Beyond manual tuning of hyperparameters

Matej Uhrín

Department of Computer Science

November 23, 2018

Outline

Introduction

Why is it important

- In machine learning, the difference between state of the art and average performance is often only the parameters.
- In testing, better chosen test cases = safer software.
- In general, less time spent by evaluation = more time spent in research.

Formal Definition

$$x^\star = argmin_x f(x) \quad x \in \mathcal{X} \tag{1}$$

Where f is a costly objective function, x is a single parameter configuration from parameter space $\mathcal{X}$. Value of the objective function $y = f(x)$ and $y^\star = f(x^\star)$

Formal Definition

Example

Problem:Classification of 8x8 digit images from sklearn.datasets.digits

Costly objective function $f(x)$ and parameter space $\mathcal{X}$:

```
def train_network(params, x_train, x_test ...):
    net = algorithms.Momentum(params)
    net.train(x_train, y_train, x_test, y_test ...)
    return net.prediction_error(x_test,y_test)
```

param	**distr**	**values**
step size	log-uniform	$x \in [0.01, 0.5]$
batch size	log-uniform-integer	$x \in [16, 512]$
activ. f.	Categorical	$x \in \{Relu, PRelu, Sigmoid...\}$
nhidden l.	Categorical	$x \in \{1, 2\}$
nunits in 1st l	unif-integer	$x \in [50, 1000]$
...	...	...

Existing Approaches

Overview

- Manual tuning
- Grid search
- Random search
- Bayesian model-based optimization
- Gradient based optimization
- Evolutionary optimization

Existing Approaches

Grid Search

- Define a set of parameter values, train model for all possible parameter combinations and select the best one. For 5 parameters, 10 values each, 10 minutes time cost for objective function:

$$10^5 \cdot 10 \approx 2 \ \textit{years} \tag{2}$$

- This method is a good choice only when model can train quickly, which is not the case for typical neural networks.

Random Search

- Only use randomly selected subsets of parameters

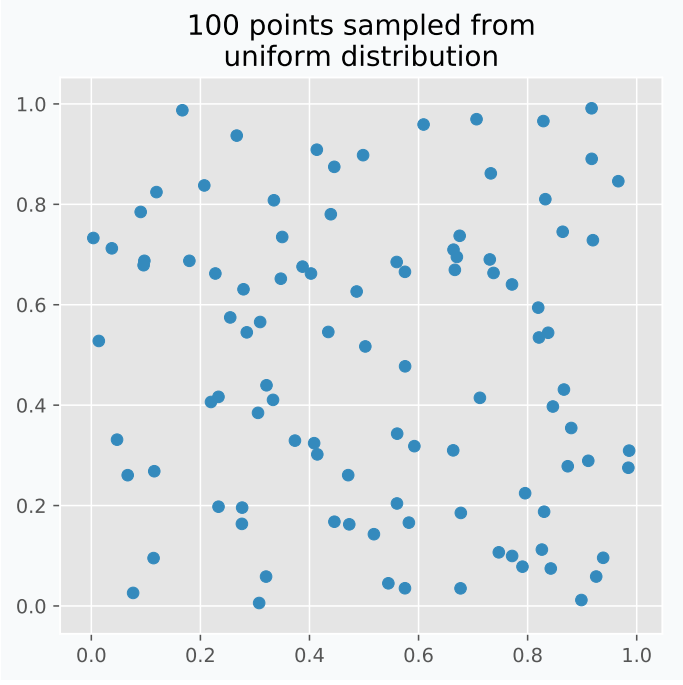


Figure 1: Two params $\in [0, 1]$ uniform distr.

Parameter space not really covered, some configurations tests may be redundant. We can improve with low-discrepancy sequences.

Random Search

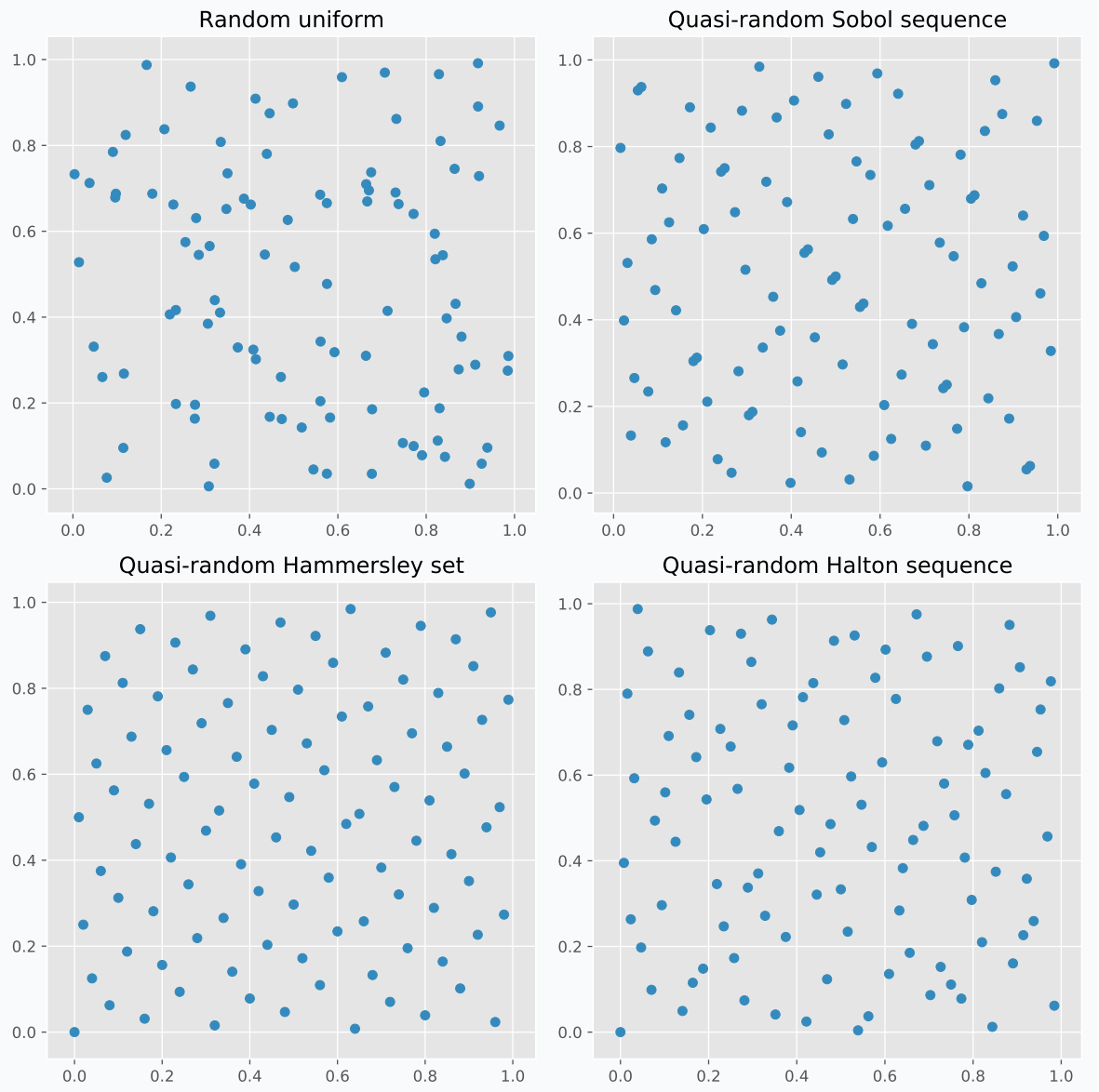


Figure 2: Random uniform comparison with quasi-random sequences

Random Search

Review

- More effective than grid search.
- May not cover parameter space well.
- We can improve the "coverage" of parameter space with quasi random sequences.
- These techniques do not provide good results in higher dimensions. E.g. Halton and Hammersley set do not work well for dimensions higher than 10.
- Not suitable for optimization of large neural networks.

