Nile University

School of Communication and Information Technology (CIT)

Master of Science/Engineering MSCIT/ MCIT Program

D-Smart ML

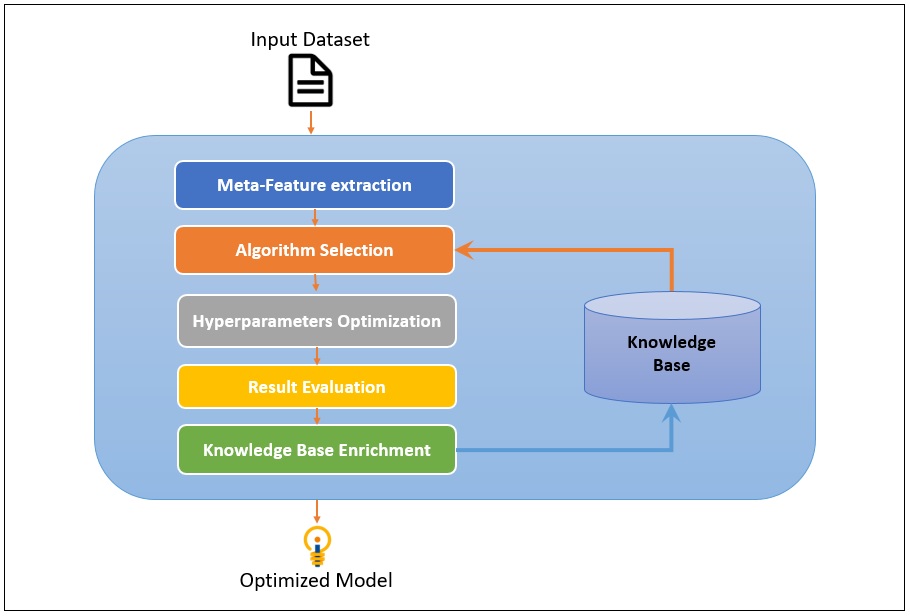
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version No.** | **Date** | **Description** | **Created By** | **e-mail** |
| 0.1 | 11-02-2019 | D-Smart ML | Ahmed Mohamed Abdel Rahman | [Robot209@gmail.com](mailto:Robot209@gmail.com) |
|  |  |  |  |  |

## Introduction

* Building Machine learning model is a highly iterative exploratory process where most scientists work hard to find the best model that meets their performance requirement.
* In practice, there is no one-model-fits-all solutions, thus, there is no single model or algorithm that can handle all data set varieties and changes in data that may occur over time. Also each machine learning algorithm require user defined inputs to achieve a balance between accuracy and generalizability (which called hyperparameters).
* So the process of selecting the Algorithm and its hyperparameters usually require a lot of trails until we find it. In case of big datasets this process will need a lot of resources (Time, Hardware, effort, …)

## what is “D-Smart ML Library”?

* The D-Smart ML library main objective is to do automatic machine learning by receiving a dataset as input and return the best Machine Learning model as output with minimum resources (ex: Time) without the need of selecting the algorithm or its hyperparameters manually.
* The library built on Apache Spark as a distributed processing platform, so the process of getting the best machine learning model can be easily distributed (like any spark application).
* The library main idea can be summarized in the below figure:

****

* The generation of the best machine learning model can be divided into two steps:
  + Selecting the best Algorithm
  + Selecting the best hyperparameters for the Selected Algorithm
* The library also can be used to do a Full Grid Search for all available algorithms, this function will try every algorithm with a predefines list of hyperparameters value and return the best one.
* The library also contains implementation for two Classifiers (Linear Discernment Analysis and Quadratic Discernment Analysis).

