Hyperparameter optimization

MIPT

2025

Model selection: coherent inference

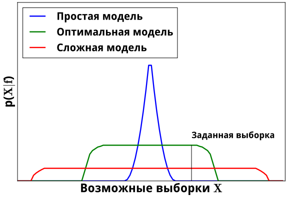
First level: select optimal parameters:

$$\mathbf{w} = \arg\max \frac{p(\mathfrak{D}|\mathbf{w})p(\mathbf{w}|\mathbf{h})}{p(\mathfrak{D}|\mathbf{h})},$$

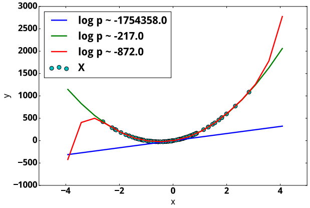
Second level: select optimal model (hyperparameters).

Evidence:

$$p(\mathfrak{D}|\mathbf{h}) = \int_{\mathbf{w}} p(\mathfrak{D}|\mathbf{w})p(\mathbf{w}|\mathbf{h})d\mathbf{w}.$$



Model selection scheme



Example: polynoms

Hyperparameters

Definition

Prior for parameters $\mathbf{w}$ and structure $\mathbf{\Gamma}$ of the model $\mathbf{f}$ is a distrubution $p(\mathbf{W}, \mathbf{\Gamma}|\mathbf{h}) : \mathbb{W} \times \mathbb{\Gamma} \times \mathbb{H} \to \mathbb{R}^+$, where $\mathbb{W}$ is a parameter space, $\mathbb{\Gamma}$ is a structure space.

Definition

Hyperparameters $\mathbf{h} \in \mathbb{H}$ of the models are the parameters of $p(\mathbf{w}, \mathbf{\Gamma}|\mathbf{h})$ (parameters of prior $\mathbf{f}$).

Laplace approximation

Nonlinear case with m objects and n features: $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{w}), \lambda^{-1}), \boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{A}^{-1})$.
Write integral:

$$p(\mathfrak{D}|\boldsymbol{h}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{A}, \lambda) = \frac{\sqrt{\lambda \cdot |\boldsymbol{A}|}}{\sqrt{(2\pi)^{m+n}}} \int_{\boldsymbol{w}} \exp(-S(\boldsymbol{w}))d\boldsymbol{w}.$$

Using Taylor serioes for S:

$$S(\boldsymbol{w}) \approx S(\hat{\boldsymbol{w}}) + \frac{1}{2}\Delta\boldsymbol{w}^{\mathsf{T}}\boldsymbol{H}\Delta\boldsymbol{w}$$

Integral reduces to the following expression:

$$\frac{\sqrt{\lambda \cdot |\boldsymbol{A}|}}{\sqrt{(2\pi)^{m+n}}} S(\hat{\boldsymbol{w}}) \int_{\boldsymbol{w}} \exp(-\frac{1}{2}\Delta\boldsymbol{w}^{\mathsf{T}}\boldsymbol{H}\Delta\boldsymbol{w})d\boldsymbol{w}$$

The expression under integral corresponds to the unnormalized Gaussian PDF.

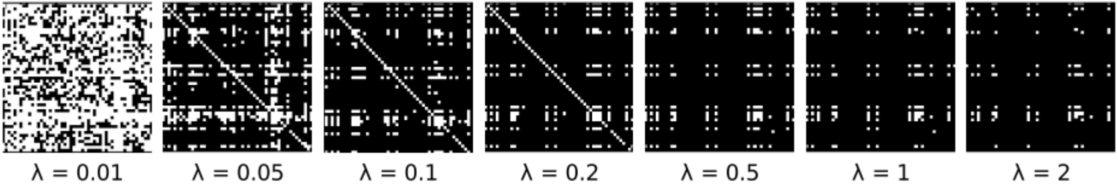
Graves, 2011

Prior: $p(\boldsymbol{w}|\sigma) \sim \mathcal{N}(\boldsymbol{\mu}, \sigma \boldsymbol{I})$.
Variational inference: $q(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{\mu}_q, \sigma_q \boldsymbol{I})$.
Greedy optimization:

$$\boldsymbol{\mu} = \hat{\mathsf{E}}\boldsymbol{w}, \quad \sigma = \hat{\mathsf{D}}\boldsymbol{w}.$$