Hand Tuning

Example

Problem: 8x8 digits classification. Goal: Selecting the best number of units in the hidden layer. $nhidden \in [10, 1000]$

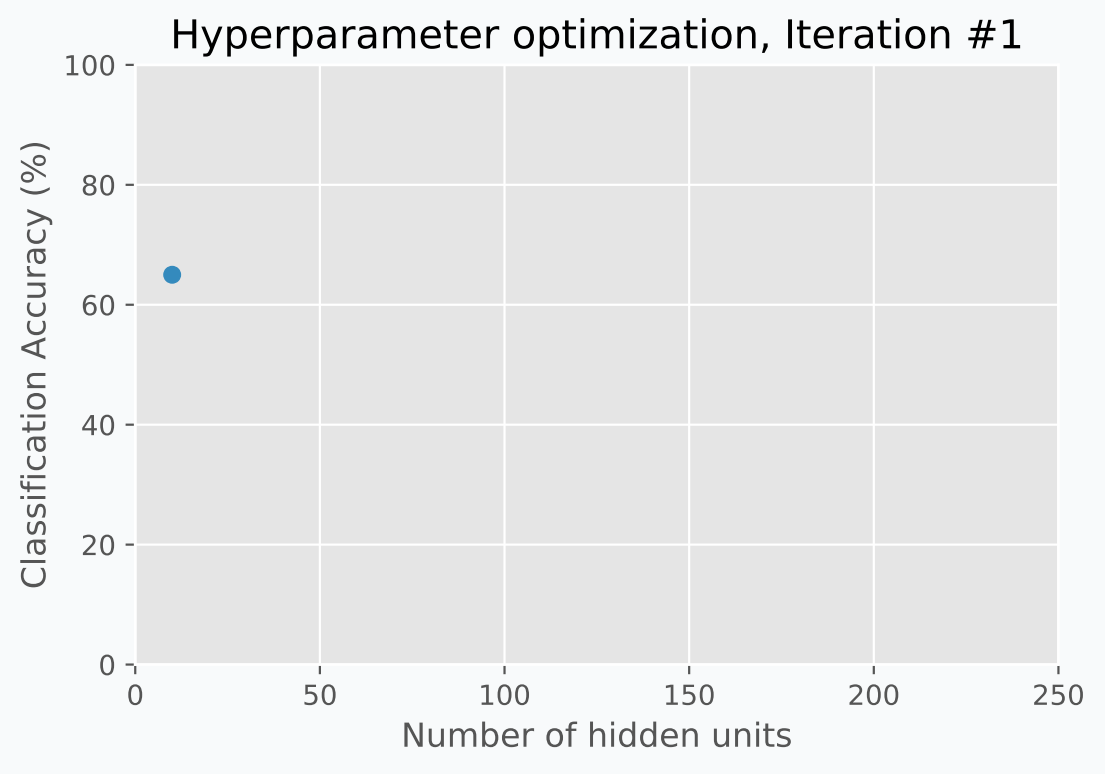


Figure 3: Our initial param config is 10 hidden units. Accuracy 65%

Hand Tuning

Observing the result for 10 units. We make a decision to skip many of the configurations and try 100.

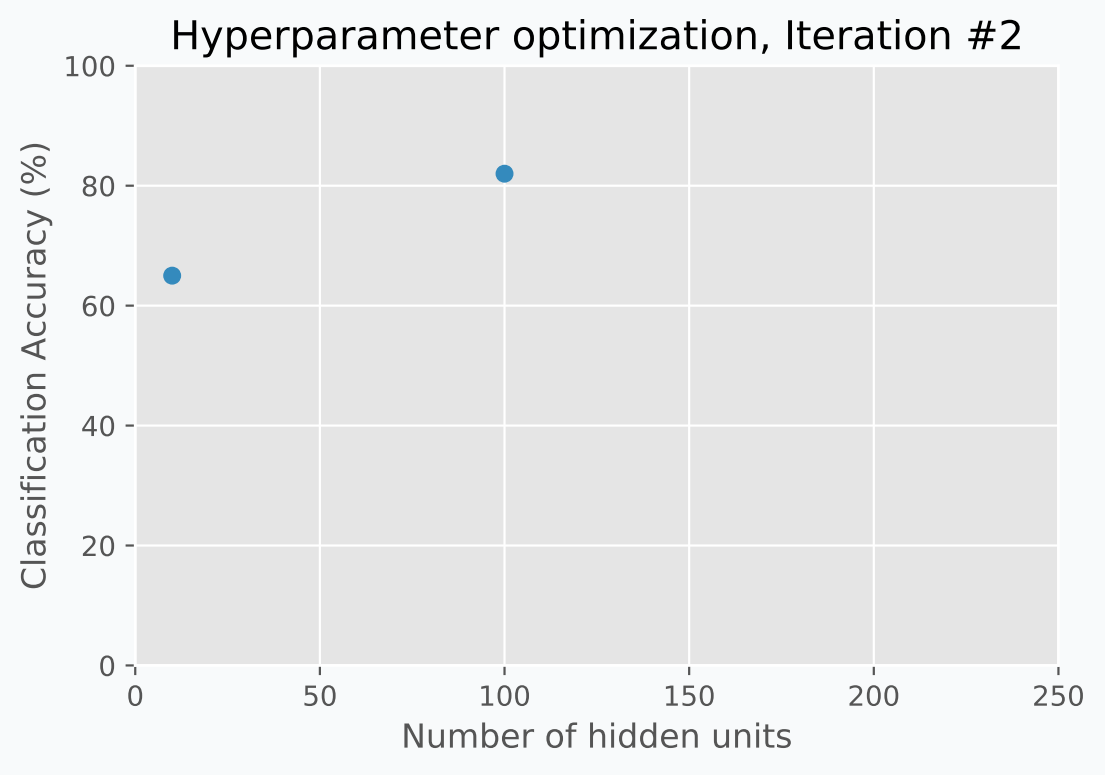


Figure 4: Configuration 100 hidden units. Accuracy 82%

Hand Tuning

Can we automate this process?

