Towards true end-to-end learning & optimization

Frank Hutter
Department of Computer Science
University of Freiburg, Germany
fh@cs.uni-freiburg.de
Motivation: Successes of Deep Learning

- Speech recognition
- Computer vision in self-driving cars
- Reasoning in games
Deep learning learns features from raw data
- Multiple layers of abstractions
- **End-to-end learning**: joint optimization of a single loss function

Visualizations of network activations taken from Zeiler [2014]
One Problem of Deep Learning

Performance is very **sensitive** to many hyperparameters

- Architectural hyperparameters

- Optimization algorithm, **learning rates**, momentum, batch normalization, batch sizes, dropout rates, weight decay, ...

- Data augmentation & preprocessing
Towards True End-to-end Learning

Current deep learning practice

Expert chooses architecture & hyperparameters → Deep learning “end-to-end”

AutoML: true end-to-end learning

Meta-level learning & optimization → Learning box

Meta-level learning & optimization → Learning box
Learning box can be any ML pipeline

• Traditional machine learning pipeline
  – Clean & preprocess the data
  – Select / engineer better features
  – Select a model family
  – Set the hyperparameters
  – Construct ensembles of models
  – …

AutoML: true end-to-end learning
Bayesian optimization

- AutoML: end-to-end learning systems
- Speeding up AutoML: beyond vanilla Bayesian optimization
- End-to-end optimization of combinatorial problem solvers
Our Workhorse: Bayesian Optimization

- Prominent method for **expensive blackbox optimization** [Mockus et al., '78]
- Recent convergence results
  [Srinivas et al, '10; Bull '11; de Freitas, Smola, Zoghi, '12, Kawaguchi et al, '15]
Bayesian Optimization Extensions for AutoML

- Structured, *discrete/continuous* input space
  - Hutter, Hoos & Leyton-Brown, LION 2011; Swersky et al, BayesOpt 2013

- **High-dimensional** input spaces
  - Wang, Zoghi, Hutter, Matheson, de Freitas, IJCAI 2013
  - https://github.com/automl/SMAC3

- Robust optimization for *noisy functions*
  - Bartz-Beielstein, GECCO 2006; Hutter, Hoos & Leyton-Brown, LION 2010

- Exploiting **parallel** resources
  - Snoek et al, NIPS 2011; Hutter et al, LION 2012; Desautels et al, JMLR 2014, ...

- **Understanding hyperparameter importance**
  - Hutter, Hoos & Leyton-Brown, ICML 2014; van Rijn & Hutter, AutoML 2017

→ **SMAC** [Hutter et al, 2009-2017]
Overview

- Bayesian optimization

AutoML: End-to-end learning systems
  - Auto-WEKA
  - Auto-sklearn
  - Auto-Net

- Speeding up AutoML: beyond vanilla Bayesian optimization

- End-to-end optimization of combinatorial problem solvers
Decades of Related Work

- **European projects**

- **Hyperparameter optimization**
  - Genetic algorithms [e.g., Goldberg & Holland, MLJ 1988]
  - Bayesian optimization [e.g., Bergstra et al, NIPS 2011; Snoek et al, NIPS 2012]

- **Model selection**
  - Racing / bandit-based approaches [e.g., Maron & Moore, NIPS 1994]
  - Meta-learning [e.g., Brazdil, Vilalta & Soares, 2009]

- **Some automation methods have been used for decades**
  - WEKA: grid search
  - Scikit-learn: grid search and random search
  - RapidMiner: grid search for entire operator chain [Fischer et al, 2002]
  - genetic programming for feature extraction [Morik & Mierswa, MLJ 2005]
Benchmark: AutoML Challenge

- **Large-scale challenge run by ChaLearn & CodaLab**
  - December 2014 – April 2016
  - 5 phases with 5 new datasets each
  - 2 tracks: code submissions / Kaggle-like human track

- **25 datasets from wide range of application areas**, e.g.
  - Medical diagnosis from laboratory analyses
  - Prediction or drug toxicity or efficacy
  - Text classification

- **Datasets were already featurized**
  - Inputs: unstructured features $X$, targets $y$
AutoML System 1: Auto-WEKA

- WEKA [Witten et al, 1999-current]
  - 27 base classifiers (with up to 10 parameters each)
  - 10 meta-methods
  - 2 ensemble methods
  - In total: 786 hyperparameters

