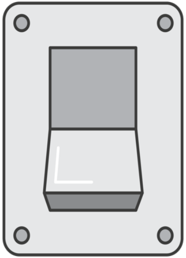
Bayesian Optimisation

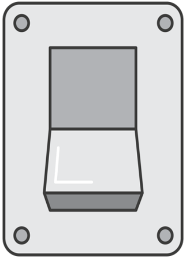
Cédric Archambeau
cedrica@amazon.com

Imperial College, London, 2017

Democratising machine learning

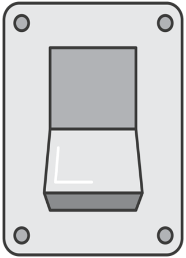


Democratising machine learning



- Abstract away algorithms
- Abstract away feature engineering

Democratising machine learning



- Abstract away algorithms
- Abstract away feature engineering
- Abstract away memory constraints
- Abstract away network constraints
- Abstract away computing infrastructure

Machine learning aims to estimate (`learn`) a statistical data model to make predictions (`generalise`) about unseen data

Machine learning aims to estimate (`learn`) a statistical data model to make predictions (`generalise`) about unseen data

1. Model the application (assuming labeled emails were collected)

   `Spam detection:`

   - Is this a (binary) classification problem?
   - Shall I use logistic regression or a SVM?
   - How should I represent emails?

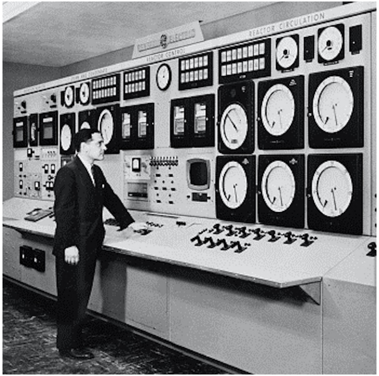
Machine learning aims to estimate (`learn`) a statistical data model to make predictions (`generalise`) about unseen data

1. Model the application (assuming labeled emails were collected)
   `Spam detection:`
   - Is this a (binary) classification problem?
   - Shall I use logistic regression or a SVM?
   - How should I represent emails?
2. Learn the model parameters from the data
   `Estimate weights of logistic regression model:`
   - Iterative least squares?
   - Stochastic gradient descent?
   - Adagrad?

Machine learning aims to estimate (`learn`) a statistical data model to make predictions (`generalise`) about unseen data

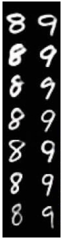
1. Model the application (assuming labeled emails were collected)
   `Spam detection:`
   - Is this a (binary) classification problem?
   - Shall I use logistic regression or a SVM?
   - How should I represent emails?
2. Learn the model parameters from the data
   `Estimate weights of logistic regression model:`
   - Iterative least squares?
   - Stochastic gradient descent?
   - Adagrad?
3. Make predictions about new data with the trained model
   `Decide if this new email is spam or not?`
   - Shall I optimise for precision?
   - Shall I optimise for recall?

The performance of machine learning depends on meta-parameters that have to be tuned with care...

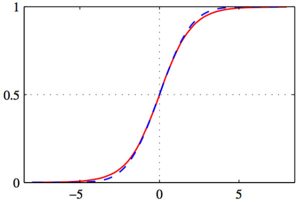


- Regularisation and (hyper)priors
- Optimisation and sampling
- Feature extraction
- Model complexity
- Decision

A toy example: digit classification with (binary) logistic regression

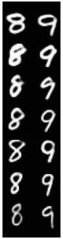


MNIST

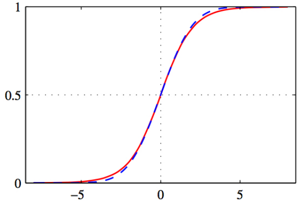


Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

A toy example: digit classification with (binary) logistic regression



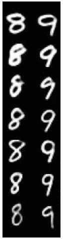
MNIST



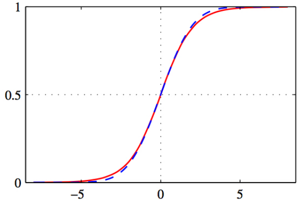
Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

- Let $\boldsymbol{x}$ be an image and t its label.

A toy example: digit classification with (binary) logistic regression



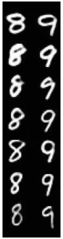
MNIST



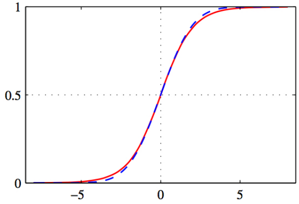
Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

- Let $\boldsymbol{x}$ be an image and t its label.
- Logistic link: $P(t = \text{“8”}) = \sigma(y(\boldsymbol{x}))$.
- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + w_0$.

A toy example: digit classification with (binary) logistic regression



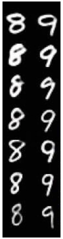
MNIST



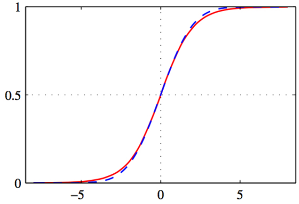
Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

- Let $\boldsymbol{x}$ be an image and t its label.
- Logistic link: $P(t = \text{“8”}) = \sigma(y(\boldsymbol{x}))$.
- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + w_0$.
- Likelihood: $P(t_1, \dots, t_n) = \prod_{i=1}^{n} \mathrm{Bernoulli}\,(P(t_n))$.
- Prior: $P(\boldsymbol{w}) = \mathrm{Gaussian}(\boldsymbol{0}, \lambda \boldsymbol{I})$.

A toy example: digit classification with (binary) logistic regression



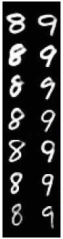
MNIST



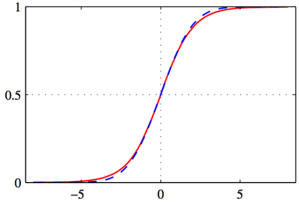
Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

- Let $\boldsymbol{x}$ be an image and t its label.
- Logistic link: $P(t = \text{“8”}) = \sigma(y(\boldsymbol{x}))$.
- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + w_0$.
- Likelihood: $P(t_1, \ldots, t_n) = \prod_{i=1}^n \mathrm{Bernoulli}\,(P(t_n))$.
- Prior: $P(\boldsymbol{w}) = \mathrm{Gaussian}(\mathbf{0}, \lambda \boldsymbol{I})$.
- Maximum a posteriori learning with stochastic gradient descent.

A toy example: digit classification with (binary) logistic regression



MNIST

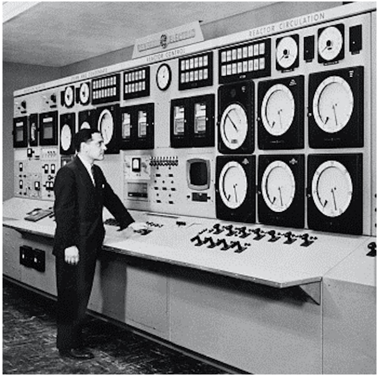


Sigmoid: $\sigma(z) = \frac{1}{1+\exp(-z)}$.

- Let $\boldsymbol{x}$ be an image and t its label.
- Logistic link: $P(t = \text{“8”}) = \sigma(y(\boldsymbol{x}))$.
- Linear discriminant: $y(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + w_0$.
- Likelihood: $P(t_1, \dots, t_n) = \prod_{i=1}^n \mathrm{Bernoulli}\,(P(t_n))$.
- Prior: $P(\boldsymbol{w}) = \mathrm{Gaussian}(\boldsymbol{0}, \lambda \boldsymbol{I})$.
- Maximum a posteriori learning with stochastic gradient descent.

$$f : \boldsymbol{x} = (\#\text{epochs, learning rate, amount of regularisation}) \mapsto f(\boldsymbol{x}) = \text{AUC}$$

The performance of machine learning depends on meta-parameters that have to be tuned with care...



- Regularisation and (hyper)priors
- Model complexity
- Optimisation and sampling
- Feature extraction
- Decision

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

A second example: Is a product review `positive` or `negative`?

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

A second example: Is a product review `positive` or `negative`?

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

- That's a binary classification problem!
- Logistic regression model with standard text features.