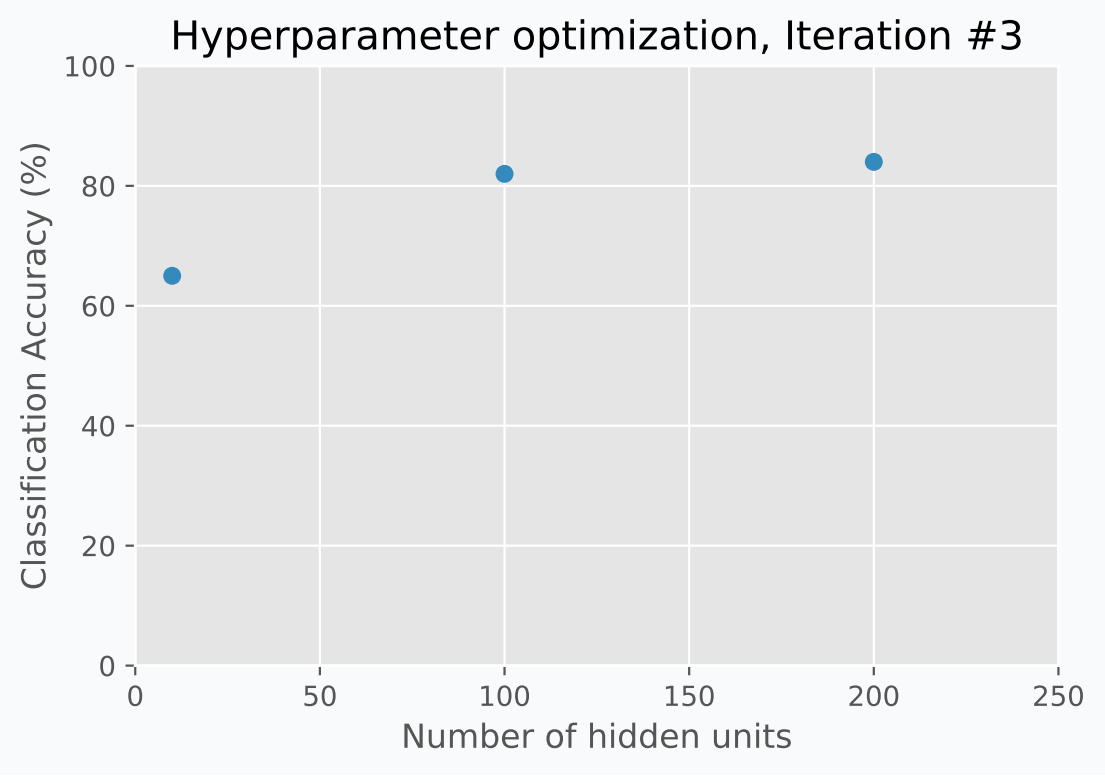


Figure 5: Configuration 200 hidden units. Accuracy 84%

Bayesian Model-Based Optimization

Bayesian approaches keep track of past evaluation results which they use to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function:

We build a model called "surrogate" for the objective function f:

$$P(score|hyperparameters) \tag{3}$$

In literature often denoted as $p(y|x)$.

Bayesian methods are derivative free, meaning we do not use the derivate information. Hence these techniques can be effective in practice even if the underlying function f being optimized is stochastic, non-convex, or even non-continuous.

Bayesian Model-Based Optimization

The process can be explained as follows:

1. Build a surrogate probability model of the objective function
2. Find the hyperparameters that perform best on the surrogate
3. Update the surrogate model incorporating the new results
4. Repeat steps 2–4 until max iterations or time is reached

The aim of Bayesian reasoning is to become "less wrong" with more data which these approaches do by continually updating the surrogate probability model after each evaluation of the objective function.

Typically, a probabilistic regression model is initialized using a set of samples from the domain $\mathcal{X}$. Following this initialization phase, new locations within the domain are sequentially selected by optimizing the acquisition function.

Visualization

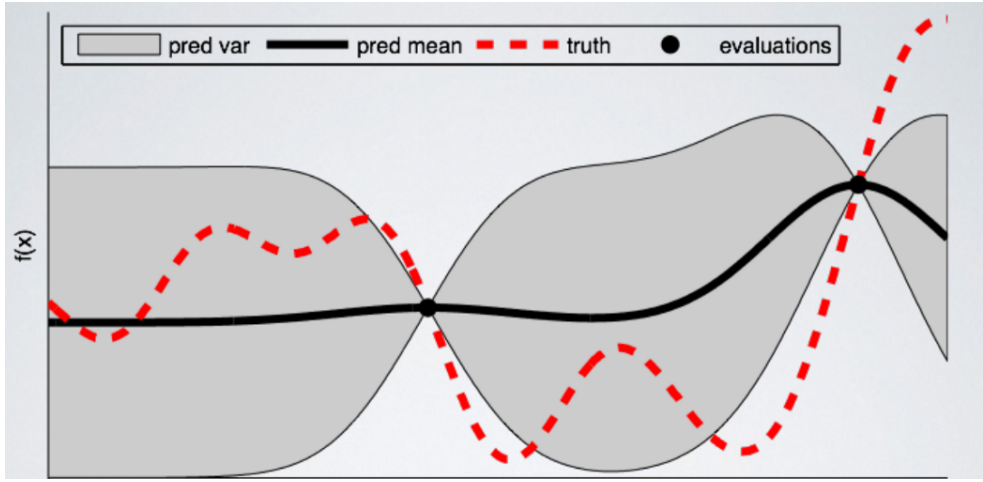


Figure 6: Initial surrogate model is very inaccurate.

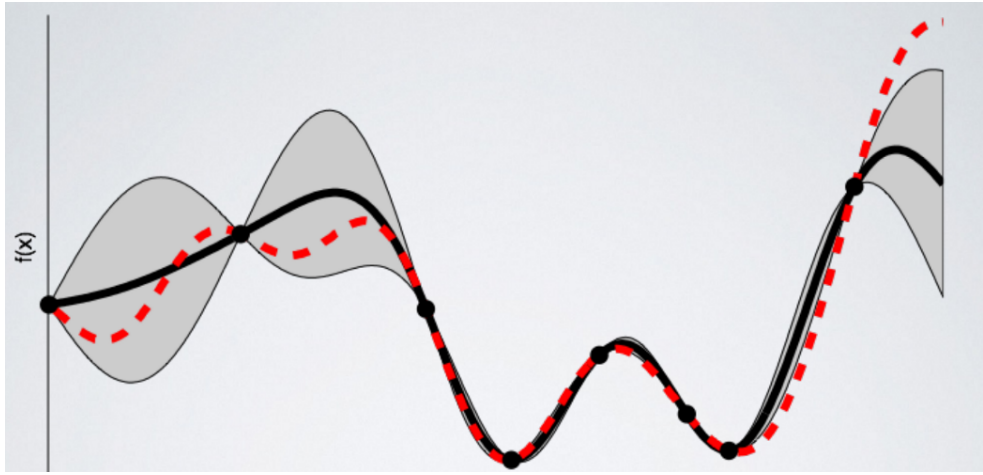


Figure 7: Evals of suggested configurations improve the surrogate m.

Sequential model-based optimization

SMBO methods are a formalization of Bayesian optimization. The sequential refers to running trials one after another, each time trying better hyperparameters by applying Bayesian reasoning and updating a probability model referring to our surrogate model.
Five aspects of SMBO:

1. A domain of hyperparameters over which to search
2. An objective function which takes in hyperparameters and outputs a score that we want to minimize (or maximize).
3. The surrogate model of the objective function.
4. A criteria, called the acquisition function, for evaluating which hyperparameters to choose next from the surrogate model.
5. A history consisting of (score, hyperparameter config) pairs used by the algorithm to update the surrogate model.

SMBO

SMBO methods differ in aspects 3–4. Differences come from how we build the surrogate of the objective function and the criteria of selecting the next hyperparameters.

Surrogate model can be modelled by:

- Gaussian Process (GP)
- Tree Parzen Estimator (TPE)
- Random Forest Regression

Selection function

Expected Improvement(EI)

$$EI_{y^\star}(x) = \int_{-\infty}^{y^\star} (y^\star - y) p(y|x) dy \tag{4}$$

- $y^\star$ is a threshold value of the objective. y the actual value
- x proposed set of hyperparams.
- $p(y|x)$ is the surrogate probability model, the probability of y given config. x

If the integral is positive, then it means that the hyperparameter configuration x is expected to yield a better result than the threshold value.

The aim is to maximize the Expected Improvement with respect to possible hyperparameter configurations $x \in \mathcal{X}$.

