Hyperparameter tuning *across* datasets

Siminole meeting

Rémi Bardenet, joint work w/ M. Brendel, B. Kégl & M. Sebag

LAL, LRI, Univ. Paris-Sud XI

October 19th 2012

Contents

1. Model-based tuning on single datasets
2. A ranking-based latent structure
3. A case-study on AdaBoost

1. **Model-based tuning on single datasets**
2. A ranking-based latent structure
3. A case-study on AdaBoost

Classical hyperparameter tuning

Algorithm	Hyperparameters
SVM, $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\ell^2}\right)$	ℓ, C
MLP	learning rate, batchsize, size of hidden layer, penalties, ...
Boosting	Number of iterations, hyper-parameters of the weak classifiers.

Classical hyperparameter tuning

Algorithm	Hyperparameters
SVM, $k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\ell^2}\right)$	ℓ, C
MLP	learning rate, batchsize, size of hidden layer, penalties, ...
Boosting	Number of iterations, hyper-parameters of the weak classifiers.

$\textsc{ExhaustiveTuning}(D, \mathcal{H} \subset \mathbb{H}, \mathcal{A})$

for $x \in \mathcal{H}$, ▷ *Outer loop*
    Train $\mathcal{A}$ on D with hyperparameters x, ▷ *Inner loop*
    Compute validation error $f(x) = R(\mathcal{A}(D, x))$,
return $\arg\min_{x \in \mathcal{H}} f(x)$.

Sequential model-based optimization

$\mathrm{SMBO}(f, \mathcal{M}_0, T, S)$

$\mathcal{O} \leftarrow \emptyset$,

For $t \leftarrow 1$ **to** T,

  $x^* \leftarrow \arg\max_x \; S(x, \mathcal{M}_{t-1})$,

  Evaluate $f(x^*)$, $\triangleright$ *Expensive step*

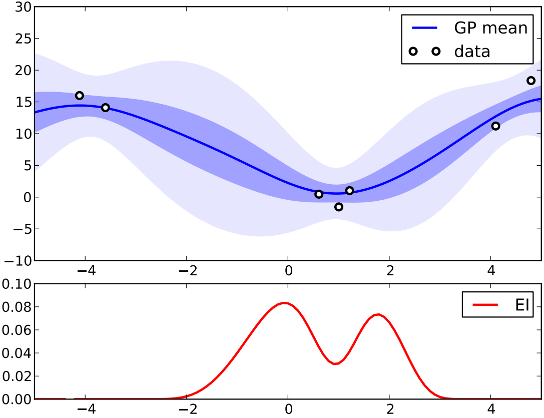
  $\mathcal{O} \leftarrow \mathcal{O} \cup (x^*, f(x^*))$,

  Fit a new model $\mathcal{M}_t$ to $\mathcal{O}$,

return $\arg\min_{\mathcal{O}} f(x)$.

- SMBO is useful when target evaluation is costly.

Gaussian Processes and Expected Improvement



- GPs are priors over functions that are closed under sampling.
- EI$(x) := \mathbb{E}\big((\min_i f(x_i) - f(x)) \wedge 0 | \mathcal{F}_n\big)$.
- There are other choices [6, 10, 8].

Sequential model-based tuning in ML

- SMBO was successfully applied to deep learning [1],
- Since then, advances were made in methodology [8], benchmarking [9], software [2].

Sequential model-based tuning in ML

- SMBO was successfully applied to deep learning [1],
- Since then, advances were made in methodology [8], benchmarking [9], software [2].

But...

All experiments have been based on single datasets, while humans have a memory of past experiments on similar datasets.

Sequential model-based tuning in ML

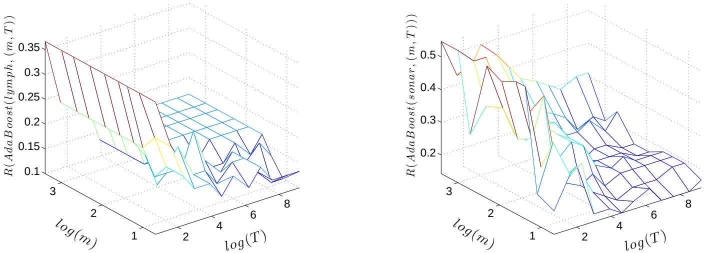
- SMBO was successfully applied to deep learning [1],
- Since then, advances were made in methodology [8], benchmarking [9], software [2].