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]

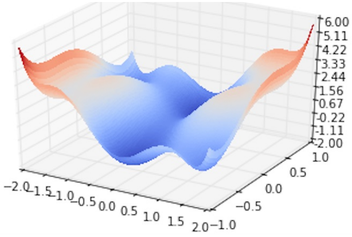
Method	Acc.
SVM-unigrams	88.62
SVM-$\{1,2\}$-grams	90.70
SVM-$\{1,2,3\}$-grams	90.68
NN-unigrams	88.94
NN-$\{1,2\}$-grams	91.10
NN-$\{1,2,3\}$-grams	91.24
LR (this work)	91.56
Bag of words CNN	91.58
Sequential CNN	92.22

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

Acc.: accuracy

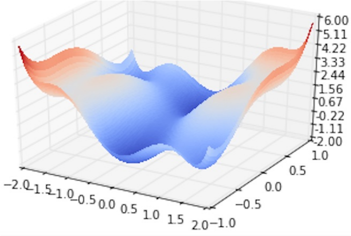
SVM: support vector machine
NN: neural network
LR: logistic regresion
CNN: convolutional neural network

Black-box optimisation



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

Black-box optimisation



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

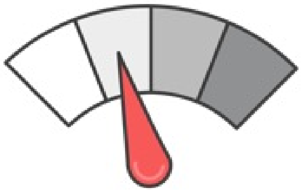
Our goal is to solve the following optimisation problem:

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

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

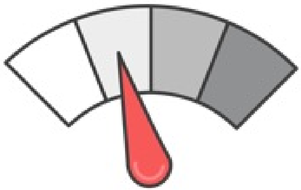
Global optimisation for hyperparameter optimisation

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



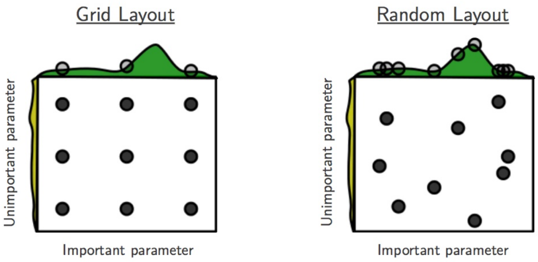
Global optimisation for hyperparameter optimisation

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



This requires iterating over hyperparameter configurations.

Two straightforward approaches



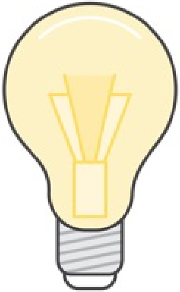
(Figure by Bergstra and Bengio, 2012)

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

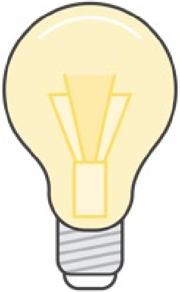
Can we do better?

(Banksy, London)

Bayesian optimisation

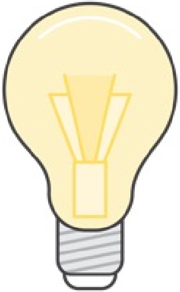


Bayesian optimisation



Global optimisation technique that adopts a probabilistic approach:

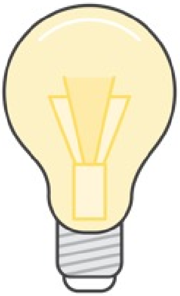
Bayesian optimisation



Global optimisation technique that adopts a probabilistic approach:

1. Builds a probabilistic model of the objective:
   - Optimises a proxy instead of the objective
   - Models the uncertainty

Bayesian optimisation



Global optimisation technique that adopts a probabilistic approach:

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

Questions?

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]

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

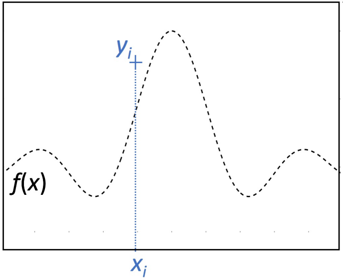
Canonical algorithm:

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

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

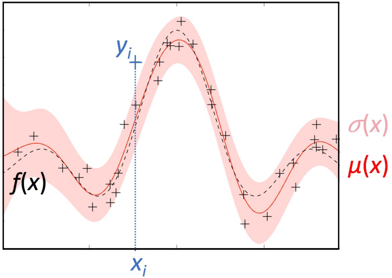


Bayesian (black-box) optimisation with Gaussian processes [JSW98]

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

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

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

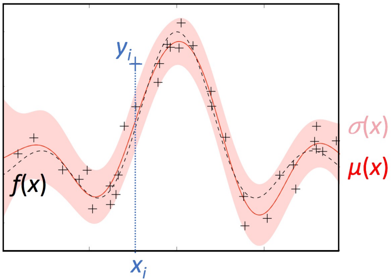


Bayesian (black-box) optimisation with Gaussian processes [JSW98]

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

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

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 D random variables based on correlations:

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

Ingredient 1: Gaussian processes for regression [RW06]

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

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

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

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

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

Ingredient 1: Gaussian processes for regression [RW06]

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

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

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

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

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

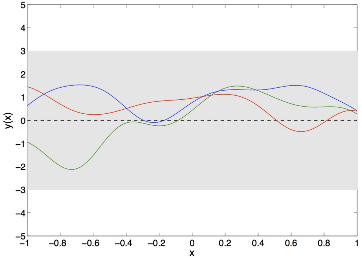
- The **posterior process** is again a Gaussian process (if Gaussian likelihood):

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

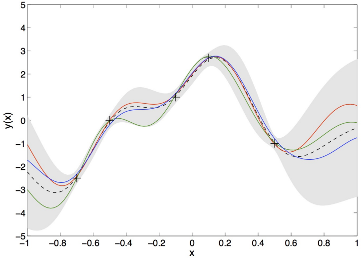
  where

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

Ingredient 1: Gaussian processes for regression [RW06]



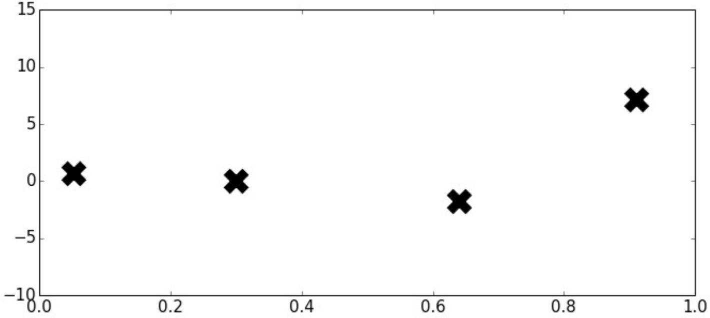
Prior.



Posterior.

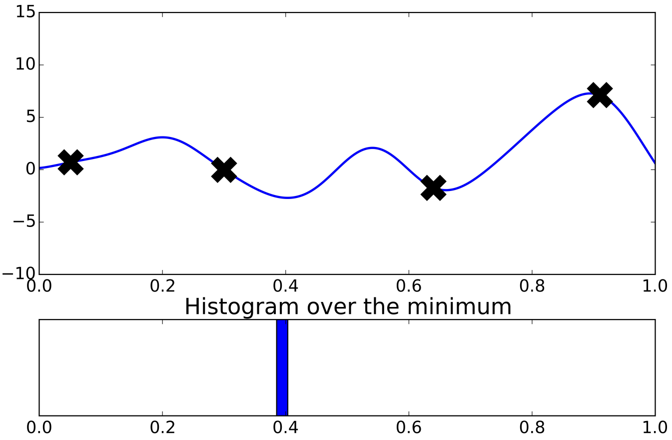
Three random functions generated from (a) the prior GP and (b) the posterior GP. An observation is indicated by a $+$, the mean function by a dashed line and the 3 standard deviation error bars by the shaded regions. We used a squared exponential covariance function.

Where is the minimum of f?