Gaussian Process

Multivariate Gaussian Distribution

Example of a bi-variate gaussian distribution. Defined by μ and Σ.

$$\boldsymbol{\mu} = \begin{bmatrix} 0.0 & 1.0 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.7 \\ 0.7 & 2.5 \end{bmatrix} \tag{5}$$

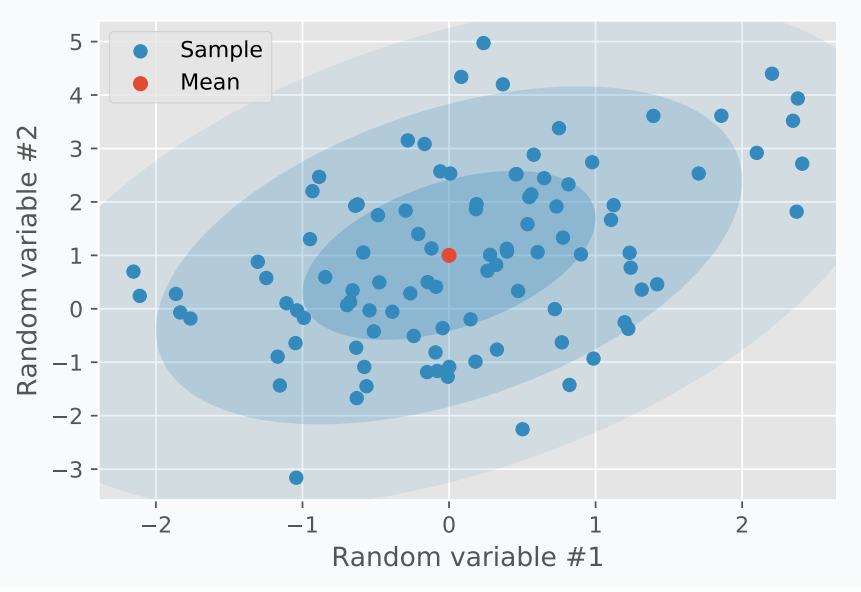
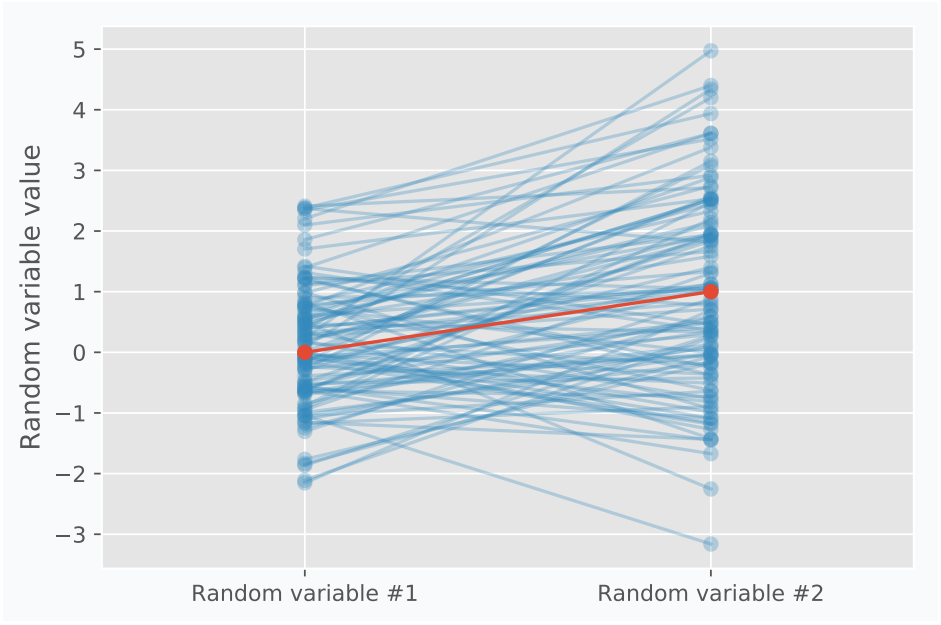
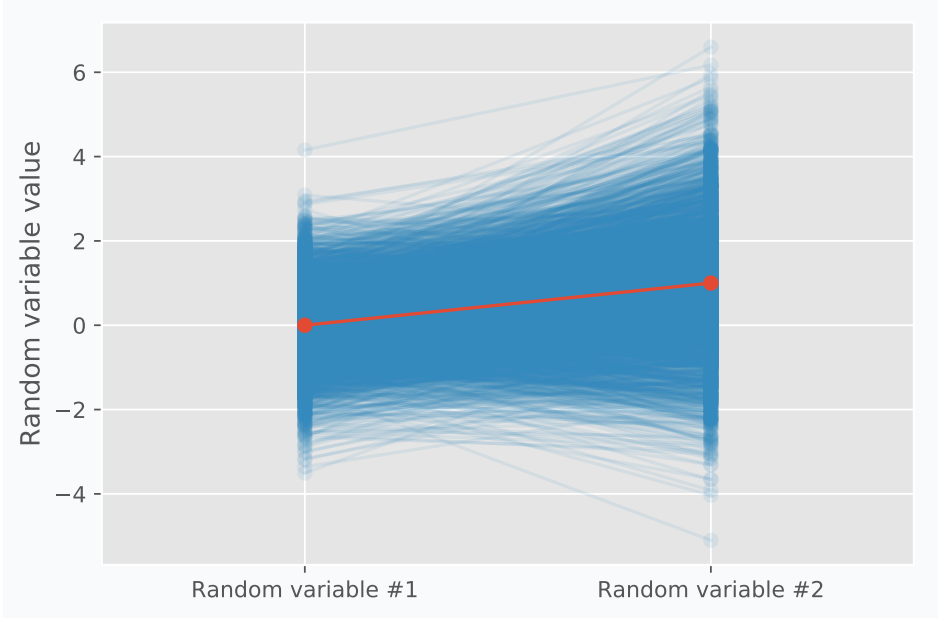


Figure 8: Bi-variate gaussian distribution example

Gaussian Process





Gaussian Process

Multivariate Gaussian Distribution

The more dimensions we add, the more it looks like a set of functions sampled from a Gaussian Process. In case of GP, number of dimensions is infinite. GP is uniquely defined by mean function μ and Covariance(Kernel) function K. $p(y|x) \sim \mathcal{GP}(\mu, K)$