But...

All experiments have been based on single datasets, while humans have a memory of past experiments on similar datasets.

- Is there something to gain by using information obtained on other datasets?
- Does the SMBO framework extend to several datasets?

1. Model-based tuning on single datasets
2. **A ranking-based latent structure**
3. A case-study on AdaBoost



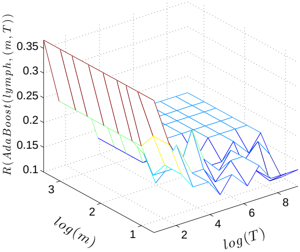
- Validation errors on 2 datasets can differ arbitrarily in scale.
- We need a target

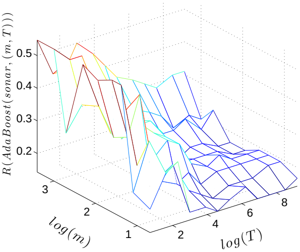
$$f_{\mathcal{A}} : \mathbb{D} \times \mathbb{H} \longrightarrow \mathbb{R}$$

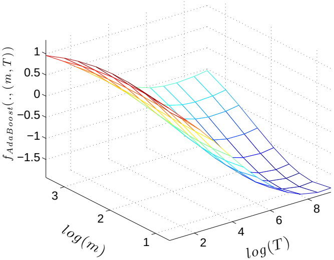
that conveys information such that

$$\textit{if } f_{\mathcal{A}}(D_1, x_1) < f_{\mathcal{A}}(D_1, x_2) \textit{ and } D_2 \textit{ is similar to } D_1,$$
$$\textit{then probably } f_{\mathcal{A}}(D_2, x_1) < f_{\mathcal{A}}(D_2, x_2),$$

A common latent structure







- SVMrank [5] tries to find a smooth function g that is monotone in the input rankings: $x \prec y \Rightarrow g(x) \leq g(y)$.

A latent ranker

- SVMrank [5] tries to find a smooth function g that is monotone in the input rankings: $x \prec y \Rightarrow g(x) \leq g(y)$.
- A new SMBO paradigm: define

$$(D, x_1) \prec (D, x_2) \Leftrightarrow R\big(\mathcal{A}(D, x_1)\big) < R\big(\mathcal{A}(D, x_2)\big),$$

and repeatedly

1. give all available rankings to SVMrank,
2. fit a GP to SVMrank's output g,
3. maximize EI : $\mathbb{H} \rightarrow \mathbb{R}_+$,
4. evaluate new point.

A latent ranker

- SVMrank [5] tries to find a smooth function g that is monotone in the input rankings: $x \prec y \Rightarrow g(x) \leq g(y)$.
- A new SMBO paradigm: define

$$(D, x_1) \prec (D, x_2) \Leftrightarrow R\big(\mathcal{A}(D, x_1)\big) < R\big(\mathcal{A}(D, x_2)\big),$$

  and repeatedly
  1. give all available rankings to SVMrank,
  2. fit a GP to SVMrank's output g,
  3. maximize $\mathrm{EI} : \mathbb{H} \to \mathbb{R}_+$,
  4. evaluate new point.
- The latent ranker of SVMrank carries all information provided by the validation errors across datasets.
- The choice of SVMrank is not unique [4].