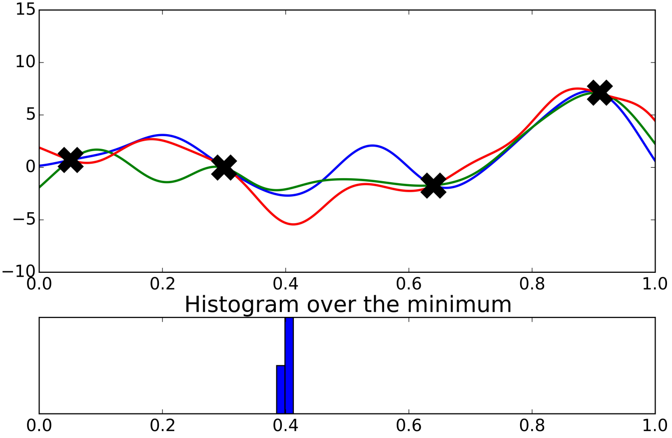
(Image credit: Javier González)

Intuitive solution



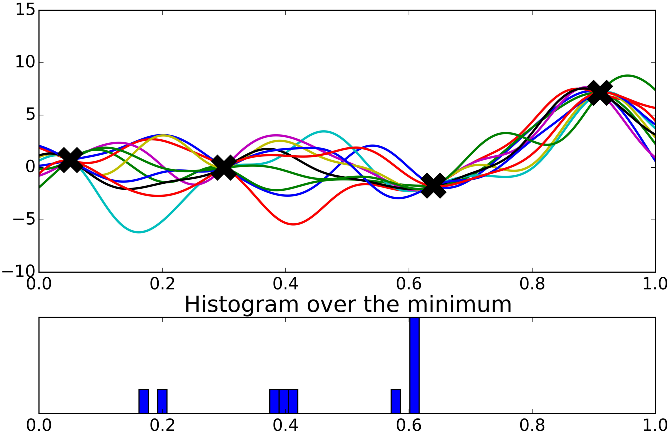
(Image credit: Javier González)

Intuitive solution



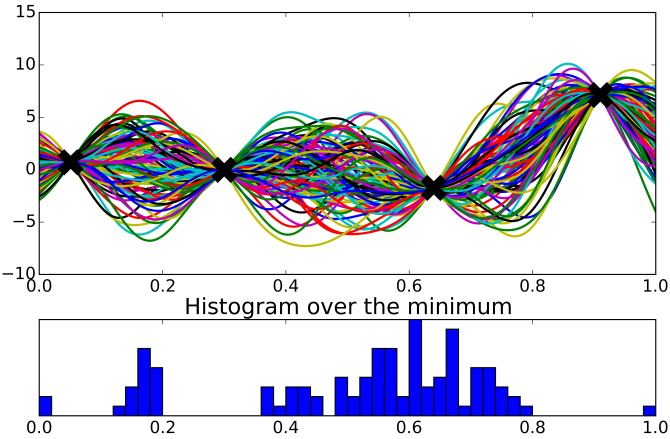
(Image credit: Javier González)

Intuitive solution



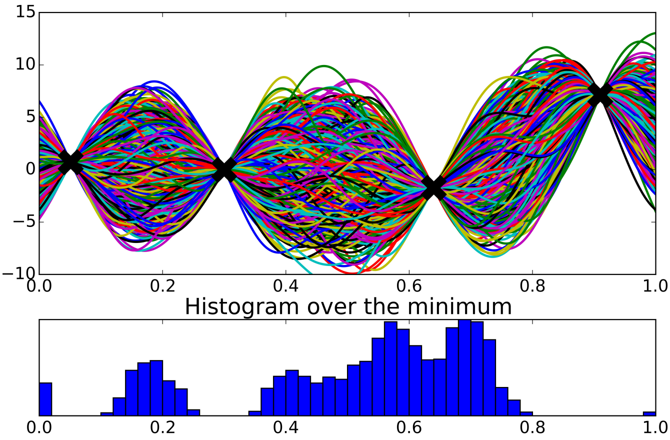
(Image credit: Javier González)

Intuitive solution



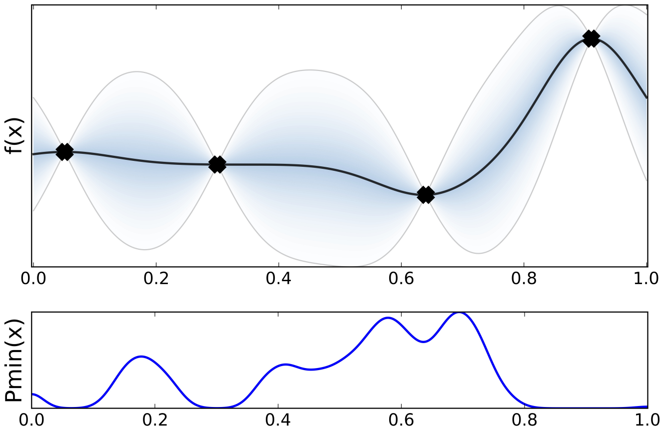
(Image credit: Javier González)

Intuitive solution



(Image credit: Javier González)

Intuitive solution



(Image credit: Javier González)

Ingredient 2: Acquisition function

Let $\mathcal{C} = \{\boldsymbol{x}_c, y_c\}$ denote a set of observed parameter-value pairs. The acquisition function is defined as follows:

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

Ingredient 2: Acquisition function

Let $\mathcal{C} = \{\boldsymbol{x}_c, y_c\}$ denote a set of observed parameter-value pairs. The acquisition function is defined as follows:

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

- Decide which are most promising regions in $\mathcal{X} \backslash \mathcal{C}$
- 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 $\mathcal{C} = \{\boldsymbol{x}_c, y_c\}$ denote a set of observed parameter-value pairs. The acquisition function is defined as follows:

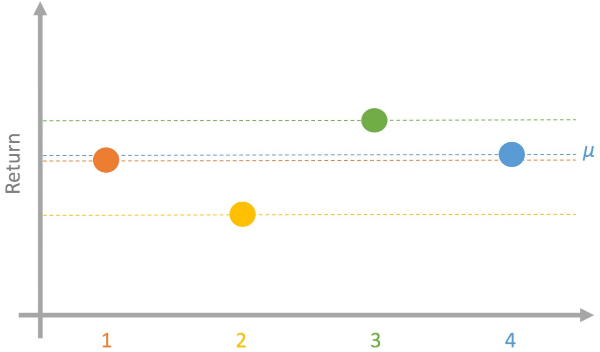
$$a : \boldsymbol{x} \mapsto a(\boldsymbol{x}|\mathcal{C}) = \mathbb{E}\left(\mathcal{A}(f, \boldsymbol{x})|\mathcal{C}\right).$$

- Decide which are most promising regions in $\mathcal{X} \backslash \mathcal{C}$
- Get as quickly as possible to "the" optimum (unlike Bayesian experimental design)
- Can be optimised with off the self solvers!
- Makes the exploration-exploitation trade-off

Exploration-exploitation trade-off

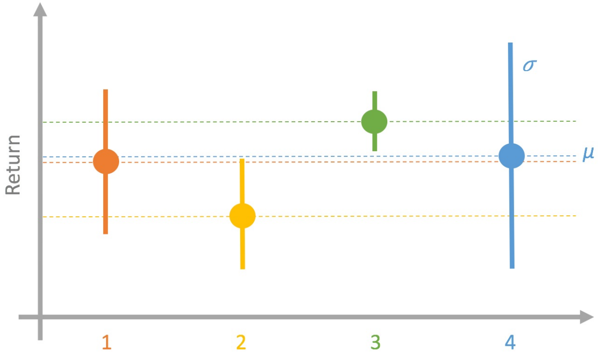
Which slot machine should I pick?

Exploration-exploitation trade-off



Which slot machine should I pick?

Exploration-exploitation trade-off



Which slot machine should I pick?

Ingredient 2: Acquisition function

- Evaluate all candidates according to an **acquisition function** $a(\boldsymbol{x})$.
- Rank them and pick the best one.