Prune w_i using relative PDF:

$$\lambda = \frac{q(\mathbf{0})}{q(\boldsymbol{\mu}_{i,q})} = \exp(-\frac{\mu_i^2}{2\sigma_i^2}).$$



Problem statement

Let $\boldsymbol{\theta} \in \mathbb{R}^s$ be the set of all the optimized parameters (including variational parameters if needed).
$L(\boldsymbol{\theta}, \boldsymbol{h})$ is a differential loss function $\boldsymbol{f}$.
$Q(\boldsymbol{\theta}, \boldsymbol{h})$ is a differential validation function.
The problem is to find optimal parameters $\boldsymbol{\theta}^*$ and hyperparameters $\boldsymbol{h}^*$ of the model that minimze

$$\boldsymbol{h}^* = \arg\max_{\boldsymbol{h} \in \mathbb{H}} Q(\boldsymbol{\theta}^*(\boldsymbol{h}), \boldsymbol{h}),$$

$$\boldsymbol{\theta}(\boldsymbol{h})^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^s} L(\boldsymbol{\theta}, \boldsymbol{h}).$$

Bayesian inference

Let $\boldsymbol{\theta} = [\boldsymbol{w}]^{\mathsf{T}}$.
First level:

$$\boldsymbol{\theta}^* = \arg\max\bigl(-L(\boldsymbol{\theta}, \boldsymbol{h})\bigr) = p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{h}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{h})}{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{h})}.$$

Second level:

$$p(\boldsymbol{h}|\boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{h})p(\boldsymbol{h}),$$

Considering $p(\boldsymbol{h})$ improper flat prior we get the following expression:

$$Q(\boldsymbol{\theta}, \boldsymbol{h}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{h}) = \int_{\boldsymbol{w}\in\mathbb{R}^u} p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{h}) \to \max_{\boldsymbol{h}\in\mathbb{H}}.$$

Cross-validation

Split the dataset $\mathfrak{D}$ into k equal (maybe stratified) parts:

$$\mathfrak{D} = \mathfrak{D}_1 \sqcup \cdots \sqcup \mathfrak{D}_k.$$

Optimize k modelsfor each data part. Let $\boldsymbol{\theta} = [\mathbf{w}_1, \dots, \mathbf{w}_k]$, where $\mathbf{w}_1, \dots, \mathbf{w}_k$ are the model parameters for optimization k.

Let L be a loss function:

$$L(\boldsymbol{\theta}, \mathbf{h}) = -\frac{1}{k}\sum_{q=1}^{k}\Big(\frac{k}{k-1}\log p(\mathbf{y} \setminus \mathbf{y}_q | \mathbf{X} \setminus \mathbf{X}_q, \mathbf{w}_q) + \log p(\mathbf{w}_q | \mathbf{h})\Big). \tag{1}$$

Let Q be a validation loss:

$$Q(\boldsymbol{\theta}, \mathbf{h}) = \frac{1}{k}\sum_{q=1}^{k} k\log p(\mathbf{y}_q | \mathbf{X}_q, \mathbf{w}_q).$$

ELBO

Let $L = -Q$:

$$\log\, p(\mathbf{y}|\mathbf{X}, \mathbf{A}) \geq \sum_{\mathbf{x},y} \log\, p(y|\mathbf{x}, \hat{\mathbf{w}}) - D_{\mathrm{KL}}\big(q(\mathbf{w})||p(\mathbf{w}|\mathbf{A})\big) = -L(\boldsymbol{\theta}, \mathbf{A}^{-1}) = Q(\boldsymbol{\theta}, \mathbf{A}^{-1}),$$

where q is a normal distribution with diagonal covariance matrix:

$$q \sim \mathcal{N}(\boldsymbol{\mu}_q, \mathbf{A}_q^{-1}),$$

$$D_{\mathrm{KL}}\big(q(\mathbf{w})||p(\mathbf{w}|\mathbf{f})\big) = \frac{1}{2}\big(\mathrm{Tr}[\mathbf{A}\mathbf{A}_q^{-1}] + (\boldsymbol{\mu} - \boldsymbol{\mu}_q)^{\mathsf{T}}\mathbf{A}(\boldsymbol{\mu} - \boldsymbol{\mu}_q) - u + \ln\, |\mathbf{A}^{-1}| - \ln\, |\mathbf{A}_q^{-1}|\big).$$