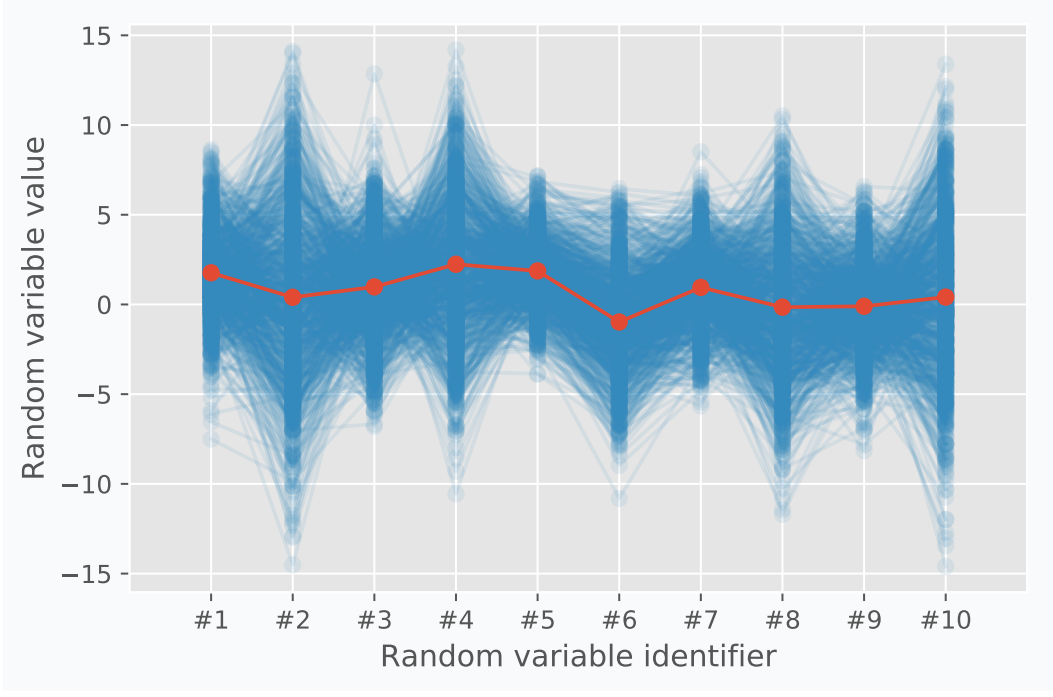


Figure 9: Random Multi-variate gaussian distribution example

Gaussian Process Regression

Number of hidden units example

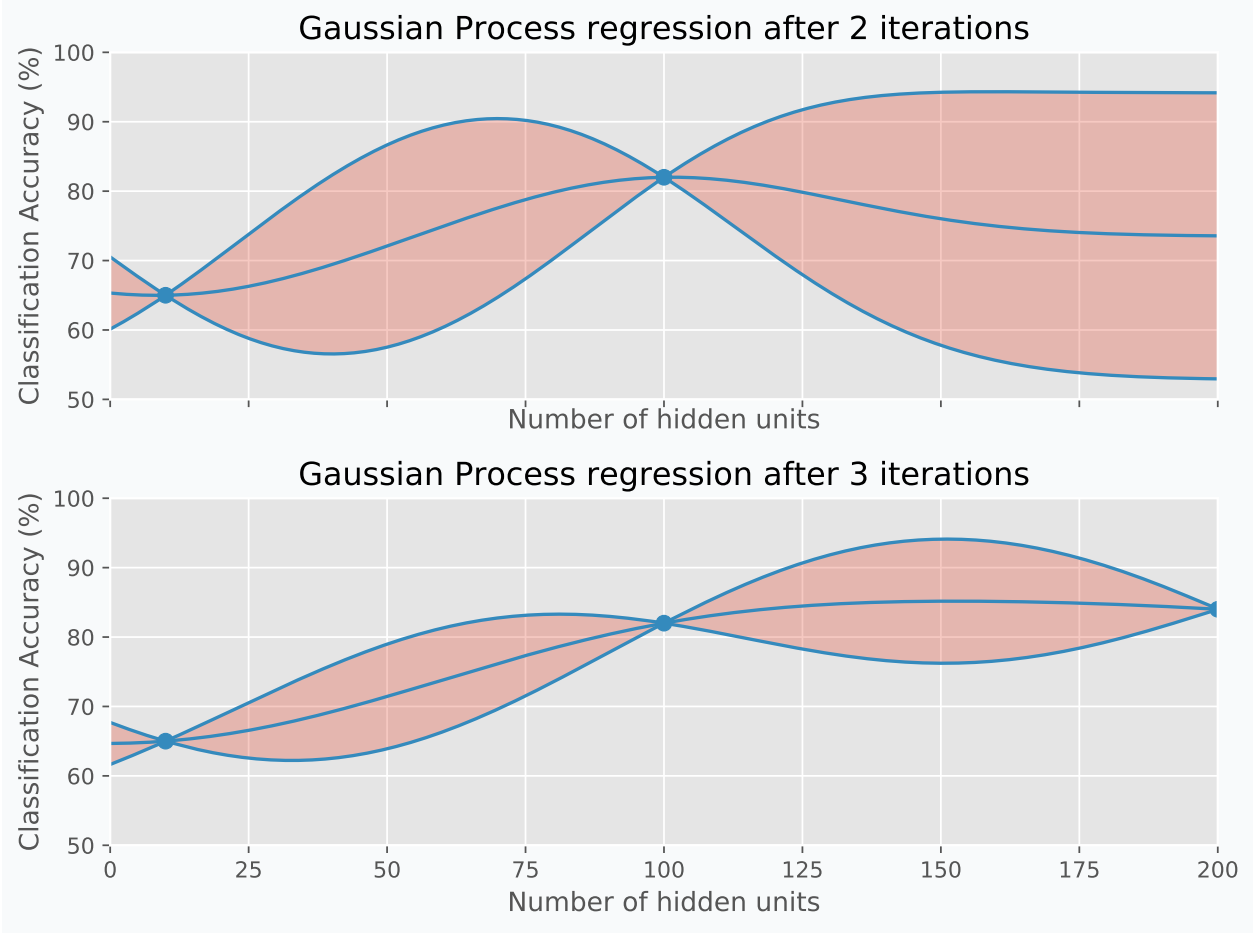


Figure 10: Gaussian Process regression on digits classification

Gaussian Process

Algorithm

1. Heuristically select some param configurations and evaluate the real, costly objective function f.
2. Fit GP on existing (score, config) pairs.
3. Use GP to suggest new hyperparameter config, config should maximize acquisition function.
4. Evaluate chosen configuration with real objective function.
5. Update GP with additional (score, config) pair.
6. Repeat 3-5

Gaussian Process

Acquisition function

When our surrogate model $p(y|x)$ is gaussian process, we can calculate the expected improvement with a closed form solution as follows:

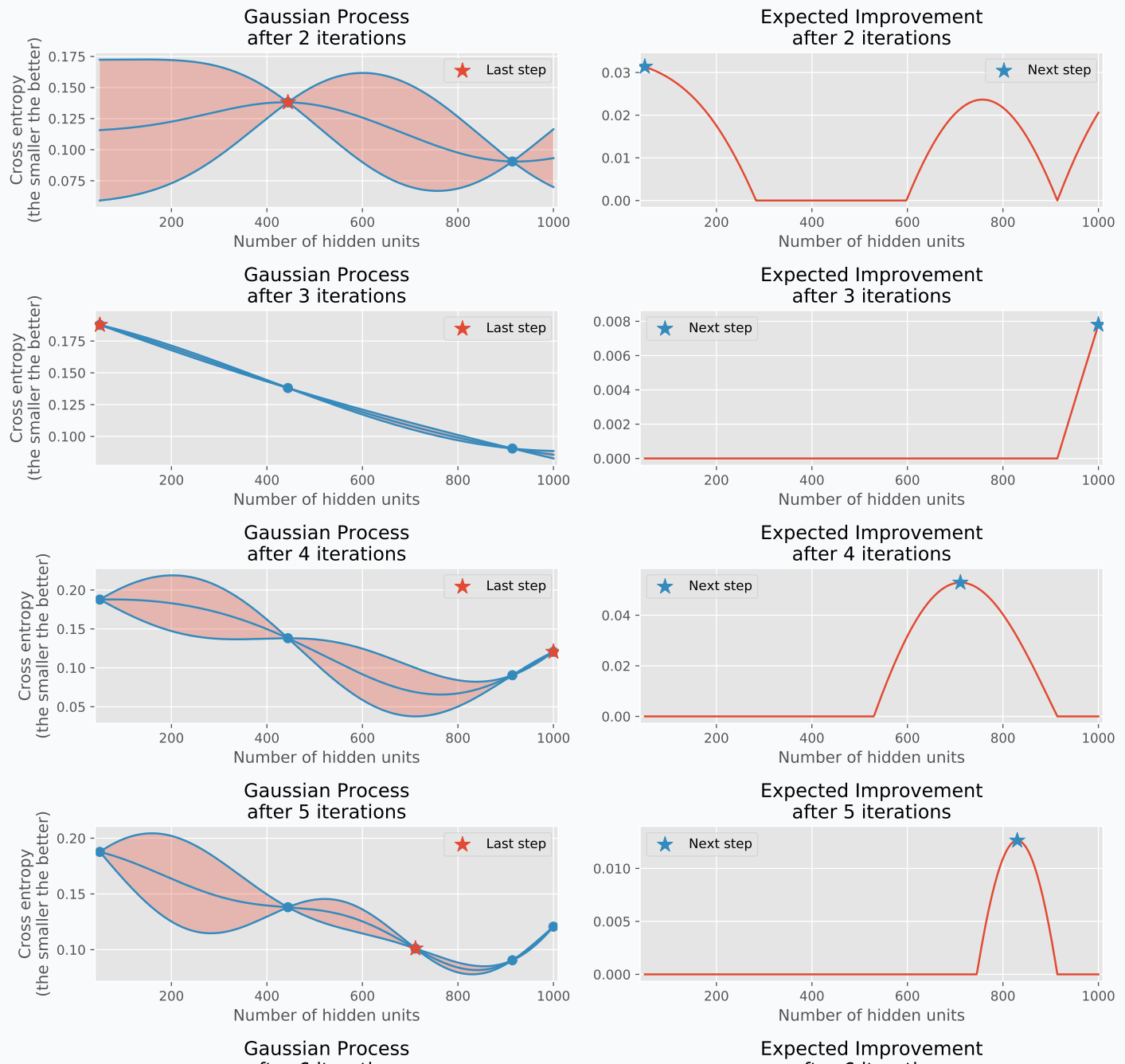
$$EI_{y^\star}(x) = \sigma_x(u \cdot \Theta(u) + \Phi(u)) \tag{6}$$