Ingredient 2: Acquisition function

- Evaluate all candidates according to an **acquisition function** $a(\boldsymbol{x})$.
- Rank them and pick the best one.
- Examples of acquisition functions:

  Let $y_\star$ be the best value observed so far and $f(\boldsymbol{x})|\boldsymbol{y} \sim \mathrm{Gaussian}\left(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})^2\right)$:

Ingredient 2: Acquisition function

- Evaluate all candidates according to an **acquisition function** $a(\boldsymbol{x})$.
- Rank them and pick the best one.
- Examples of acquisition functions:

  Let $y_\star$ be the best value observed so far and $f(\boldsymbol{x})|\boldsymbol{y} \sim \mathrm{Gaussian}\left(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})^2\right)$:

  - Lower confidence bound (GP-LCB) [SKKS09]:

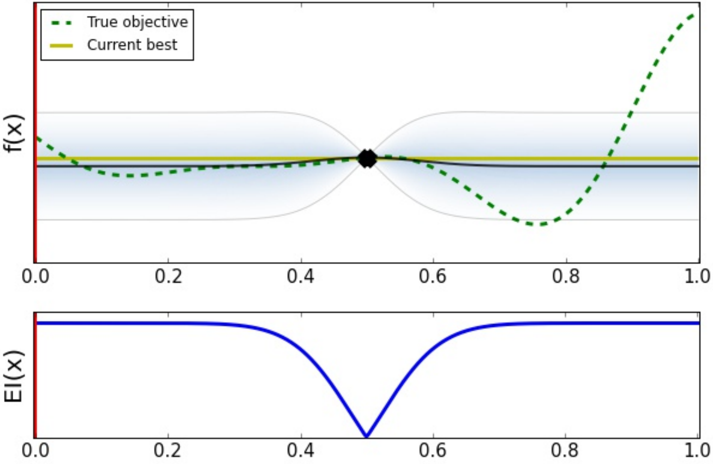
  $$a(\boldsymbol{x}) = -\mu(\boldsymbol{x}) + \alpha\sigma(\boldsymbol{x}) \quad (\alpha \geqslant 0).$$

  - **Expected improvement** (EI):

  $$a(\boldsymbol{x}) = \mathbb{E}\left(\max\{0, y_\star - f(\boldsymbol{x})\}\right).$$

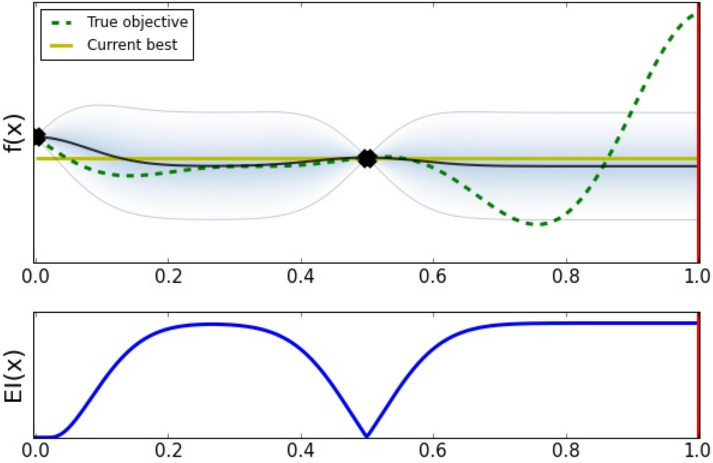
  - Probability of improvement, Thompson sampling, entropy search, etc.

Bayesian optimisation in action



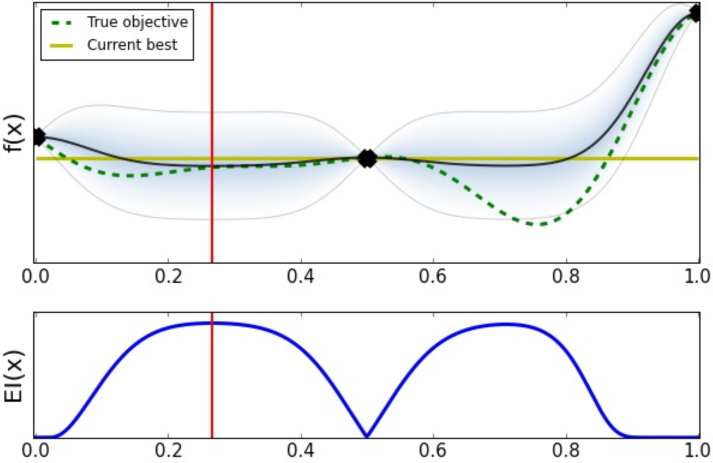
(Image credit: Javier González)

Bayesian optimisation in action



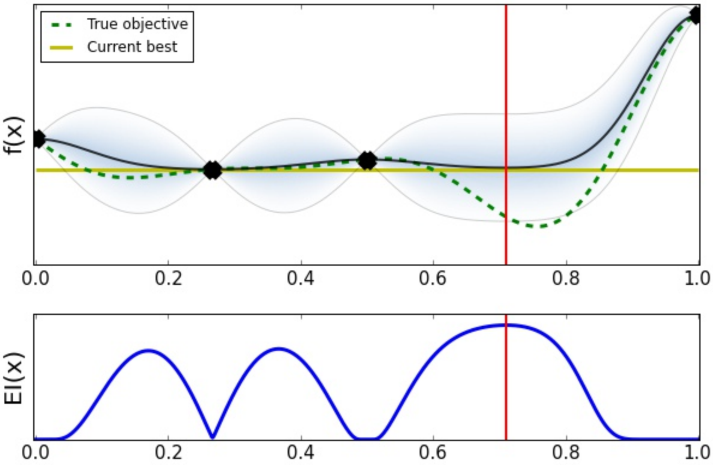
(Image credit: Javier González)

Bayesian optimisation in action



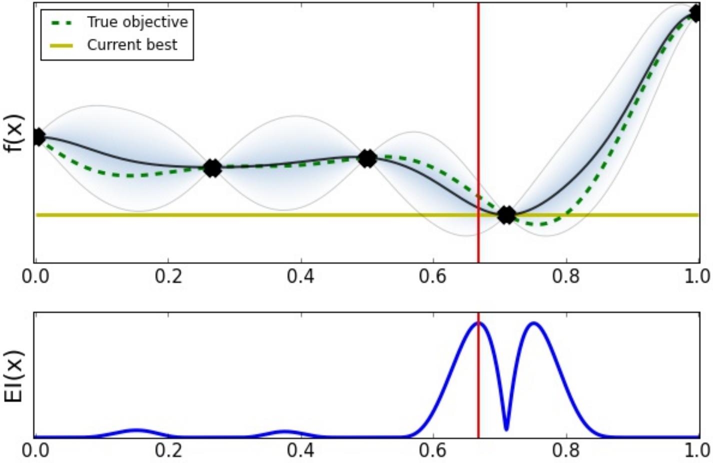
(Image credit: Javier González)

Bayesian optimisation in action



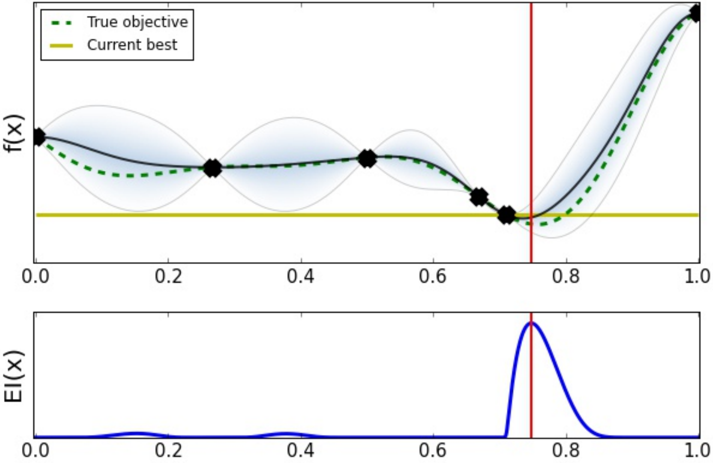
(Image credit: Javier González)

Bayesian optimisation in action



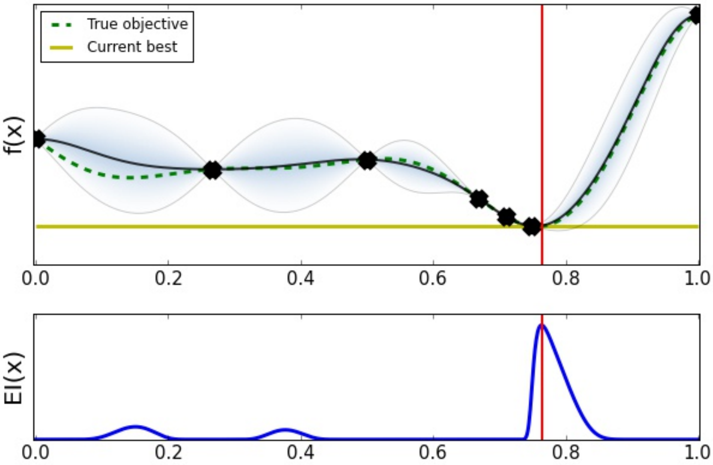
(Image credit: Javier González)