Use variational parameters of q as a vector of optimized parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = [\alpha_1, \ldots, \alpha_u, \mu_1, \ldots, \mu_u].$$

Evidence vs CV

Evidece estimation:

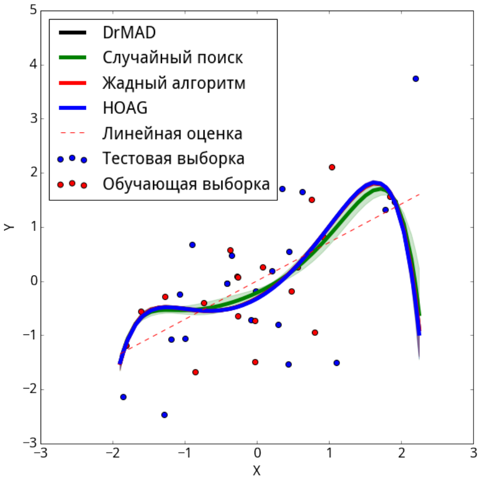
$$\log\, p(\mathfrak{D}|\boldsymbol{f}) = \log\, p(\mathfrak{D}_1|\boldsymbol{f}) + \log\, p(\mathfrak{D}_2|\mathfrak{D}_1, \boldsymbol{f}) + \dots + \log\, p(\mathfrak{D}_n|\mathfrak{D}_1, \dots, \mathfrak{D}_{n-1}, \boldsymbol{f}).$$

Leave-one-out estimation:

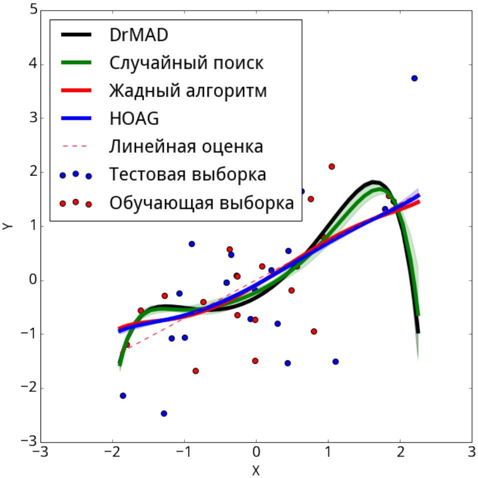
$$\mathrm{LOU} = \mathrm{E}\log\, p(\mathfrak{D}_n|\mathfrak{D}_1, \dots, \mathfrak{D}_{n-1}, \boldsymbol{f}).$$

Cross-validation uses expected values of the last term $p(\mathfrak{D}_n|\mathfrak{D}_1, \dots, \mathfrak{D}_{n-1}, \boldsymbol{f})$ as a complexity estimation.
Evidence considers **full** complexity.

Experiment: polynoms



CV



Evidence

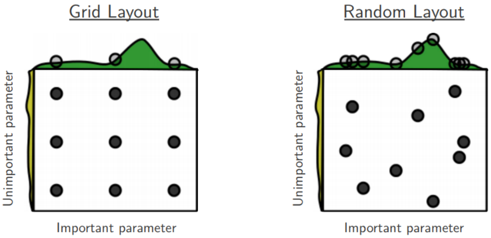
Basic methods of hyperparameter optimization

Variants:

- Grid search;
- random search.

Both methods suffer from curse of dimensionality.

The random search can be more effective if the hyperparameter space is degenerate.