## Algorithm Selection

1. Introduction
2. KB & Metadata Extraction
3. Using ML to get best Algorithm

## Hyper-parameters optimization

1. Introduction
2. Hyperband

## Implementation on Spark

1. Adding new Classifiers (LDA & QDA)
2. Implementing Grid Search
3. Parallelism in Hyper-Parameters Optimization

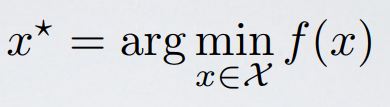
## D-Smart ML vs. Grid Search vs. Transmogrifai

1. Grid Search
2. TransmogrifAI
3. D-Smart ML
4. Comparison

## References

## Appendix

* KB Datasets List
* D-Smart ML vs. Grid Search Details
* D-Smart Example
* The aim of hyperparameter optimization in machine learning is to find the hyperparameters of a given machine learning algorithm that return the best performance as measured on a validation set. (Hyperparameters, in contrast to model parameters, are set by the machine learning engineer before training.
* The number of trees in a random forest is a hyperparameter while the weights in a neural network are model parameters learned during training. I like to think of hyperparameters as the model settings to be tuned.)



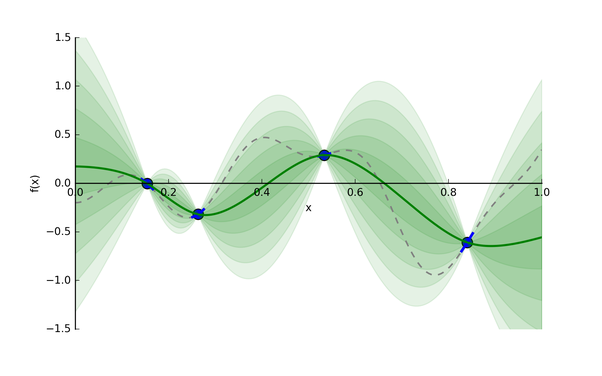
* Here f(x) represents an objective score to minimize— such as RMSE or error rate— evaluated on the validation set; x\* is the set of hyperparameters that yields the lowest value of the score, and x can take on any value in the domain X.
* In simple terms, we want to **find the model hyperparameters that yield the best score on the validation set metric**.
* The problem with hyperparameter optimization is that evaluating the objective function to find the score is extremely expensive. Each time we try different hyperparameters, we have to train a model on the training data, make predictions on the validation data, and then calculate the validation metric. With a large number of hyperparameters and complex models such as ensembles or deep neural networks that can take days to train, this process quickly becomes intractable to do by hand!
* Grid search and [random search are slightly better](http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf) than manual tuning because we set up a grid of model hyperparameters and run the train-predict -evaluate cycle automatically in a loop while we do more productive things (like[feature engineering](https://www.featuretools.com/)).
* However, even these methods are relatively inefficient because they do not choose the next hyperparameters to evaluate based on previous results. **Grid and random search are completely** *uninformed***by past evaluations,**and as a result, often spend a significant amount of time evaluating “bad” hyperparameters.

## Bayesian Optimization (using GP)

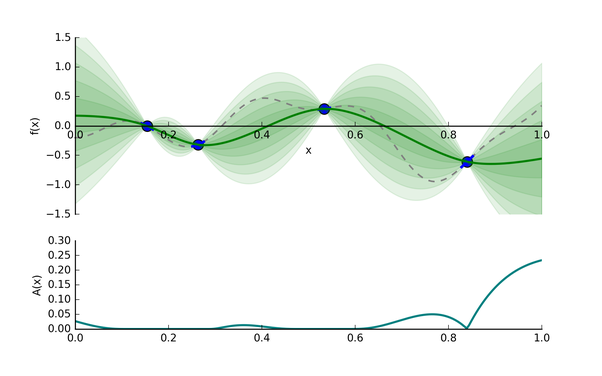
* [Bayesian hyperparameter optimization](https://sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf) is: build a probability model of the objective function and use it to select the most promising hyperparameters to evaluate in the true objective function.
* [Bayesian approaches](https://www.iro.umontreal.ca/~bengioy/cifar/NCAP2014-summerschool/slides/Ryan_adams_140814_bayesopt_ncap.pdf), in contrast to random or grid search, 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:

https://cdn-images-1.medium.com/max/800/1*u00KlxHhd1fz6-Jaeou6PA.png

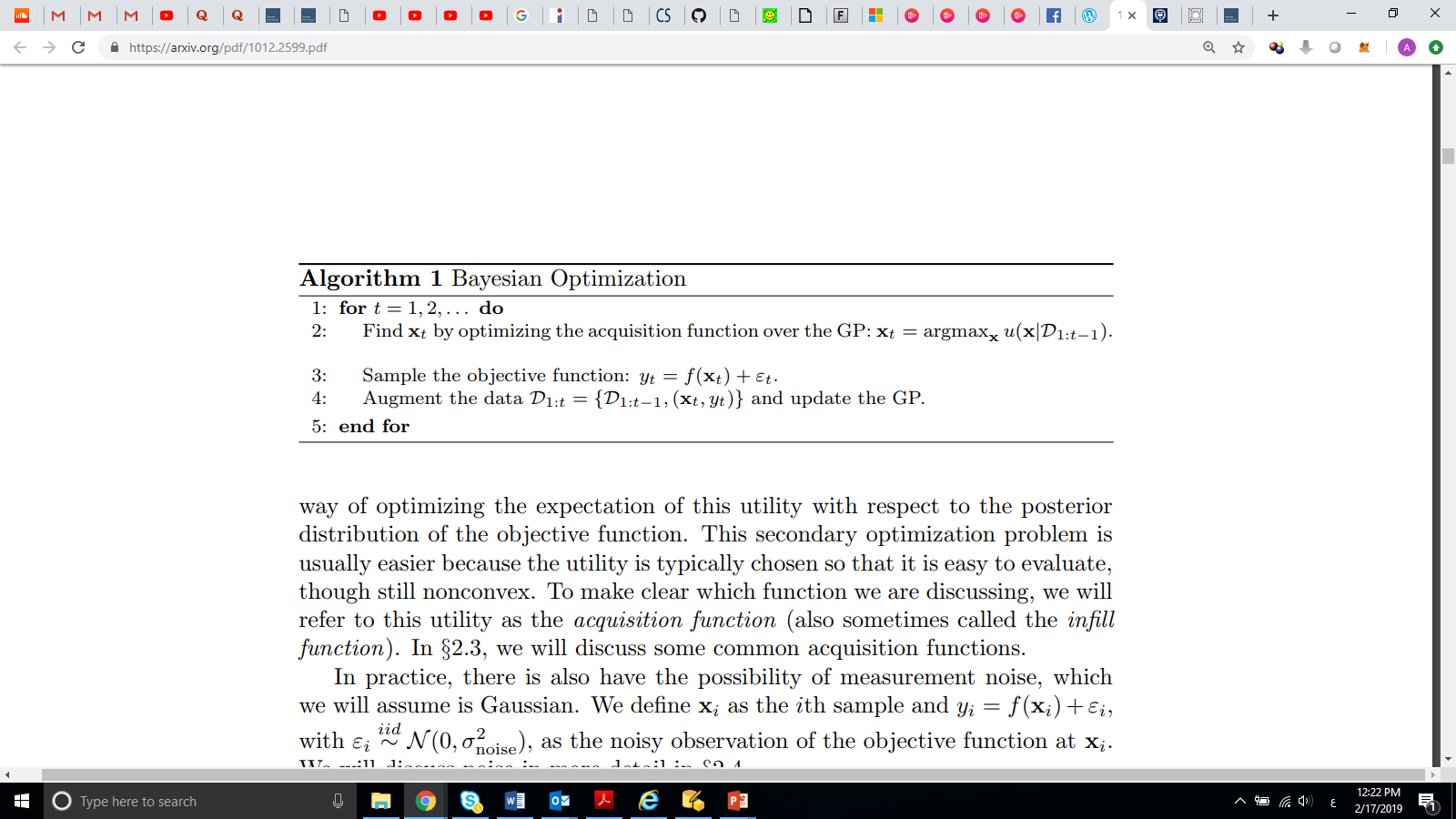
* this model is called a “surrogate” for the objective function and is represented as p(y | x). The surrogate is much easier to optimize than the objective function and Bayesian methods work by finding the next set of hyperparameters to evaluate on the actual objective function by selecting hyperparameters that perform best on the surrogate function.
* In other words:
  1. Build a surrogate probability model of the objective function
  2. Find the hyperparameters that perform best on the surrogate
  3. Apply these hyperparameters to the true objective function
  4. Update the surrogate model incorporating the new results
  5. Repeat steps 2–4 until max iterations or time is reached
* he basic idea is: **spend a little more time selecting the next hyperparameters in order to make fewer calls to the objective function.**
* In practice, the time spent selecting the next hyperparameters is inconsequential compared to the time spent in the objective function.
* Bayesian optimization is a powerful strategy for finding the extrema of objective functions that are expensive to evaluate.
* Sample some input-outputs (ex: less than 10) and use them to guess the true function with 'Gaussian Process' (or ‘GP’ here).
* Then use that guessed function to determine where to evaluate next.
* Evaluate that point, add it to our set of input-outputs and infer the guessed function once again.
* Repeat this until you've exhausted your budget of evaluations (or some other stopping criteria). If the GP is any good at guessing the true function, we’ll do better than random sampling.



