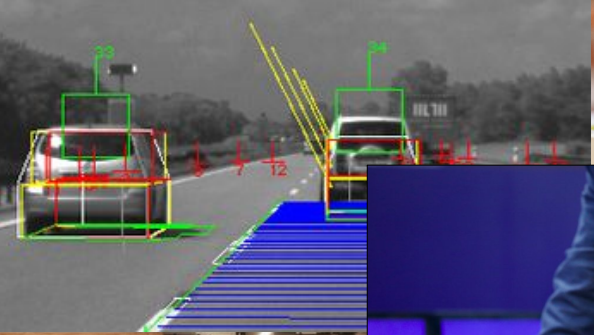
Automated Hyperparameter and Architecture Tuning

Cédric Archambeau
Amazon Web Services, Berlin
cedrica@amazon.com

From HPO to NAS: Automated Deep Learning
CVPR, June 2020

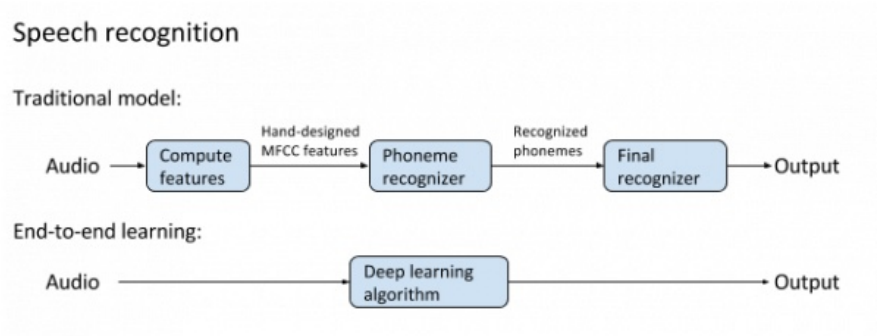



amazon

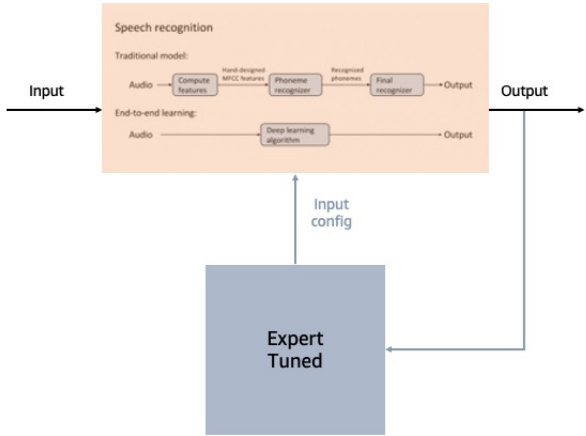

amazon go
BAGS


arch

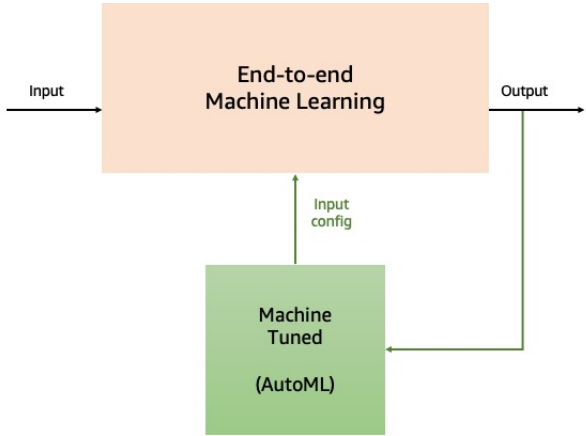
End-to-end machine learning



Is this the end of the story?



Automated end-to-end machine learning



SVM classifier: validation error for different hyperparameters [JT16]

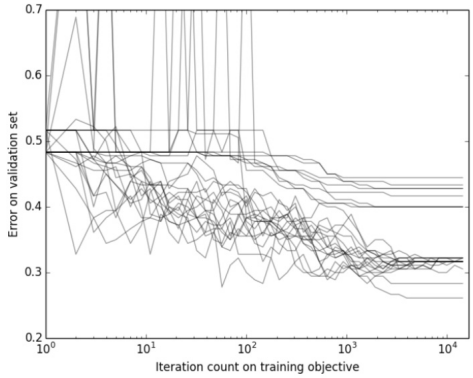


Figure 1: Validation error for various hyperparameter configurations for a classification task trained using stochastic gradient descent.

On the impact of hyperparameter tuning in practice

A Knight of the Seven Kingdoms (A Song of Ice and Fire)

$20.43 | In Stock. Ships from and sold by Amazon.com. Gift-wrap available.

★★★★★ **I quickly became absorbed in the tales of "Dunk and Egg" and the ancestors of the great houses of Westeros**

By Amazon Customer on November 24, 2015

Format: Kindle Edition | Verified Purchase

After reading Martin's other series, I was eager to find any and all related materials. This story is set about 100 years before the main action in Westeros and introduces some new characters and fills in some blanks on ones who were referred to in the later story. I quickly became absorbed in the tales of "Dunk and Egg" and the ancestors of the great houses of Westeros. I loved the angle of Egg traveling around living as a regular child instead of a prince of the realm. This book holds three short tales of adventures they have together and different lessons they both learn. My only issue with it was that it was too short! I wanted more; I wanted to see how Dunk developed as a person because he had a lot to learn about how noble power players might use hapless knights such as he. I hope there are plans to continue this series because I'd like to see how Egg learned from his experiences living among the people and how that changed the man he would become.

Comment | 3 people found this helpful. Was this review helpful to you? Yes No Report abuse

★★★☆☆ **A taste of game of thrones before 6th book!**

By Steven M McLaughlin on December 5, 2015

Format: Kindle Edition | Verified Purchase

I tore through the game of thrones series and have been waiting and waiting for the latest book. An associate told me about this book and I was psyched. I was traveling and was thankful to download onto my kindle for a long flight home. It was entertaining but I got lost with all the characters and couldn't really keep up with who was doing what. Might need to go back and read again slowly to try to comprehend what happened! Didn't have this problem with the other game of thrones books...

Comment | 2 people found this helpful. Was this review helpful to you? Yes No Report abuse

On the impact of hyperparameter tuning in practice

A Knight of the Seven Kingdoms (A Song of Ice and Fire)

$20.43 | In Stock. Ships from and sold by Amazon.com. Gift-wrap available.

★★★★★ **I quickly became absorbed in the tales of "Dunk and Egg" and the ancestors of the great houses of Westeros**

By Amazon Customer on November 24, 2015

Format: Kindle Edition | Verified Purchase

After reading Martin's other series, I was eager to find any and all related materials. This story is set about 100 years before the main action in Westeros and introduces some new characters and fills in some blanks on ones who were referred to in the later story. I quickly became absorbed in the tales of "Dunk and Egg" and the ancestors of the great houses of Westeros. I loved the angle of Egg traveling around living as a regular child instead of a prince of the realm. This book holds three short tales of adventures they have together and different lessons they both learn. My only issue with it was that it was too short! I wanted more; I wanted to see how Dunk developed as a person because he had a lot to learn about how noble power players might use hapless knights such as he. I hope there are plans to continue this series because I'd like to see how Egg learned from his experiences living among the people and how that changed the man he would become.

Comment | 3 people found this helpful. Was this review helpful to you? Yes No Report abuse

★★★☆☆ **A taste of game of thrones before 6th book!**

By Steven M McLaughlin on December 5, 2015

Format: Kindle Edition | Verified Purchase

I tore through the game of thrones series and have been waiting and waiting for the latest book. An associate told me about this book and I was psyched. I was traveling and was thankful to download onto my kindle for a long flight home. It was entertaining but I got lost with all the characters and couldn't really keep up with who was doing what. Might need to go back and read again slowly to try to comprehend what happened! Didn't have this problem with the other game of thrones books...

Comment | 2 people found this helpful. Was this review helpful to you? Yes No Report abuse

Is a product review `positive` or `negative`?

- Simple sentiment analysis as a binary classification problem.
- Logistic regression model with standard text features.
- Model trained via stochastic gradient descent (SGD).

Revisiting sentiment analysis [YS15]

Hyperparameter	**Values**
n_{min}	$\{1, 2, 3\}$
n_{max}	$\{n_{min}, \ldots, 3\}$
weighting scheme	{tf, tf-idf, binary}
remove stop words?	{True, False}
regularization	$\{\ell_1, \ell_2\}$
regularization strength	$[10^{-5}, 10^{5}]$
convergence tolerance	$[10^{-5}, 10^{-3}]$

Revisiting sentiment analysis [YS15]

Method	Acc.
SVM-unigrams	88.62
SVM-$\{1, 2\}$-grams	90.70
SVM-$\{1, 2, 3\}$-grams	90.68
NN-unigrams	88.94
NN-$\{1, 2\}$-grams	91.10
NN-$\{1, 2, 3\}$-grams	91.24
LR (this work)	91.56
Bag of words CNN	91.58
Sequential CNN	92.22

Table 5: Comparisons on the Amazon electronics dataset. Scores are as reported by Johnson and Zhang (2014).

Acc.: accuracy

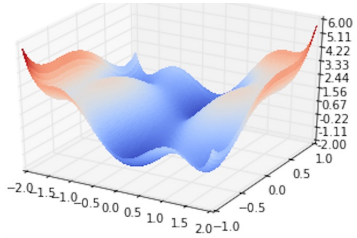
SVM: support vector machine
NN: neural network
LR: logistic regresion
CNN: convolutional neural network

Automated Hyperparameter and Architecture Tuning

- Black-box global optimisation
- Bayesian optimisation
  - Gaussian processes for regression
  - Acquisition functions
  - SMAC, TPE, etc.
- Multi-fidelity black-box optimisation
  - Successive halving, Hyperband, etc.
  - Model-based extensions
  - Asynchronous extensions

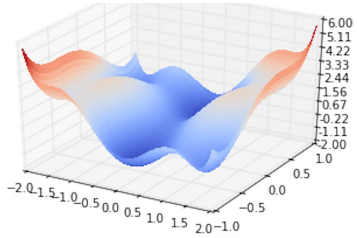
Black-box global optimisation

Black-box global optimisation



- The function f we wish to optimise can be non-convex.
- The number of hyperparameters p is moderate (typically < 20).

