Algorithm Recommendation as Collaborative Filtering

Michèle Sebag & Mustafa Misir & Philippe Caillou

TAO, CNRS – INRIA – Université Paris-Sud

AutoML Wshop, ICML 2015
Control layer in algorithmic platforms

Goal

deliver peak performance on any/most problem instances

A general issue

- In constraint programming Rice 76
- In stochastic optimization Grefenstette 87
- In machine learning (meta-learning) Bradzil 93

Scope: Selection and Calibration

- Offline control
  Portfolio algorithm selection, optimal hyper-parameter setting
- Online control
  adjusting hyper-parameters during the run
Control layer in algorithmic platforms

Goal

deliver peak performance on any/most problem instances

A general issue

▶ In constraint programming Rice 76
▶ In stochastic optimization Grefenstette 87
▶ In machine learning (meta-learning) Bradzil 93

Scope: Selection and Calibration

▶ Offline control
  Portfolio algorithm selection, optimal hyper-parameter setting
▶ Online control
  adjusting hyper-parameters during the run
An optimization problem
Given a problem instance, find

$$\theta^* = \text{arg opt} \{ \text{Performance} (\theta, \text{pb instance}) \}$$

with $\theta$: algorithm and hyper-parameters thereof

Learn objective function “Performance”

- Learn it (surrogate optimisation)
  
  Hutter et al. 11; Thornton et al. 13

- Learn a monotonous transformation thereof
  
  Bardenet et al. 13; this talk

See also Reversible learning

McLaurin et al. 15
Control: A meta-learning problem

Procedure

- Gather problem instances (benchmark suite)
- Design descriptive features for pb instances
- Run algorithms on pb instances
- Build meta-training set:

\[ \mathcal{E} = \{(\text{desc. of } i\text{-th pb instance, perf. of } j\text{-th algo})\} \]

- Learn \( \hat{h} \) from \( \mathcal{E} \)
- Decision making (predict, optimize)
Some advances in CP and SAT

- **CPHydra**
  - case-based reasoning; kNN
  - O'Mahony et al. 08

- **Satzilla**
  - learn runtime(inst,alg); select argmin runtime
  - Xu et al. 08

- **ParamILS**
  - learn perf(hyper-param); optimize perf
  - Hutter et al. 09

- **Programming by optimization**
  - http://www.prog-by-opt.net/
  - Holger Hoos, 12

100 Features

**Static features**

- Problem definition: density, tightness
- Variable size and degree (min, max, average, variance)
- Constraint degree and cost category (exp, cubic, quadratic, lin. cheap, lin. expensive)

**Dynamic features**

- Heuristic criteria(variable): wdeg, domdeg, impact: min, max, average
- Constraint weight (wdeg): min, max, average
- Constraint filtering: min, max, average of number of times called by propagation

Hutter et al. 06, 07
ML control, the bottleneck

\[ \mathcal{E} = \{ (\text{desc. of } i\text{-th pb instance, perf. of } j\text{-th algo}) \} \]

**Bottleneck: design good cheap descriptive features**

Tentative interpretation

- SAT: “high level” problem instance
- ML: a problem instance is a dataset \( \equiv \) distribution. Learning distribution parameters is expensive
Some advances in ML

- **Matchbox**
  Collaborative filtering + Bayesian learning
  
  Stern et al. 10

- **SCOT**
  \( \hat{\text{perf}}(\text{hyper-param}); \text{optimize} \hat{\text{perf}} \)
  where \( \hat{\text{perf}} \) is learned using learning-to-rank.

  Bardenet et al. 13

- **AutoWeka**
  SMAC (Sequential Model-based Algorithm Configuration) applied on the top of Weka.

  Thornton et al. 13
Overview

Context

**ALORS:** Algorithm Recommender System

Empirical evaluation
- Collaborative filtering performance
- Cold start performance

Visualizing the problem/alg landscape
**Main idea**

Recommender systems

- Set of users, set of products
- Users like/dislike a few products
- A sparse matrix

<table>
<thead>
<tr>
<th>USERS</th>
<th>MOVIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 3</td>
</tr>
<tr>
<td>2</td>
<td>1 3</td>
</tr>
<tr>
<td>3</td>
<td>3 1</td>
</tr>
</tbody>
</table>

*Stern et al. 10 Netflix challenge*
Main idea

Recommender systems
- Set of users, set of products
- Users like/dislike a few products
- A sparse matrix

Algorithm selection
- Set of problem instances, set of algorithms
- Pb instance likes “better”
  algorithms that behave “better” on instance
Differences