* The solid green line is our guess of the true function. Each additional band of green is another half standard deviation on the output distribution.
* We have to develop “*acquisition function*” that balance between exploration-exploitation (That is, we should evaluate on points where the solid green line is low and We should check areas we know less about).
* This function is cheap, so we can optimize it and use that x as our next point to search next.

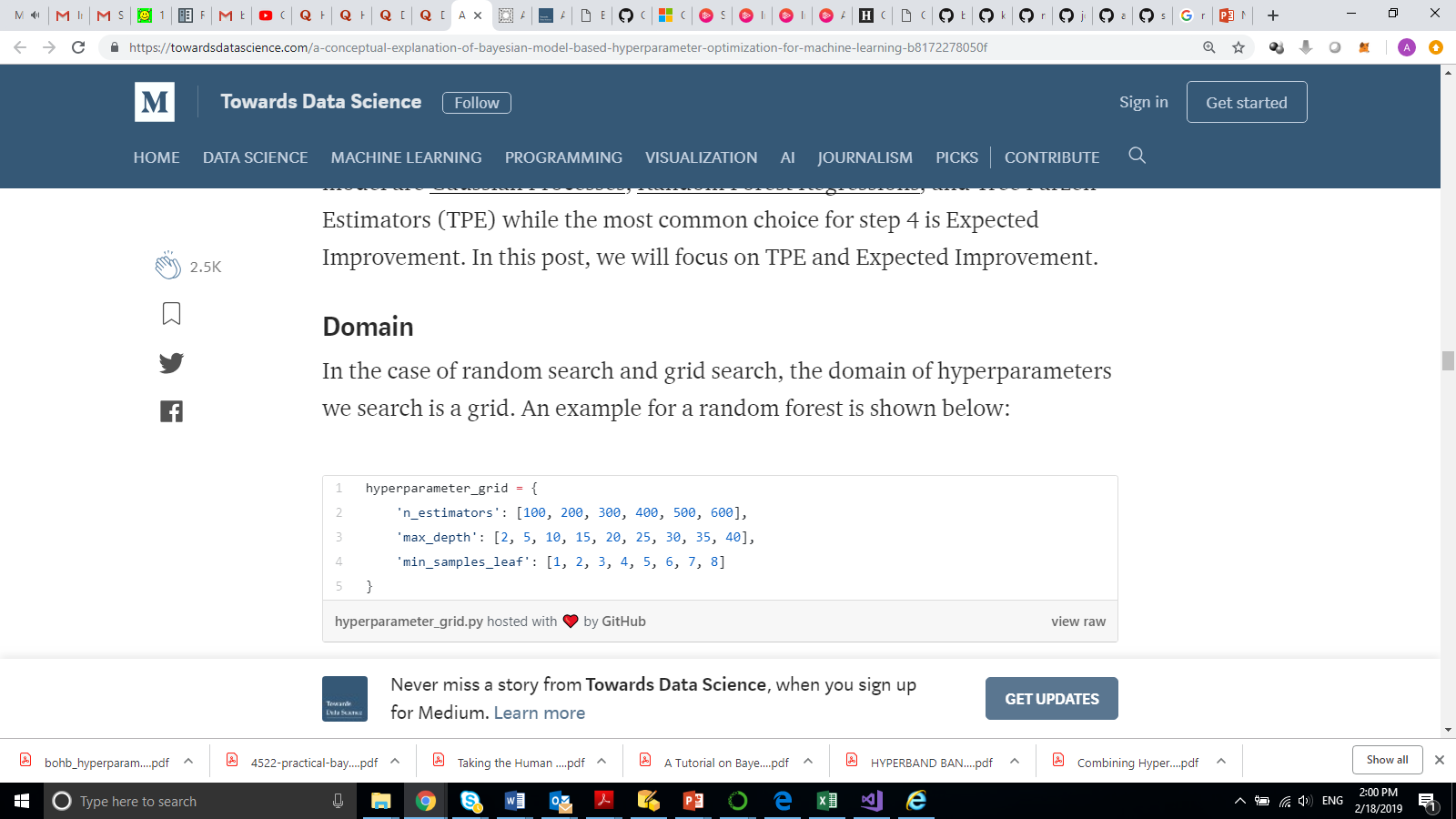


* Generally, the optimization is over more than just one dimension. GPs scale to more than one dimension naturally,



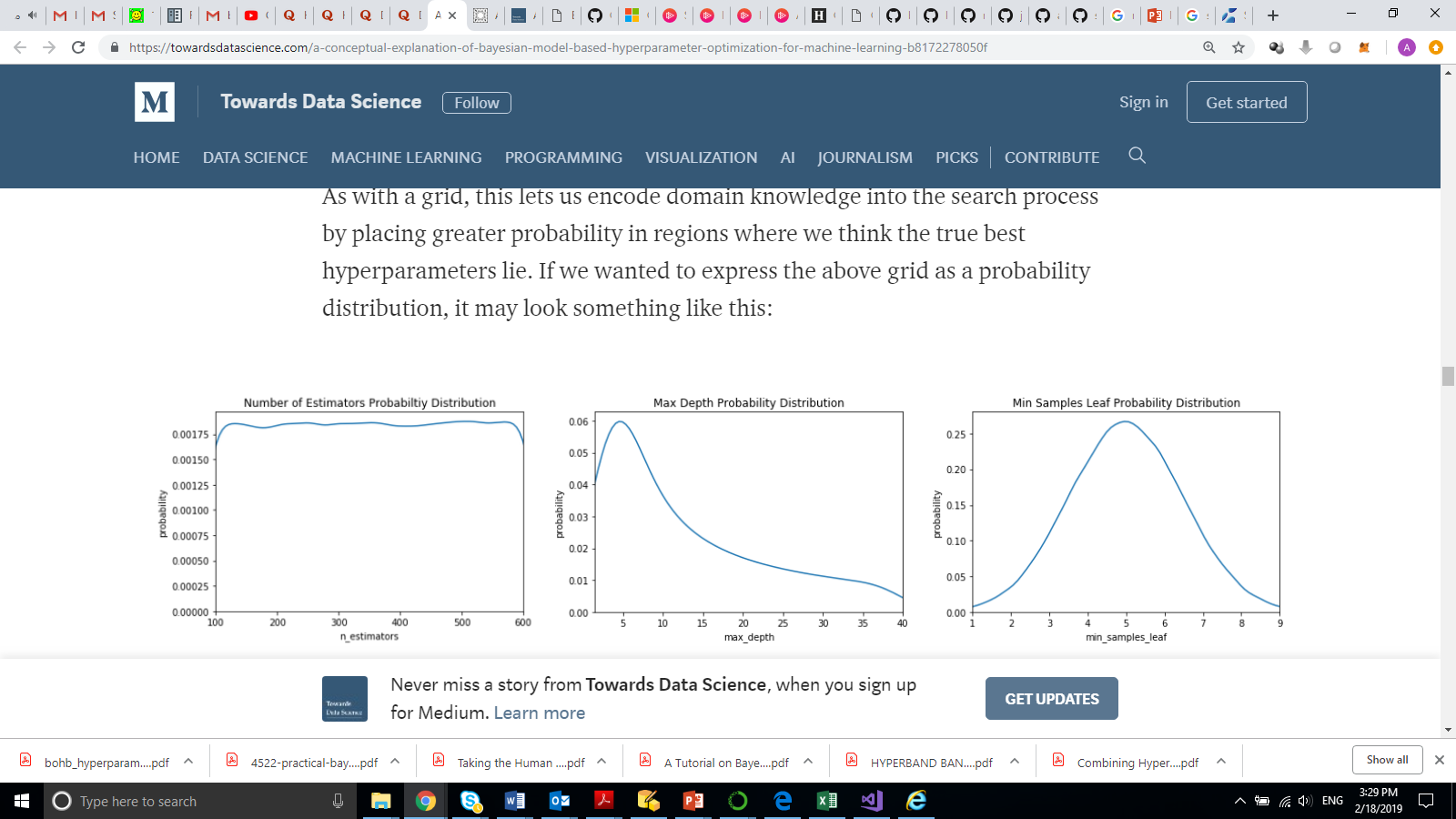
## Sequential Model-Based Optimization (SMBO)