- Optimize CV performance by SMAC
  - 5x speedup of 10-fold CV
  - Better than each base classifier; better than TPE & random search
**Scikit-learn** [Pedregosa et al, 2011]

- 15 classifiers with a total of 59 hyperparameters
- 13 feature preprocessors
- 4 data preprocessors
- In total: 110 hyperparameters

<table>
<thead>
<tr>
<th>classifier</th>
<th>#λ</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost (AB)</td>
<td>4</td>
</tr>
<tr>
<td>Bernoulli naïve Bayes</td>
<td>2</td>
</tr>
<tr>
<td>decision tree (DT)</td>
<td>4</td>
</tr>
<tr>
<td>extreml. rand. trees</td>
<td>5</td>
</tr>
<tr>
<td>Gaussian naïve Bayes</td>
<td>-</td>
</tr>
<tr>
<td>gradient boosting (GB)</td>
<td>6</td>
</tr>
<tr>
<td>kNN</td>
<td>3</td>
</tr>
<tr>
<td>LDA</td>
<td>4</td>
</tr>
<tr>
<td>linear SVM</td>
<td>4</td>
</tr>
<tr>
<td>kernel SVM</td>
<td>7</td>
</tr>
<tr>
<td>multinomial naïve Bayes</td>
<td>2</td>
</tr>
<tr>
<td>passive aggressive</td>
<td>3</td>
</tr>
<tr>
<td>QDA</td>
<td>2</td>
</tr>
<tr>
<td>random forest (RF)</td>
<td>5</td>
</tr>
<tr>
<td>Linear Class. (SGD)</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>preprocessor</th>
<th>#λ</th>
</tr>
</thead>
<tbody>
<tr>
<td>extreml. rand. trees prepr.</td>
<td>5</td>
</tr>
<tr>
<td>fast ICA</td>
<td>4</td>
</tr>
<tr>
<td>feature agglomeration</td>
<td>4</td>
</tr>
<tr>
<td>kernel PCA</td>
<td>5</td>
</tr>
<tr>
<td>rand. kitchen sinks</td>
<td>2</td>
</tr>
<tr>
<td>linear SVM prepr.</td>
<td>3</td>
</tr>
<tr>
<td>no preprocessing</td>
<td>-</td>
</tr>
<tr>
<td>nystroem sampler</td>
<td>5</td>
</tr>
<tr>
<td>PCA</td>
<td>2</td>
</tr>
<tr>
<td>polynomial</td>
<td>3</td>
</tr>
<tr>
<td>random trees embed.</td>
<td>4</td>
</tr>
<tr>
<td>select percentile</td>
<td>2</td>
</tr>
<tr>
<td>select rates</td>
<td>3</td>
</tr>
<tr>
<td>one-hot encoding</td>
<td>2</td>
</tr>
<tr>
<td>imputation</td>
<td>1</td>
</tr>
<tr>
<td>balancing</td>
<td>1</td>
</tr>
<tr>
<td>rescaling</td>
<td>1</td>
</tr>
</tbody>
</table>
AutoML System 2: Auto-sklearn

[Feurer, Klein, Eggensperger, Springenberg, Blum, Hutter; NIPS 2015]

- Optimize CV performance by SMAC
  - **Meta-learning** to warmstart Bayesian optimization
    - Reasoning over different datasets
    - Dramatically speeds up the search
  - Automated **posthoc ensemble construction** to combine the models Bayesian optimization evaluated
    - Efficiently re-uses its data; improves robustness
• Ranking plots over **140 datasets from OpenML**
  [Vanschoren et al, SIGKDD Explorations 2013]
  
  – Ranks have to add up to 3 (=1+2)
  – Meta-learning helps right from the start
Ensembling Helps!

- Ensembles help more when more good models available
• With meta-learning, ensembling starts helping earlier
  – More good models available earlier
Winning approach in the AutoML challenge

- Auto-track: overall winner, 1st place in 3 phases, 2nd place in 1
  - Human track: always in top-3 vs. 150 teams of human experts
  - Final two rounds: won both tracks

https://github.com/automl/auto-sklearn

Trivial to use:

```python
import autosklearn.classification as cls
automl=cls.AutoSklearnClassifier(include_estimators = ['lda', 'decision_tree'])
automl.fit(X_train, y_train)
y_hat = automl.predict(X_test)
```
Application: Robotic Object Handling

- Collaboration with Andreas Eitel & Wolfram Burgard
- Binary classification task for object placement: \textit{will the object fall over?}