▶ Meta-Learning is not (yet) a Big Data problem
(500,000 users, 180,000 movies in Netflix)

▶ The main issue is: dealing with a brand new problem instance:
cold start
Milestones

Acquire data
  - Run a few alg. on problem instances

Collaborative filtering
  - Content-based
  - Model-based

Cold start
  - Handle a brand new pb instance

Sparse matrix

Fill the matrix
Collaborative filtering

Matrix decomposition

- $U: P \times k$, $k = \text{nb latent factors}$
- $V: A \times k$
- s.t. $M \approx UV'$

$m_{i,j} \approx \langle u_i, v_j \rangle$
Bayesian Collaborative filtering

Matchbox

- Define priors on $U$ and $V$ independent Gaussian
- Finite number of perf. levels (1, 2, 3)
- Learn thresholds from $\langle u, v \rangle$ to perf. level
- Latent features = linear combinations of initial features
Specificities

- Include a bias $r_{i,j} \approx \langle u_i, v_j \rangle + b_i$
- Include a threshold-based rank decoding

$$m_{i,j} \approx f(r_{i,j}, \text{thresholds})$$

Motivations

- Non stationary phenomenons
- Fast approximation possible, using a single propagation
**ALORS: Algorithm Recommender System**

Standard SVD: Find $U (n_i, k)$, $V (n_a, k)$

$$\arg\min \text{Loss}(\mathcal{M}, UV')$$

**Loss**

- **RMSE**  
  root mean square error
- **MAE**  
  mean absolute error
- **Rank loss**  
  Cofirank, Weimer et al. 07
Criterion NDCG

\[ DCG(\pi, k) = \sum_{i=1}^{k} \frac{2^\pi(i) - 1}{\log(i + 2)} \]

\[ NDCG(\pi, k) = \frac{DCG(\pi, k)}{DCG(\pi^*, k)} \]

Non convex!

- Use a linear convex upper bound
- Alternate minimization (opt. \( U \) with fixed \( V \); then opt. \( V \) with fixed \( U \))
Cold start in **ALORS**: the cornerstone of meta-learning

Assuming descriptive features $X$

- Use matrix decomposition to build latent features $U$
- Learn $U \approx \phi(X)$
Overview

Context

ALORS: Algorithm Recommender System

Empirical evaluation
  Collaborative filtering performance
  Cold start performance

Visualizing the problem/alg landscape
Experimental setting

Goals of experiments

- Comparison with Matchbox
- Sensitivity study wrt $M$ sparsity
- Performance of cold-start
- Inspecting latent features

Domains

- Satisfiability benchmark SAT 2011
- Constraint programming challenge CP 2008
- Black-box optimization benchmark BBOB 2012
- Machine learning Joaquin Vanschoren

Experimental setting

- Sparsity in 10% - 90% (at least 1 non-missing performance on each line)
- Cold start: 10-fold CV
Comparison with Matchbox

On OpenML, SAT 2011 - 2012, CSP 2008
   no significant differences

Varying the 1st rank threshold in Matchbox (5%, 10%, 33%):
   no significant differences

An artificial problem
200 pb instances \times 30 algorithms
\( x_i, y_j \sim U[-10, 10]^3 \)

\[ m_{i,j} = d(x_i, y_j) + \mathcal{N}(0, \epsilon) \]

Where \( d(x_i, y_j) \) is the Euclidean distance over three coordinates of \( x_i \) and \( y_j \)
Comparison with Matchbox, 2

Average rank of recommended system

1st axis: 4 * noise + sparsity
Comparison with Matchbox, 3

A more fair comparison, providing Matchbox with features $x^{(\ell)}$ and feature products $x^{(\ell)} \times x^{(k)}$

Average rank of recommended system

![Graph showing average rank of recommended system for Alors-SVM, Alors-NN, Matchbox. The x-axis represents epsilon-sparsity, and the y-axis represents average rank. The graph compares the performance of different models across varying epsilon-sparsity values.](image-url)
Collaborative filtering performance

- kNN-\text{Alors} >> \text{CF-Alors} for low sparsity
- Then \text{CF-Alors} catches up
- Low sensitivity to \# latent factors $k \leq 10$
On SAT 2011
On SAT 2011, followed
3P-single best very moderately improves on the single best: the single best already is an ensemble learner (bagging).
Overview