Black-box global optimisation



- The function f we wish to optimise can be non-convex.
- The number of hyperparameters p is moderate (typically < 20).

Our goal is to solve the following optimisation problem:

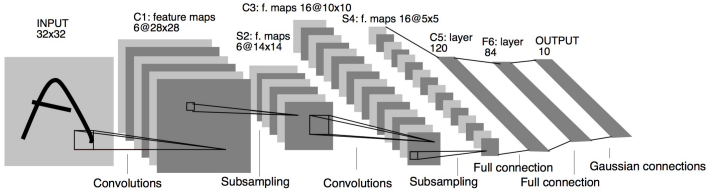
$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}).$$

- Evaluating $f(\boldsymbol{x})$ is expensive.
- No analytical form or gradient.
- Evaluations may be noisy.

Hyperparameter tuning as a global optimisation problem

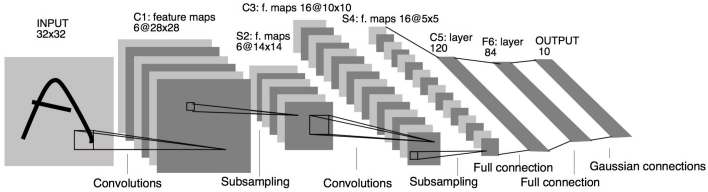
1. Define an objective or metric to optimise
   E.g.: validation error
2. Identify the knobs that impact this objective
   E.g.: hyperparameters
3. Measure the quality of the configurations
   E.g.: cross-validation

Example: Tuning a deep neural net [SLA12, SRS+15, KFB+17]



LeNet5 [LBBH98]

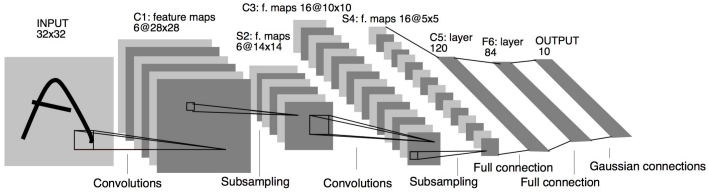
Example: Tuning a deep neural net [SLA12, SRS+15, KFB+17]



LeNet5 [LBBH98]

- $f(\mathbf{x})$ measures the quality of the neural net as a function of its hyperparameters $\mathbf{x}$.
- Evaluating $f(\mathbf{x})$ is very **costly**

Example: Tuning a deep neural net [SLA12, SRS+15, KFB+17]



LeNet5 [LBBH98]

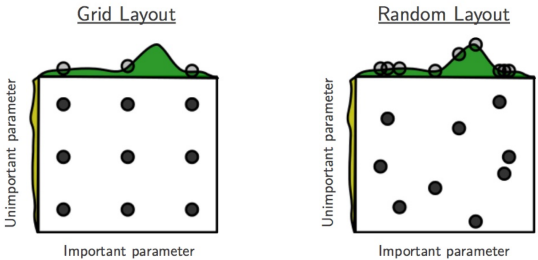
- $f(\mathbf{x})$ measures the quality of the neural net as a function of its hyperparameters $\mathbf{x}$.
- Evaluating $f(\mathbf{x})$ is very **costly**
- The search space $\mathcal{X}$ is large and diverse:
  - Architecture: # hidden layers, activation functions, ...
  - Model complexity: regularisation, dropout, ...
  - Optimisation parameters: learning rates, momentum, batch size, ...

Hyperparameter tuning as a global optimisation problem

1. Define an objective or metric to optimise
   E.g.: validation error
2. Identify the knobs that impact this objective
   E.g.: hyperparameters
3. Measure the quality of the configurations
   E.g.: cross-validation

This requires evaluating **many** hyperparameter configurations.

Two straightforward approaches



(Figure by Bergstra and Bengio, 2012)

- Exhaustive search on a regular or random grid
- Complexity is exponential in p
- Wasteful of resources, but easy to parallelise

Can we do better?

Bayesian optimisation

A probabilistic approach to global optimisation:

Bayesian optimisation

A probabilistic approach to global optimisation:

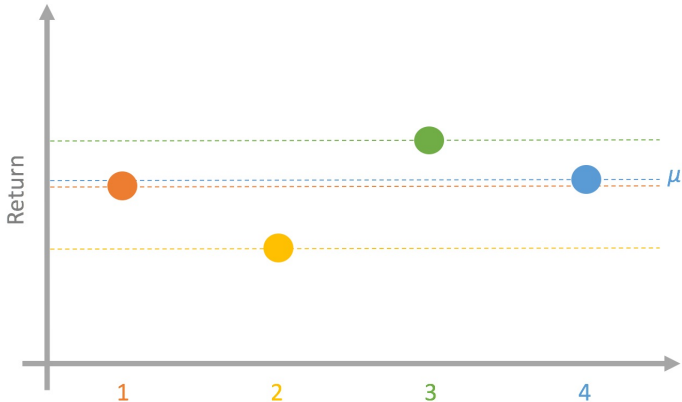
1. Builds a probabilistic model of the objective:
   - Optimises this surrogate instead of the objective
   - Captures the **uncertainty**

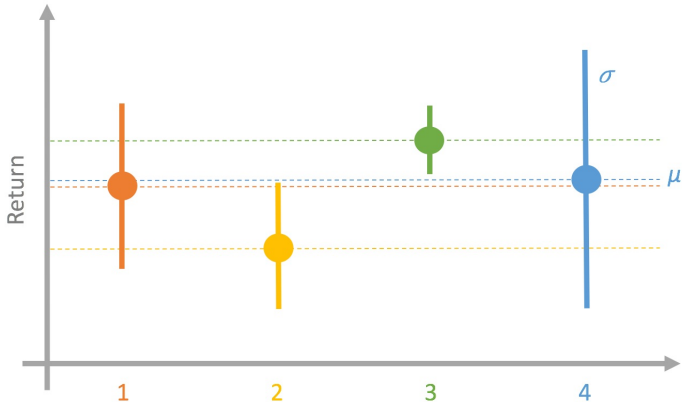
Bayesian optimisation

A probabilistic approach to global optimisation:

1. Builds a probabilistic model of the objective:
   - Optimises this surrogate instead of the objective
   - Captures the **uncertainty**
2. Performs an efficient grid search by balancing exploration against exploitation!

Which slot machine should I pick?


Return
μ
1
2
3
4


Return
σ
μ
1
2
3
4

Bayesian black-box optimisation with Gaussian processes

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$$

Bayesian (black-box) optimisation [MTZ78, SSW+16]

$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$$

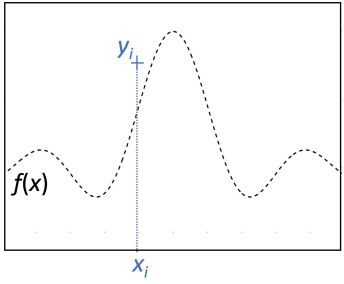
Canonical algorithm:

- Surrogate model $\mathcal{M}$ of f `#cheaper to evaluate`
- Set of evaluated candidates $\mathcal{C} = \{\}$
- While some `BUDGET` available:
  - Select candidate $\mathbf{x}_{\text{new}} \in \mathcal{X}\backslash\mathcal{C}$ based on $\mathcal{M}$ `#exploration/exploitation`
  - Collect evaluation y_{new} of f at $\mathbf{x}_{\text{new}}$ `#time-consuming`
  - Update $\mathcal{C} = \mathcal{C} \cup \{\mathbf{x}_{\text{new}}\}$
  - Update $\mathcal{M}$ with $(\mathbf{x}_{\text{new}}, y_{\text{new}})$ `#Update surrogate model`
  - Update `BUDGET`

Bayesian (black-box) optimisation with Gaussian processes [JSW98]

1. Learn a probabilistic model of f, which is cheap to evaluate:

$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

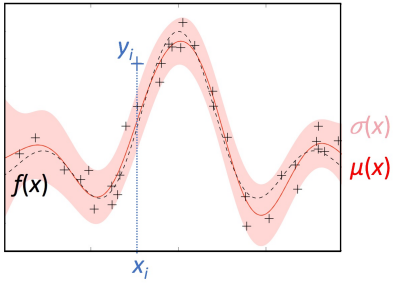


Bayesian (black-box) optimisation with Gaussian processes [JSW98]

1. Learn a probabilistic model of f, which is cheap to evaluate:

$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

2. Given the observations $\boldsymbol{y} = (y_1, \ldots, y_n)$, estimate the posterior mean and the posterior standard deviation:

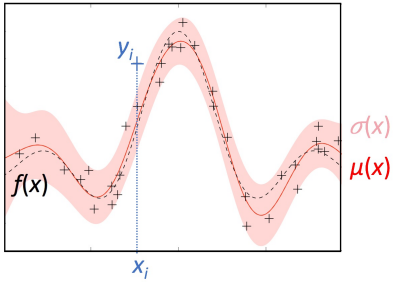


Bayesian (black-box) optimisation with Gaussian processes [JSW98]

1. Learn a probabilistic model of f, which is cheap to evaluate:

$$y_i | f(\boldsymbol{x}_i) \sim \text{Gaussian}\left(f(\boldsymbol{x}_i), \varsigma^2\right), \qquad f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}).$$

2. Given the observations $\boldsymbol{y} = (y_1, \ldots, y_n)$, estimate the posterior mean and the posterior standard deviation:



3. Repeatedly query f by balancing **exploitation** against **exploration**!

Ingredient 1: Gaussian processes for regression [RW06]

- A **multivariate Gaussian** is density over p random variables based on correlations:

$$\boldsymbol{f} \equiv (f_1, \ldots, f_p)^\top \sim \text{Gaussian}\left(\boldsymbol{m}, \boldsymbol{K}\right).$$

- A **Gaussian process** generalises the a multivariate Gaussian to infinitely many variables:

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \cdot)\right).$$

  - The joint density of any finite subset is a consistent Gaussian density.
  - It defines a probability measure over random functions.

