Yandex

From Hyper Parameter Optimization Towards the Best Model Selection

MLHEP 2017

Andrey Ustyuzhanin

Predictive Model in a Nutshell

- Blackbox: $F(\theta, \mathbf{x}) = y$, where $\mathbf{x}$, y are taken from dataset $D = \{\mathbf{X}, \mathbf{y}\}$
  - reads input data
  - outputs predictions
  - could be of different nature
  - depends on vector of **hyper parameters** θ, selected a priori
- Figure of Merit
  - Efficiency at certain working point or Area under ROC curve or logloss
  - usually non-convex and non-differentiable over θ
  - ...

"This just isn't doing it for me. Could we go back to using the crystal ball?"

General Procedure to search for the best model

Specify FOM Q (i.e. ROC AUC)
Split D into D_{train}, D_{test}

- Do not touch D_{test} until very latest stage (comparison)

Split D_{train} into K folds

- K should be large enough so the sample of $(K\text{-}1)$ folds would not differ significantly from D_{train} (i.e. $\mathrm{K} > 3$)

Sample θ_i from $\boldsymbol{\Theta}^{\mathrm{n}}$, train model F_i^j (j - refers to fold number)

- apply those models to corresponding folds,
- record mean & variation of the $\mathrm{Q}(\boldsymbol{\theta}_i)$ on the folds
- **repeat sampling** until satisfied, fix $\theta^* = \mathrm{argmax}\ (\mathrm{Q}(\boldsymbol{\theta}))$

Train model $F(\theta^*)$ on full D_{train}, apply to D_{test}, calculate Q

Hyper Parameter Sampling problem

- Θ^n is large:
  - BDT para
    BoostTyp
    UseNvar
    tmva.sou
  - Evaluatic
  - Some pa
- What is the
  top model (function is non-convex, non-smooth)?

Popular and easy approaches

Expert guess
Grid search - good coverage, slow
Manual coordinate descent
Random search - fast, scalable

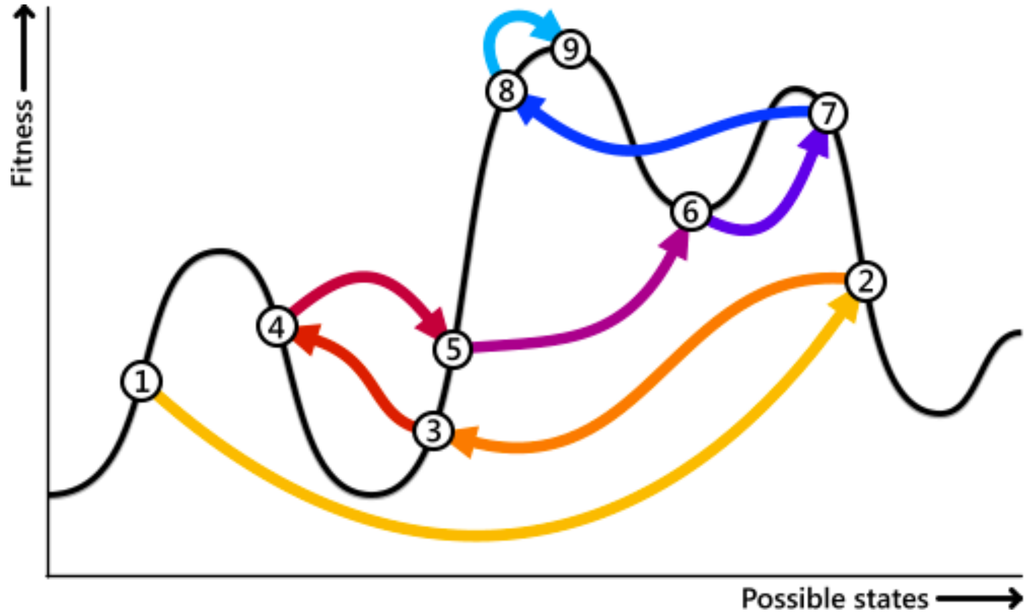
> if at least 5% of the points on the grid yield a close-to-optimal solution, then random search with 60 trials will find that region with high probability:
>
> $1 - (1 - 0.05)^n > 0.95 \rightarrow n >= 60$

"Random Search for Hyper-Parameter Optimization." James Bergstra and Yoshua Bengio. *Journal of Machine Learning Research*, 2012

https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/5/hyperparameter-tuning

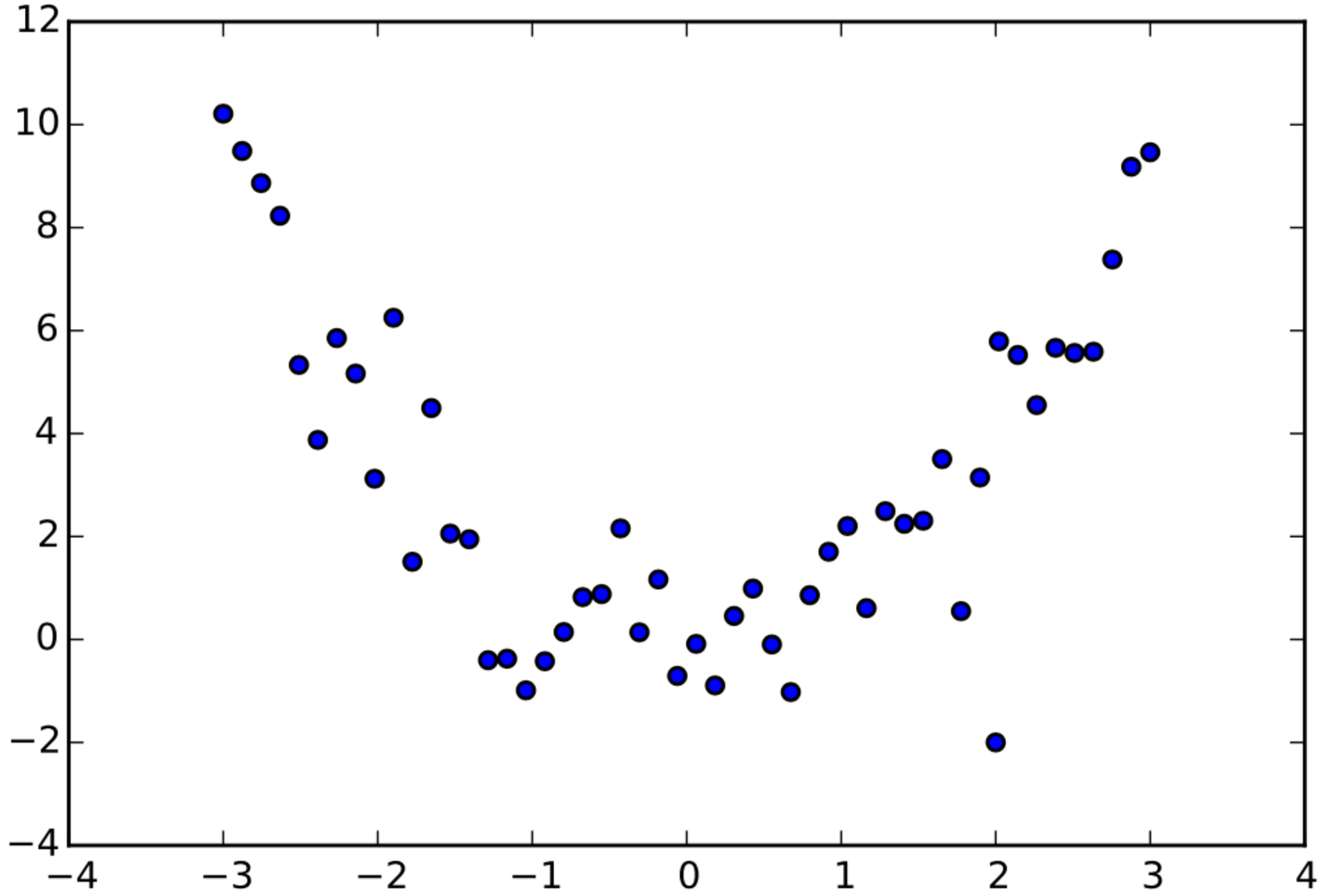
Gradient-free and global optimisations

- Generic algorithm
- Simulated annealing
- Nelder-Mead, http://bit.ly/zheng-bilenko
- Response surface