where $u = \frac{y^\star - \mu_x}{\sigma_x}$ and Φ is probability function and Θ is cummulative density function of the standard normal distribution. It corresponds to the following simple utility function

$$U(x) = max\{0, y^\star - y\} \tag{7}$$

Where again $y^\star$ is the objective $f(x)$ value of configuration x that performed the best. The y is the expected value of the objective function. That is, we receive a reward equal to the "improvement" and no reward otherwise.

Gaussian Process



Gaussian Process

Review

- Using GP as a surrogate model gives us the ability to reason about the quality of experiments before they are run.
- It doesn't work well for categorical parameters.
- It produces the best results for continuous parameter spaces.
- It can be difficult to select right hyperparameters for Gaussian Process. Gaussian Process has lots of different kernel types. In addition you can construct more complicated kernels using simple kernels as a building block... Sq. Exponential: Char. Landscape l, σ_f, σ_n
- It works slower when number of hyperparameters increases.

Tree Structured Parzen Estimator

The methods of SMBO differ in how they construct the surrogate model $p(y|x)$. The Tree-structured Parzen Estimator(TPE) builds a model by applying Bayes rule. Instead of directly representing $p(y|x)$, it uses the following notoriously known rule:

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)} \tag{8}$$

$p(x|y)$, which is the probability of the hyperparameters given the score of the objective function, in turn is expressed:

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^\star \\ g(x) & \text{if } y \geq y^\star \end{cases} \tag{9}$$

We make two different distributions for the hyperparameters:

- Distribution $l(x)$ of parameter configurations which are "likely to improve the score", one where the value of the objective function is less than the threshold.
- Distribution $g(x)$ of the other configurations

TPE

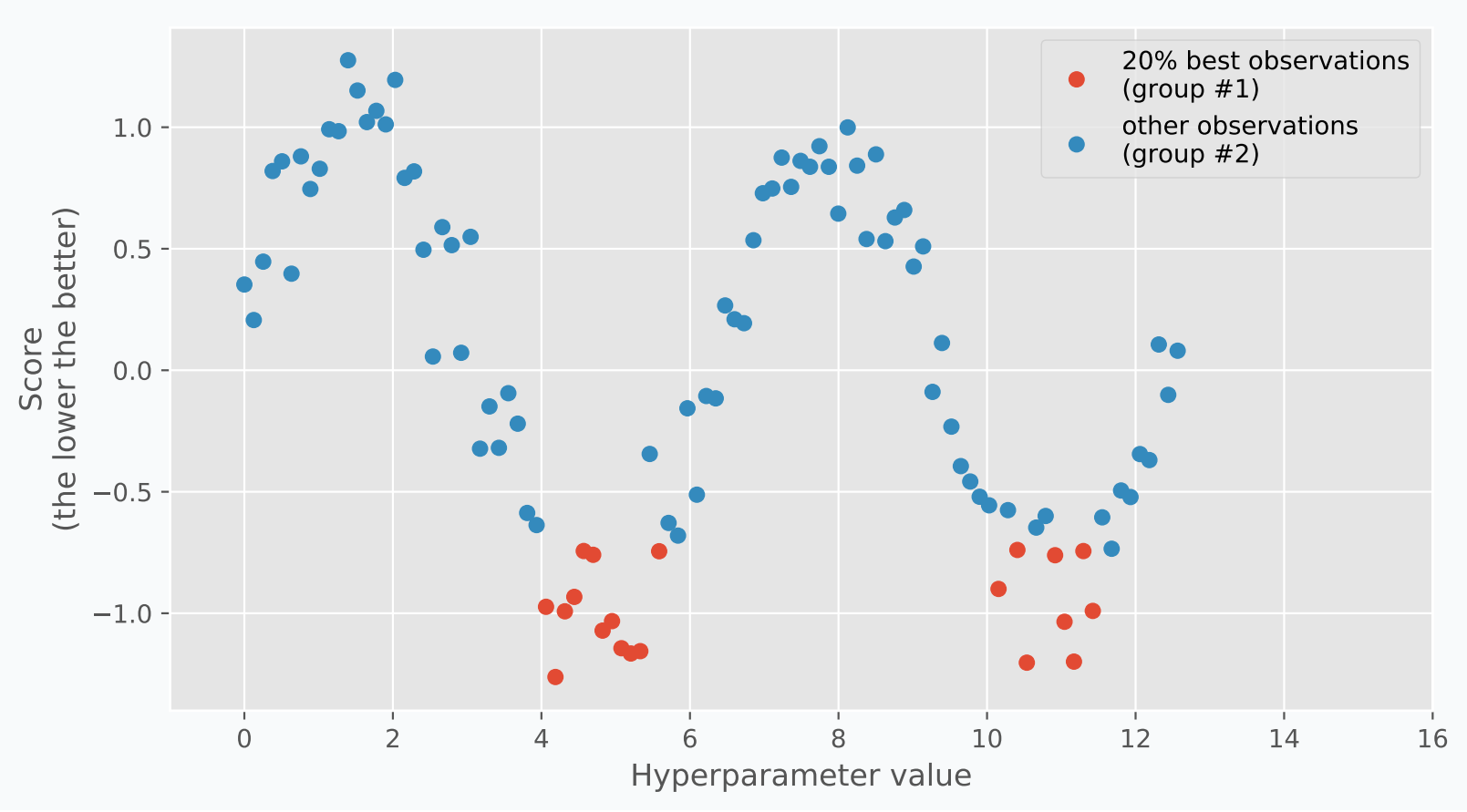


Figure 12: $l(x)$ the improvement group red, $g(x)$ all other configurations, blue

TPE

We can then "simplify" the EI as follows:

$$EI_{y^\star}(x) = \frac{\gamma y^\star l(x) - l(x)\int_{-\infty}^{y^\star} p(y)d(y)}{\gamma l(x) + (1-\gamma)g(x)} \propto (\gamma + \frac{g(x)}{l(x)}(1-\gamma))^{-1} \tag{10}$$

where *gamma* is a percentage parameter. E.g. "Use 20% of the best observations to estimate the next set of parameters"

- The expected improvement criteria allows the model to balance exploration versus exploitation.
- $l(x)$ is a distribution and not a single value which means that the hyperparameters drawn are likely close but not exactly at the maximum of the expected improvement.
- The surrogate is just an estimate of the objective function, the selected hyperparameters may not actually yield an improvement when evaluated and the surrogate model will have to be updated.