Ingredient 1: Gaussian processes for regression [RW06]

- A **multivariate Gaussian** is density over p random variables based on correlations:

$$\boldsymbol{f} \equiv (f_1, \ldots, f_p)^\top \sim \text{Gaussian}\,(\boldsymbol{m}, \boldsymbol{K})\,.$$

- A **Gaussian process** generalises the a multivariate Gaussian to infinitely many variables:

$$f(\boldsymbol{x}) \sim \mathcal{GP}\,(m(\boldsymbol{x}), k(\boldsymbol{x}, \cdot))\,.$$

  - The joint density of any finite subset is a consistent Gaussian density.
  - It defines a probability measure over random functions.

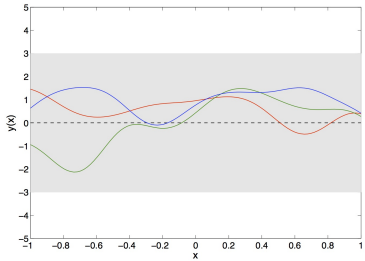
- The **posterior process** is again a Gaussian process (if Gaussian likelihood):

$$f(\boldsymbol{x})|\mathcal{D} \sim \mathcal{GP}\,(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}, \cdot))\,,$$

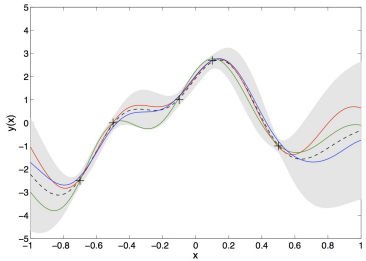
where

$$\mu(\boldsymbol{x}) = \boldsymbol{k}_n^\top(\boldsymbol{x})(\boldsymbol{K}_n + \varsigma^2 \boldsymbol{I}_n)^{-1}\boldsymbol{y},$$
$$\sigma^2(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_n^\top(\boldsymbol{x})(\boldsymbol{K}_n + \varsigma^2 \boldsymbol{I}_n)^{-1}\boldsymbol{k}_n(\boldsymbol{x}).$$

Ingredient 1: Gaussian processes for regression [RW06]



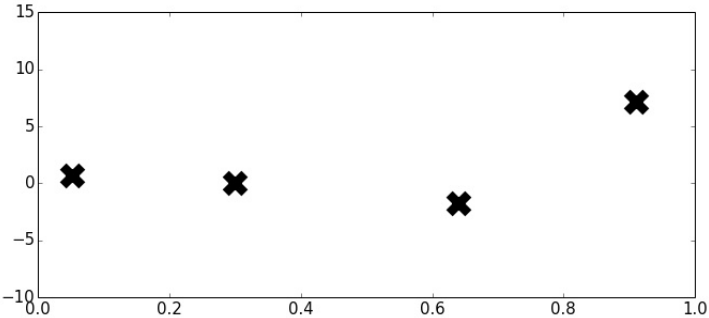
Prior.



Posterior.

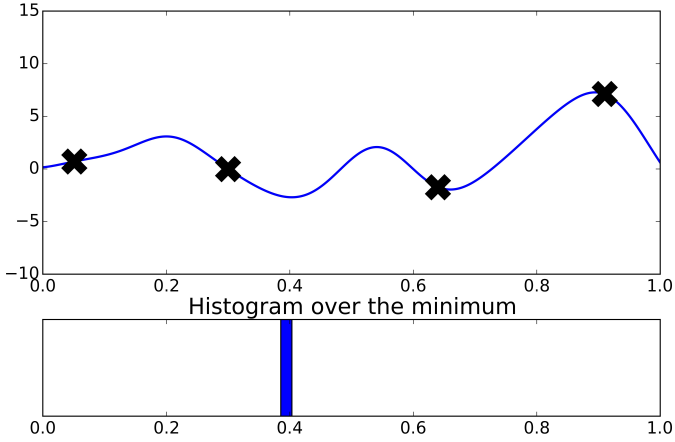
Three random functions generated from (a) the prior GP and (b) the posterior GP. An observation is indicated by a $+$, the mean function by a dashed line and the 3 standard deviation error bars by the shaded regions. We used a squared exponential covariance function.

Where is the minimum of f?



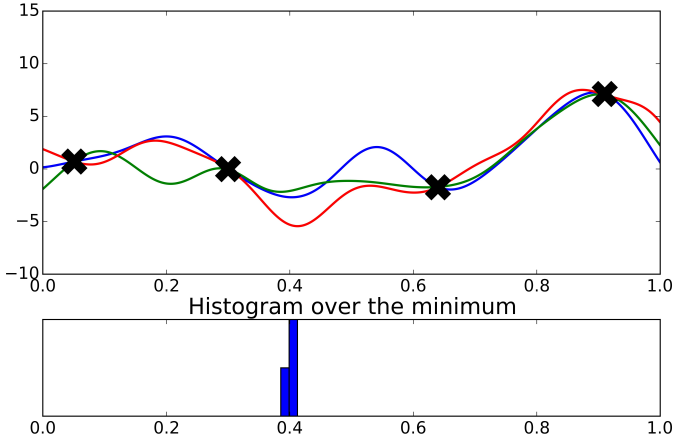
(Image credit: Javier González)

Intuitive solution



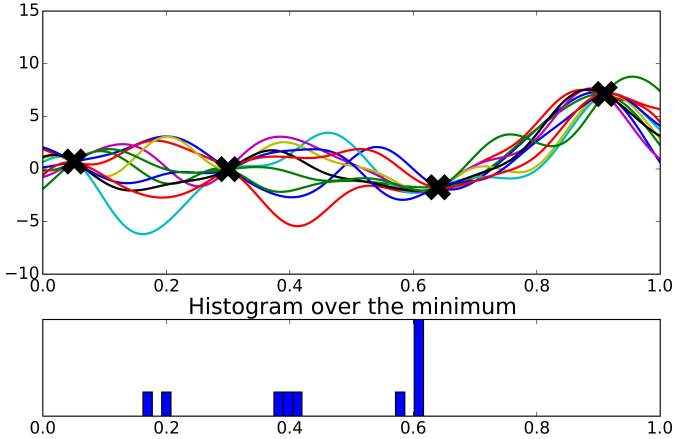
(Image credit: Javier González)

Intuitive solution



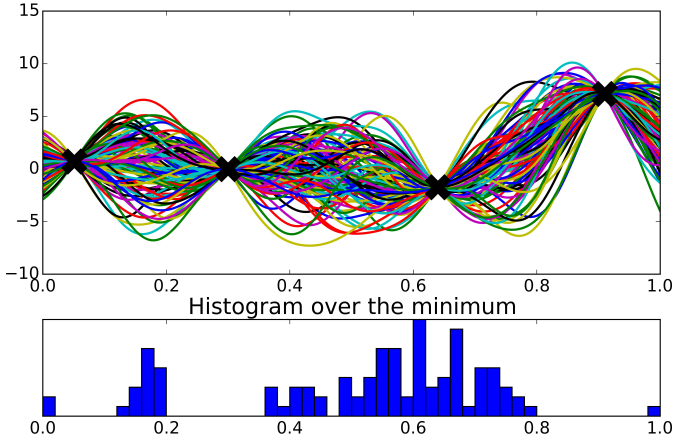
(Image credit: Javier González)

Intuitive solution



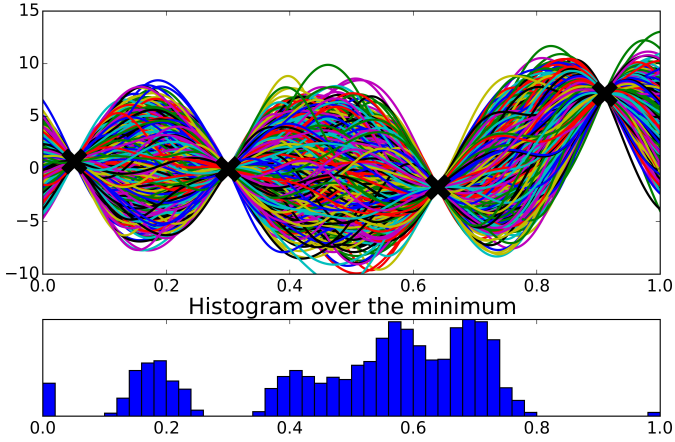
(Image credit: Javier González)

Intuitive solution



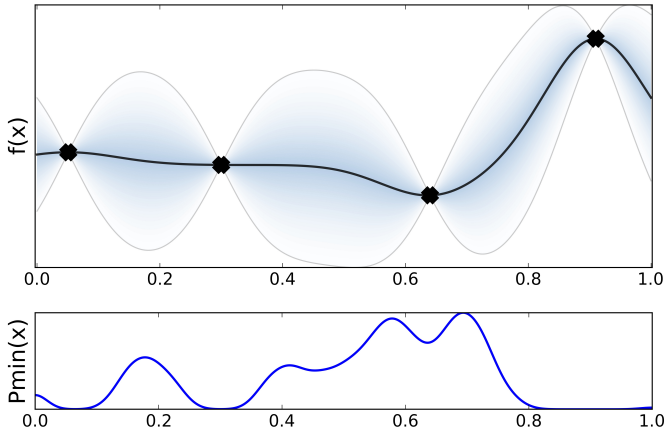
(Image credit: Javier González)

Intuitive solution



(Image credit: Javier González)

Intuitive solution



(Image credit: Javier González)

Ingredient 2: Acquisition function

The acquisition function trades exploitation for exploration:

$$a : \boldsymbol{x} \mapsto a(\boldsymbol{x}|\mathcal{D}) = \mathbb{E}\left(\mathcal{A}(f, \boldsymbol{x})|\mathcal{D}\right).$$

Ingredient 2: Acquisition function

The acquisition function trades exploitation for exploration:

$$a : \boldsymbol{x} \mapsto a(\boldsymbol{x}|\mathcal{D}) = \mathbb{E}\left(\mathcal{A}(f, \boldsymbol{x})|\mathcal{D}\right).$$

- Decide which are most promising regions in $\mathcal{X}\backslash\mathcal{C}$:

$$\max_{\boldsymbol{x}} a(\boldsymbol{x}|\mathcal{D}).$$

- Get as quickly as possible to "the" optimum (unlike Bayesian experimental design)
- Can be optimised with off-the-self solvers!