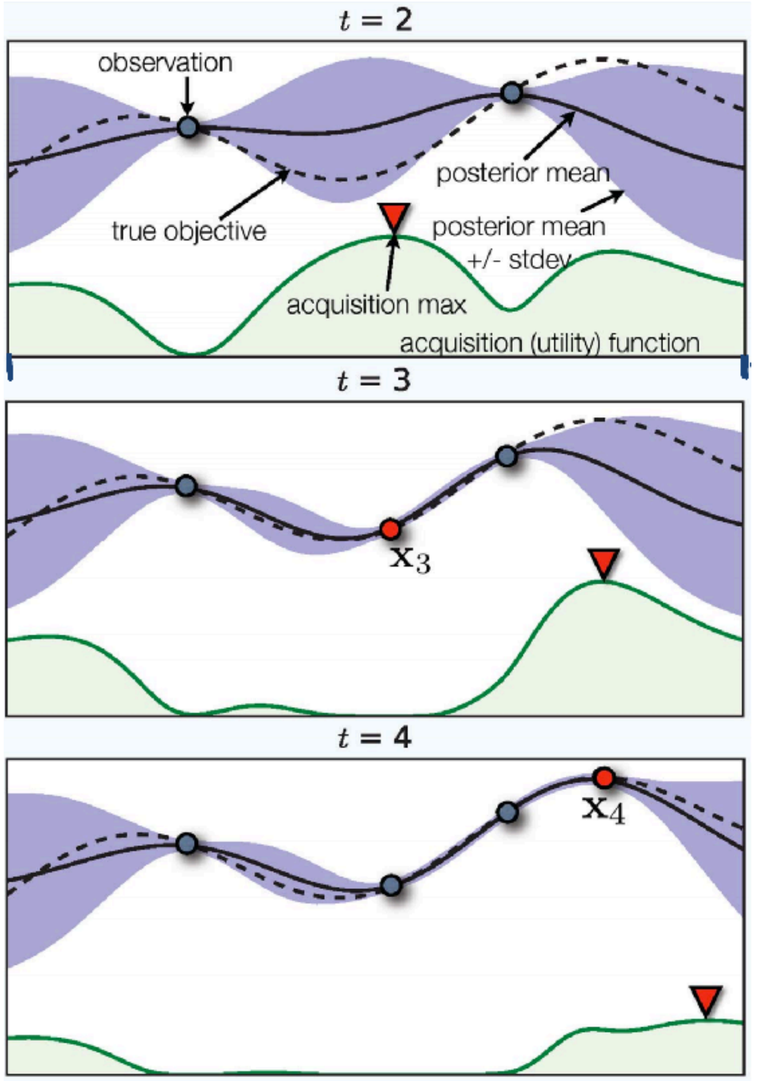


Simulated annealing

Stochastic functions



Bayesian Optimization



Algorithm 1 Bayesian Optimization

1: **for** $t = 1, 2, \ldots$ **do**
2: Find $\mathbf{x}_{t+1} \in \mathbb{R}^D$ by optimizing the acquisition function u: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} u(\mathbf{x} | \mathcal{D}_t)$.
3: Augment the data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1}))\}$
4: **end for**

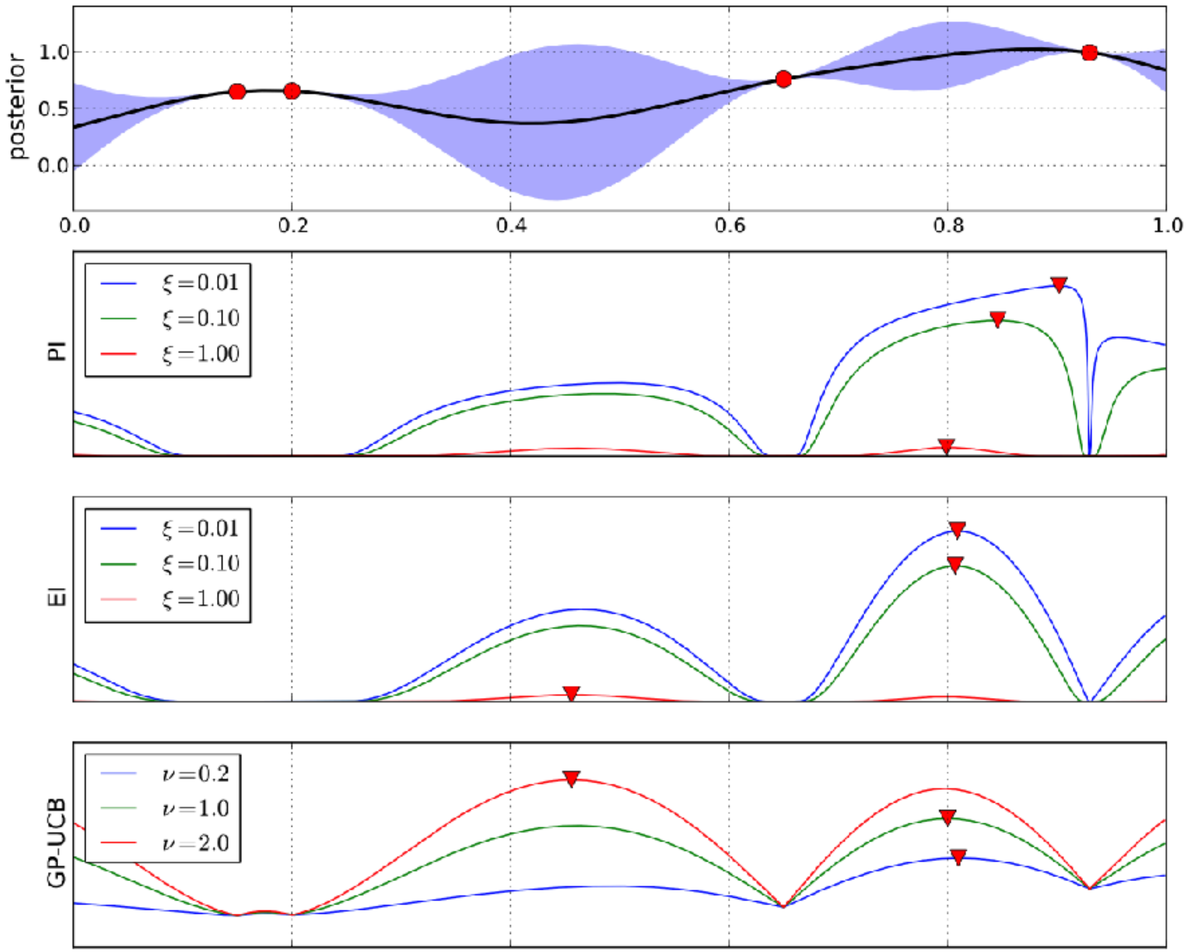
Acquisition function

Exploration: Seek places with high variance

Exploitation: Seek places with low (high) mean

The acquisition function balances Exploration and Exploitation for our surrogate optimization to determine the next evaluation point.

Acquisition functions examples



Built surrogate

Probability of improvement (PI)

$$PI(x) = P(f(x) \geq f(x^+) + \xi)$$

Expected improvement (EI)

$$EI(x) = \mathbb{E}(\max\{0, f(x) - f(x^+) - \xi\})$$

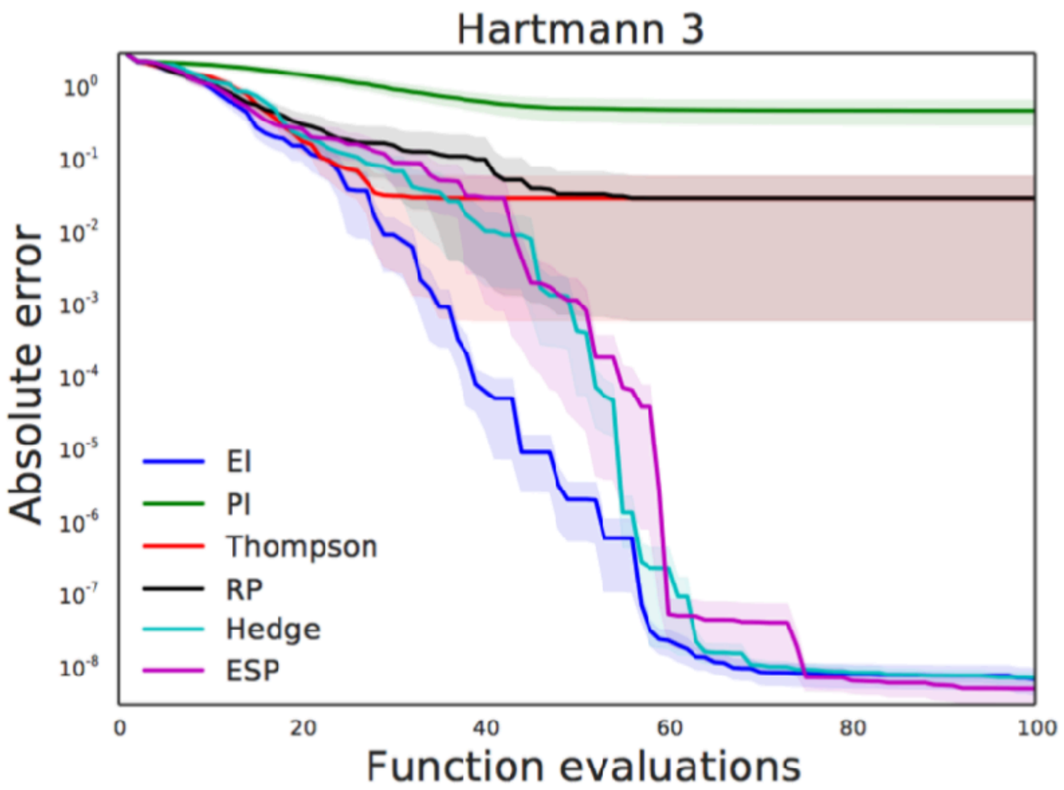
Upper Confidence Bound (UCB)