Bergstra et al., 2012

Sequential model-based optimization (SMBO)

Algorithm Framework 1: Sequential Model-Based Optimization (SMBO)

$\mathbf{R}$ keeps track of all target algorithm runs performed so far and their performances (*i.e.*, SMBO's training data $\{([\boldsymbol{\theta}_1, \boldsymbol{x}_1], o_1), \ldots, ([\boldsymbol{\theta}_n, \boldsymbol{x}_n], o_n)\}$), $\mathcal{M}$ is SMBO's model, $\vec{\boldsymbol{\Theta}}_{new}$ is a list of promising configurations, and t_{fit} and t_{select} are the runtimes required to fit the model and select configurations, respectively.

Input : Target algorithm A with parameter configuration space $\boldsymbol{\Theta}$; instance set Π; cost metric $\hat{c}$

Output : Optimized (incumbent) parameter configuration, $\boldsymbol{\theta}_{inc}$

$[\mathbf{R}, \boldsymbol{\theta}_{inc}] \leftarrow \mathit{Initialize}(\boldsymbol{\Theta}, \Pi)$

repeat

    $[\mathcal{M}, t_{fit}] \leftarrow \mathit{FitModel}(\mathbf{R})$

    $[\vec{\boldsymbol{\Theta}}_{new}, t_{select}] \leftarrow \mathit{SelectConfigurations}(\mathcal{M}, \boldsymbol{\theta}_{inc}, \boldsymbol{\Theta})$

    $[\mathbf{R}, \boldsymbol{\theta}_{inc}] \leftarrow \mathit{Intensify}(\vec{\boldsymbol{\Theta}}_{new}, \boldsymbol{\theta}_{inc}, \mathcal{M}, \mathbf{R}, t_{fit} + t_{select}, \Pi, \hat{c})$

until *total time budget for configuration exhausted*

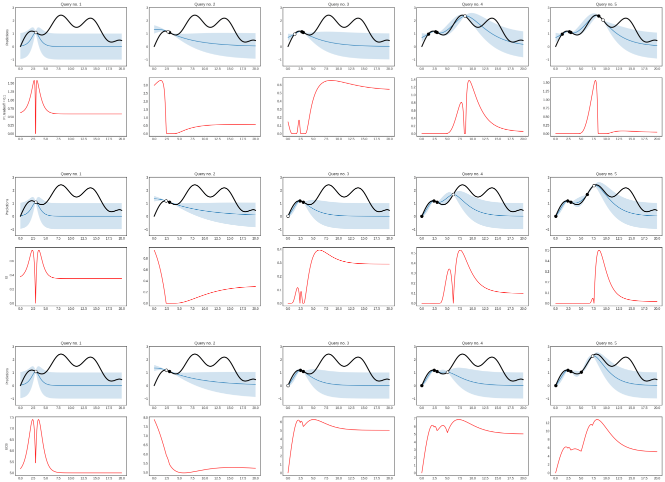
return $\boldsymbol{\theta}_{inc}$

Next point to estimate

Next point selection is done using *Acquisition function*:

- Upper confidence level
- Probability of Improvement: $P(I(\mathbf{h} > 0)), I(\mathbf{h}) = \max\left(L(\mathbf{h}) - L(\mathbf{h}^*), 0\right)$
- Expected improvement $\mathrm{E}I(\mathbf{h})$

Acquisition functions



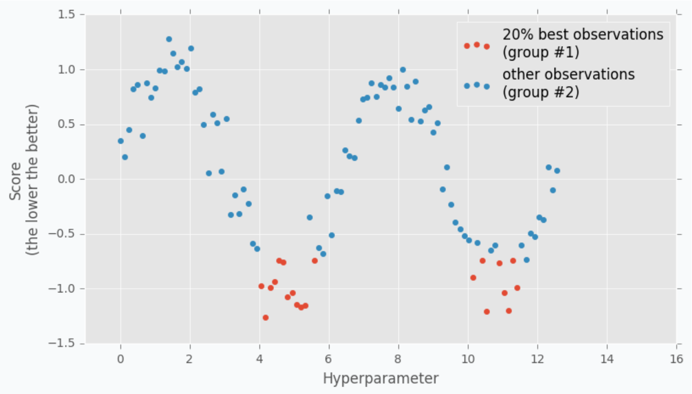
From modAL python library: PI, EI, UCB.