Ingredient 2: Acquisition function

Let $y_\star$ and $\boldsymbol{x}_\star$ be respectively the global minimum and global minimizer. Further, let $\hat{y}_\star$ be the best value observed so far and $f(\boldsymbol{x})|\mathcal{D} \sim \text{Gaussian}\left(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})^2\right)$.

Ingredient 2: Acquisition function

Let $y_\star$ and $\boldsymbol{x}_\star$ be respectively the global minimum and global minimizer. Further, let $\hat{y}_\star$ be the best value observed so far and $f(\boldsymbol{x})|\mathcal{D} \sim \text{Gaussian}\left(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})^2\right)$.

Improvement based:

- Probability of improvement (PI) [Kus64]: $a(\boldsymbol{x}) = P(f(\boldsymbol{x}) < \hat{y}_\star|\mathcal{D})$.
- Expected improvement (EI) [MTZ78]: $a(\boldsymbol{x}) = \mathbb{E}\left(\max\{0, \hat{y}_\star - f(\boldsymbol{x})\}|\mathcal{D}\right)$.
- Lower confidence bound (GP-LCB) [SKKS09]: $a(\boldsymbol{x}) = -\mu(\boldsymbol{x}) + \alpha\sigma(\boldsymbol{x}) \quad (\alpha \geqslant 0)$.

Ingredient 2: Acquisition function

Let $y_\star$ and $\boldsymbol{x}_\star$ be respectively the global minimum and global minimizer. Further, let $\hat{y}_\star$ be the best value observed so far and $f(\boldsymbol{x})|\mathcal{D} \sim \text{Gaussian}\left(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})^2\right)$.

Improvement based:

- Probability of improvement (PI) [Kus64]: $a(\boldsymbol{x}) = P(f(\boldsymbol{x}) < \hat{y}_\star|\mathcal{D})$.
- Expected improvement (EI) [MTZ78]: $a(\boldsymbol{x}) = \mathbb{E}\left(\max\{0, \hat{y}_\star - f(\boldsymbol{x})\}|\mathcal{D}\right)$.
- Lower confidence bound (GP-LCB) [SKKS09]: $a(\boldsymbol{x}) = -\mu(\boldsymbol{x}) + \alpha\sigma(\boldsymbol{x}) \quad (\alpha \geqslant 0)$.

Entropy based:

- Entropy search (ES) [HS12]: $a(\boldsymbol{x}) = \mathbb{H}\left(p(\boldsymbol{x}_\star|\mathcal{D})\right) - \mathbb{E}\left(\mathbb{H}\left(p(\boldsymbol{x}_\star|\mathcal{D} \cup \{(\boldsymbol{x}, y)\})\right)\right)$.
- Predictive entropy search (PES) [HLHG14]: $a(\boldsymbol{x}) = \mathbb{H}\left(p(y|\mathcal{D}, \boldsymbol{x})\right) - \mathbb{E}\left(\mathbb{H}\left(p(y|\mathcal{D}, \boldsymbol{x}, \boldsymbol{x}_\star)\right)\right)$.
- Max-value entropy search (MES) [WJ17]: $a(\boldsymbol{x}) = \mathbb{H}\left(p(y|\mathcal{D}, \boldsymbol{x})\right) - \mathbb{E}\left(\mathbb{H}\left(p(y|\mathcal{D}, \boldsymbol{x}, y_\star)\right)\right)$.

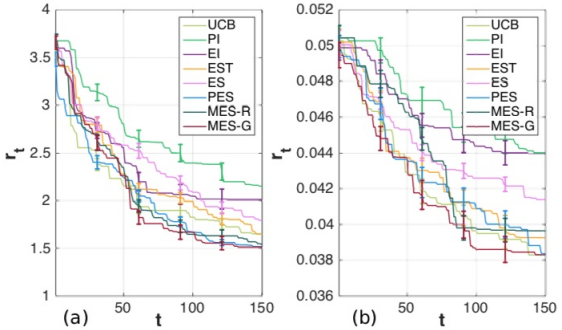
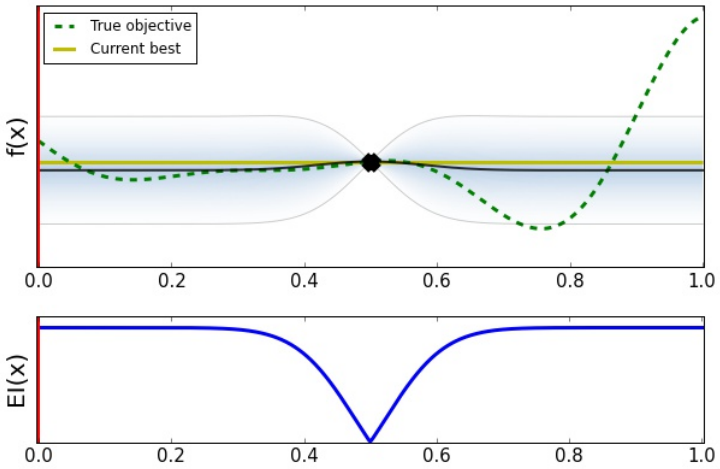


Figure 2. Tuning hyper-parameters for training a neural network, (a) Boston housing dataset; (b) breast cancer dataset. MES methods perform better than other methods on (a), while for (b), MES-G, UCB, PES perform similarly and better than others.

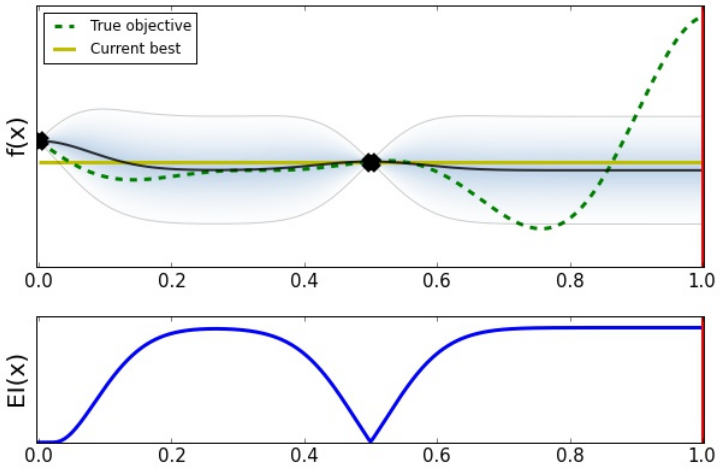
(Figure by Wang and Jegelka [WJ17])

Bayesian optimisation in action



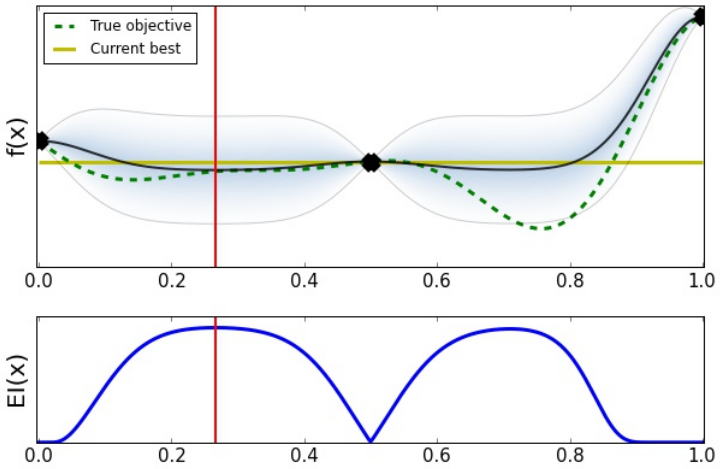
(Image credit: Javier González)

Bayesian optimisation in action



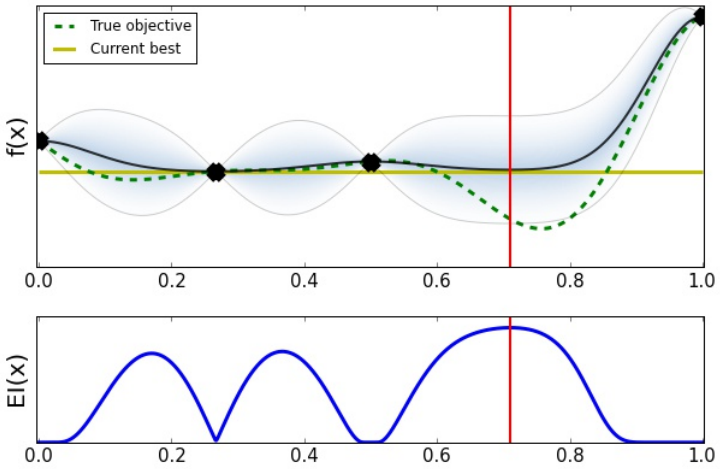
(Image credit: Javier González)

Bayesian optimisation in action



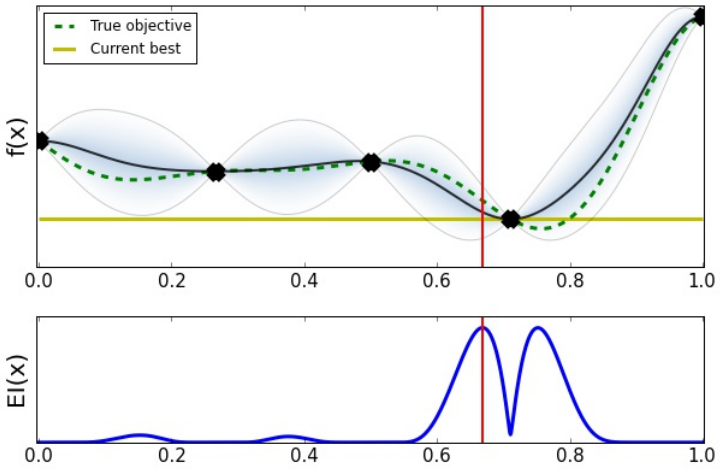
(Image credit: Javier González)