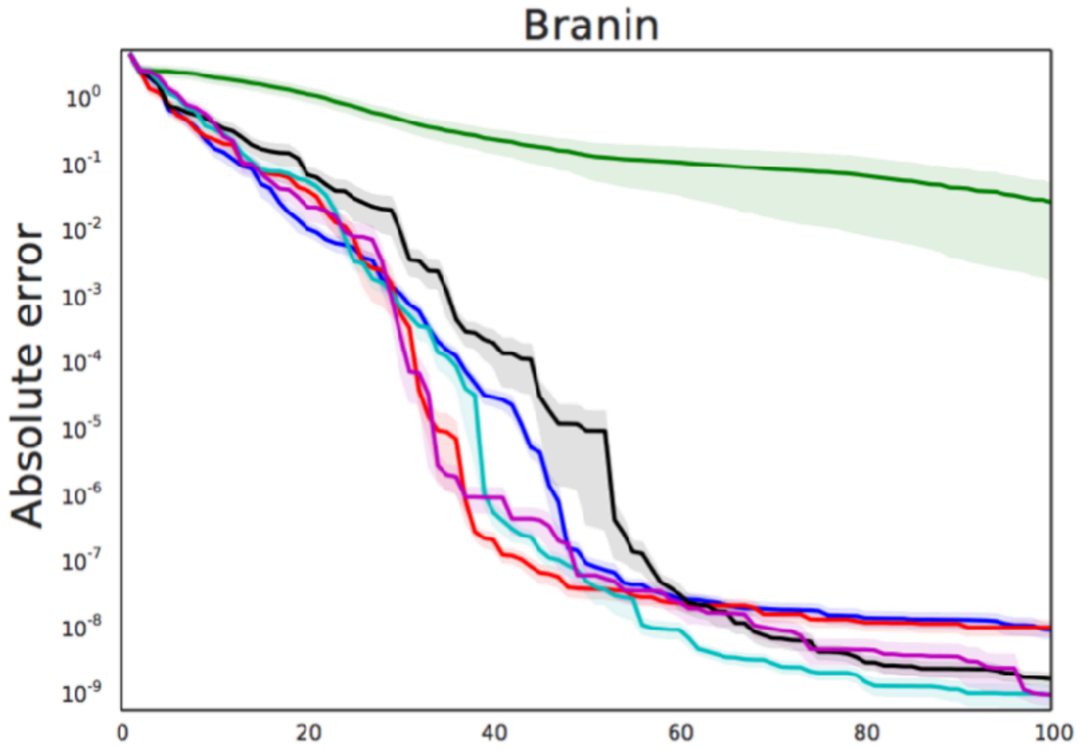
$$UCB(x) = \mathbb{E}f(x) + \nu\sigma(x)$$

Hutter, Frank, Jörg Lücke, and Lars Schmidt-Thieme. "Beyond Manual Tuning of Hyperparameters." 2015.

Shahriari, Bobak, et al. "Taking the Human Out of the Loop: A Review of Bayesian Optimization." 2015.

Comparison of acquisition functions





Shahriari, Bobak, et al. "Taking the Human Out of the Loop: A Review of Bayesian Optimization." 2015.

Common algorithm design

- Initial design — evaluate the black box in some points
  - Best from previous experiments
  - Ask for experts (hand
  - Random
  - Grid
  - Several optimal design criteria https://en.wikipedia.org/wiki/Optimal_design
- Adaptive design
  1. Build a surrogate
  2. Find the argmax of Expected Improvement
  3. Evaluate the black box in this point
  4. Go to step 2

Types of surrogates

- Gaussian Processes (GP),

  http://www.robots.ox.ac.uk/~mebden/reports/GPtutorial.pdf

- Tree of Parzen Estimators (TPE),

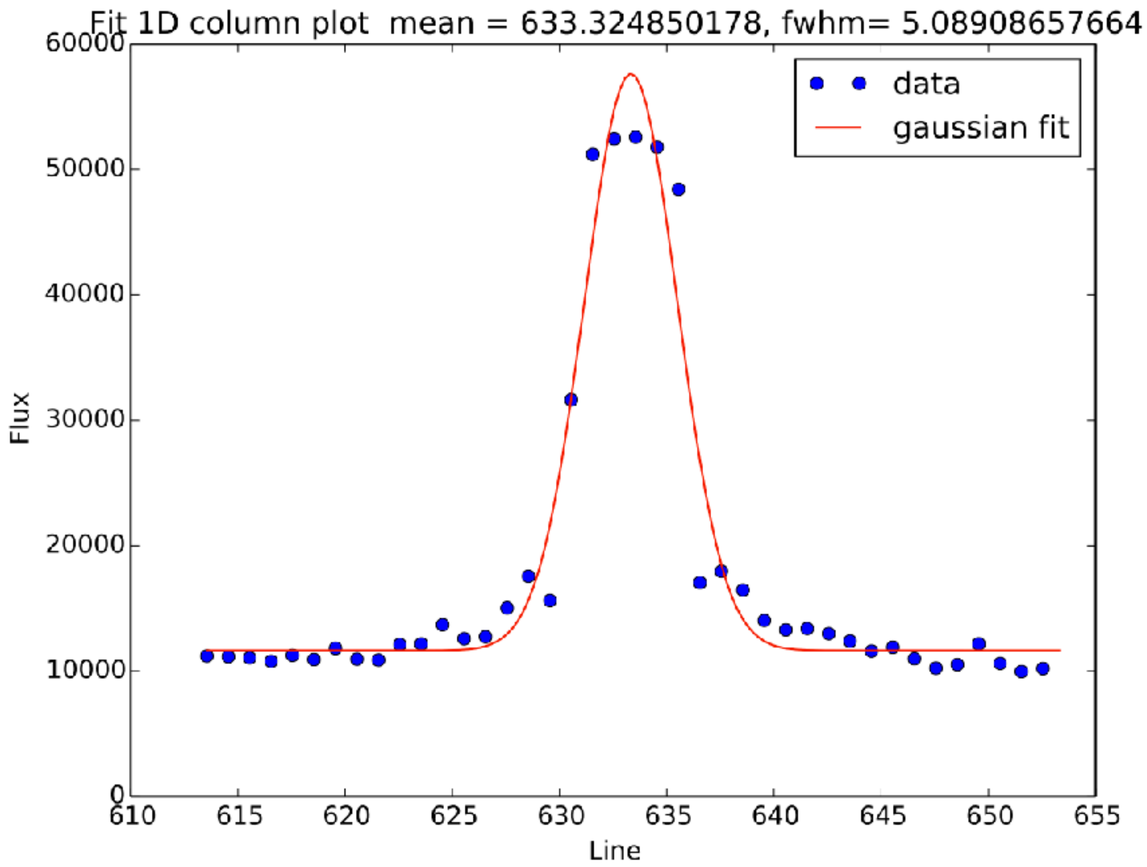
  http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf

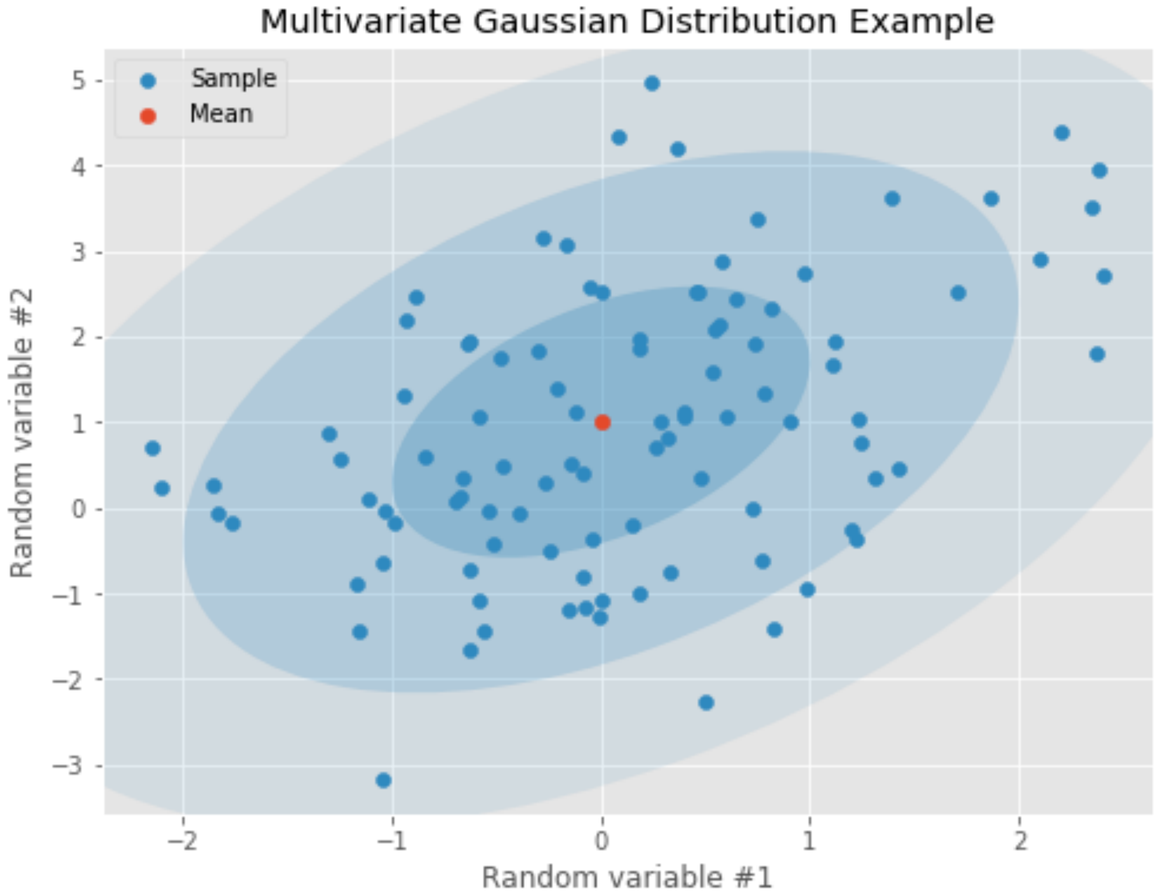
- Random Forest (Sequential Model-based Algorithm Configuration (SMAC)),