Context

**ALORS**: Algorithm Recommender System

**Empirical evaluation**
- Collaborative filtering performance
- Cold start performance

Visualizing the problem/alg landscape
Cold start performance

On SAT

<table>
<thead>
<tr>
<th>Method</th>
<th>Phase 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APP</td>
</tr>
<tr>
<td>Oracle</td>
<td>22.8 ± 2.5</td>
</tr>
<tr>
<td>SingleBest</td>
<td>17.4 ± 3.0</td>
</tr>
<tr>
<td>Random</td>
<td>13.0 ± 2.2</td>
</tr>
<tr>
<td>model-CF + SVM-CS</td>
<td>17.2 ± 2.5</td>
</tr>
<tr>
<td>memory-CF + SVM-CS</td>
<td>17.7 ± 2.6</td>
</tr>
<tr>
<td>model-CF + NN-CS</td>
<td>16.9 ± 2.9</td>
</tr>
<tr>
<td>memory-CF + NN-CS</td>
<td>17.2 ± 2.8</td>
</tr>
<tr>
<td>3P-SingleBest</td>
<td>20.4 ± 2.3</td>
</tr>
<tr>
<td>3P-Random</td>
<td>17.7 ± 2.2</td>
</tr>
<tr>
<td>3P-(model-CF + SVM-CS)</td>
<td>19.6 ± 2.4</td>
</tr>
<tr>
<td>3P-(memory-CF + SVM-CS)</td>
<td>19.6 ± 2.4</td>
</tr>
<tr>
<td>3P-(model-CF + NN-CS)</td>
<td>19.4 ± 2.5</td>
</tr>
<tr>
<td>3P-(memory-CF + NN-CS)</td>
<td>19.3 ± 2.6</td>
</tr>
</tbody>
</table>
On SAT, followed

<table>
<thead>
<tr>
<th>Method</th>
<th>Phase 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APP</td>
</tr>
<tr>
<td>Oracle</td>
<td>25.3 ± 2.1</td>
</tr>
<tr>
<td>SingleBest</td>
<td>21.5 ± 3.6</td>
</tr>
<tr>
<td>Random</td>
<td>19.3 ± 2.9</td>
</tr>
<tr>
<td>model-CF + SVM-CS</td>
<td>21.5 ± 3.1</td>
</tr>
<tr>
<td>memory-CF + SVM-CS</td>
<td>21.3 ± 3.2</td>
</tr>
<tr>
<td>model-CF + NN-CS</td>
<td>21.5 ± 3.1</td>
</tr>
<tr>
<td>memory-CF + NN-CS</td>
<td>21.5 ± 3.3</td>
</tr>
<tr>
<td>3P-SingleBest</td>
<td>23.5 ± 3.0</td>
</tr>
<tr>
<td>3P-Random</td>
<td>22.9 ± 2.1</td>
</tr>
<tr>
<td>3P-(model-CF + SVM-CS)</td>
<td>23.4 ± 2.6</td>
</tr>
<tr>
<td>3P-(memory-CF + SVM-CS)</td>
<td>23.5 ± 2.6</td>
</tr>
<tr>
<td>3P-(model-CF + NN-CS)</td>
<td>23.5 ± 2.5</td>
</tr>
<tr>
<td>3P-(memory-CF + NN-CS)</td>
<td>23.8 ± 2.6</td>
</tr>
</tbody>
</table>
## Cold start performance on CSP