Bayesian optimisation in action



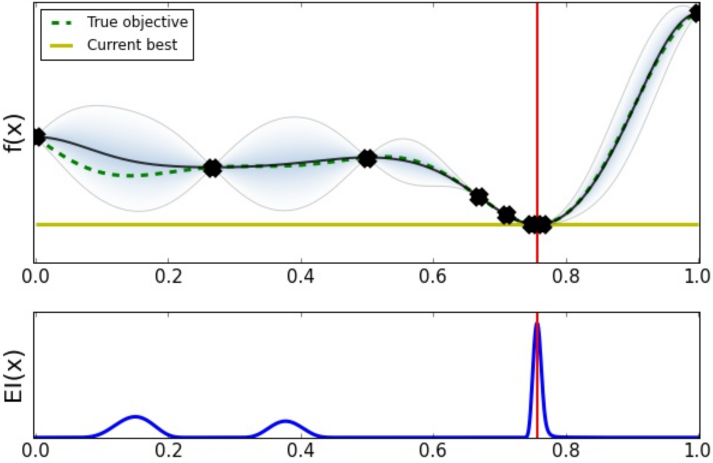
(Image credit: Javier González)

Bayesian optimisation in action



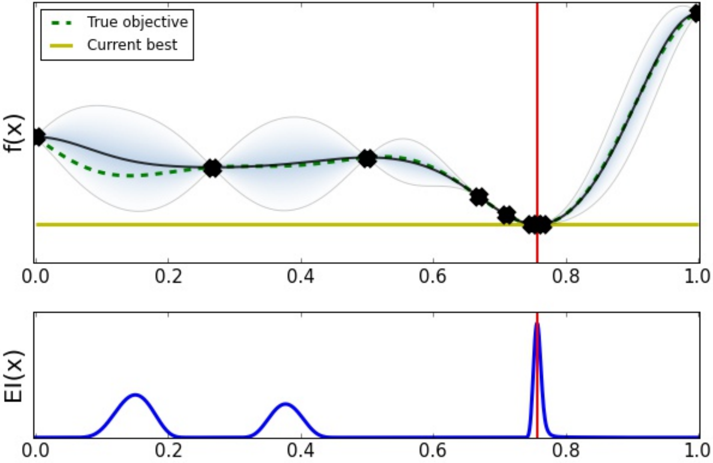
(Image credit: Javier González)

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 $\mathcal{C} = \mathcal{C} \cup \{(\mathbf{x}_{\text{new}}, y_{\text{new}})\}$
  - Update $\mathcal{M}$ with $\mathcal{C}$ `#GP posterior`
  - Update `BUDGET`

Questions?

How do we handle the hyperparameters of the surrogate model?

How do we 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 $\boldsymbol{\theta}$.

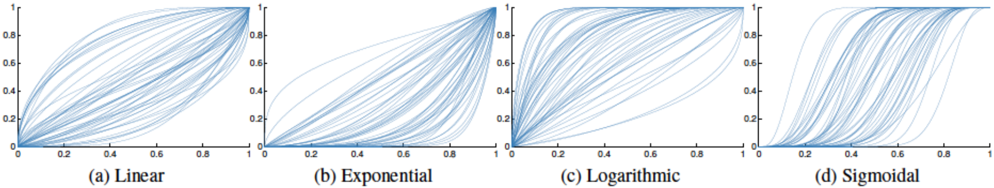
The **log-marginal likelihood** is given by

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

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

Can we handle hyperparameter transformations?

Can we handle hyperparameter transformations?



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

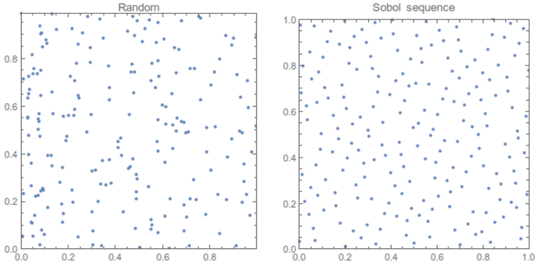
- Automatic input **warping** [SSZA14]:

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

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

How do we fill the hyperparameter space $\mathcal{X}$?

How do we fill the hyperparameter space $\mathcal{X}$?



(Image credit: Wikipedia)

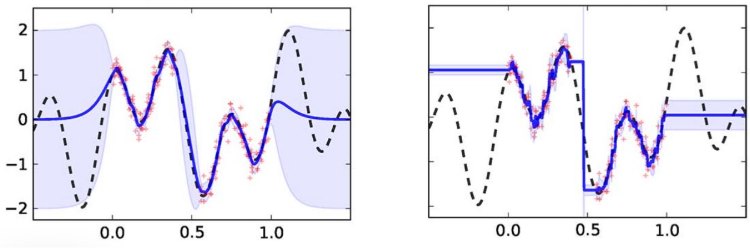
- Populate hypercube $[0,1]^D$ as densely as possible (as well as it's lower dimensional faces):

$$\text{Find sequence } \{\boldsymbol{x}_i\} \text{ such that } \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) = \int_{[0,1]^D} f(\boldsymbol{x}).$$

- Quasi random sequence generators, such as **Sobol sequences**, are better than random.

Are there other choices for the surrogate model?

Are there other choices for the surrogate model?



(Image credit: [SSW+16])

- Bayesian (black-box) optimisation with **Random Forests** [HHLB11]:

$$y(\boldsymbol{x}) = \mathrm{RF}, \qquad\qquad f(\boldsymbol{x})|\boldsymbol{y} \sim \mathrm{Gaussian}\left(\mu(\boldsymbol{x}), \Sigma(\mu(\boldsymbol{x}))\right).$$

  where $\mu(\boldsymbol{x}) \approx \frac{1}{B}\sum_i y_i(\boldsymbol{x})$ and $\Sigma(\boldsymbol{x}) \approx \frac{1}{B-1}\sum_i (y_i(\boldsymbol{x}) - \mu(\boldsymbol{x}))^2$.
- But very competitive baseline!

Reference material

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 Ryan Adams (2014): A Tutorial on Bayesian Optimization for Machine Learning. *CIFAR NCAP Summer School*.

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

Very brief historical overview

- Closely related to optimal design of experiments, dating back to *Kirstine Smith* (1918).
- As Bayesian optimisation, studied first by *Kushner* (1964), then by *Mockus* (1978), and more recently by *Jones, et al.* (1998).

Very brief historical overview

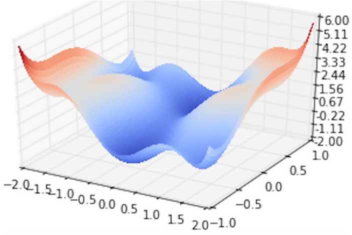
- Closely related to optimal design of experiments, dating back to *Kirstine Smith* (1918).
- As Bayesian optimisation, studied first by *Kushner* (1964), then by *Mockus* (1978), and more recently by *Jones, et al.* (1998).
- Multiple workshops at NIPS (`bayesopt.com`) and ICML (`www.automl.org`)

Very brief historical overview

- Closely related to optimal design of experiments, dating back to *Kirstine Smith* (1918).
- As Bayesian optimisation, studied first by *Kushner* (1964), then by *Mockus* (1978), and more recently by *Jones, et al.* (1998).
- Multiple workshops at NIPS (`bayesopt.com`) and ICML (`www.automl.org`)
- Open source software:
  SMAC (`http://www.cs.ubc.ca/labs/beta/Projects/SMAC/`) – RF,
  HyperOpt (`http://jaberg.github.io/hyperopt/`) – TPE,
  Spearmint (`https://github.com/JasperSnoek/spearmint`) – GP,
  GPyOpt (`https://github.com/SheffieldML/GPyOpt`) – GP,
  BayesOpt (`http://rmcantin.bitbucket.org/`) – GP,
- Challenges and benchmarks (HPOLib: `www.automl.org/hpolib.html`)!

Questions?