  https://github.com/automl/pysmac

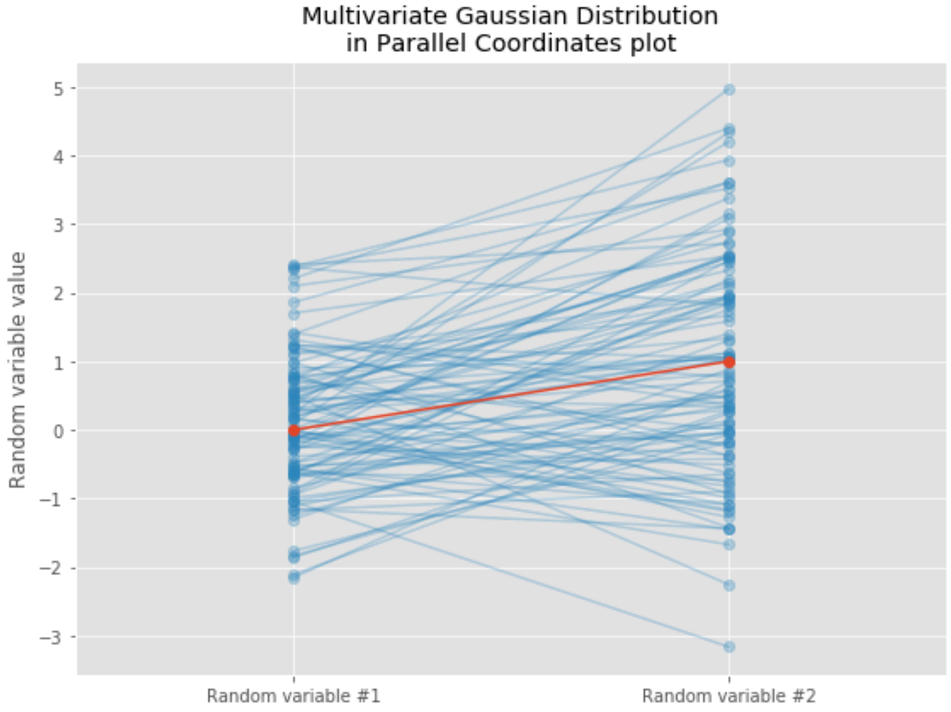
- Gradient Boosted Decision Trees

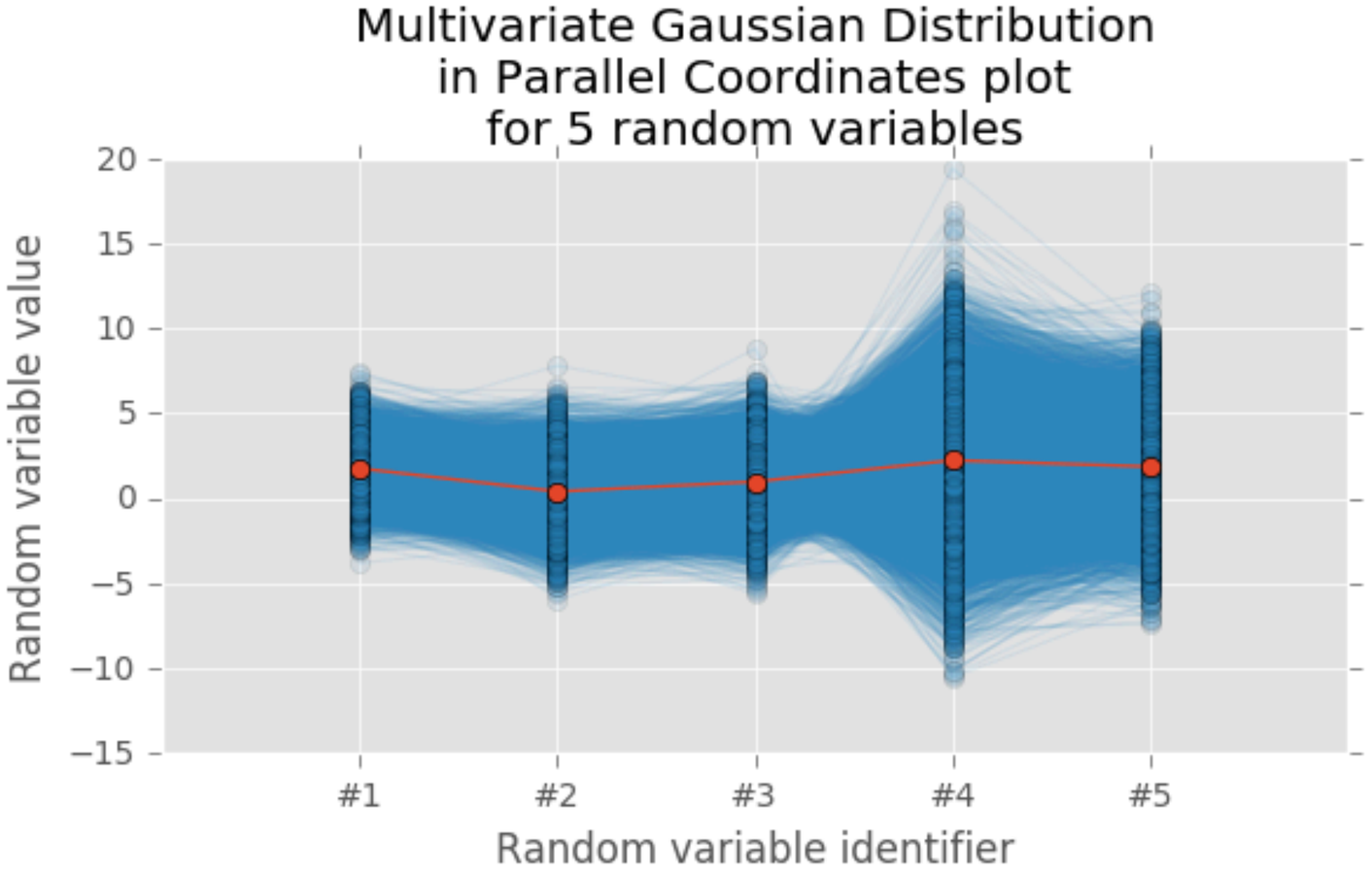
Gaussian Process: from 1D, 2D ad infinitum