<table>
<thead>
<tr>
<th>Method</th>
<th>GLOBAL</th>
<th>k-ARY-INT</th>
<th>2-ARY-EXT</th>
<th>N-ARY-EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>49.3 ± 3.2</td>
<td>130.2 ± 3.2</td>
<td>62.0 ± 1.3</td>
<td>44.9 ± 2.3</td>
</tr>
<tr>
<td>Random</td>
<td>32.4 ± 3.2</td>
<td>86.2 ± 4.9</td>
<td>47.4 ± 3.2</td>
<td>29.2 ± 2.4</td>
</tr>
<tr>
<td>SingleBest</td>
<td>41.6 ± 5.2</td>
<td><strong>116.5 ± 5.8</strong></td>
<td><strong>57.2 ± 2.3</strong></td>
<td><strong>43.1 ± 2.8</strong></td>
</tr>
<tr>
<td>model-CF + SVM-CS</td>
<td>39.5 ± 5.1</td>
<td>111.5 ± 7.4</td>
<td>56.2 ± 2.9</td>
<td>42.2 ± 2.0</td>
</tr>
<tr>
<td>memory-CF + SVM-CS</td>
<td>43.6 ± 4.8</td>
<td><strong>115.3 ± 6.9</strong></td>
<td><strong>57.1 ± 2.9</strong></td>
<td><strong>43.4 ± 2.4</strong></td>
</tr>
<tr>
<td>model-CF + NN-CS</td>
<td>39.4 ± 5.1</td>
<td>110.8 ± 6.9</td>
<td>56.1 ± 2.9</td>
<td>42.1 ± 2.1</td>
</tr>
<tr>
<td>memory-CF + NN-CS</td>
<td><strong>44.1 ± 4.4</strong></td>
<td>115.0 ± 6.4</td>
<td><strong>57.4 ± 2.9</strong></td>
<td><strong>43.4 ± 2.6</strong></td>
</tr>
<tr>
<td>3P-Random</td>
<td>44.6 ± 3.2</td>
<td>115.9 ± 4.4</td>
<td>57.2 ± 2.3</td>
<td>41.3 ± 2.0</td>
</tr>
<tr>
<td>3P-SingleBest</td>
<td><strong>47.4 ± 3.8</strong></td>
<td>117.6 ± 5.6</td>
<td><strong>58.3 ± 2.5</strong></td>
<td><strong>44.2 ± 2.3</strong></td>
</tr>
<tr>
<td>3P-(model-CF + SVM-CS)</td>
<td>44.0 ± 4.0</td>
<td>119.6 ± 5.8</td>
<td>57.4 ± 2.4</td>
<td>43.8 ± 2.2</td>
</tr>
<tr>
<td>3P-(memory-CF + SVM-CS)</td>
<td><strong>47.0 ± 3.6</strong></td>
<td>122.2 ± 5.8</td>
<td><strong>58.3 ± 2.2</strong></td>
<td><strong>44.2 ± 2.2</strong></td>
</tr>
<tr>
<td>3P-(model-CF + NN-CS)</td>
<td>43.9 ± 4.0</td>
<td>119.1 ± 5.7</td>
<td>57.3 ± 2.5</td>
<td>43.8 ± 2.2</td>
</tr>
<tr>
<td>3P-(memory-CF + NN-CS)</td>
<td><strong>46.9 ± 3.5</strong></td>
<td>121.9 ± 5.4</td>
<td><strong>58.6 ± 2.3</strong></td>
<td><strong>44.2 ± 2.2</strong></td>
</tr>
</tbody>
</table>
## Cold start performance on OpenML

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>0.121 ± 0.000</td>
</tr>
<tr>
<td>SingleBest</td>
<td>0.170 ± 0.000</td>
</tr>
<tr>
<td>Random</td>
<td>0.253 ± 0.000</td>
</tr>
<tr>
<td>memory-CF + SVM-CS</td>
<td>0.180 ± 0.008</td>
</tr>
<tr>
<td>memory-CF + NN-CS</td>
<td>0.184 ± 0.008</td>
</tr>
<tr>
<td>3P-SingleBest</td>
<td>0.166 ± 0.000</td>
</tr>
<tr>
<td>3P-Random</td>
<td>0.179 ± 0.000</td>
</tr>
<tr>
<td>3P-(memory-CF + SVM-CS)</td>
<td><strong>0.160 ± 0.004</strong></td>
</tr>
<tr>
<td>3P-(memory-CF + NN-CS)</td>
<td>0.163 ± 0.003</td>
</tr>
</tbody>
</table>

Not much margin of improvement: single best close to oracle.
Overview

Context

**ALORS:** Algorithm Recommender System

Empirical evaluation

- Collaborative filtering performance
- Cold start performance

Visualizing the problem/alg landscape
Where we learn something about the field

Each pb instance: a vector in $\mathbb{R}^d$; mapped onto $\mathbb{R}^2$ using Multi-dimensional scaling.

Left: initial features
Right: Latent features

On SAT
On SAT, followed
On CSP
On CSP, followed
On Black-Box Optimization functions
On OpenML

Datasets
On OpenML, 2

Algorithms
On OpenML, 3

Algorithms and Datasets
Conclusion

- Algorithm recommender system works
- Cold start requires initial features
  - These can be poorly informative (BBOB)
  - Current ML features are not informative enough
- Provides educated (latent) features
Short and mid-term perspectives

Use latent features in order to
- Assess a benchmark suite (diversity);
- Assess a validation procedure (coverage of the benchmark suite used to validate a new algorithm);
- Assess novelty of an algorithm

Learn descriptive features
- using clusters based on latent features
Longer-term perspectives

- Intrinsic description of alg / problems
- Certification of portfolios
- Understand what makes it hard (new cues for parameterized complexity)

**A typology of problems and algorithms**