* [Sequential model-based optimization (SMBO) methods](https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf)  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 (surrogate).
* There are five aspects of model-based hyperparameter optimization:
  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 a selection function, for evaluating which hyperparameters to choose next from the surrogate model
  5. A history consisting of (score, hyperparameter) pairs used by the algorithm to update the surrogate model
* There are several variants of [SMBO methods that differ](https://sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf) in steps 3–4, namely, how they build a surrogate of the objective function and the criteria used to select the next hyperparameters.
* Several common choices for the surrogate model are
  + [Gaussian Processes](https://en.wikipedia.org/wiki/Gaussian_process)
  + [Random Forest Regressions](http://aad.informatik.uni-freiburg.de/papers/13-GECCO-BBOB_SMAC.pdf)
  + Tree Parzen Estimators (TPE)
* **Domain:**
  + In the case of random search and grid search, the domain of hyperparameters we search is a grid. For example

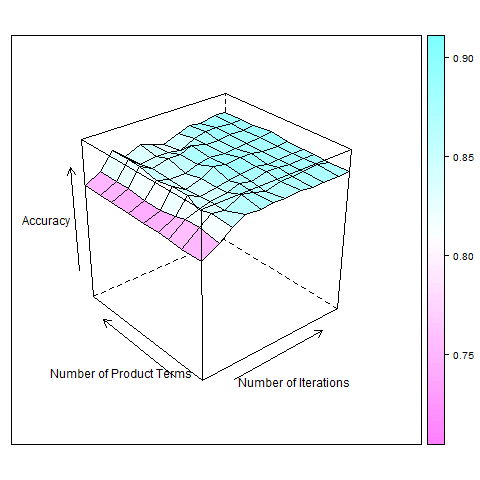


* + For a model-based approach, the domain consists of *probability distributions*.

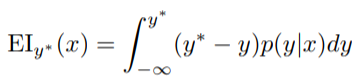
If we wanted to express the above grid as a probability distribution, it may look something like this:



* + Here we have a uniform, log-normal, and normal distribution.
* **Objective Function**
  + The objective function takes in hyperparameters and outputs a single real-valued score that we want to minimize (or maximize).
  + The entire concept of Bayesian model-based optimization is to reduce the number of times the objective function needs to be run by choosing only the most promising set of hyperparameters to evaluate based on previous calls to the evaluation function.
* **Surrogate Function (Probability Model)**
  + The surrogate function, also called the response surface, is the probability representation of the objective function built using previous evaluations.
  + This is called sometimes called a response surface because it is a high-dimensional mapping of hyperparameters to the probability of a score on the objective function.