Tree Parzen estimator

Basic idea

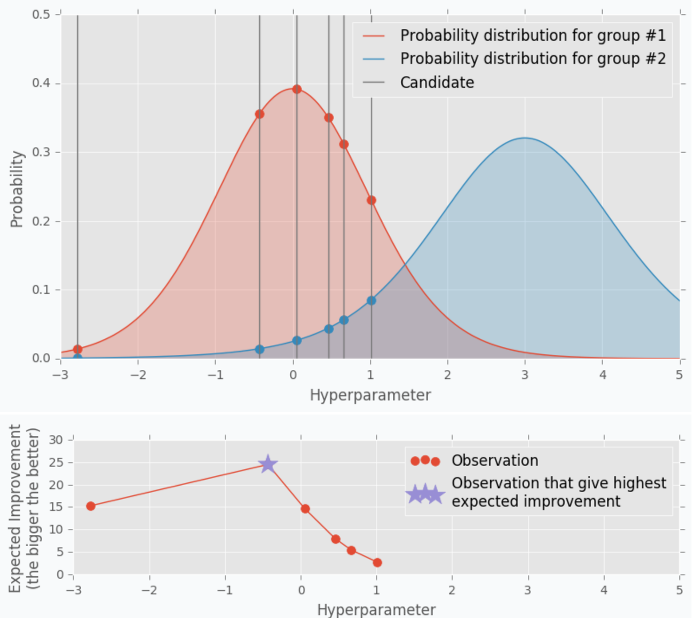
- Sample multiple hyperparameter instances $\mathbf{h}_i$
- Fit models $\mathbf{f}_i$ using $\mathbf{h}_i$
- Select models from λ-quantile of model results Loss_λ and fit adaptive Parzen estimator p_1
- Select remaining models and fit adaptive Parzen estimator p_2
- Sample new hyperparameter $\mathbf{h}$ that maximizes Expected improvement: $\mathrm{E}I(\mathbf{h})$.

TPE



From NeurPy library.

TPE


Probability distribution for group #1
Probability distribution for group #2
Candidate
Probability
Hyperparameter
Observation
Observation that give highest expected improvement
Expected Improvement (the bigger the better)
Hyperparameter

Gaussian process, definition (wiki)

- A random process f_t with continuous time is gaussian if and only if for each finite set of indices $t_1, \ldots, t_k$: $f_{t_1}, \ldots, f_{t_k}$ is a mutlivariative Gaussian variable.
- Each linear combination $f_{t_1}, \ldots, f_{t_k}$ is a univariative Gaussian.

Definition (simplified)

Define a Gaussian process $\mathcal{GP}(\boldsymbol{m(x)}, \boldsymbol{k(x, x')})$ to be a distribution on the set of functions that for each $\boldsymbol{x}, \boldsymbol{x'}$: $\mathcal{GP}(\boldsymbol{m(x)}, \boldsymbol{k(x, x')})$ is a Gaussian distribution.

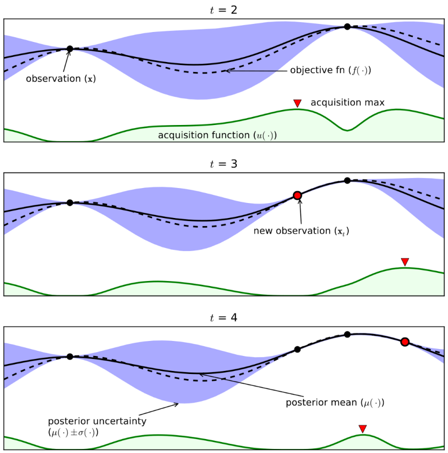
Gaussian process

Idea: Model $Q(\boldsymbol{\theta}(\boldsymbol{h})^*, \boldsymbol{h})$ using Gaussian process depending on $\boldsymbol{h}$.