Black-box optimisation with (tree-structured) dependencies [JAGS17]



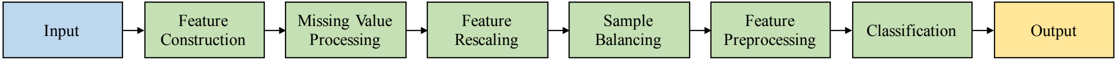
- The function f we wish to optimise can be non-concave.
- The number of hyperparameters p is moderate (typically < 20).

Our goal is to solve the following optimisation problem:

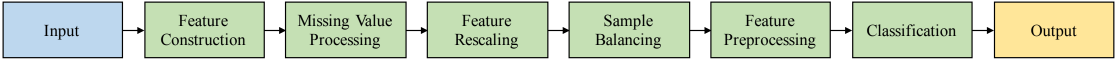
$$\boldsymbol{x}_\star = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}).$$

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

Example 1: Data analytics pipeline [THHLB13, FKE+15, ZBSS16]

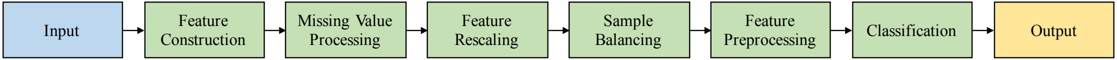


Example 1: Data analytics pipeline [THHLB13, FKE+15, ZBSS16]



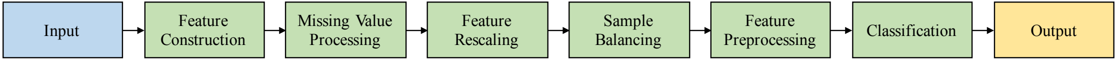
- $f(\mathbf{x})$ measures the quality of entire pipeline with hyperparameter(s) $\mathbf{x}$

Example 1: Data analytics pipeline [THHLB13, FKE+15, ZBSS16]



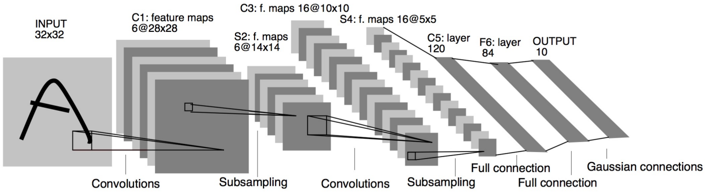
- $f(\mathbf{x})$ measures the quality of entire pipeline with hyperparameter(s) $\mathbf{x}$
- Evaluating $f(\mathbf{x})$ is possibly **costly**

Example 1: Data analytics pipeline [THHLB13, FKE+15, ZBSS16]



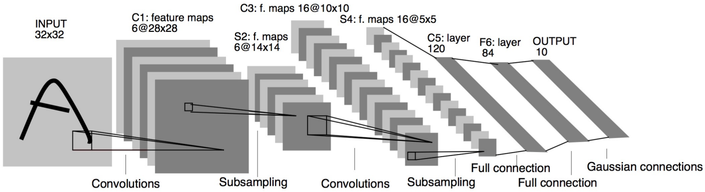
- $f(\mathbf{x})$ measures the quality of entire pipeline with hyperparameter(s) $\mathbf{x}$
- Evaluating $f(\mathbf{x})$ is possibly **costly**
- The search space $\mathcal{X}$ can be large:
  - Feature processing parameters
  - Dimensionality reduction method
  - Dimensionality reduction parameters
  - Classifier type
  - Classifier hyperparameters
  - ...

Example 2: Deep learning [SLA12, SRS$^+$15, KFB$^+$16]



LeNet5 [LBBH98]

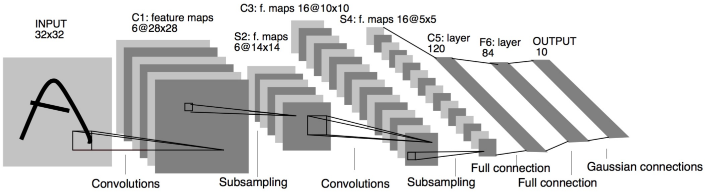
Example 2: Deep learning [SLA12, SRS+15, KFB+16]



LeNet5 [LBBH98]

- $f(\mathbf{x})$ measures the quality of deep neural network with hyperparameter(s) $\mathbf{x}$

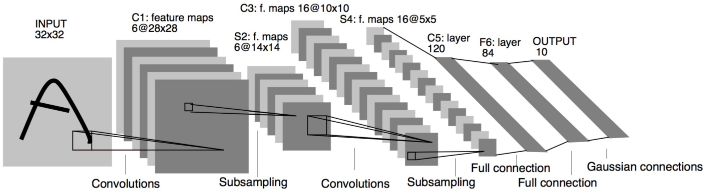
Example 2: Deep learning [SLA12, SRS+15, KFB+16]



LeNet5 [LBBH98]

- $f(\mathbf{x})$ measures the quality of deep neural network with hyperparameter(s) $\mathbf{x}$
- Evaluating $f(\mathbf{x})$ is very **costly** $\approx$ up to weeks

Example 2: Deep learning [SLA12, SRS+15, KFB+16]



LeNet5 [LBBH98]

- $f(\mathbf{x})$ measures the quality of deep neural network with hyperparameter(s) $\mathbf{x}$
- Evaluating $f(\mathbf{x})$ is very **costly** $\approx$ up to weeks
- The search space $\mathcal{X}$ can be large:
  - Architecture: # hidden layers, activation functions, ...
  - Model complexity: regularization, dropout, ...
  - Optimisation parameters: learning rates, momentum, batch size, ...

What is a structured search space $\mathcal{X}$?

What is a structured search space $\mathcal{X}$?

- The search space $\mathcal{X}$ exhibits **conditional** relationships, such that

$$\mathcal{X} = \mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_K.$$

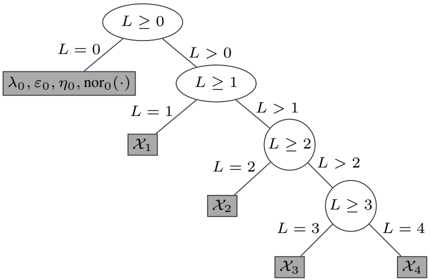
- Depending on some values in $\mathcal{X}_i$, values in $\mathcal{X}_j$ are irrelevant:
  - **Data analytics pipeline:**

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

  - **Feedforward neural nets:**

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

Tuning of feedforward neural nets



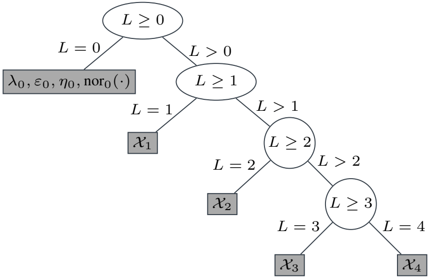
$$\mathcal{X}_k \triangleq \left\{ \lambda_k, \{u_j^{(k)}\}_{j=1}^{k}, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot) \right\}$$

- L: Number of hidden layers in $\{0, 1, 2, 3, 4\}$
- λ: Regularization parameter
- u_j: Number of units in j-th layer
- ε, η: Stopping criterion and learning rate of `Adam` [KB14]
- $\mathrm{nor}(\cdot)$: Normalization of the dataset
- $\mathrm{act}(\cdot)$: Activation function

Naive approach: Agnostic to the structure

For $\mathbf{x} \in \mathcal{X}$,

$$
\begin{aligned}
f(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}) \\
y|f(\mathbf{x}) &\sim \mathcal{N}(f(\mathbf{x}), \varsigma^2)
\end{aligned}
$$

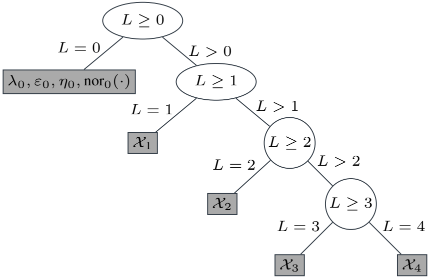


$$\mathcal{X}_k \triangleq \left\{ \lambda_k, \{u_j^{(k)}\}_{j=1}^{k}, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot) \right\}$$

Naive approach: Agnostic to the structure

For $\mathbf{x} \in \mathcal{X}$,

$$
\begin{aligned}
f(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}) \\
y|f(\mathbf{x}) &\sim \mathcal{N}(f(\mathbf{x}), \varsigma^2)
\end{aligned}
$$



$$\mathcal{X}_k \triangleq \left\{ \lambda_k, \{u_j^{(k)}\}_{j=1}^{k}, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot) \right\}$$

- Single, joint model
- Ignores conditional dependencies: $\mathcal{X} = \cancel{\mathcal{X}_0 \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_K}$.
- Complexity: $\mathcal{O}\big((\sum_p n_p)^3\big)$.