TPE

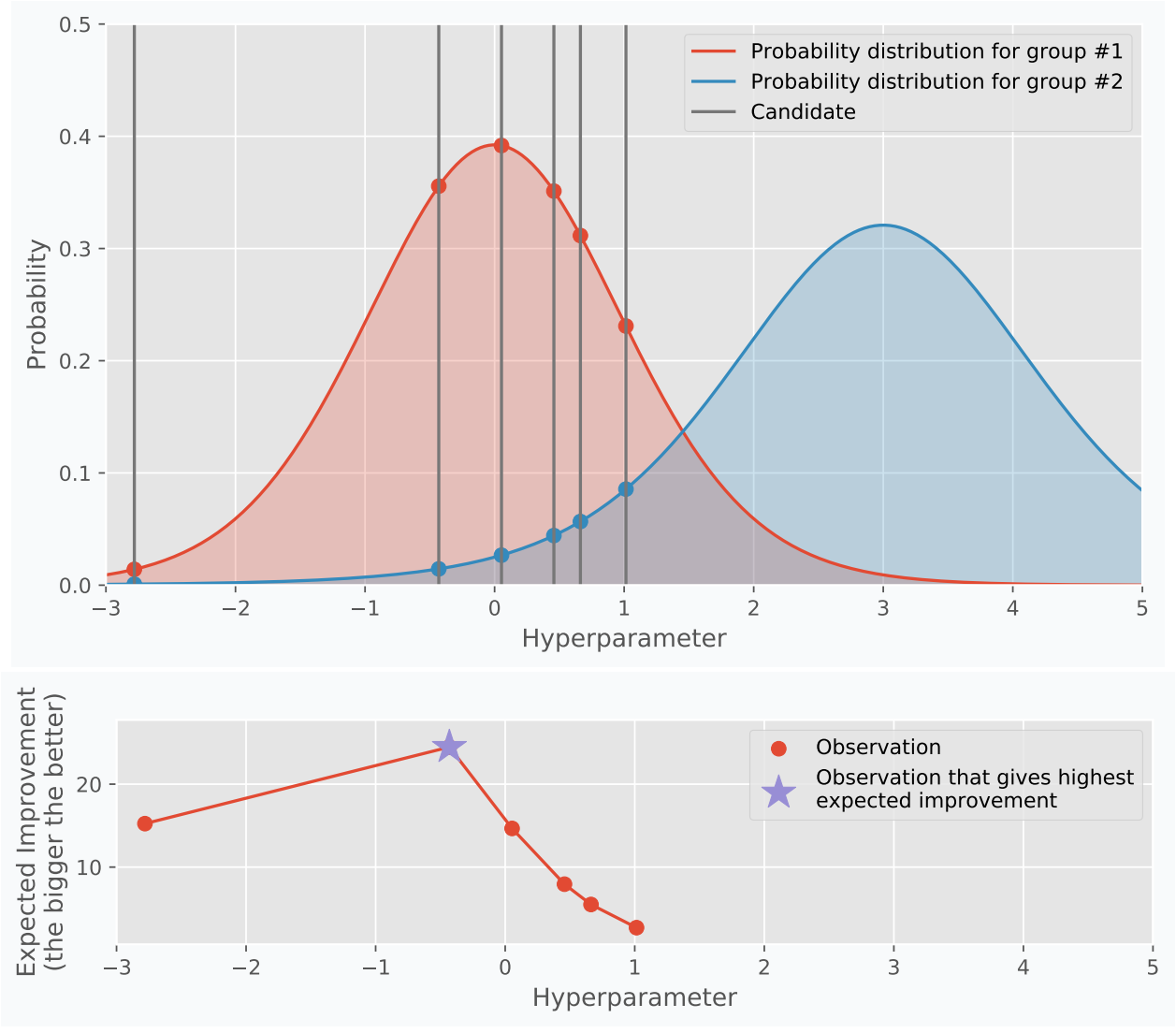


Figure 13: Caption goes here.

TPE

Review

- Based on acquisition function an the surrogate model algorithm proposes a new set of candidate hyperparameters,
- It evaluates them with the actual objective function and records the result in a pair (score, parameters).
- These records form a history. The algorithm builds distributions $l(x)$ and $g(x)$ using the history to come up with a probability model of the objective function(surrogate) that improves with each iteration.