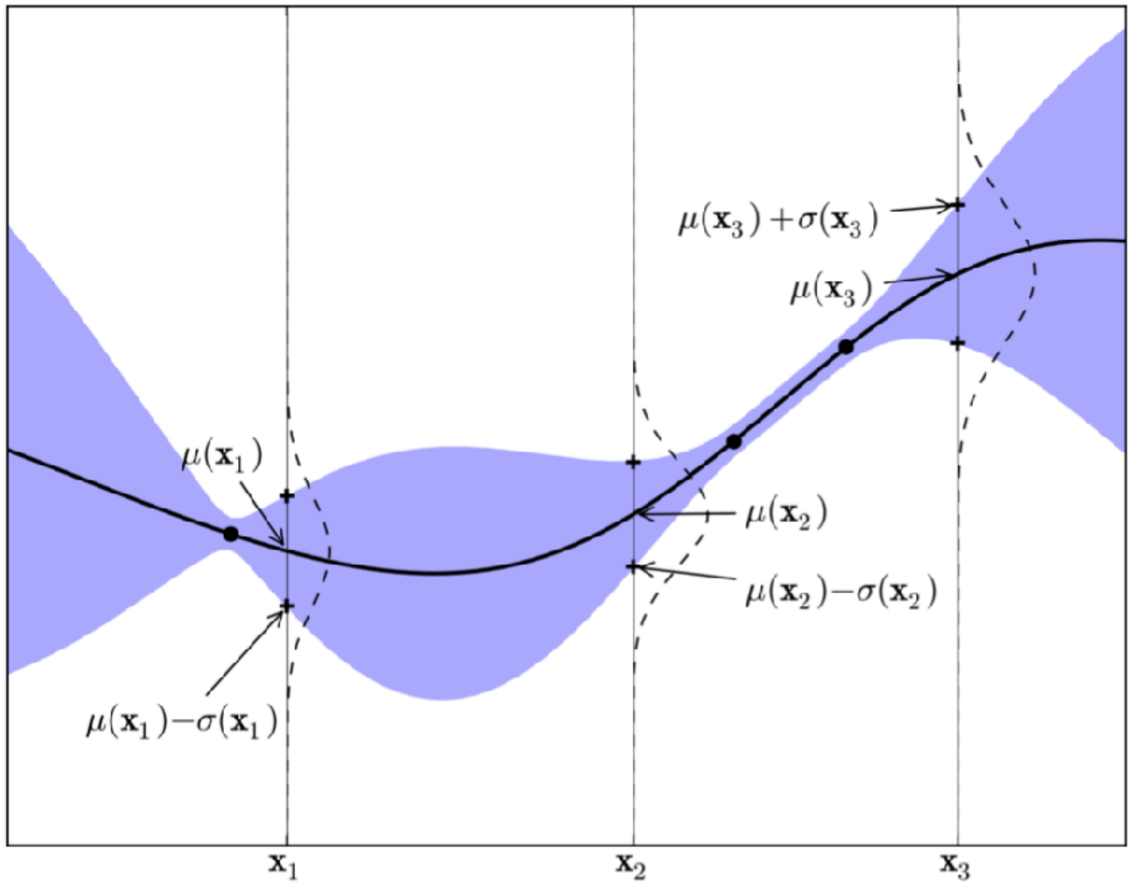
… ad infinitum. Intuition





Gaussian Process

Gaussian processes (GPs) are probability distributions over functions for which inference task is tractable. They can be seen as a generalisation of the Gaussian probability distribution to the space of functions. That is a multivariate Gaussian distribution defines a distribution over a finite set of random variables, a Gaussian process defines a distribution over an infinite set of random variables (for example the real numbers). (Rasmussen and Williams, 2006).



Gaussian process regression: formulas

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \end{bmatrix}\right)$$

GP is analogous to a function, but instead of returning a scalar f(x) for an arbitrary x, it returns **the mean** and **the variance** of a normal distribution over the possible values of f at x.

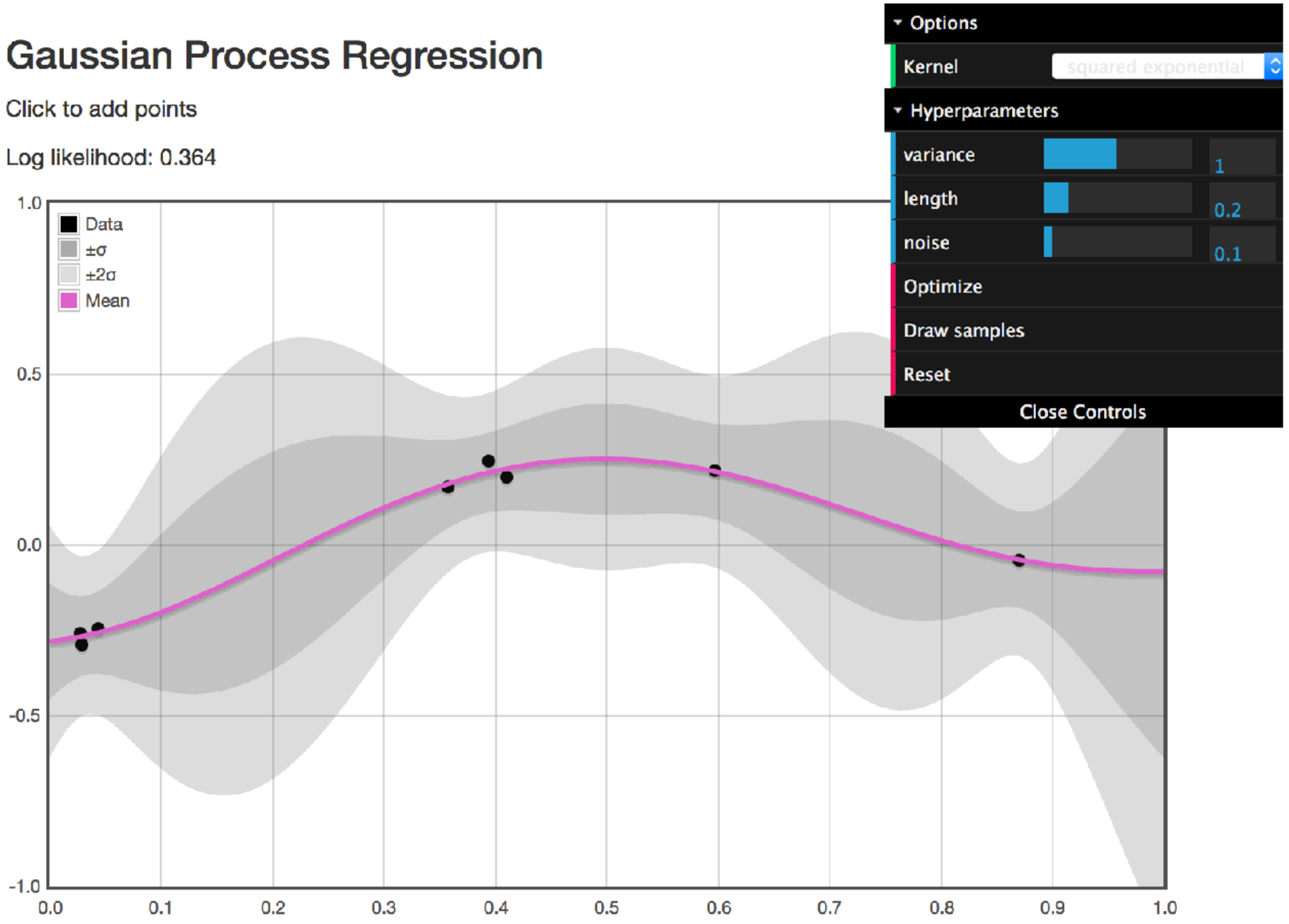
$$P(f_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}\left(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1})\right)$$

$$\begin{aligned} \mu_t(\mathbf{x}_{t+1}) &= \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:t} \\ \sigma_t^2(\mathbf{x}_{t+1}) &= k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} \end{aligned}$$

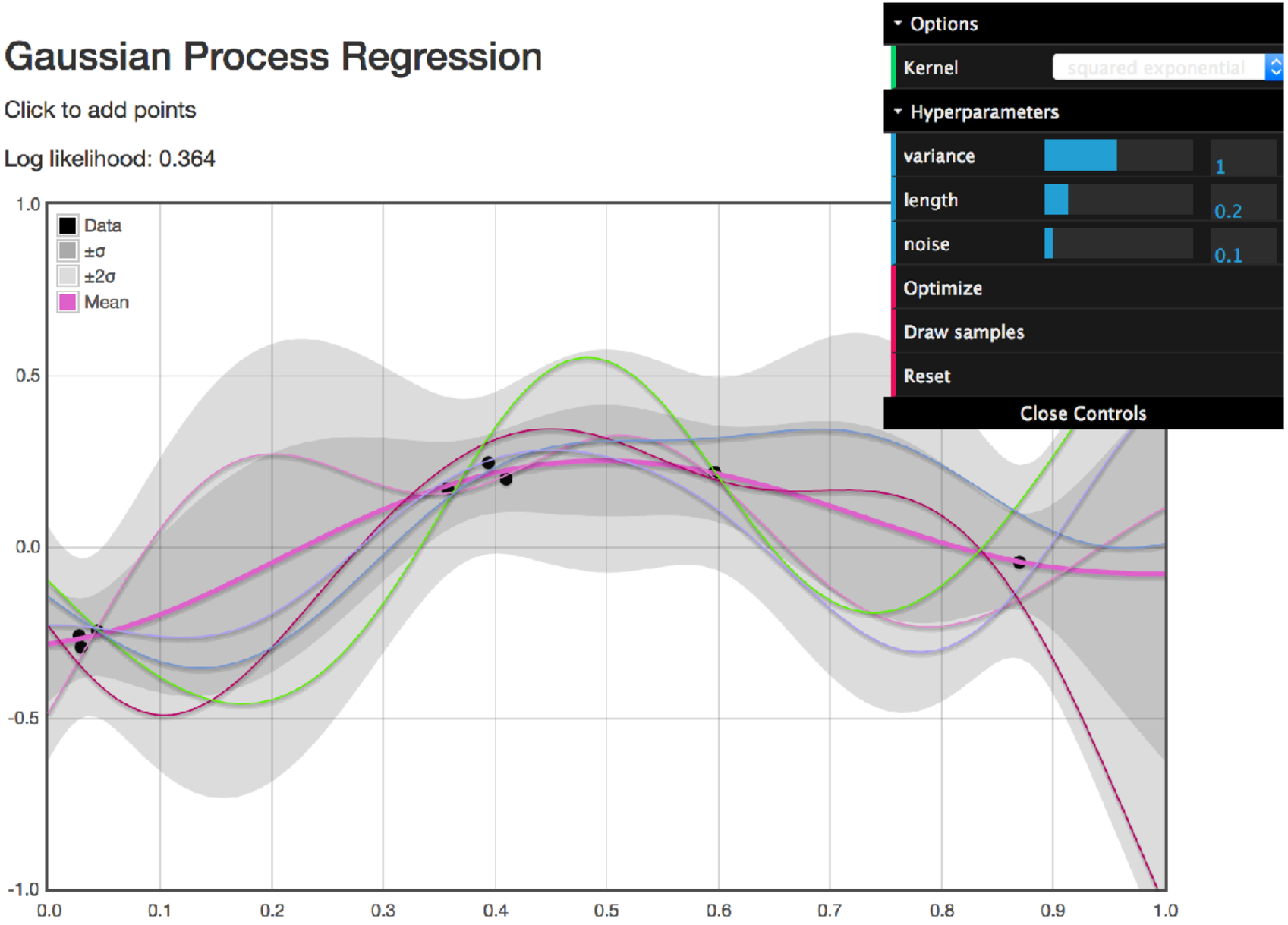
$$\mathcal{D}_{1:t} = \{\mathbf{x}_{1:t}, y_{1:t}\}$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \dots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}$$