Baseline: Independent models [BBBK11]

For $\mathbf{x} \in \mathcal{X}_{p_0}$,

$$\begin{aligned} f_{p_0}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_0}) \\ y|f_{p_0}(\mathbf{x}) &\sim \mathcal{N}(f_{p_0}(\mathbf{x}), \varsigma_{p_0}{}^2) \end{aligned}$$

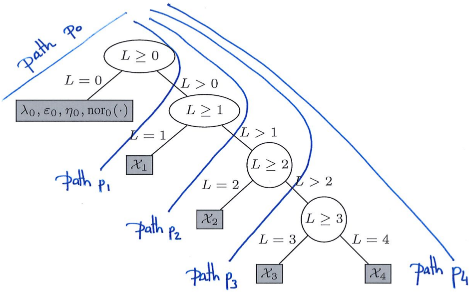
For $\mathbf{x} \in \mathcal{X}_{p_1}$,

$$\begin{aligned} f_{p_1}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_1}) \\ y|f_{p_1}(\mathbf{x}) &\sim \mathcal{N}(f_{p_1}(\mathbf{x}), \varsigma_{p_1}{}^2) \end{aligned}$$

$\vdots$

For $\mathbf{x} \in \mathcal{X}_{p_4}$,

$$\begin{aligned} f_{p_4}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_4}) \\ y|f_{p_4}(\mathbf{x}) &\sim \mathcal{N}(f_{p_4}(\mathbf{x}), \varsigma_{p_4}{}^2) \end{aligned}$$



$$\mathcal{X}_k \triangleq \left\{\lambda_k, \{u_j^{(k)}\}_{j=1}^k, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot)\right\}$$

Baseline: Independent models [BBBK11]

For $\mathbf{x} \in \mathcal{X}_{p_0}$,

$$\begin{aligned} f_{p_0}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_0}) \\ y|f_{p_0}(\mathbf{x}) &\sim \mathcal{N}(f_{p_0}(\mathbf{x}), \varsigma_{p_0}{}^2) \end{aligned}$$

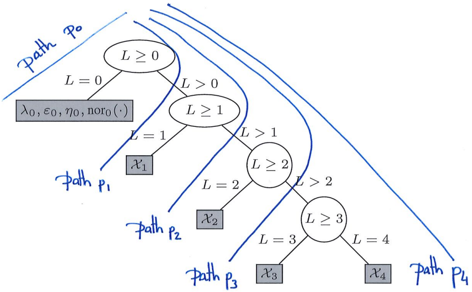
For $\mathbf{x} \in \mathcal{X}_{p_1}$,

$$\begin{aligned} f_{p_1}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_1}) \\ y|f_{p_1}(\mathbf{x}) &\sim \mathcal{N}(f_{p_1}(\mathbf{x}), \varsigma_{p_1}{}^2) \end{aligned}$$

$\vdots$

For $\mathbf{x} \in \mathcal{X}_{p_4}$,

$$\begin{aligned} f_{p_4}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_4}) \\ y|f_{p_4}(\mathbf{x}) &\sim \mathcal{N}(f_{p_4}(\mathbf{x}), \varsigma_{p_4}{}^2) \end{aligned}$$

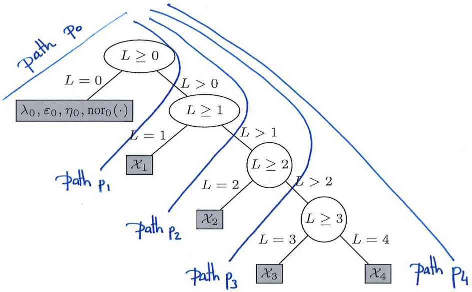


$$\mathcal{X}_k \triangleq \left\{\lambda_k, \{u_j^{(k)}\}_{j=1}^k, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot)\right\}$$

- No sharing of information across leaves
- Compare leaves via utility functions
- $\mathcal{O}\left(\sum_p n_p^3\right)$ vs. $\mathcal{O}\left((\sum_p n_p)^3\right)$.

Tree-structured sharing

Joint prior on the mean:: $\mathbf{c} = [c_1, \ldots, c_V] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{c}})$



$$\mathcal{X}_k \triangleq \left\{\lambda_k, \{u_j^{(k)}\}_{j=1}^{k}, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot)\right\}$$

Tree-structured sharing

Joint prior on the mean:: $\mathbf{c} = [c_1, \ldots, c_V] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{c})$

For $\mathbf{x} \in \mathcal{X}_{p_0}$,

$$\begin{aligned} f_{p_0}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_0}) \\ y|f_{p_0}(\mathbf{x}), \mathbf{c} &\sim \mathcal{N}(f_{p_0}(\mathbf{x}) + \textstyle\sum_{v \in p_0} c_v, \varsigma_{p_0}{}^2) \end{aligned}$$

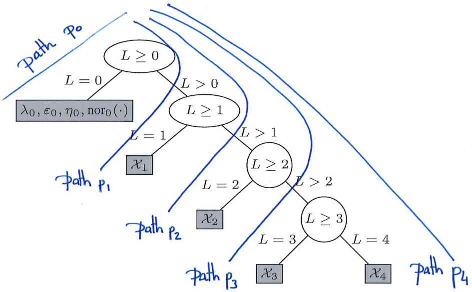
For $\mathbf{x} \in \mathcal{X}_{p_1}$,

$$\begin{aligned} f_{p_1}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_1}) \\ y|f_{p_1}(\mathbf{x}), \mathbf{c} &\sim \mathcal{N}(f_{p_1}(\mathbf{x}) + \textstyle\sum_{v \in p_1} c_v, \varsigma_{p_1}{}^2) \end{aligned}$$

$\vdots$

For $\mathbf{x} \in \mathcal{X}_{p_4}$,

$$\begin{aligned} f_{p_4}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_4}) \\ y|f_{p_4}(\mathbf{x}), \mathbf{c} &\sim \mathcal{N}(f_{p_4}(\mathbf{x}) + \textstyle\sum_{v \in p_4} c_v, \varsigma_{p_4}{}^2) \end{aligned}$$



$$\mathcal{X}_k \triangleq \left\{ \lambda_k, \{u_j^{(k)}\}_{j=1}^k, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot) \right\}$$

Tree-structured sharing

Joint prior on the mean:: $\mathbf{c} = [c_1, \ldots, c_V] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{c}})$

For $\mathbf{x} \in \mathcal{X}_{p_0}$,

$$\begin{aligned} f_{p_0}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_0}) \\ y|f_{p_0}(\mathbf{x}), \mathbf{c} &\sim \mathcal{N}(f_{p_0}(\mathbf{x}) + \textstyle\sum_{v \in p_0} c_v, \varsigma_{p_0}{}^2) \end{aligned}$$

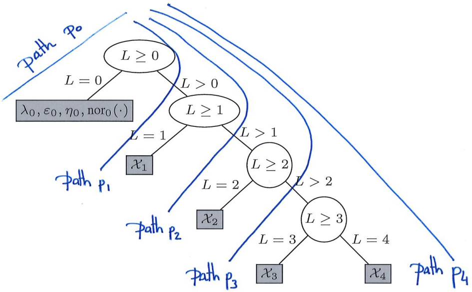
For $\mathbf{x} \in \mathcal{X}_{p_1}$,

$$\begin{aligned} f_{p_1}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_1}) \\ y|f_{p_1}(\mathbf{x}), \mathbf{c} &\sim \mathcal{N}(f_{p_1}(\mathbf{x}) + \textstyle\sum_{v \in p_1} c_v, \varsigma_{p_1}{}^2) \end{aligned}$$

$\vdots$

For $\mathbf{x} \in \mathcal{X}_{p_4}$,

$$\begin{aligned} f_{p_4}(\mathbf{x}) &\sim \mathcal{GP}(0, \mathcal{K}_{p_4}) \\ y|f_{p_4}(\mathbf{x}), \mathbf{c} &\sim \mathcal{N}(f_{p_4}(\mathbf{x}) + \textstyle\sum_{v \in p_4} c_v, \varsigma_{p_4}{}^2) \end{aligned}$$



$$\mathcal{X}_k \triangleq \left\{\lambda_k, \{u_j^{(k)}\}_{j=1}^k, \varepsilon_k, \eta_k, \mathrm{nor}_k(\cdot), \mathrm{act}_k(\cdot)\right\}$$

- **Sharing** of information across leaves: if p similar to p', $\sum_{v \in p} c_v \approx \sum_{v \in p'} c_v$.
- $\mathcal{O}\left(\sum_p n_p^3 + V^3\right)$ vs. $\mathcal{O}\left((\sum_p n_p)^3\right)$.