- Dataset
  - Based on BigBIRD and YCB Object and Model Set
  - 30000 data points
  - 50 features -- manually defined [BSc thesis AIS, Hauff 2015]

- Performance
  - Caffe deep learning framework: 2\% error rate
  - Auto-sklearn: 0.6\% error rate (within 30 minutes)
AutoML System 3: Auto-Net

- CV performance optimized by SMAC

- Joint optimization of:
  - Neural network architecture
  - Neural network hyperparameters

\[ k = \sum_{i=1}^{k} \text{i} \]
Application 1: Object Recognition

[Domhan, Springenberg, Hutter, IJCAI 2015]

- Parameterized the Caffe framework [Jia, 2013]
  - Convolutional neural network with up to 6 layers
  - 81 hyperparameters
    - 9 network hyperparameters
    - 12 layer-wise hyperparameters for each of the 6 layers

- Results for CIFAR-10
  - New best result for CIFAR-10 without data augmentation
  - SMAC outperformed TPE (only other applicable hyperparameter optimizer)
Convolutional neural network for motor-execution data
- Tap fingers on left hand / right hand / do nothing / clench toes
- EEG data from 128 channels

Results for Auto-Net
- Automatically selected useful subset of channels
- Outperformed manual solution, by 10% relative error
- Per-patient optimization: cross-validation error rates reduced by factor of 2
• Unstructured data → fully-connected network
  – Up to 5 layers (with 3 layer hyperparameters each)
  – 14 network hyperparameters, in total **29 hyperparameters**
  – Optimized for 18h on 5GPUs
  – Timeout of 30 minutes per network (≈500 networks evaluated)

• Auto-Net won several datasets against human experts
  – E.g., Alexis data set:
    • 54491 data points, 5000 features, 18 classes
    • Test set AUC 90%
    • All other (manual) approaches < 80%
  – First automated deep learning system to win a ML competition data set against human experts
Since then: many works on architecture search

- Most prominent: **Neural Architecture Search by Reinforcement Learning** [Zoph & Le, ICLR 2017]
  - Large computational demand
    - 800 GPUs for 2 weeks
    - 12,800 architectures evaluated
  - In the end: hyperparameter optimization by full grid search
  - Performance: close to state-of-the-art on CIFAR-10

- Many good ideas, but there is still much to do
  - No focus on **joint** architecture & hyperparameter search
  - No **efficient** method that achieves **state-of-the-art** performance
Overview

- Bayesian optimization
- AutoML: end-to-end learning systems

Speeding up AutoML: beyond vanilla Bayesian optimization

- End-to-end optimization of combinatorial problem solvers
• **Sum of little black boxes**
  – Each little black box is fast but very noisy  
    [Thornton, Hutter, Hoos & Leyton-Brown, KDD 2013]

  $$\sum_{i=1}^{k} \blackbox_{i}$$

• **Transfer learning from related blackboxes**
  [Bardenet et al, ICML 2013; Swersky et al, NIPS 2013; Feurer, Springenberg, Hutter, AAAI 2015]

• **Graybox optimization**
  terminate poor partial evaluations early  
  [Domhan, Springenberg, Hutter; IJCAI 2015]

• **Using cheaper, approximate variants of the blackbox**
  [Klein, Bartels, Falkner, Hennig, Hutter, AISTATS 2017]
Iterative Training with SGD: Typical Learning Curves
Probabilistic Learning Curve Model

Parametric model, e.g. \( \frac{a}{1 + \left( \frac{x}{cb} \right)^c} \)

MCMC: to quantify model uncertainty

K = 11 parametric models
Convex Combination of these models:

\[
y_t = \sum_{k=1}^{K} w_k f_k(t | \theta_k) + \varepsilon, \; \varepsilon \sim N(0, \sigma^2)
\]
Predictive Termination

\[ P(y_m > y_{best} \mid y_{1:n}) \geq 5\% \]

\[ \text{continue training...} \]
Predictive Termination

\[ P( y_m > y_{best} | y_{1:n} ) < 5\% \]

Terminate!
Qualitative Analysis

All learning curves

Learning curves with early termination
Quantitative Results

- 2-fold speedup of DNN structure & hyperparameter optimization
  - For several network architectures, including state-of-the-art
    - For several optimizers (SMAC, TPE, random search)
    - New state-of-the-art for CIFAR-10 without data augmentation

https://github.com/automl/pylearningcurvepredictor
• **Sum of little black boxes**
  - Each little black box is fast but very noisy
    [Thornton, Hutter, Hoos & Leyton-Brown, KDD 2013]