$$\mathbf{k} = \begin{bmatrix} k(\mathbf{x}_{t+1}, \mathbf{x}_1) & k(\mathbf{x}_{t+1}, \mathbf{x}_2) & \cdots & k(\mathbf{x}_{t+1}, \mathbf{x}_t) \end{bmatrix}$$



http://chifeng.scripts.mit.edu/stuff/gp-demo/



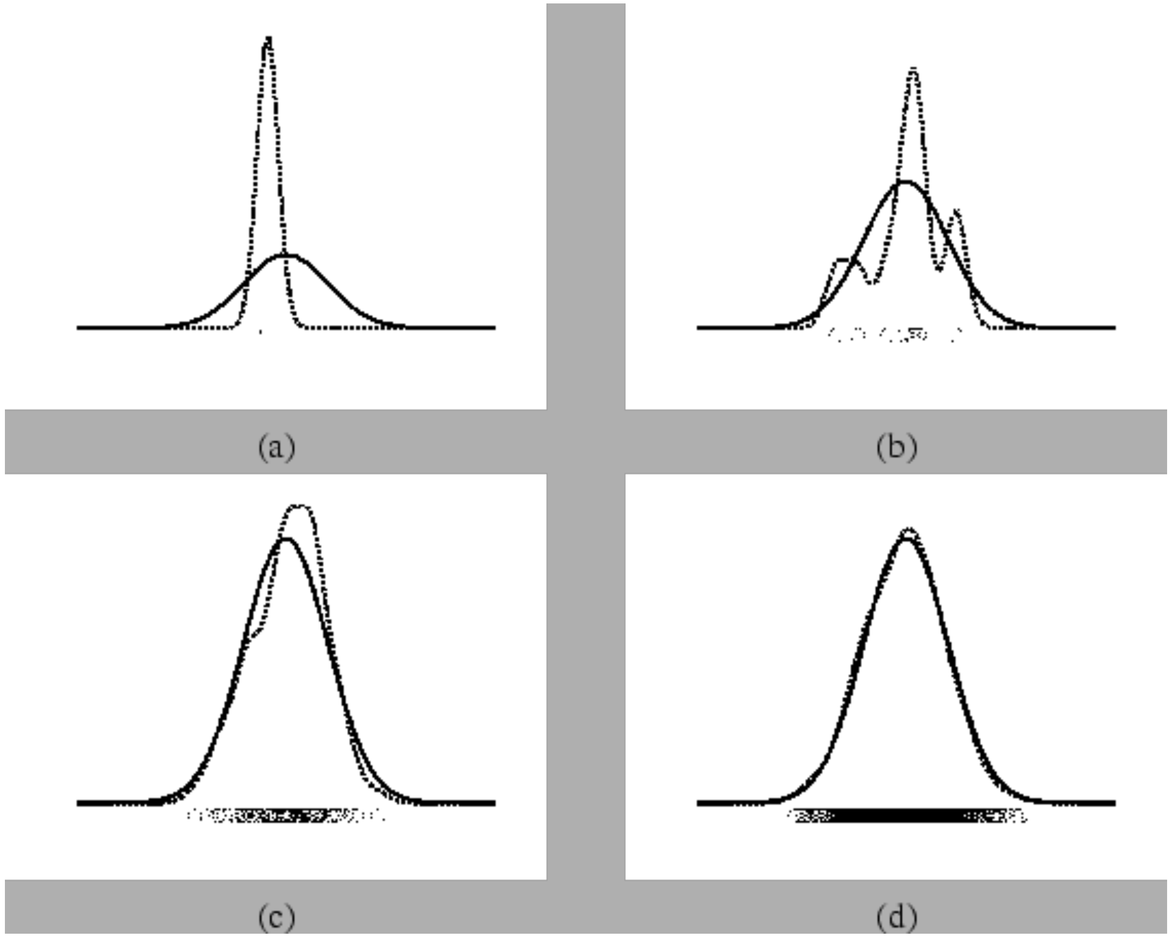
http://chifeng.scripts.mit.edu/stuff/gp-demo/

Tree-based Parzen Estimator

Estimation of density based on Parzen window density estimators (named after Emanuel Parzen, 1962).

$$P(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n^d}K\left(\frac{x - x_i}{h_n}\right),$$

K(x) - window function (kernel)
h_n - window width, hyperparameter

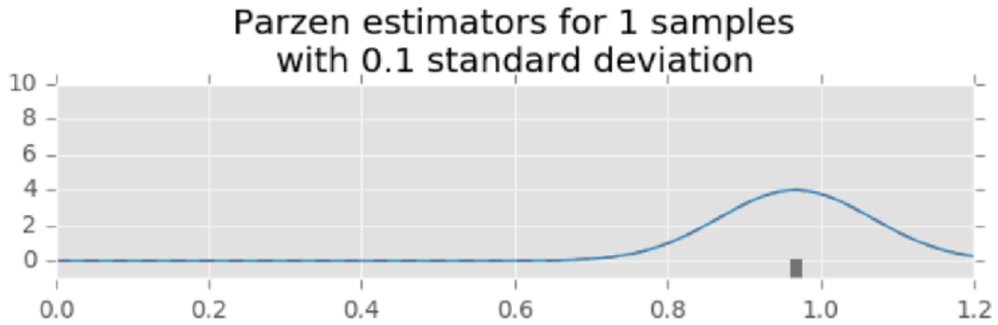


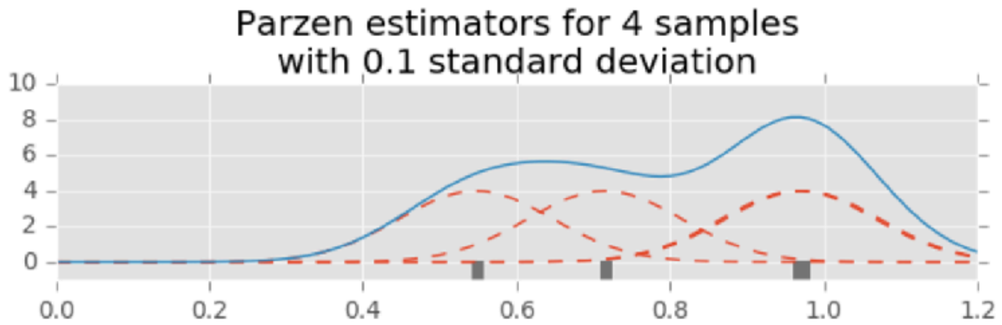
The Parzen-window PDF estimate (dotted curve), for a Gaussian PDF, with a ***Gaussian*** kernel and a sample size of (a) 1, (b) 10, (c) 100, and (d) 1000. The circles indicate the observations in the sample.