The surrogate tuning algorithm

$\mathrm{ST}\big(D, T, \mathcal{O} = (\mathcal{D}, \mathcal{H}, \mathcal{R}), \mathcal{A}, \mathcal{B}\big)$

$\mathcal{O}_0 \leftarrow \mathcal{O}$,

For $t \leftarrow 0$ **to** $T - 1$,

Compute rankings $\mathcal{P}_t$ defined by $\prec$ from $\mathcal{O}_t$,

$\widehat{\mathbf{f}}_t \leftarrow$ surrogate model built by $\mathcal{B}$ called on

$(\mathcal{D}_t, \mathcal{H}_t)$ with rankings $\mathcal{P}_t$,

$M_{t-1} \leftarrow$ Posterior GP on $\widehat{f}_t$ knowing

$((\mathcal{D}_t, \mathcal{H}_t), \widehat{\mathbf{f}}_t)$,

$x^* \leftarrow \mathrm{argmax}_{x \in \mathbb{H}} \; EI(D, x)$,

$R^* \leftarrow R(\mathcal{A}(D, x^*))$, $\triangleright$ *Run learning algo.*

$\mathcal{O}_{t+1} \leftarrow \mathcal{O}_t \cup (D, x^*, R^*)$,

return $\mathcal{O}_T$.

The surrogate collaborative tuning algorithm

SCoT$\big((D_1, \ldots, D_M), T, \mathcal{O} = (\mathcal{D}, \mathcal{H}, \mathcal{R}), \mathcal{A}, \mathcal{B}\big)$

$\mathcal{O}_0 \leftarrow \mathcal{O}$.

For $t \leftarrow 0$ **to** $T - 1$,

  For $i \leftarrow 1$ **to** M,

    Compute rankings $\mathcal{P}_t$ defined by $\prec$ from $\mathcal{O}_t$,

    $\widehat{\mathbf{f}}_t \leftarrow$ surrogate model built by $\mathcal{B}$ called on

      $(\mathcal{D}_t, \mathcal{H}_t)$ with rankings $\mathcal{P}_t$,

    $M_{t-1} \leftarrow$ Posterior GP on $\widehat{f}_t$ knowing

      $\big((\mathcal{D}_t, \mathcal{H}_t), \widehat{\mathbf{f}}_t\big)$,

    $x^* \leftarrow \mathrm{argmax}_{x \in \mathbb{H}} \; EI(D_i, x)$,

    $R^* \leftarrow R(\mathcal{A}(D_i, x^*))$, $\triangleright$ *Run learning algo.*

    $\mathcal{O}_{t+1} \leftarrow \mathcal{O}_t \cup (D_i, x^*, R^*)$,

return $\mathcal{O}_T$.

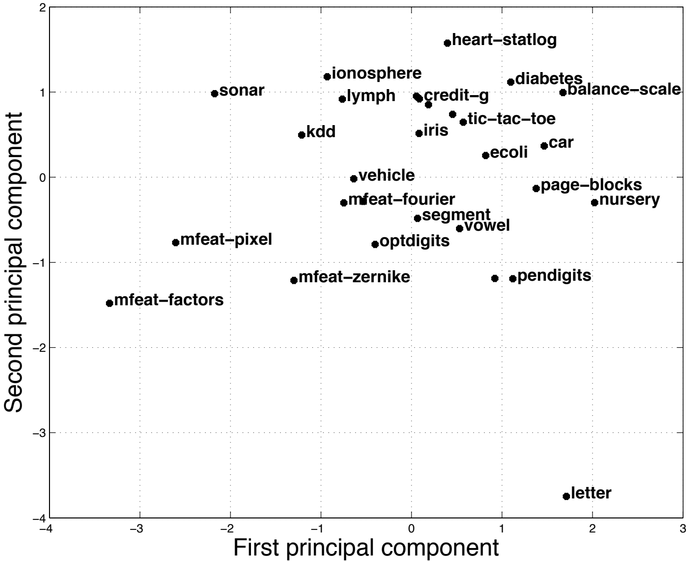
1. Model-based tuning on single datasets
2. A ranking-based latent structure
3. **A case-study on AdaBoost**

Setup

- AdaBoost with decision products as weak learners [7] has two hyperparameters: number of iterations T and number of product terms m.

- AdaBoost with decision products as weak learners [7] has two hyperparameters: number of iterations T and number of product terms m.
- The small number of hyperparameters allows to set a grid on $\mathbb{H}$ and pre-compute all validation errors.

