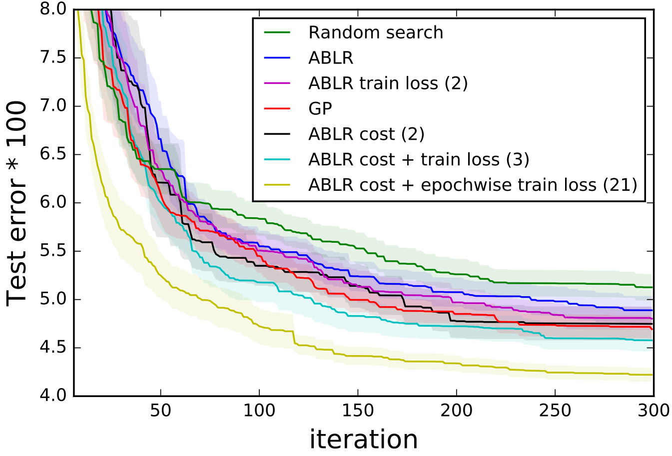
Transfer learning in XGBoost.

Learning better representations by jointly modelling multiple signals



- Tasks are now auxiliary signals (related to the target function to optimise).
- The target is still the validation loss $f(\mathbf{x})$.
- Examples are training cost or training loss.

A representation for transfer learning across OpenML data sets

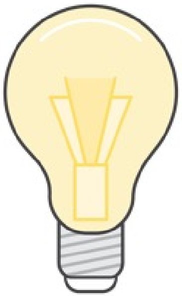


Transfer learning accross LIBSVM data sets.

Conclusion

Bayesian optimisation automates machine learning:

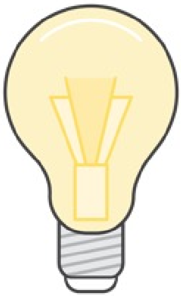
- Algorithm tuning
- Model tuning
- Pipeline tuning



Conclusion

Bayesian optimisation automates machine learning:

- Algorithm tuning
- Model tuning
- Pipeline tuning



Bayesian optimisation is a model-based approach that can leverage auxiliary information:

- It can exploit dependency structures [JAGS17]
- It can be extended to warm-start HPO jobs [PJSA17]
- It is a key building block of **meta-learning**

Conclusion

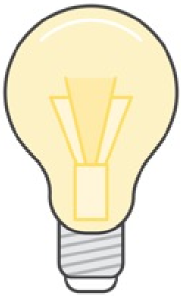
Bayesian optimisation automates machine learning:

- Algorithm tuning
- Model tuning
- Pipeline tuning

Bayesian optimisation is a model-based approach that can leverage auxiliary information:

- It can exploit dependency structures [JAGS17]
- It can be extended to warm-start HPO jobs [PJSA17]
- It is a key building block of **meta-learning**

Effective representations for HPO transfer learning can be learned.



cedrica@amazon.com

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.

Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger.
Bayesian optimization with tree-structured dependencies.
In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1655–1664, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Donald R Jones, Matthias Schonlau, and William J Welch.
Efficient global optimization of expensive black-box functions.
Journal of Global Optimization, 13(4):455–492, 1998.

References II

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.

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.

References III

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.

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.

References IV

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.