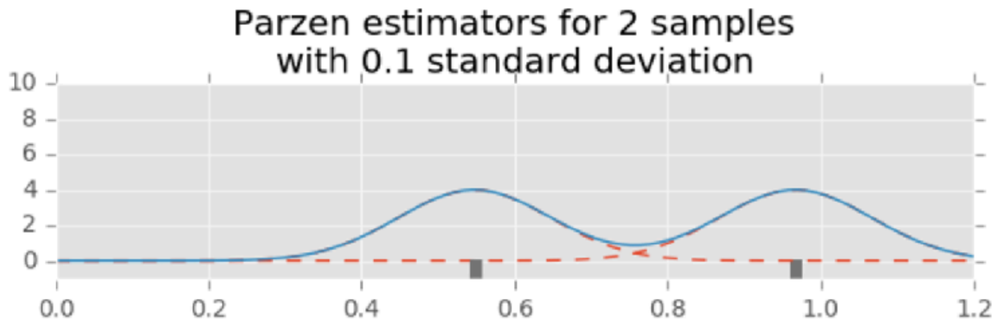
E. Parzen.
On the estimation of a probability density function and the mode.
Annals of Math. Stats., 33:1065-1076, 1962

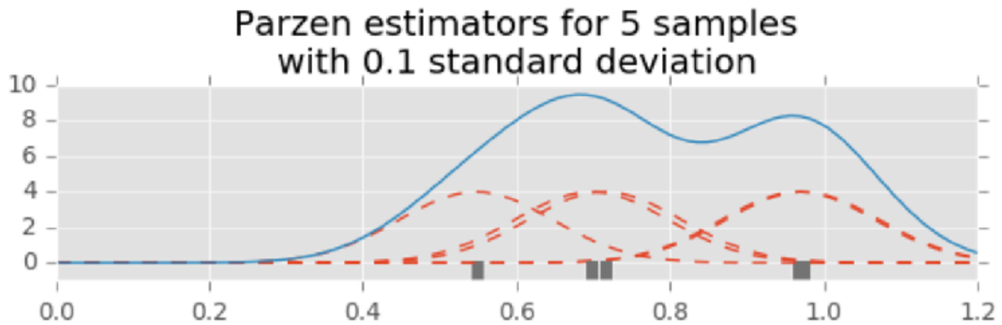
https://www.cs.utah.edu/~suyash/Dissertation_html/node11.html

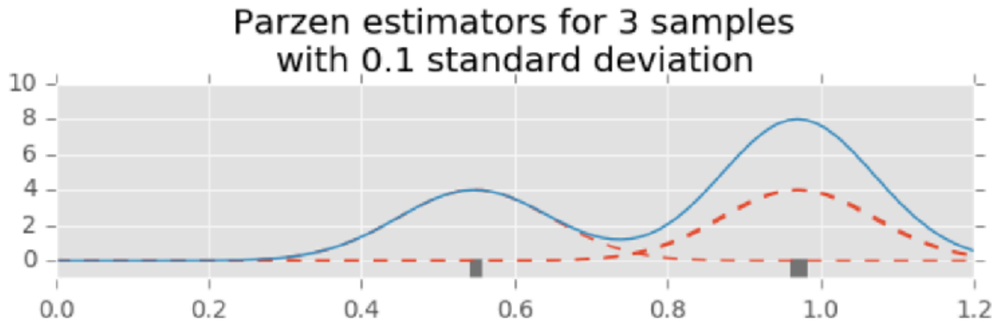
Parzen estimators dynamics

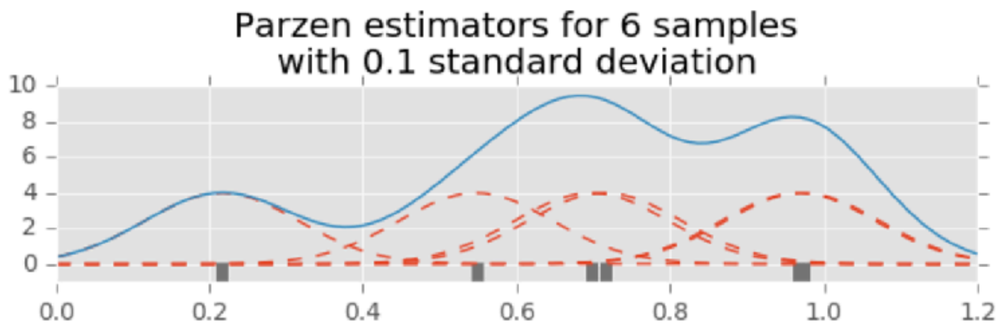












Surrogate optimization comparison

dataset	WEKA-DEF	RANDOM			SMAC			TPE		
		PREV	NEW	FULL	PREV	NEW	FULL	PREV	NEW	FULL
abalone	73.18	74.88	**72.92** ↑	73.76 ↑	73.51	73.41 ↑	73.16 ↑	72.94	73.04	73.26
amazon	**28.44**	41.11	39.22 ↑	58.94	33.99	36.26	49.90	36.59	35.69 ↑	62.52
car	0.7700	0.0100	0.1300	1.8400	0.4000	0.0526 ↑	0.1958 ↑	0.1800	**0.0075** ↑	0.0484 ↑
cifar10	64.27	69.72	58.23 ↑	67.81 ↑	61.15	55.54 ↑	69.99	66.01	**54.88** ↑	70.06
cifar10small	65.91	66.12	59.84 ↑	72.93	56.84	57.87	72.64	57.01	**56.41** ↑	72.58
convex	25.96	31.20	24.75 ↑	36.87	23.17	**21.31** ↑	31.05	25.59	22.62 ↑	34.88
dexter	8.89	9.18	8.29 ↑	12.59	7.49	7.31 ↑	10.14	8.89	**6.90** ↑	7.99 ↑
dorothea	6.96	5.22	5.27	5.37	6.21	**5.12** ↑	5.51 ↑	6.15	5.25 ↑	5.15 ↑
germancredit	27.33	29.03	**25.40** ↑	26.88 ↑	28.24	25.42 ↑	26.68	27.54	25.49 ↑	26.61 ↑
gisette	2.81	4.62	2.28 ↑	3.83 ↑	**2.24**	2.34	2.85	3.94	2.37 ↑	3.08 ↑
kddcup09app	1.7405	1.74	**1.72** ↑	2.05	1.7358	1.74	1.74	1.7381	1.74	1.74
krvskp	0.31	0.58	0.34 ↑	0.39 ↑	0.31	**0.23** ↑	0.39	0.54	0.36 ↑	0.33 ↑
madelon	21.38	24.29	19.10 ↑	24.48	21.56	**16.80** ↑	18.26 ↑	21.12	16.91 ↑	18.35 ↑
mnist	5.19	5.05	4.00 ↑	13.92	**3.64**	4.10	22.70	12.28	3.96 ↑	25.48
mnistr	63.14	66.4	57.16 ↑	69.67	57.04	**54.86** ↑	67.03	70.20	56.31 ↑	70.04 ↑
secom	8.09	8.03	7.88 ↑	8.20	8.01	7.87 ↑	7.87 ↑	8.10	**7.84** ↑	7.87 ↑
semeion	8.18	6.10	**4.78** ↑	8.27	5.08	5.10	5.46	8.26	4.91 ↑	6.31 ↑
shuttle	0.0138	0.0157	0.0071 ↑	0.0219	0.0130	0.0070 ↑	0.0075	0.0145	**0.0069** ↑	0.0077 ↑
waveform	14.40	14.27	14.26 ↑	14.28	14.42	14.17 ↑	**13.99** ↑	14.23	14.34	14.04 ↑
wineqw	37.51	34.41	32.99	36.72	33.95	**32.90** ↑	34.14	33.56	32.93 ↑	34.09
yeast	40.45	43.15	37.68 ↑	40.86 ↑	40.67	**37.60** ↑	39.02 ↑	40.10	37.89 ↑	38.92 ↑

https://www.slideshare.net/draxus/automating-machine-learning-is-it-feasible-62661182

Popular optimization libraries, Python mostly

hyperopt, https://github.com/hyperopt/hyperopt, TPE
SMAC3, https://github.com/SMAC
GPyOpt (https://github.com/SheffieldML/GPyOpt)
spearmint, GP,
scikit-optimize, GP, GBDT, RandomForest
bayesopt (C++, https://bitbucket.org/rmcantin/bayesopt/)

Difference:

sequential/distributed/parallel execution
surrogate family
support from the developers