The induced kernel corresponds to the intersection kernel

Let $\mathbf{H} = [\mathbf{H}_p] \in \mathbb{R}^{V \times n}$ be stacked binary masks and $\mathbf{K}^{\text{block}} \in \mathbb{R}^{n \times n}$ be the block-diagonal matrix with blocks $\mathbf{K}_p$.

The induced kernel corresponds to the intersection kernel

Let $\mathbf{H} = [\mathbf{H}_p] \in \mathbb{R}^{V \times n}$ be stacked binary masks and $\mathbf{K}^{\text{block}} \in \mathbb{R}^{n \times n}$ be the block-diagonal matrix with blocks $\mathbf{K}_p$.

The marginal likelihood is given by

$$P(\mathbf{y}) = \int_{\mathbf{f},\mathbf{c}} P(\mathbf{y}, \mathbf{f}, \mathbf{c}) = \mathcal{N}\left(\mathbf{0}, \mathbf{H}^\top \boldsymbol{\Sigma}_c \mathbf{H} + \mathbf{K}^{\text{block}} + \text{diag}\{\varsigma^2\}\right).$$

The induced kernel corresponds to the intersection kernel

Let $\mathbf{H} = [\mathbf{H}_p] \in \mathbb{R}^{V \times n}$ be stacked binary masks and $\mathbf{K}^{\text{block}} \in \mathbb{R}^{n \times n}$ be the block-diagonal matrix with blocks $\mathbf{K}_p$.

The marginal likelihood is given by

$$P(\mathbf{y}) = \int_{\mathbf{f},\mathbf{c}} P(\mathbf{y}, \mathbf{f}, \mathbf{c}) = \mathcal{N}\left(\mathbf{0}, \mathbf{H}^\top \boldsymbol{\Sigma}_c \mathbf{H} + \mathbf{K}^{\text{block}} + \mathrm{diag}\{\varsigma^2\}\right).$$

If we assume that $\boldsymbol{\Sigma}_c = \sigma_c^2 \mathbf{I}_V$, then

$$\mathbf{H}^\top \boldsymbol{\Sigma}_c \mathbf{H} = \left[\sigma_c^2 (\mathbf{h}_p^\top \mathbf{h}_{p'}) \mathbf{1}_{n_p} \mathbf{1}_{n_{p'}}^\top\right]_{p,p'}.$$

- Diagonal blocks are proportional to the length of path p.
- Off-diagonal blocks are proportional to the path overlap between p and p'.

Two-step acquisition function to reduce complexity

$$(\boldsymbol{x}_\star, p_\star) = \operatorname*{argmax}_{p \in \mathcal{P}, \boldsymbol{x} \in \mathcal{X}_p} a(\boldsymbol{x}, p | \mathcal{D}_n).$$

Two-step acquisition function to reduce complexity

$$(\boldsymbol{x}_\star, p_\star) = \underset{p\in\mathcal{P}, \boldsymbol{x}\in\mathcal{X}_p}{\operatorname{argmax}} \ a(\boldsymbol{x}, p|\mathcal{D}_n).$$

- Exploit the tree structure through a **path acquisition** function:

$$p_\star = \underset{p\in\mathcal{P}}{\operatorname{argmax}} \ a(p|\mathcal{D}_n), \quad \boldsymbol{x}_\star = \underset{\boldsymbol{x}\in\mathcal{X}_{p_\star}}{\operatorname{argmax}} \ a(\boldsymbol{x}, p_\star|\mathcal{D}_n).$$

Two-step acquisition function to reduce complexity

$$(\boldsymbol{x}_\star, p_\star) = \underset{p \in \mathcal{P}, \boldsymbol{x} \in \mathcal{X}_p}{\operatorname{argmax}} \; a(\boldsymbol{x}, p | \mathcal{D}_n).$$

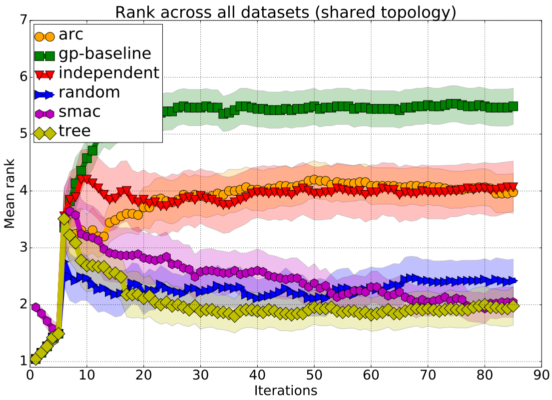
- Exploit the tree structure through a **path acquisition** function:

$$p_\star = \underset{p \in \mathcal{P}}{\operatorname{argmax}} \, a(p | \mathcal{D}_n), \quad \boldsymbol{x}_\star = \underset{\boldsymbol{x} \in \mathcal{X}_{p_\star}}{\operatorname{argmax}} \, a(\boldsymbol{x}, p_\star | \mathcal{D}_n).$$

- The path EI is given by

$$a(p | \mathcal{D}_n) = \mathbb{E}\left(\max\{0, y_\star - \mathbf{h}_p^\top \mathbf{c}\} \right).$$

Experiment with feedforward neural network for classification



- Binary classification: 45 datasets from LIBSVM repository
- Mean rank based on mean classification accuracy for each dataset (25 replications)
- `Arc` [SDS+14], `Smac` [HHLB11], `Random` [BB12]

Conclusion

Bayesian optimisation automates machine learning:

- Algorithm tuning
- Model tuning
- Pipeline tuning

Conclusion

Bayesian optimisation automates machine learning:

- Algorithm tuning
- Model tuning
- Pipeline tuning

Bayesian optimisation is a model-based approach that can leverage side information:

- For example, it can exploit dependency structure
- Approach can leverage **shared variables** (aka features) at inner nodes `#see paper [JAGS17]`

Conclusion

Bayesian optimisation automates machine learning:

- Algorithm tuning
- Model tuning
- Pipeline tuning

Bayesian optimisation is a model-based approach that can leverage side information:

- For example, it can exploit dependency structure
- Approach can leverage **shared variables** (aka features) at inner nodes `#see paper [JAGS17]`

https://sheffieldml.github.io/GPyOpt/

References I

James Bergstra and Yoshua Bengio.
Random search for hyper-parameter optimization.
Journal of Machine Learning Research, 13:281–305, 2012.

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.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter.
Efficient and robust automated machine learning.
In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970, 2015.

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

Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger.
Bayesian optimization with tree-structured dependencies.
In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1655–1664, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

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.

Diederik Kingma and Jimmy Ba.
Adam: A method for stochastic optimization.
Technical report, preprint arXiv:1412.6980, 2014.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter.
Fast Bayesian optimization of machine learning hyperparameters on large datasets.
Technical report, preprint arXiv:1605.07079, 2016.

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.

References III

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

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

Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, and Michael A Osborne.
Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces.
Technical report, preprint arXiv:1409.4011, 2014.

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.

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.

References IV

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.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown.
Auto-weka: Combined selection and hyperparameter optimization of classification algorithms.
In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

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

References V

Yuyu Zhang, Mohammad Taha Bahadori, Hang Su, and Jimeng Sun.
Flash: Fast bayesian optimization for data analytic pipelines.
In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2065–2074. ACM, 2016.

cedrica@amazon.com