Learning Representations for Hyperparameter Transfer Learning

Cédric Archambeau
cedrica@amazon.com

DALI 2018
Lanzarote, Spain
Tuning deep neural nets for optimal performance

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

- Architecture: $\#$ hidden layers, activation functions, $\ldots$
- Model complexity: regularization, dropout, $\ldots$
- Optimisation parameters: learning rates, momentum, batch size, $\ldots$
Two straightforward approaches

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

(Figure by Bergstra and Bengio, 2012)
Can we do better?
Hyperparameter transfer learning
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:

$$\mathbf{x}_* = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}).$$

- Evaluating $f(\mathbf{x})$ is expensive.
- No analytical form or gradient.
- Evaluations may be noisy.
Example: tuning deep neural nets \[\text{SLA12, SRS}^{+15, \text{KFB}}^{+16}\]

- $f(x)$ is the validation loss of the neural net as a function of its hyperparameters $x$.
- Evaluating $f(x)$ is very costly $\approx$ up to weeks!
Bayesian (black-box) optimisation [MTZ78, SSW\textsuperscript{+}16]

\[ x_\star = \arg\min_{x \in \mathcal{X}} f(x) \]
Bayesian (black-box) optimisation \cite{MTZ78, SSW+16}

\[
x_\star = \arg\min_{x \in \mathcal{X}} f(x)
\]

Canonical algorithm:

- **Surrogate model** $M$ of $f$ \#cheaper to evaluate
- Set of evaluated candidates $C = \{\}$
Bayesian (black-box) optimisation \cite{MTZ78,SSW+16}

\[
x_\ast = \underset{x \in \mathcal{X}}{\text{argmin}} \ f(x)
\]

Canonical algorithm:

- Surrogate model $\mathcal{M}$ of $f$ \#cheaper to evaluate
- Set of evaluated candidates $\mathcal{C} = \{\}$
- While some BUDGET available:
  - Select candidate $x_{\text{new}} \in \mathcal{X}$ using $\mathcal{M}$ and $\mathcal{C}$ \#exploration/exploitation
Bayesian (black-box) optimisation [MTZ78, SSW⁺16]

\[ x_\star = \arg\min_{x \in \mathcal{X}} f(x) \]

**Canonical algorithm:**

- **Surrogate model** \( \mathcal{M} \) of \( f \) #cheaper to evaluate
- Set of evaluated candidates \( \mathcal{C} = {} \)
- While some BUDGET available:
  - Select candidate \( x_{\text{new}} \in \mathcal{X} \) using \( \mathcal{M} \) and \( \mathcal{C} \) #exploration/exploitation
  - Collect evaluation \( y_{\text{new}} \) of \( f \) at \( x_{\text{new}} \) #time-consuming
Bayesian (black-box) optimisation [MTZ78, SSW\textsuperscript{+}16]

\[
x_* = \arg\min_{x \in \mathcal{X}} f(x)
\]

**Canonical algorithm:**

- **Surrogate model** $\mathcal{M}$ of $f$ \#cheaper to evaluate
- **Set of evaluated candidates** $\mathcal{C} = \{\}$
- **While some BUDGET available:**
  - Select candidate $x_{\text{new}} \in \mathcal{X}$ using $\mathcal{M}$ and $\mathcal{C}$ \#exploration/exploitation
  - Collect evaluation $y_{\text{new}}$ of $f$ at $x_{\text{new}}$ \#time-consuming
  - Update $\mathcal{C} = \mathcal{C} \cup \{(x_{\text{new}}, y_{\text{new}})\}$
  - Update $\mathcal{M}$ with $\mathcal{C}$ \#Update surrogate model
  - Update BUDGET
Bayesian (black-box) optimisation with Gaussian processes [JSW98]

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

$$y_i | f(x_i) \sim \text{Gaussian} \left( f(x_i), \sigma^2 \right), \quad f(x) \sim \mathcal{GP}(0, \mathcal{K}).$$
Bayesian (black-box) optimisation with Gaussian processes [JSW98]

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

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

2. Given the observations $\mathbf{y} = (y_1, \ldots, y_n)$, 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:

$$y_i \mid f(x_i) \sim \text{Gaussian } (f(x_i), \varsigma^2), \quad f(x) \sim \mathcal{GP}(0, K).$$

2. Given the observations $y = (y_1, \ldots, y_n)$, 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)
What is wrong with the Gaussian process surrogate?

Scaling is $\mathcal{O}(N^3)$. 
Adaptive Bayesian linear regression (ABLR) [Bis06]

The model:

\[ P(y|w, z, \beta) = \prod_n \mathcal{N}(\phi_z(x_n)w, \beta^{-1}), \]
\[ P(w|\alpha) = \mathcal{N}(0, \alpha^{-1}I_D). \]

The predictive distribution:

\[
P(y^*|x^*, \mathcal{D}) = \int P(y^*|x^*, w)P(w|\mathcal{D})dw
= \mathcal{N}(\mu_t(x^*), \sigma_t^2(x^*))
\]
Multi-task ABLR for transfer learning

1. Multi-task extension of the model:

\[
P(y_t|w_t, z, \beta_t) = \prod_{n_t} \mathcal{N}(\phi_z(x_{n_t})w_t, \beta_t^{-1}), \quad P(w_t|\alpha_t) = \mathcal{N}(0, \alpha_t^{-1}I_D).
\]

2. Shared features \(\phi_z(x)\):
   - Explicit features set (e.g., RBF)
   - Random kitchen sinks \([RR^+07]\)
   - Learned by feedforward neural net

3. Multi-task objective:

\[
\rho\left(z, \{\alpha_t, \beta_t\}_{t=1}^T\right) = -\sum_{t=1}^T \log P(y_t|z, \alpha_t, \beta_t)
\]
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].

\[ f_t(x) = \frac{a_{2,t}}{2} \|x\|^2 + a_{1,t}1^\top x + a_{0,t} \]
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 $f(x)$.
- 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
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
James Bergstra and Yoshua Bengio.  
Random search for hyper-parameter optimization.  

C. M. Bishop.  
*Pattern Recognition and Machine Learning*.  

Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger.  
Bayesian optimization with tree-structured dependencies.  

Donald R Jones, Matthias Schonlau, and William J Welch.  
Efficient global optimization of expensive black-box functions.  


References III

Ali Rahimi, Benjamin Recht, et al.
Random features for large-scale kernel machines.

Matthias Seeger, Asmus Hetzel, Zhenwen Dai, and Neil D Lawrence.
Auto-differentiating linear algebra.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter.
Bayesian optimization with robust Bayesian neural networks.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams.
Practical Bayesian optimization of machine learning algorithms.