Bayesian optimisation in action



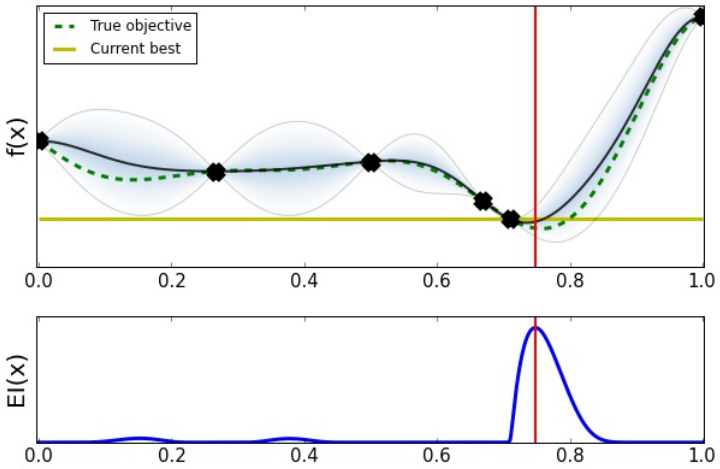
(Image credit: Javier González)

Bayesian optimisation in action



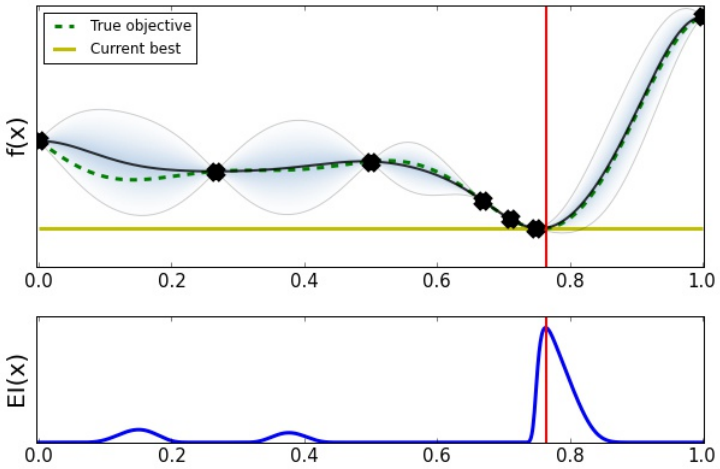
(Image credit: Javier González)

Bayesian optimisation in action



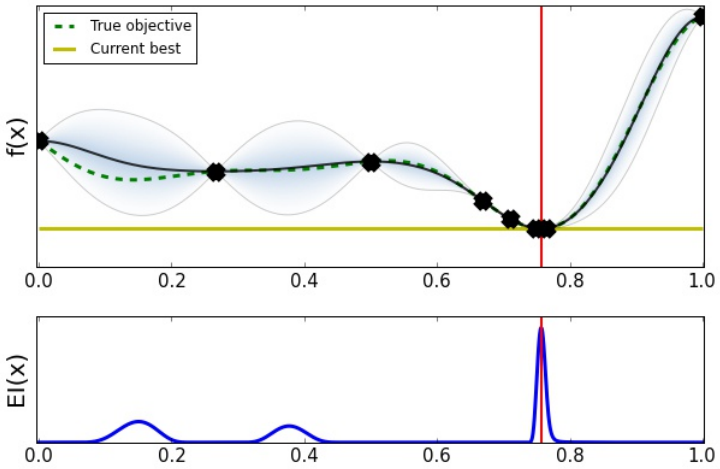
(Image credit: Javier González)

Bayesian optimisation in action



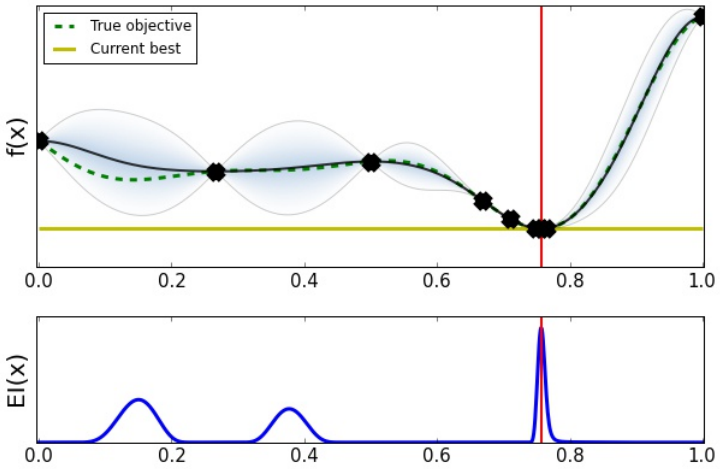
(Image credit: Javier González)

Bayesian optimisation in action



(Image credit: Javier González)

Bayesian optimisation in action



(Image credit: Javier González)

Summary

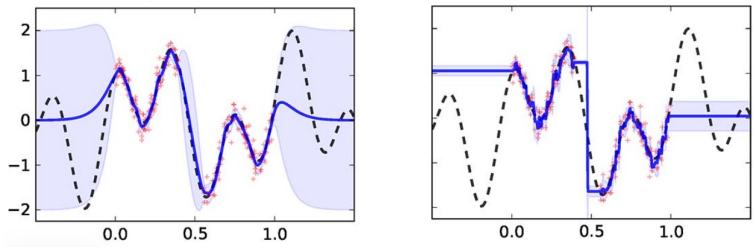
$$\boldsymbol{x}_\star = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}}\ f(\boldsymbol{x})$$

Bayesian optimisation algorithm:

- Surrogate model $\mathcal{M}$ of f `#cheaper to evaluate`
- Set of evaluated candidates $\mathcal{C} = \{\}$
- While some `BUDGET` available:
  - Select candidate $\mathbf{x}_{\text{new}} \in \mathcal{X}$ using $\mathcal{M}$ and $\mathcal{C}$ `#acquisition`
  - Collect evaluation y_{new} of f at $\mathbf{x}_{\text{new}}$ `#time-consuming`
  - Update $\mathcal{C} = \mathcal{C} \cup \{\mathbf{x}_{\text{new}}\}$
  - Update $\mathcal{M}$ with $(\mathbf{x}_{\text{new}}, y_{\text{new}})$ `#GP posterior`
  - Update `BUDGET`

Are there other choices for the surrogate model?

SMAC [HHLB11]



(Image credit: [SSW+16])

Bayesian (black-box) optimisation with **Random Forests**:

$$y(\boldsymbol{x}) = \mathrm{RF}(\boldsymbol{x}), \qquad f(\boldsymbol{x})|\mathcal{D} \sim \text{Gaussian}\left(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})\right).$$

where $\mu(\boldsymbol{x}) \approx \frac{1}{B}\sum_i y_i(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x}) \approx \frac{1}{B-1}\sum_i (y_i(\boldsymbol{x}) - \mu(\boldsymbol{x}))^2$.

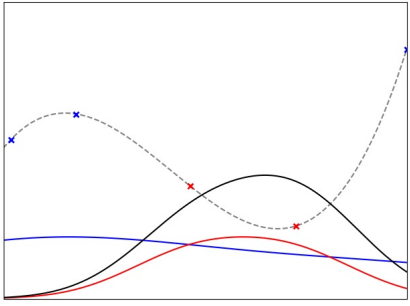
Tree of Parzen estimators (TPE) [BBBK11]

- Parzen window estimator to model good and bad configurations:

$$p(\boldsymbol{x}|y) = \begin{cases} g(\boldsymbol{x}) & \text{if } y \geqslant \alpha, \\ l(\boldsymbol{x}) & \text{if } y < \alpha. \end{cases}$$

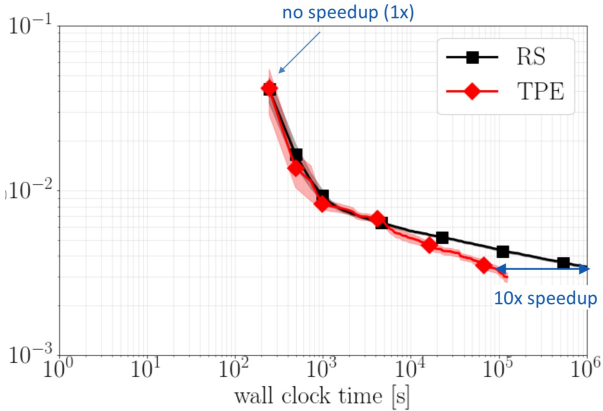
- How to choose kernel bandwidth and threshold α?

$$p(y < y_\star) \approx p(y < \alpha).$$



(Image credit: Aaron Klein)

Tree Parzen estimator vs random search



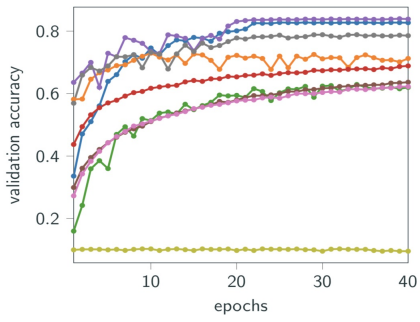
(Image credit: Frank Hutter)

Multi-fidelity black-box optimisation

Multi-fidelity hyperparameter optimisation

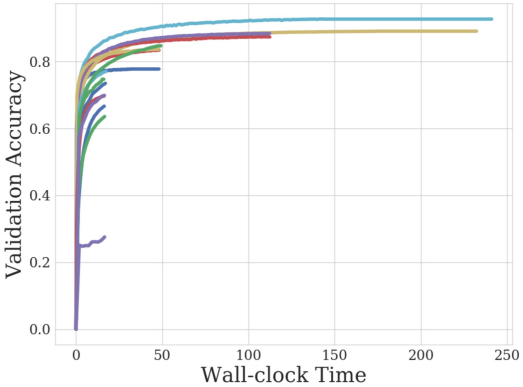
- Bayesian optimisation is sample efficient, but sequential.
- Random search is embarrassingly parallel.
- Training ML is inherently iterative and possibly very costly.
- Can we use cheap-to-evaluate approximations of the target $f(\boldsymbol{x})$?