Pros:

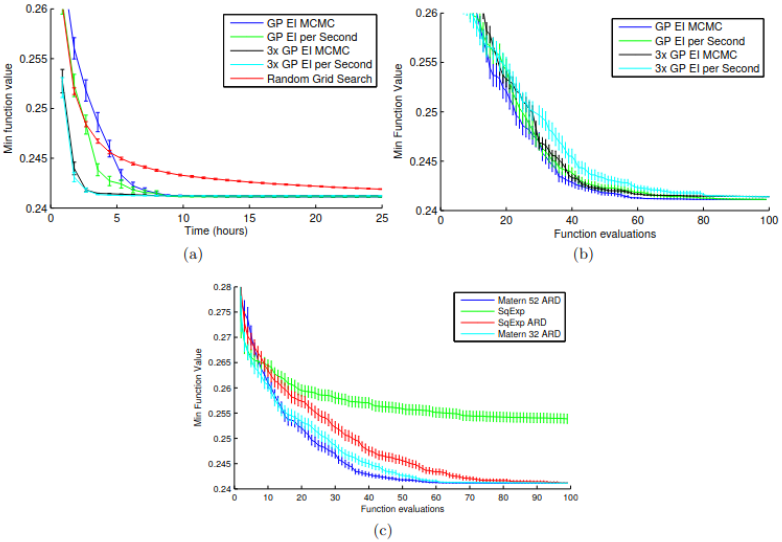
- Flexibility.
- Probabilistic model, cheaper than exhaustive search.

Cons: cubic complexity on the sample number.

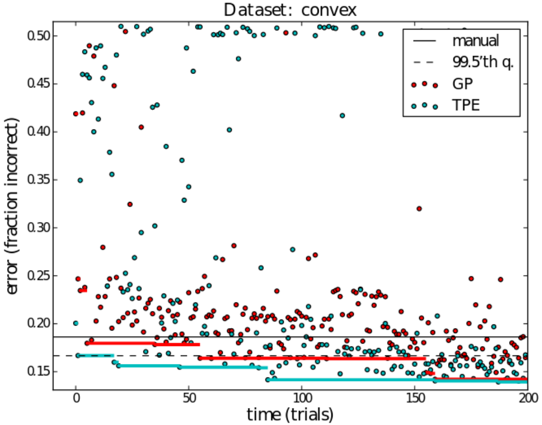


Shahriari et. al, 2016. GP example.

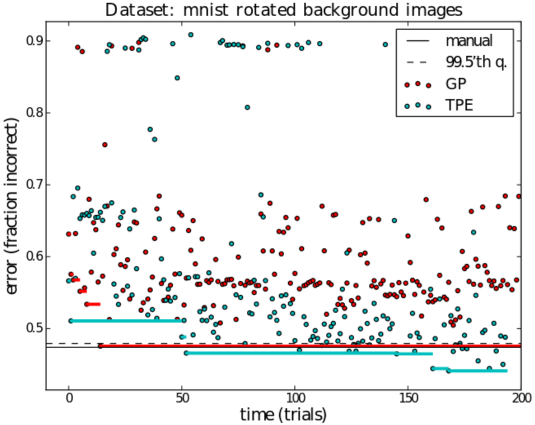
Hyperparameter optimization



TPE vs GP

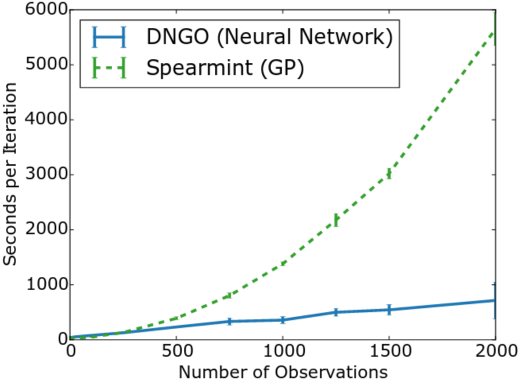


Dataset: convex
error (fraction incorrect)
time (trials)
manual
99.5'th q.
GP
TPE
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
50
100
150
200



Dataset: mnist rotated background images
error (fraction incorrect)
time (trials)
manual
99.5'th q.
GP
TPE
0.9
0.8
0.7
0.6
0.5
0
50
100
150
200