Hyperopt quick intro

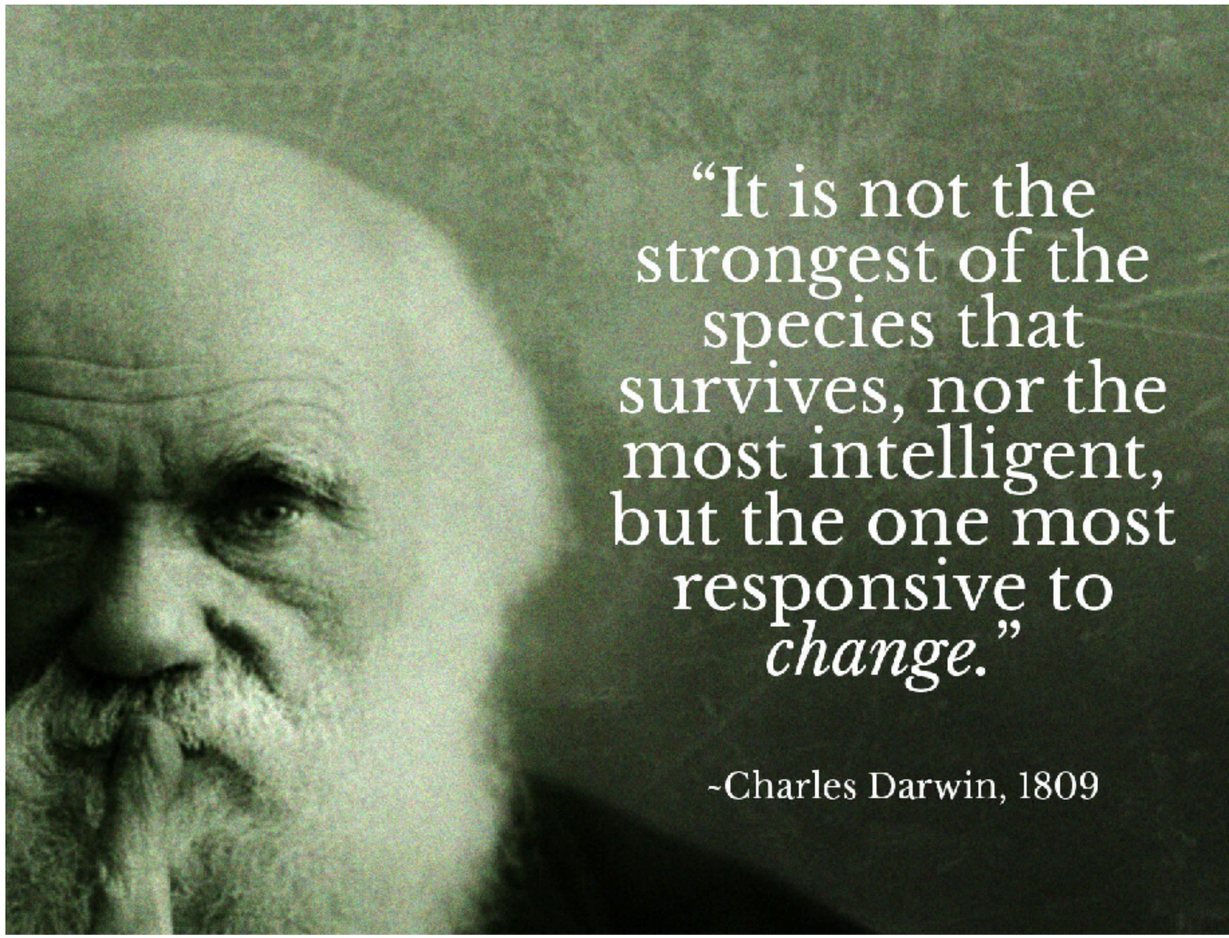
```python
# define an objective function
def objective(args):
    case, val = args
    if case == 'case 1':
        return val
    else:
        return val ** 2

# define a search space
from hyperopt import hp
space = hp.choice('a',
    [
        ('case 1', 1 + hp.lognormal('c1', 0, 1)),
        ('case 2', hp.uniform('c2', -10, 10))
    ])

# minimize the objective over the space
from hyperopt import fmin, tpe
best = fmin(objective, space, algo=tpe.suggest, max_evals=100)

print best
# -> {'a': 1, 'c2': 0.01420615366247227}
print hyperopt.space_eval(space, best)
# -> ('case 2', 0.01420615366247227}
```

```python
from hyperopt import hp
space = hp.choice('classifier_type', [
    {
        'type': 'naive_bayes',
    },
    {
        'type': 'svm',
        'C': hp.lognormal('svm_C', 0, 1),
        'kernel': hp.choice('svm_kernel', [
            {'ktype': 'linear'},
            {'ktype': 'RBF', 'width': hp.lognormal('svm_rbf_width', 0, 1)},
            ]),
    },
    {
        'type': 'dtree',
        'criterion': hp.choice('dtree_criterion', ['gini', 'entropy']),
        'max_depth': hp.choice('dtree_max_depth',
            [None, hp.qlognormal('dtree_max_depth_int', 3, 1, 1)]),
        'min_samples_split': hp.qlognormal('dtree_min_samples_split', 2, 1, 1),
    },
    ])
```

Model Selection != hyper parameter tuning

Evaluate several models

- Wrap all models to the same interface
- Specify search space for each model
- Monitor intermediate results
- Handle parallel execution
- Comparison



Model Gym

- Reference implementation of model selection framework
- Main classes

  - Model
    - XGBoostModel (XGBoost)
    - LGBModel (LightGBM)
  - Trainer
    - hyperopt
  - Tracker
    - file
    - MongoDB

https://github.com/yandexdataschool/modelgym

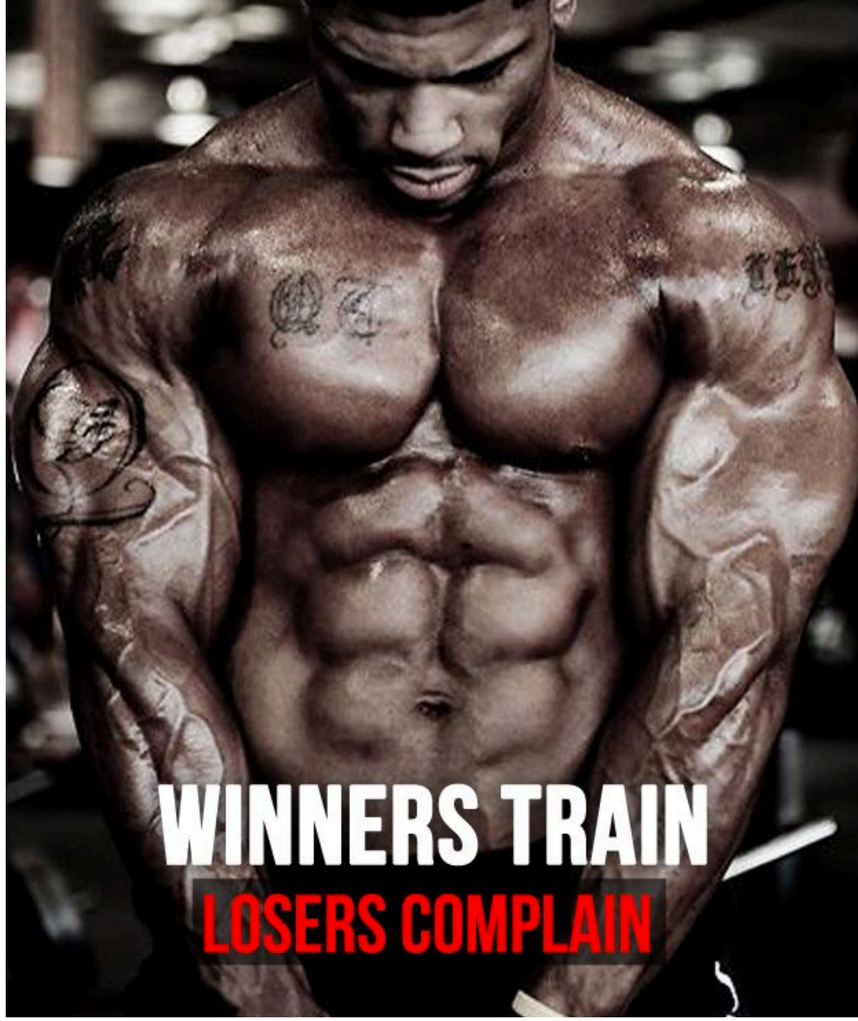
Model Gym features

Out of the box

- Support for Docker
- Support for Jupyter
- Runs online (everware)

Roadmap

- TMVA (anyone?)
- Support for distributed execution
- Works well with CI



Summary

〉 We usually need to optimize stochastic functions;

〉 Use surrogate models to speed up things;

〉 It takes knowledge & efforts of many to train a single winner. Automate it!

〉 Model gym - a reference for automation, work is ongoing;

PS Random search is much better than you thought!

References

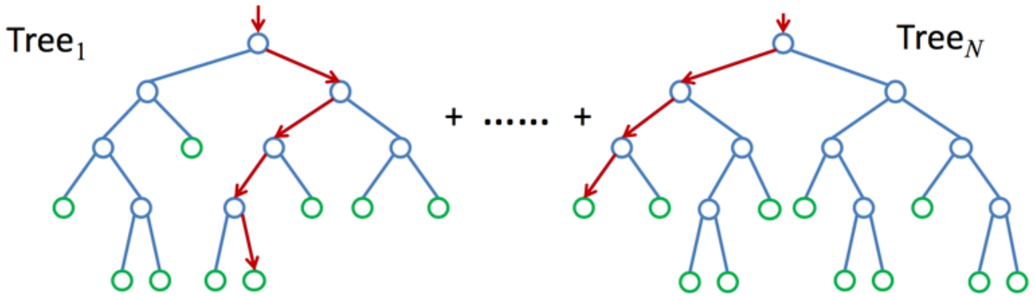
- Hutter, Frank, Jörg Lücke, and Lars Schmidt-Thieme. "Beyond Manual Tuning of Hyperparameters.” 2015.
- http://www.cs.ubc.ca/~hutter/papers/13-BayesOpt_EmpiricalFoundation.pdf
- http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf
- Brochu, Eric, Vlad M. Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning." arXiv preprint arXiv:1012.2599 (2010).
- https://github.com/hyperopt/hyperopt/wiki/FMin
- https://bayesopt.github.io/papers/2015/gonzalez-batch.pdf

Network structure optimzation

- Ensemble Bayesian Optimization, https://arxiv.org/pdf/1706.01445.pdf
- Hyperparameter Optimization: A Spectral Approach, https://arxiv.org/pdf/1706.00764v1.pdf
- Automated deep neural network design via genetic programming, https://github.com/joeddav/devol

Backup

Sequential Model-based Algorithm Configuration (SMAC)



- Mean in each point — prediction of Random Forest (RF)
- Variance in each point — variance of predictions of separate trees from RF

Hutter, Frank, Jörg Lücke, and Lars Schmidt-Thieme. "Beyond Manual Tuning of Hyperparameters." 2015.