• **Transfer learning from related blackboxes**
  [Bardenet et al, ICML 2013; Swersky et al, NIPS 2013; Feurer, Springenberg, Hutter, AAAI 2015]

**Graybox optimization**
terminate poor partial evaluations early
[Domhan, Springenberg, Hutter; IJCAI 2015]

• **Using cheaper, approximate variants of the blackbox**
  [Klein, Bartels, Falkner, Hennig, Hutter, AISTATS 2017]
• **Problem**: training is very slow for large datasets

• **Solution approach**: scaling up from subsets of the data

• **Example: SVM**
  – Computational cost grows quadratically in dataset size $s$
  – Error shrinks smoothly with $s$
Our Approach

- **Automatically choose dataset size for each evaluation**
  - Include extra dimension in probabilistic model to capture dependence on dataset size $s$: $f(\lambda, s)$
  - Construct a second model for computational cost: $c(\lambda, s)$
  - Trade off information gain about global optimum vs. cost

- **Entropy Search** [Hennig & Schuler, JMLR 2012]
  - Based on a probability distribution of where the maximum lies:

  $$p_{max}(\lambda|D) = p\left(\lambda = \arg\max_{\lambda' \in \Lambda} f(\lambda', s_{max}) | D\right)$$

- **Strategy**: pick configuration & data size pair $(\lambda, s)$ to maximally decrease entropy of $p_{max}$ per time spent:

  $$a(\lambda, s) = \frac{H[p_{max}(\lambda|D_n)] - \mathbb{E}_p(f(\lambda, s)|D_n, \lambda, s)\left[H[p_{max}(\lambda|D_n \cup \{(\lambda, s, f(\lambda, s))\})]\right]}{c(\lambda, s)}$$
Results

10x-50x fold speedup for optimizing SVM hyperparameters
5x-10x fold speedup for optimizing convolutional networks

https://github.com/automl/RoBO
Putting it all together?

- **Transfer learning** from other datasets $D$
  $$\rightarrow f(\lambda, D)$$

- Graybox optimization: **data at each time step $t$**
  $$\rightarrow f(\lambda, t)$$

- **Large datasets**: start from small **data subsets of size $s$**
  $$\rightarrow f(\lambda, s)$$

- How about $f(\lambda, D, t, s)$?

- This will produce **a lot (!) of data points**
  - Expensive blackbox evaluation $\rightarrow$ Cheap incremental evaluations
  - Current Gaussian process models **will not scale!**
Robust weight uncertainty in neural networks using Stochastic Gradient Hamiltonian Monte Carlo

A fit of the sinc function using SGHMC

-sinc(x)
-sinc_SGHMC
Not all types of Bayesian neural networks are as robust
Empirical Evaluation

- **Most related work:**
  - Scalable Bayesian Optimization Using Deep Neural Networks (DNGO) [Snoek et al, ICML 2015]
  - Standard DNNs, with Bayesian linear regression in last layer

- **Results:**
  - Both algorithms effective
  - SGHMC more robust
Empirical Evaluation

• For standard benchmarks:
  as good as Bayesian optimization with GPs

But much more flexible

  – E.g., reasoning over many related datasets

https://github.com/automl/RoBO
Overview

• Bayesian optimization
• AutoML: end-to-end learning systems
• Speeding up AutoML: beyond vanilla Bayesian optimization

End-to-end optimization of combinatorial problem solvers
End-to-end Optimization

- Choices to be made at different levels of solving
  - Problem formulation: e.g., SAT / CP / MIP / TSP / AI planning
  - Encoding
  - Preprocessing
  - Solver and its parameters (sometimes hundreds)
  - Low-level compiler flags
• **The classical manual approach**
  - Make each choice independently, based on small-scale experiments
  - Avoid exposing parameters

• **An end-to-end approach**
  - Embrace all choices & expose them as parameters
  - Use automated methods to jointly optimize these parameters

→ Algorithm configuration / Programming by Optimization
End-to-end ‘Deep’ Optimization

- Many similarities with Deep Learning
  - Very highly parameterized models / solving pipelines
  - Joint optimization of a single loss function
  - Use lots of data to avoid overfitting
  - Paradigm-changing & very strong empirical results