Low-fidelity via early stopping



(Image credit: Aaron klein)

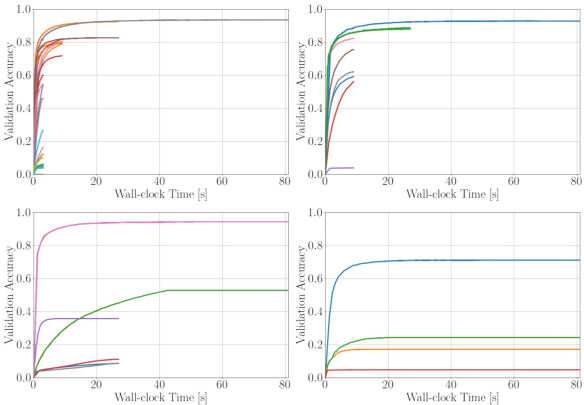
Successive halving [KKS13]



(Image credit: Aaron Klein)

- Apply median rule to stop the evaluation of configurations.
- Outperforms uniform the allocation of resources.
- Early stopping is simple and robust.

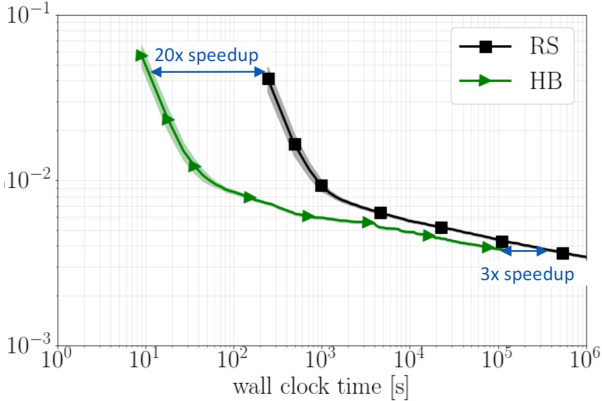
Hyperband [LJD+16]



(Image credit: Aaron Klein)

- Ressource levels: n versus B/n
- Strong anytime performance
- Easy to parallize

Hyperband vs random search



(Image credit: Frank Hutter)

Can we do better than uniform sampling?

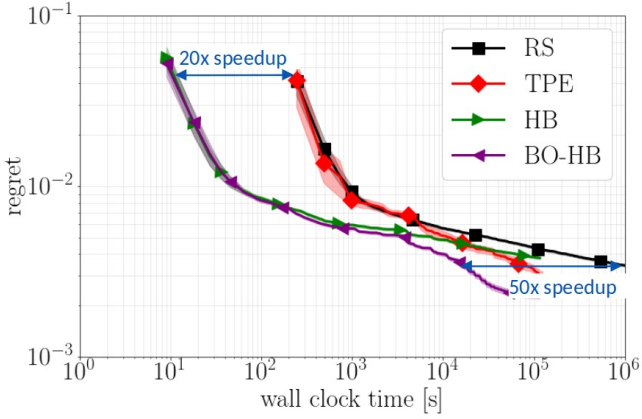
Model-based multi-fidelity optimisation

- Each fidelity is parameterised by a resource level r:
  - if $r = r_{max}$: then $f(\boldsymbol{x}, r_{max}) = f(\boldsymbol{x})$
  - if $r < r_{max}$: then $f(\boldsymbol{x}, r)$ is a low-fidelity approximation of $f(\boldsymbol{x})$.
- Bayesian optimisation-based:

$$f(\boldsymbol{x}, r) \sim \mathcal{GP}\left(m(\boldsymbol{x}, r), k(\boldsymbol{x}, r, \cdot, \cdot)\right).$$

  - Auto-correlation within each fidelity
  - Cross-correlation between fidelities
- (Original) BOHB [FKH18]: independent TPE-based hyperband for each fidelity r.

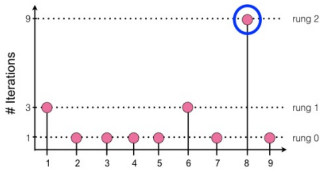
Uniform vs non-uniform multi-fidelity optimisation



(Image credit: Frank Hutter)

Asynchronous extensions

Asynchronous hyperband



(a) Visual depiction of the promotion scheme for bracket $s = 0$.

bracket s	rung i	n_i	r_i	total budget
0	0	9	1	9
	1	3	3	9
	2	1	9	9
1	0	9	3	27
	1	3	9	27
2	0	9	9	81

(b) Promotion scheme for different brackets s.

(Image credit: [LJR+18])

- Promotion variant (ASHA) [LJR+18]: promote when there is a configuration in top $1/\eta$.
- Stopping variant (Ray Tune): continue as long as configuration in top $1/\eta$.
- Loop over/parallelise brackets (aka early stopping rates)

Asynchronous model-based multi-fidelity optimisation [TKAS20]

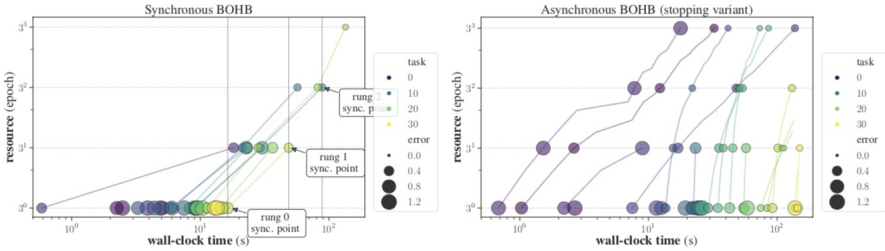
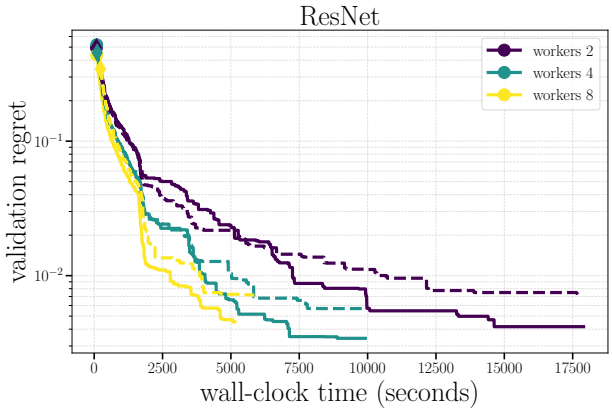


Figure 1: Evolution of synchronous (left) and asynchronous (right) BOHB on OpenML MLP tuning (see Section 4). x: wall-clock time (150 secs), y: training epochs done (log scale). Coloured lines trace training jobs for HP configurations, with marker size proportional to validation error (smaller is better). Lines not reaching the maximum number of authorized epochs (27) are stopped early. For synchronous BOHB, rungs (epochs 1, 3, 9) need to be filled completely before any configuration can be promoted, and each synchronization results in some idle time. In asychronous BOHB, configurations proceed to higher epochs faster. Without synchronization overhead, more configurations are promoted to the maximum number of epoch in the same wall-clock time (figure best seen in colours).

Asynchronous uniform vs non-uniform multi-fidelity optimisation



NASbench201 [DY20]

- Tabular benchmark for Neural Architecture Search (NAS): function evaluations are replaced by cheap table look-ups.
- It contains the precomputed performance of 15,625 architectures on 3 different image classification datasets.

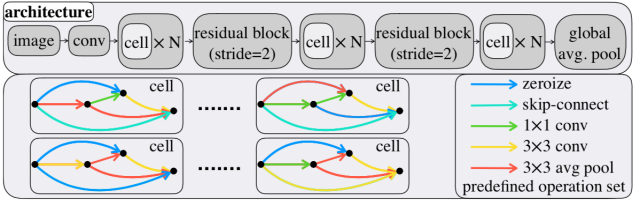


Figure 1: **Top**: the macro skeleton of each architecture candidate. **Bottom-left**: examples of neural cell with 4 nodes. Each cell is a directed acyclic graph, where each edge is associated with an operation selected from a predefined operation set as shown in the **Bottom-right**.

(Image credit: [DY20])

Comparing methods on NASbench201

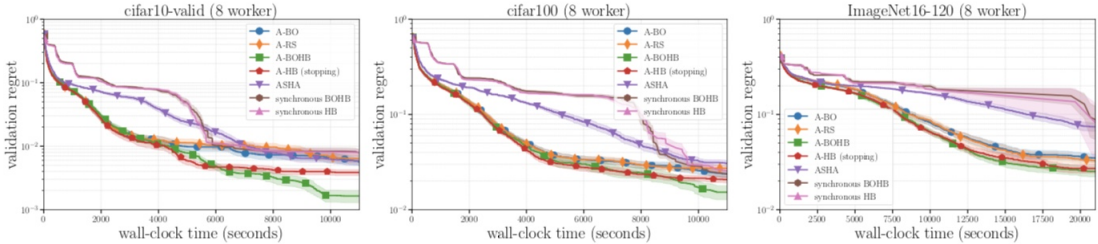


Figure 4: Comparison of state-of-the-art distributed HNAS methods on different NASBench201 datasets, run with 8 workers. The synchronous methods clearly require more time to reach good accuracies than the asynchronous ones. ASHA is initially slowed down by its more conservative promotion scheme. At higher wall-clock times, our A-BOHB clearly outperforms all other methods.

Further reading

NeurIPS tutorial by Frank Hutter and Joaquin Vanschoren (2018): Automatic Machine Learning (AutoML).

Review paper by Shahriari, et al. (2016): Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE 104(1):148–175*.

Slides by Peter Frazier (2010): Tutorial: Bayesian Methods for Global and Simulation Optimization. *INFORMS Annual Meeting*.

What's next?

1. AutoGluon Toolkit Overview (Hang Zhang)
2. AutoML for TinyML with Once-for-All Network (Song Han)
3. Hands-on tutorials:
   - Multi-fidelity optimisation (Thibaut Lienart)
   - Proxyless NAS (Chongruo Wu)
   - Fast Auto-augment (Hang Zhang)

References I


James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl.
Algorithms for hyper-parameter optimization.
In *Advances in Neural Information Processing Systems*, volume 24, pages 2546–2554, 2011.


T. Chen, E. B. Fox, and C. Guestrin.
Stochastic gradient hamiltonian monte carlo.
In *International Conference on Machine Learning*, 2014.