* + There are several different forms of the surrogate function:
    - [Gaussian Processes](https://en.wikipedia.org/wiki/Gaussian_process)
    - [Random Forest Regressions](http://aad.informatik.uni-freiburg.de/papers/13-GECCO-BBOB_SMAC.pdf)
    - Tree Parzen Estimators (TPE)
* **Selection Function**
  + The selection function is the criteria by which the next set of hyperparameters are chosen from the surrogate function.
  + The most common choice of criteria is Expected Improvement:



## Hyperband Algorithm and Successive Halving algorithm

* **Successive Halving**
  + uniformly allocate a budget to a set of hyperparameter configurations, evaluate the performance of all configurations, throw out the worst half, and repeat until one configuration remains.
  + The algorithm allocates exponentially more resources to more promising configurations.
* Hyperband
  + It considers several possible values of n (hyperparameters) for a fixed Budget B (e.g., an hour of training time to choose a hyperparameter configuration) [it is like performing a grid search over feasible value of n]
  + Associated with each value of n is a minimum resource r that is allocated to all configurations before some are discarded; a larger value of n corresponds to a smaller r and hence more aggressive early-stopping
  + There are two components to Hyperband;
    1. the inner loop invokes Successive Halving for fixed values of n and r and
    2. the outer loop iterates over different values of n and r

## Libraries and Framework for optimizing hyperparams

* SMAC
  + SMAC: Sequential Model-based Algorithm Configuration
  + <http://www.cs.ubc.ca/labs/beta/Projects/SMAC/>
  + (Java)
* [spearmint](https://github.com/JasperSnoek/spearmint)
  + Spearmint is a package to perform Bayesian optimization
  + <https://github.com/JasperSnoek/spearmint>
  + (Python)
* HyperOpt
  + <http://hyperopt.github.io/hyperopt/>
  + (Python)
* REMBO
  + <https://github.com/ziyuw/rembo>
  + (Matlab)
* HPOLib
  + It provides a common interface to three state of the art hyperparameter optimization packages: SMAC, spearmint and hyperopt
  + <https://github.com/automl/HPOlib>
* Google Vizier
  + A Service for Black-Box Optimization
  + <https://ai.google/research/pubs/pub46180>
  + <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46180.pdf>
* BOHB
  + ROBUST AND EFFICIENT HYPERPARAMETER OPTIMIZATION AT SCALE
  + <https://www.automl.org/blog_bohb/>
* auto-sklearn
  + *auto-sklearn* is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator:
  + <https://automl.github.io/auto-sklearn/master/>

## References

* Gaussian Multivariate Distribution

<http://cs229.stanford.edu/section/gaussians.pdf>

* Gaussian Process
* <http://cs229.stanford.edu/section/cs229-gaussian_processes.pdf>
* http://www.stat.columbia.edu/~rdavis/papers/VAG002.pdf
* <https://www.youtube.com/watch?v=mqOdIYxxNCs>

* Bayesian Optimization
* <https://arxiv.org/pdf/1012.2599.pdf>
* <https://www.cs.ox.ac.uk/people/nando.defreitas/publications/BayesOptLoop.pdf>
* https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf
* <https://www.youtube.com/watch?v=-Ekte3my510>
* <https://www.youtube.com/watch?v=vz3D36VXefI>
* <https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f>
* Black Box Optimization in machine Learning
* <https://engineering.lehigh.edu/sites/engineering.lehigh.edu/files/_DEPARTMENTS/ise/pdf/tech-papers/17/17T_005.pdf>
* Bayesian Optimization Implementation Example
* <http://krasserm.github.io/2018/03/19/gaussian-processes/>
* <https://github.com/joefarned/SparkGaussianProcess/blob/master/GaussianProcess.scala>
* Hyperband
* <https://arxiv.org/pdf/1603.06560.pdf>
* Distributed Hyperparameter optimization
* <http://www.cs.ubc.ca/labs/beta/Projects/SMAC/papers/parallel_smac_pre_conference_CRC.pdf>
* Sequential Model-Based Optimization for General Algorithm Configuration
* <https://www.cs.ubc.ca/~hutter/papers/10-TR-SMAC.pdf>
* <http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf>
* Introduction
  + <https://arimo.com/data-science/2016/bayesian-optimization-hyperparameter-tuning/>