- Dramatic speedups of combinatorial problem solvers
  - 500x for software verification [Hutter, Babic, Hoos, Hu, FMCAD 2007]
    - Won QF_BV category in 2007 SMT competition
  - 50x for MIP [Hutter, Hoos, Leyton-Brown, CPAIOR 2011]
  - 100x for finding better domain encoding in AI planning
    [Vallati, Hutter, Chrpa, McCluskey, IJCAI 2015]
The Choice of Meta-level Optimizer Matters

Example: Optimizing CPLEX on combinatorial auctions (Regions 100)

Random search
Random search + racing
Random search + racing + adaptive capping
SMAC

≥ 1.000x speedup
400x speedup
200x speedup
Algorithm Portfolios

• Algorithm Selection
  – Learn a policy to map instance features to the best algorithm
  – **SATzilla** [Xu, Hutter, Hoos, Leyton-Brown; JAIR 2008 & SAT 2012]
    • Won SAT competitions 2007, 2009, 2012 (every time we entered)
    • Won ICON algorithm selection challenge 2015

• End-to-end portfolio construction
  – **AutoFolio** [Lindauer, Hoos, Hutter; JAIR 2016]
  – **Hydra** [Xu, Hutter, Hoos, Leyton-Brown, RCRA 2011]
  – **Cedalion** [Seipp, Siefert, Helmert, Hutter, AAAI 2015]
    • Won IPC 2014 Planning & Learning Track with FD-Cedalion
Hard combinatorial problems

- SAT, MIP, TSP, AI planning, ASP, time-tabling, Protein Folding, ...
- → key to strong performance in practice

Applications in Freiburg

- BCI pipelines
- Motion tracking
- People tracking
- RNA sequence-structure alignment

Very neat applications by others

- Game theory: kidney exchange
- Linear algebra subroutines
- Improving Java garbage collection

Applications

- FCC spectrum auction
- Mixed integer programming
- Analytics & Optimization
- Social gaming
- Scheduling and Resource Allocation
Example Application: FCC Spectrum Auction

- Wireless frequency spectra: demand increases
  - US Federal Communications Commission (FCC) just held a 13-month auction

- **Key computational problem:**
  feasibility testing based on interference constraints
  - A hard **graph colouring problem**
  - 2991 stations (nodes) & 2.7 million interference constraints
  - Need to solve many different instances
  - More instances solved: higher revenue

- **Best solution:** based on end-to-end (‘deep’) optimization of SAT solvers
  - Configuration with SMAC; portfolios with SATzilla
  - Improved #instances solved from 73% to 99.6%
    [Frechette, Newman & Leyton-Brown, AAAI'16]
  - Net income for US government: $7 billion (used to pay down national debt)
Conclusion

• Bayesian optimization enables **true end-to-end learning**
  – Auto-WEKA, Auto-sklearn & Auto-Net

• Large speedups by going beyond blackbox optimization
  – Learning across datasets
  – Dataset subsampling
  – Learning curve extrapolation

• Similar techniques enable **end-to-end ‘deep’ optimization**

• Links to code: [http://ml4aad.org](http://ml4aad.org)
• **AutoML tutorial & workshop**, this Friday here at ECML
  – With Pavel Brazdil, Holger Hoos and Joaquin Vanschoren

• **Open postdoc positions in my group**
  – Efficient architecture & hyperparameter search for neural nets
  – Deep learning for decoding EEG data
  – Deep RL for parameter control and deep optimization
Thanks!

Funding sources

My fantastic team

Other collaborators

**UBC**: Steve Ramage, Chris Thornton, Holger Hoos, Kevin Leyton-Brown, Kevin Murphy

**Oxford/Google DeepMind**: Ziyu Wang, Michael Osborne, Nando de Freitas

**MPI Tübingen**: Philipp Hennig

**Uni Freiburg**: Tobias Springenberg, Robin Schirrmeister, Andreas Eitel, Michael Tangermann, Tonio Ball, Thomas Brox, Wolfram Burgard
Conclusion

- Bayesian optimization enables **true end-to-end learning**
  - Auto-WEKA, Auto-sklearn & Auto-Net

- Large speedups by going beyond blackbox optimization
  - Learning across datasets
  - Dataset subsampling
  - Learning curve extrapolation

- Similar techniques enable **end-to-end ‘deep’ optimization**

- Links to code: [http://ml4aad.org](http://ml4aad.org)