GP: complexity challenge



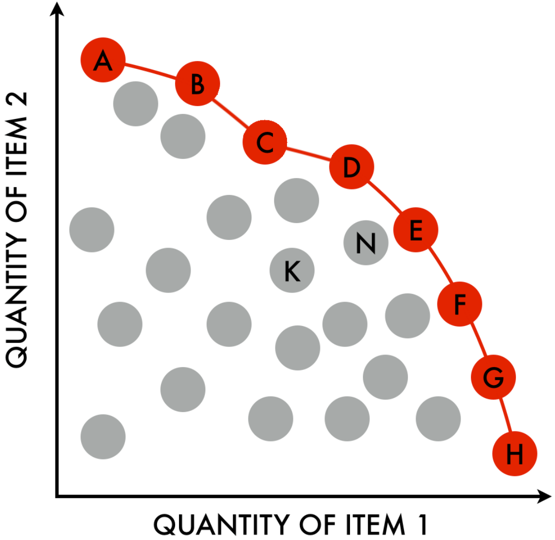
(Snoek, 2015)

Muilti-objective optimization

Can we use multiple criteria for optimization task?

Muilti-objective optimization

Can we use multiple criteria for optimization task?



Next hometask

- The projects are assigned
- Please fill the activities
- Basic code activity is renamed $\rightarrow$ Proof of concept
  - A very simple demonstration of the problem is expected (with a very naive baselines)
  - The dataset, basic plots and evulation criteria must be selected
- The project schedule is on the page course

Next hometask: presentation

- For all teams and their members: make presentations of your projects
- The presentation must cover
  - Project description (maybe more detailed than I gave to you)
  - Name of the project library
  - Scheme of the project (what will be the classes, how it will be integrated, what's the stack)
  - Brief algorithm description (from 1 to 4 slides for all the algorithms, other people must be able to understand the idea of all the algorithms)
  - Idea for demo/basic code
- Time limit: 10 min

Next hometask: for people who are wrapping the library

- Create a repository in intsystems Please make it w.r.t. to the manual
- Think about your project stack (DS libraries, testing, docummentation)

Next hometask: for people who are planning the library

- Create a document in the repository with the following information (the same as in the presentation, but maybe with more details)
  - Project name
  - Architecture of the project: what classes must be implemented? How they should interact?
  - Describe all the public functions and class methods you are planing to implement, with annotations.
  - What are the libraries you are planning to use and/or integrate?
- In perfect case, the member who is implementing the algorithm can write the code just by your architecture description.
- Note, the document can be improved/changed in the future, but I will score you and other members of the team on the correspondence of the proposed structure and the final algorithm implementation.

References

- Bishop C. M. Pattern recognition //Machine learning. – 2006. – Т. 128. – №. 9.
- Bakhteev O. Y., Strijov V. V. Comprehensive analysis of gradient-based hyperparameter optimization algorithms //Annals of Operations Research. – 2020. – Т. 289. – №. 1. – С. 51-65.
- Graves A. Practical variational inference for neural networks //Advances in neural information processing systems. – 2011. – Т. 24.
- Bergstra et al., Random Search for Hyper-Parameter Optimization, 2012
- Bobak Shahriari et. al, Taking the Human Out of the Loop: A Review of Bayesian Optimization, 2016
- Daniel McDuff, Gaussian Processes, https://courses.media.mit.edu/2010fall/mas622j/ProblemSets/slidesGP.pdf
- Chris Williams, Gaussian Processes for Machine Learning, https://www.newton.ac.uk/files/seminar/20070809140015001-150844.pdf
- Ed Snelson, utorial: Gaussian process models for machine learning, https://mlg.eng.cam.ac.uk/tutorials/06/es.pdf
- Snoek J., Larochelle H., Adams R. P. Practical bayesian optimization of machine learning algorithms //Advances in neural information processing systems. – 2012. – Т. 25.
- Snoek J. et al. Scalable bayesian optimization using deep neural networks //International conference on machine learning. – PMLR, 2015. – С. 2171-2180.
- modAL: A modular active learning framework for Python3
- NeurPY