X. Dong and Y. Yang.
Nas-bench-201: Extending the scope of reproducible neural architecture search.
arXiv:2001.00326 [cs.CV], 2020.


S. Falkner, A. Klein, and F. Hutter.
BOHB: Robust and efficient hyperparameter optimization at scale.
In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.


F. Hutter, H. H. Hoos, and K. Leyton-Brown.
Sequential model-based optimization for general algorithm configuration.
In *Proceedings of LION-5*, page 507?523, 2011.

References II


José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani.
Predictive entropy search for efficient global optimization of black-box functions.
Technical report, preprint arXiv:1406.2541, 2014.


Philipp Hennig and Christian J Schuler.
Entropy search for information-efficient global optimization.
Journal of Machine Learning Research, 98888(1):1809–1837, 2012.


Donald R Jones, Matthias Schonlau, and William J Welch.
Efficient global optimization of expensive black-box functions.
Journal of Global optimization, 13(4):455–492, 1998.


Kevin Jamieson and Ameet Talwalker.
A bayesian approach to filtering junk e-mail.
In *International conference on Aritificial Intelligence and Statistics (AIStats) 19*, 2016.


Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter.
Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets.
In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

References III


Zohar Karnin, Tomer Koren, and Oren Somekh.
Almost optimal exploration in multi-armed bandits.
In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1238–1246, 2013.


Harold J Kushner.
A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise.
Journal of Fluids Engineering, 86(1):97–106, 1964.


Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.


Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar.
Hyperband: A novel bandit-based approach to hyperparameter optimization.
Technical report, preprint arXiv:1603.06560, 2016.

References IV


L. Li, K. Jamieson, A. Rostamizadeh, K. Gonina, M. Hardt, B. Recht, and A. Talwalkar.
Massively parallel hyperparameter tuning.
arXiv:1810.05934 [cs.LG], 2018.


D.J.C. MacKay.
Bayesian non-linear modeling for the prediction competition.
In G.R. Heidbreder, editor, *Maximum Entropy and Bayesian Methods. Fundamental Theories of Physics*. Springer, 1996.


Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas.
The application of Bayesian methods for seeking the extremum.
Towards Global Optimization, 2(117-129):2, 1978.


Valerio Perrone, Rodolphe Jenatton, Cedric Archambeau, and Matthias Seeger.
Constrained Bayesian optimisation with max-value entropy search.
Technical report, preprint arXiv:XX, 2019.

References V


Valerio Perrone, Rodolphe Jennaton, Matthias Seeger, and Cédric Archambeau.
Scalable hyperparameter transfer learning.
In *Advances in Neural Information Processing Systems*, 2018.


Ali Rahimi, Benjamin Recht, et al.
Random features for large-scale kernel machines.
In *Advances in Neural Information Processing Systems*, 2007.


Carl Rasmussen and Chris Williams.
Gaussian Processes for Machine Learning.
MIT Press, 2006.


Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter.
Bayesian optimization with robust bayesian neural networks.
In *Advances in Neural Information Processing Systems*, pages 4134–4142, 2016.


Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger.
Gaussian process optimization in the bandit setting: No regret and experimental design.
Technical report, preprint arXiv:0912.3995, 2009.

References VI


Jasper Snoek, Hugo Larochelle, and Ryan P Adams.
Practical Bayesian optimization of machine learning algorithms.
In *Advances in Neural Information Processing Systems*, pages 2960–2968, 2012.


Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md Patwary, Mostofa Ali, Ryan P Adams, et al.
Scalable Bayesian optimization using deep neural networks.
Technical report, preprint arXiv:1502.05700, 2015.


Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas.
Taking the human out of the loop: A review of Bayesian optimization.
Proceedings of the IEEE, 104(1):148–175, 2016.


Jasper Snoek, Kevin Swersky, Richard S Zemel, and Ryan P Adams.
Input warping for Bayesian optimization of non-stationary functions.
Technical report, preprint arXiv:1402.0929, 2014.

References VII


L. C. Tiao, A. Klein, C. Archambeau, and M. Seeger.
Model-based asynchronous hyperparameter optimization.
arXiv:2003.10865 [cs.LG], 2020.

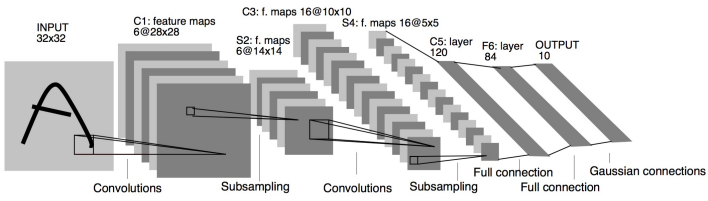

Zi Wang and Stefanie Jegelka.
Max-value entropy search for efficient bayesian optimization.
In *International Conference on Machine Learning (ICML)*, 2017.


Dani Yogatama and Noah A Smith.
Bayesian optimization of text representations.
Technical report, preprint arXiv:1503.00693, 2015.

Additional material

Types of hyperparameters

- Continuous (e.g., learning rate, regulariser)
- Integral (e.g., number of layers, number of units)
- Categorical (e.g., algorithm, activation, operator)



LeNet5 [LBBH98]

Conditional hyperparameters

The search space $\mathcal{X}$ exhibits **conditional** relationships: $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$. Depending on some values in $\mathcal{X}_i$, values in $\mathcal{X}_j$ are irrelevant:

Conditional hyperparameters

The search space $\mathcal{X}$ exhibits **conditional** relationships: $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$. Depending on some values in $\mathcal{X}_i$, values in $\mathcal{X}_j$ are irrelevant:

- **Feedforward neural nets:**

$$\mathcal{X} = \underbrace{\mathcal{X}_0}_{\text{\# hidden layers}} \times \overbrace{\mathcal{X}_1}^{\text{hyperparams layer 1}} \times \underbrace{\mathcal{X}_2}_{\text{hyperparams layer 2}} \times \cdots \times \mathcal{X}_p$$

Conditional hyperparameters

The search space $\mathcal{X}$ exhibits **conditional** relationships: $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$. Depending on some values in $\mathcal{X}_i$, values in $\mathcal{X}_j$ are irrelevant:

- **Feedforward neural nets:**

$$\mathcal{X} = \underbrace{\mathcal{X}_0}_{\text{\# hidden layers}} \times \overbrace{\mathcal{X}_1}^{\text{hyperparams layer 1}} \times \underbrace{\mathcal{X}_2}_{\text{hyperparams layer 2}} \times \cdots \times \mathcal{X}_p$$

- **Optimizer:**

$$\mathcal{X} = \underbrace{\mathcal{X}_0}_{\text{\# type of optimizer}} \times \overbrace{\mathcal{X}_1}^{\text{hyperparams SGD}} \times \underbrace{\mathcal{X}_2}_{\text{hyperparams ADAM}} \times \cdots \times \mathcal{X}_p$$

Conditional hyperparameters

The search space $\mathcal{X}$ exhibits **conditional** relationships: $\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$. Depending on some values in $\mathcal{X}_i$, values in $\mathcal{X}_j$ are irrelevant:

- **Feedforward neural nets:**

$$\mathcal{X} = \underbrace{\mathcal{X}_0}_{\text{\# hidden layers}} \times \overbrace{\mathcal{X}_1}^{\text{hyperparams layer 1}} \times \underbrace{\mathcal{X}_2}_{\text{hyperparams layer 2}} \times \cdots \times \mathcal{X}_p$$

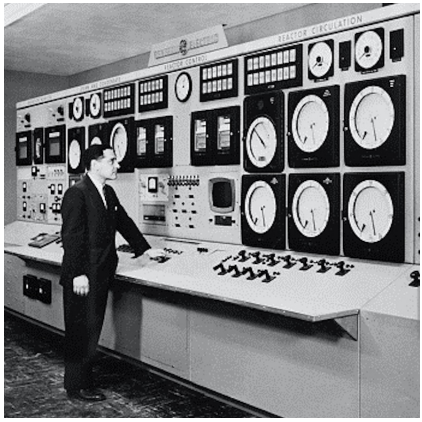
- **Optimizer:**

$$\mathcal{X} = \underbrace{\mathcal{X}_0}_{\text{\# type of optimizer}} \times \overbrace{\mathcal{X}_1}^{\text{hyperparams SGD}} \times \underbrace{\mathcal{X}_2}_{\text{hyperparams ADAM}} \times \cdots \times \mathcal{X}_p$$

- **Data analytics pipeline:**

$$\mathcal{X} = \underbrace{\mathcal{X}_0}_{\text{classifier type}} \times \overbrace{\mathcal{X}_1}^{\text{hyperparams LR}} \times \underbrace{\mathcal{X}_2}_{\text{hyperparams RF}} \times \cdots \times \mathcal{X}_p$$

The performance of machine learning depends on configuration parameters

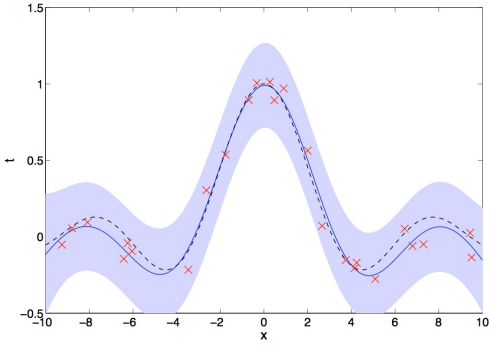


- Regularisation or (hyper)priors
- Optimisation or sampling
- Data pre-processing
- Model complexity
- Decision rules

These parameters are known as hyperparameters or system parameters and are tuned by human experts.

Neural networks with Bayesian linear output layer [SRS+15, PJSA18]

$$p(y_n|\mathbf{x}_n, \mathbf{w}; \mathbf{z}, \beta) = \mathcal{N}(\phi_{\mathbf{z}}(\mathbf{x}_n)\mathbf{w}, \beta^{-1}), \qquad p(\mathbf{w}; \alpha) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I}_p).$$

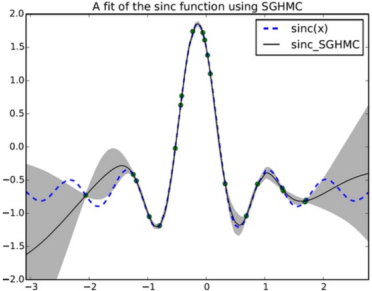


- The predictive distribution:
$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w}$$
$$= \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)).$$
- Learned features $\phi_{\mathbf{z}}(\mathbf{x})$:
  - Explicit features set (e.g., RBF)
  - Random Fourier features [RR+07]
  - Deep neural network features
- Trained with SGD or Newton's method

Scaling is $\mathcal{O}\left(p^3\right)$.