- AdaBoost with decision products as weak learners [7] has two hyperparameters: number of iterations T and number of product terms m.
- The small number of hyperparameters allows to set a grid on $\mathbb{H}$ and pre-compute all validation errors.
- We downloaded 29 classification problems from Weka, and instantiated $\mathbb{D}$ with the following features:
  - Number of classes K,
  - dimension d,
  - number of samples n,
  - $\rho = d'/d$, where d' is the smallest integer such that the first d' principal components of the dataset explain 95% of its variance.


heart−statlog
ionosphere
diabetes
sonar
lymph
credit−g
balance−scale
tic−tac−toe
kdd
iris
car
ecoli
vehicle
page−blocks
mfeat−fourier
nursery
segment
vowel
mfeat−pixel
optdigits
mfeat−zernike
pendigits
mfeat−factors
letter
Second principal component
First principal component
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3

We used a 5-fold CV on the 29 datasets and compared the following strategies:

Global default Always use the hyperparameter that minimizes the average error over the meta-train problems.

We used a 5-fold CV on the 29 datasets and compared the following strategies:

Global default Always use the hyperparameter that minimizes the average error over the meta-train problems.

Collaborative default Do one iteration of SCoT only: fit a GP on the meta train problems and take, for each meta-test problem, the hyperparameter with the best posterior mean.

Comparing tuning strategies

We used a 5-fold CV on the 29 datasets and compared the following strategies:

Global default Always use the hyperparameter that minimizes the average error over the meta-train problems.

Collaborative default Do one iteration of SCoT only: fit a GP on the meta train problems and take, for each meta-test problem, the hyperparameter with the best posterior mean.

Separate surrogate tuning Use independent two-dimensional GP for each meta-test problem,

We used a 5-fold CV on the 29 datasets and compared the following strategies:

Global default Always use the hyperparameter that minimizes the average error over the meta-train problems.

Collaborative default Do one iteration of SCoT only: fit a GP on the meta train problems and take, for each meta-test problem, the hyperparameter with the best posterior mean.

Separate surrogate tuning Use independent two-dimensional GP for each meta-test problem,

SCoT Use all available information in a single GP.

We used a 5-fold CV on the 29 datasets and compared the following strategies:

Global default Always use the hyperparameter that minimizes the average error over the meta-train problems.

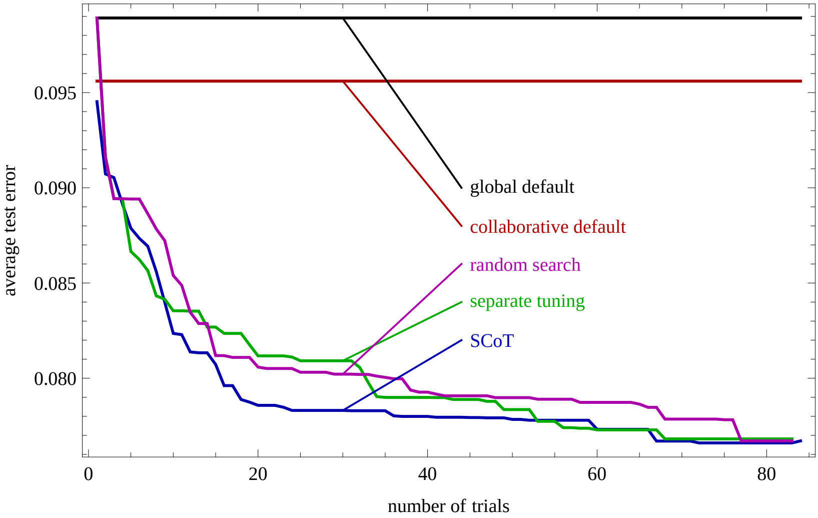
Collaborative default Do one iteration of SCoT only: fit a GP on the meta train problems and take, for each meta-test problem, the hyperparameter with the best posterior mean.