TPE vs Random Search for NNs

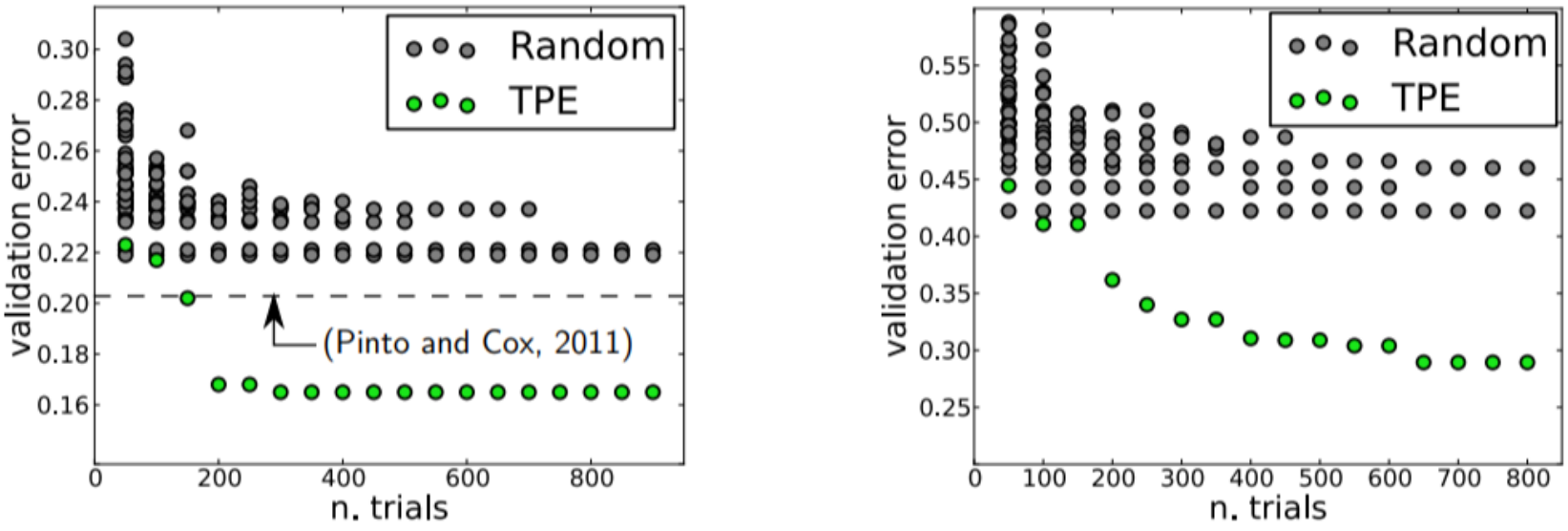
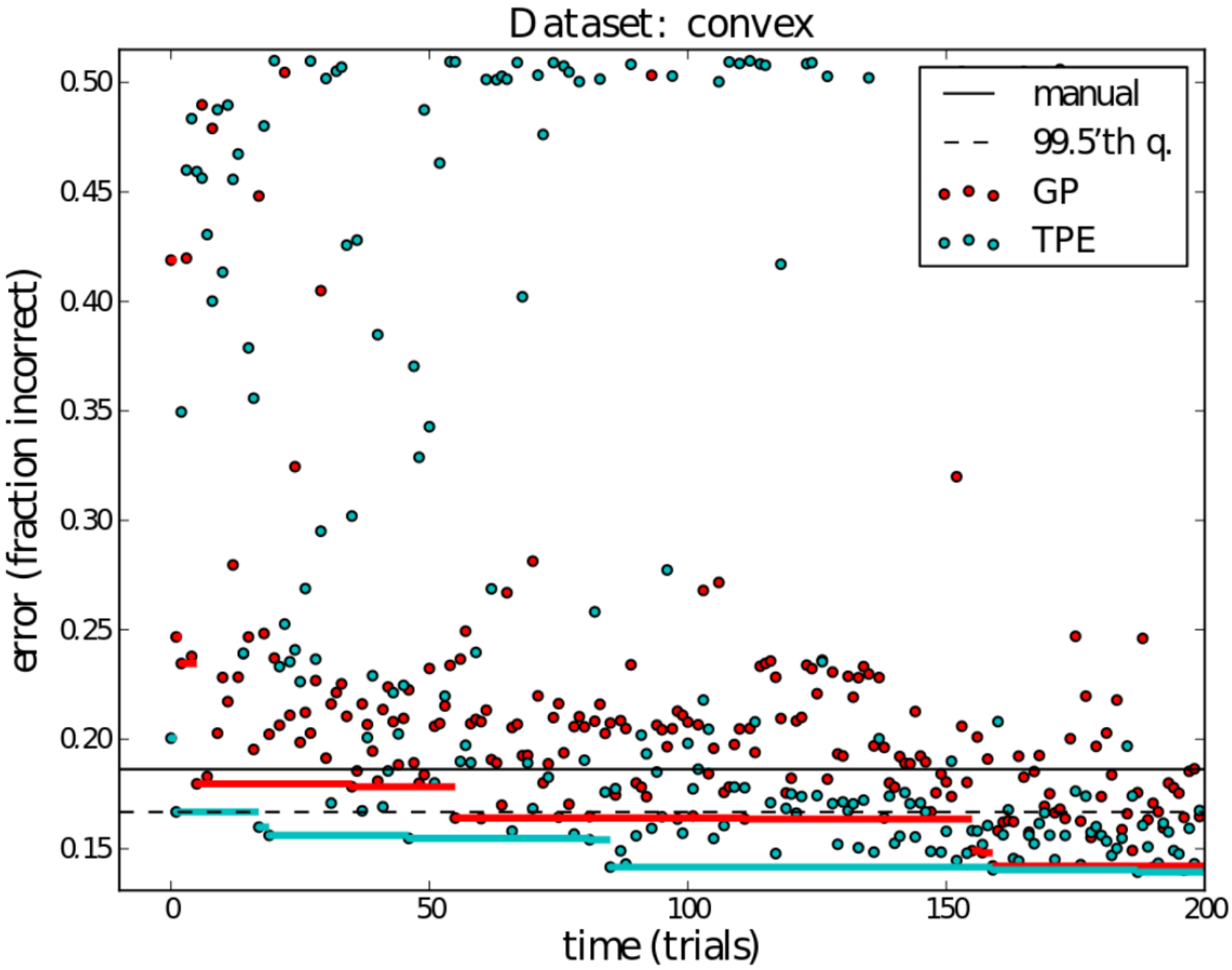
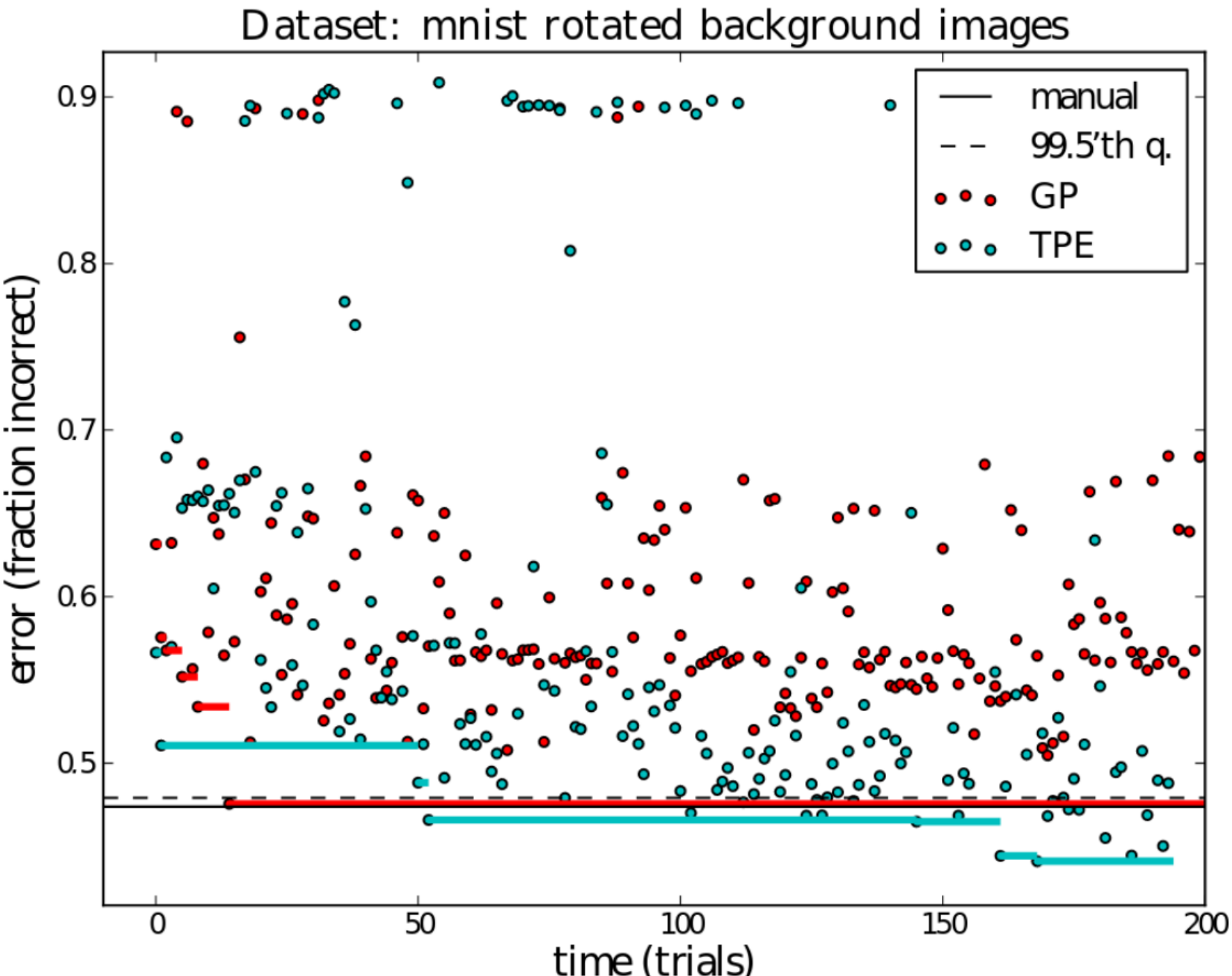


Figure 14: Image classification(NN) validation error with random search in grey and TPE in green.

TPE vs GP for DBNs on convex



TPE vs GP for DBNs on MRBI



Conclusion

- At a high-level, Bayesian optimization methods are efficient because they choose the next hyperparameters in an informed manner.
- The **central idea** is: spend a little more time selecting the next hyperparameters in order to make fewer calls to the costly objective function.
- In practice, the time spent selecting the next hyperparameters is inconsequential compared to the time spent in the objective function.
- Bayesian methods can find better model settings than random search in fewer iterations because they reason about the best set of hyperparameters to evaluate based on past trials.

Experiment

Classification

- Iris plants database
- Number of instances: 150 (50 in each of the three classes)
- Number of attributes: 4 numeric: sepal-length, sepal-width, petal-length, petal-width in cm
- Number of classes: 3: Iris-Setosa, Iris-Versicolour, Iris-Virginica