Bayesian neural networks [SKFH16]

$$p(y_n|\mathbf{x}_n;\mathbf{z},\beta) = \mathcal{N}(f(\mathbf{x}_n;\mathbf{z}),\beta^{-1}), \qquad p(\mathbf{z},\beta).$$



(Image credit: Frank Hutter)

- The predictive distribution:

$$p(y^*|\mathbf{x}^*,\mathcal{D}) = \int p(y^*|\mathbf{x}^*,\mathbf{z},\beta)p(\mathbf{z},\beta|\mathcal{D})d\mathbf{z}$$
$$\approx \mathcal{N}(\hat{\mu}(\mathbf{x}^*),\hat{\sigma}^2(\mathbf{x}^*)).$$

  where $\hat{\mu}(\mathbf{x}^*) = \frac{1}{m}\sum_{j=1}^{m} f(\mathbf{x}^*;\mathbf{z}_j)$ and $\hat{\sigma}(\mathbf{x}^*) = \ldots$
- Scalded version of stochastic gradient Hybrid Monte Carlo [CFG14]
- Burn-in used to adjust the many hyperparameters; sensitive to step-size.

Hyperparameter optimisation as non-stochastic best arm identification [JT16]

Best Arm Problem for Multi-armed Bandits

input: n arms where $\ell_{i,k}$ denotes the loss observed on the kth pull of the ith arm

initialize: $T_i = 1$ for all $i \in [n]$

for $t = 1, 2, 3, \ldots$

Algorithm chooses an index $I_t \in [n]$

Loss $\ell_{I_t, T_{I_t}}$ is revealed, $T_{I_t} = T_{I_t} + 1$

Algorithm outputs a recommendation $J_t \in [n]$

Receive external stopping signal, otherwise continue

- Each configuration corresponds to an arm.
- Sequence of losses of each arm is known and converges.
- The cost of obtaining the loss can be more costly than pulling it.

Practical considerations related to Bayesian optimisation

The function values y_n and $\boldsymbol{y}_{\setminus n}$ are jointly Gaussian:

$$
\begin{aligned}
p(y_n, \boldsymbol{y}_{\setminus n}) &= \text{Gaussian}\left(\begin{bmatrix} m(\boldsymbol{x}_n) \\ \boldsymbol{m}_{\setminus n} \end{bmatrix}, \begin{bmatrix} k(\boldsymbol{x}_n, \boldsymbol{x}_n) & \boldsymbol{K}_n \\ \boldsymbol{K}_n^\top & \boldsymbol{K}_{\setminus n} \end{bmatrix}\right) \\
&= \text{Gaussian}\left(\boldsymbol{m}, \boldsymbol{K}\right).
\end{aligned}
$$

The conditional $p(y_n|\boldsymbol{y}_{\setminus n})$ is Gaussian with the **conditional mean** and the **conditional variance** respectively given by

$$
\begin{aligned}
\widetilde{m}_n &= m(\boldsymbol{x}_n) + \boldsymbol{K}_n \boldsymbol{K}_{\setminus n}^{-1}(\boldsymbol{y}_{\setminus n} - \boldsymbol{m}_{\setminus n}), \\
\tilde{\sigma}_n^2 &= k(\boldsymbol{x}_n, \boldsymbol{x}_n) - \boldsymbol{K}_n \boldsymbol{K}_{\setminus n}^{-1} \boldsymbol{K}_n^\top.
\end{aligned}
$$

How to handle the hyperparameters of the surrogate model?

Let us denote the kernel parameters by $\boldsymbol{\theta}$. We view the latent functions as nuisance parameters and maximise the log-marginal wrt ς^2 and $\boldsymbol{\theta}$.

The **log-marginal likelihood** is given by

$$\ln p(\boldsymbol{y}|\varsigma, \boldsymbol{\theta}) = -\frac{n}{2}\ln 2\pi \underbrace{-\frac{1}{2}\ln|\boldsymbol{K}(\boldsymbol{\theta}) + \varsigma^2\boldsymbol{I}_n|}_{\text{complexity penality}} \underbrace{-\frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{K}(\boldsymbol{\theta}) + \varsigma^2\boldsymbol{I}_n)^{-1}\boldsymbol{y}}_{\text{data fit}}.$$

The negative log-marginal surface is non-convex and the computational complexity for its evaluation is $\mathcal{O}(n^3)$.

Which covariance function to pick?

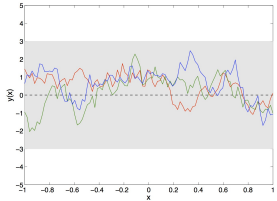
Matérn covariance function:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\ |\boldsymbol{x} - \boldsymbol{x}'|}{\ell} \right)^{\nu} \mathrm{K}_{\nu} \left(\frac{\sqrt{2\nu}\ |\boldsymbol{x} - \boldsymbol{x}'|}{\ell} \right),$$

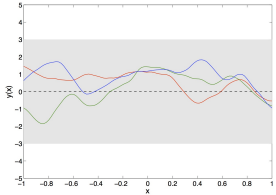
where $\ell > 0$ is the length scale and $\mathrm{K}_{\nu}(\cdot)$ is the *modified Bessel function of the second kind*.

The order $\nu > 0$ defines the roughness of the random functions as they are $\lfloor \nu - 1 \rfloor$ times differentiable:

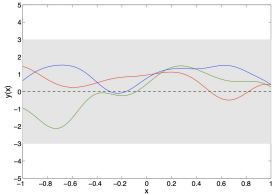
- For $\nu = p + \frac{1}{2}$ with $p \in \mathbb{N}$, the covariance function takes the form of a product of an exponential and a polynomial of order p.
- The squared exponential kernel is recovered when $\nu \to \infty$.



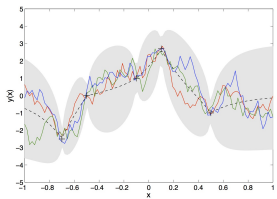
Prior $\nu = \frac{1}{2}$.



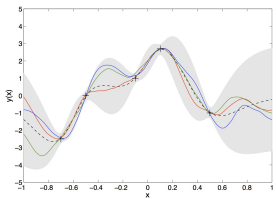
Prior $\nu = \frac{5}{2}$.



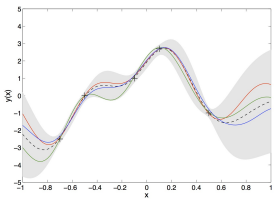
Prior $\nu \to \infty$.



Prior $\nu = \frac{1}{2}$.



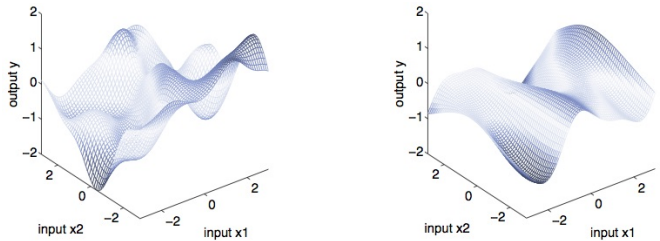
Prior $\nu = \frac{5}{2}$.



Prior $\nu \to \infty$.

Three random functions generated from (a) the prior GP and (b) the posterior GP with the Matérn kernel ($\ell = 0.25$). The observations are indicated by $+$, the means by a dashed lines and the 3 standard deviation error bars by the shaded regions.

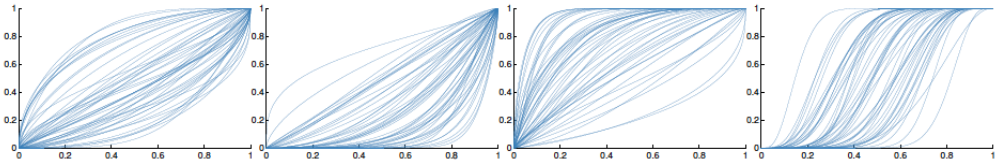
Can we identify the hyperparameters that matter?



Automatic relevance determination (**ARD**) [Mac96]:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \sum_j \frac{|x_j - x_j'|}{\ell_j} \right)^{\nu} \mathrm{K}_\nu \left(\sqrt{2\nu} \sum_j \frac{|x_j - x_j'|}{\ell_j} \right).$$

Can we handle hyperparameter transformations?



(Image credit: Snoek, et al., 2014)

- Automatic input **warping** [SSZA14]:

$$\omega : x \mapsto \omega(x) = \mathrm{BetaCDF}(x; \alpha, \beta).$$

- Learn α and β as the hyperparameters of the Gaussian process.
- Many alternatives, such as Kumaraswamy distribution: $\omega(x) = 1 - (1 - x^{\alpha})^{\beta}$.

Can we handle constraints?

$$\boldsymbol{x}_\star = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}} \{f(\boldsymbol{x}) \| c(\boldsymbol{x}) \leqslant \delta\} \qquad \text{or} \qquad \boldsymbol{x}_\star = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmin}} \{f(\boldsymbol{x}) \| P(z_c = 1|\boldsymbol{x})) \leqslant \zeta\}$$

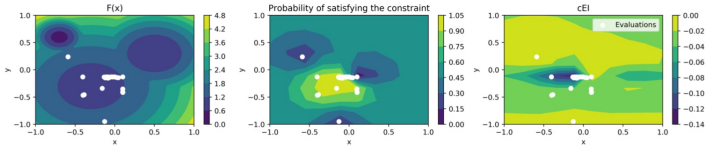


Figure 2: 2D toy example. Visualization of the evaluations proposed by cEI (above) and cMES (below). Both methods were initialized with the same 5 random points. The areas with objective values greater than 1.2 are unfeasible, and centers of the three dark valleys correspond to two local and one global optima.