Separate surrogate tuning Use independent two-dimensional GP for each meta-test problem,

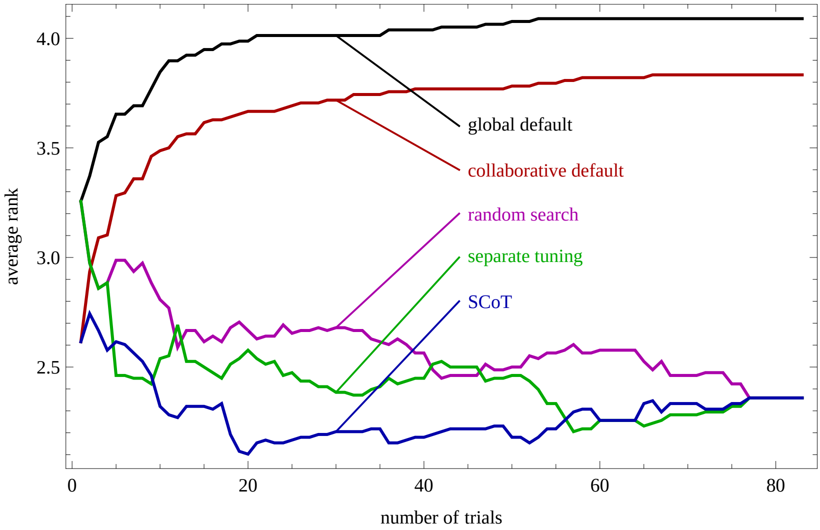
SCoT Use all available information in a single GP.

Random search It was shown to perform well in such settings [3].

Comparing average meta-test errors

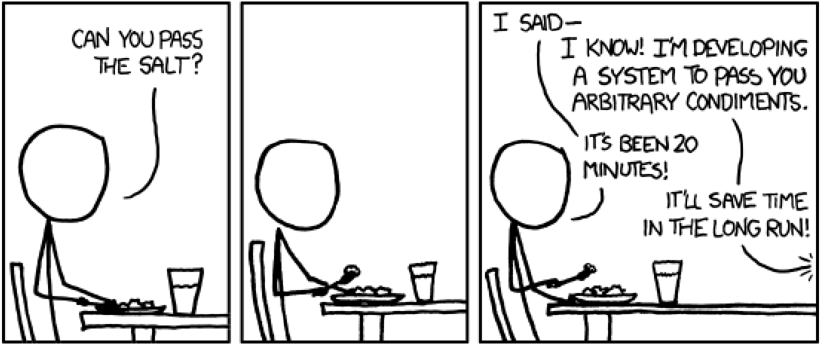


Comparing average meta-test rankings



Conclusions

- SCoT performs hyperparameter tuning using information gathered with the same algorithm on other datasets.
- It is a novel Bayesian optimization algorithm, which targets a function up to a monotone transformation.
- We are currently performing experiments with MLPs and more statistical features.
- Future work should address asynchronous tuning, feature construction, and scalable surrogate models, closing the gap to a fully automatic collaborative tuner!


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio.
Algorithms for hyperparameter optimization.
In *Advances in Neural Information Processing Systems*, volume 24. The MIT Press, 2011.

J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio.
Implementations of algorithms for hyper-parameter optimization.
In *NIPS Workshop on Bayesian optimization*, 2011.

J. Bergstra and Y. Bengio.
Random search for hyper-parameter optimization.
Journal of Machine Learning Research, 2012.

W. Chu and Z. Ghahramani.
Preference learning with Gaussian processes.
In *Proceedings of the 22nd International Conference on Machine Learning*, pages 137–144, 2005.

T. Joachims.
Optimizing search engines using clickthrough data.
In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

D. R. Jones.
A taxonomy of global optimization methods based on response surfaces.
Journal of Global Optimization, 21:345–383, 2001.

B. Kégl and R. Busa-Fekete.
Boosting products of base classifiers.
In *International Conference on Machine Learning*, volume 26, pages 497–504, Montreal, Canada, 2009.

J. Snoek, H. Larochelle, and R. P. Adams.
Practical Bayesian optimization of machine learning algorithms.
In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown.
Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms.
Technical Report TR-2012-05, University of British Columbia, Department of Computer Science, 2012.

J. Villemonteix, E. Vazquez, and E. Walter.
An informational approach to the global optimization of expensive-to-evaluate functions.
Journal of Global Optimization, 2006.