Introduction to AutoML

Dr. Srijith Rajamohan

ABOUT ME

- Computational Scientist at Virginia Tech
- Principal Scientist at Piemont (https://www.piemontsolutions.com)
- Ph.D. in Computational Engineering
- Background in HPC, ML and Computational Science
- Current interests are
  - Weak Supervision for NLP
  - Dimensionality Reduction
  - Maintainable HPC using Python

CONTENT OUTLINE

What is AutoML?

Hyperparameter Optimization

Comet.ml for Hyperparameter Optimization

Bayesian Optimization

Neural Architecture Search

H2O.ai for AutoML

WHERE DO I LOOK?

- Visit AutoML.org at https://www.ml4aad.org/automl/ for updated information on the state-of-the-art.
- They define it as:

"Automated Machine Learning provides methods and processes to make Machine Learning available for non-Machine Learning experts, to improve efficiency of Machine Learning and to accelerate research on Machine Learning."

WHAT IS AUTOML?

- Machine Learning is inherently an iterative process.
- A lot of time spent selecting the right learning technique/model family and then fine-tuning the right hyperparameters
- AutoML is a suite of tools and techniques.
- Minimize the amount of time spent performing model selection and optimizing hyperparameters.

WHAT IS AUTOML?

- What do people generally mean when they say AutoML?
  - Hyperparameter Optimization
    - E.g. learning rate in Neural Networks (NN), number of clusters in k-Nearest Neighbors(kNN)
  - Model Search
    - Identify the best performing model, e.g. NN vs Logistic Regression vs Support Vector Machine for classification
  - Neural Architecture Search
    - Identify the best architecture for a NN, i.e. number of layers, nodes per layers etc.

WHAT IS AUTOMATED?

- As complexity of problems evolve, off-the-shelf packages become popular
- Not quite AI: Humans still need to be in the loop
- However, a lot of the things can be automated such as model selection, hyperparameter optimization and validation.
- Goal: Minimize the involvement of ML experts in the model development cycle, humans still need to critically analyze the results

WHAT IS AUTOMATED?

- What Machine Learning algorithm to use?
- How to preprocess the data?
- How to set the hyperparameters? E.g. step size in gradient descent, tree depth in decision trees, regularization coefficient etc.
- Auto-WEKA calls it the Combined Algorithm Selection and Hyperparameter optimization

MODEL SELECTION TOOLS

- Open-source tools
  - Auto-sklearn builds upon the functionality of scikit-learn
  - Hyperopt uses Bayesian optimization whereas TPOT uses genetic programming
  - H2O AutoML does model selection and hyperparameter optimization
  - Auto-Keras performs architecture search, allows you to automate the generation of models in Keras
- Commercial tools are AWS SageMaker and Google AutoML

HYPERPARAMETER SEARCH

- Grid Search: Systematically try every combination of hyperparameters on a grid, rigorous and computationally expensive
- Random Search: Sample randomly from your search space, known to work as well or better than grid search
- Evolutionary algorithms
- Bayesian optimization: Use historical information to refine the search space, shown to be quite efficient

SOME BAYESIAN OPTIMIZATION TOOLS

- Gaussian Process-based approaches work best in low-dimensional space while Tree-based models work best in high-dimensional space
- For Tree-based approaches, you have Random-Forest-based SMAC and Tree-structured Parzen estimators (TPE)
- Hyperopt uses TPE
- Auto-sklearn uses SMAC

BAYESIAN OPTIMIZATION

MATH ALERT

- Bayesian optimization uses probabilistic modeling to estimate the relationship between hyperparameter space and the model's performance
- Response surface or 'Surrogate function' used to approximate ML model
- This surrogate function has to be easier to evaluate than the ML model
- Iteratively build up the model by sampling various points 'x' and computing the function f(x) (loss, accuracy etc.) - use this history to refine the model

BAYESIAN OPTIMIZATION

MATH ALERT

- Classified based on a 'Surrogate function' choice
- Gaussian Process, Random Forests, Tree-structured Parzen Estimators
- ‘Acquisition functions’ used to define a balance between exploitation and exploration of the hyperparameter space
- Common to use the Expected Improvement criterion 'which is high in regions of low predictive mean and high predictive variance’ (assuming low f(x) is better)

Gaussian Processes

MATH ALERT

- GPs are distributions of functions
- Think of it as a way to perform regression but where you also get an estimate of the uncertainty associated with each point ’x’
- GP returns the mean value(similar to linear regression) and the variance at any desired point
- The farther you are from an x with a known f(x), the larger the uncertainty
- Computes P(score|configuration) , i.e. P(f(x)|x)
- Expected Improvement, Probability of Improvement etc.

Gaussian Processes

MATH ALERT

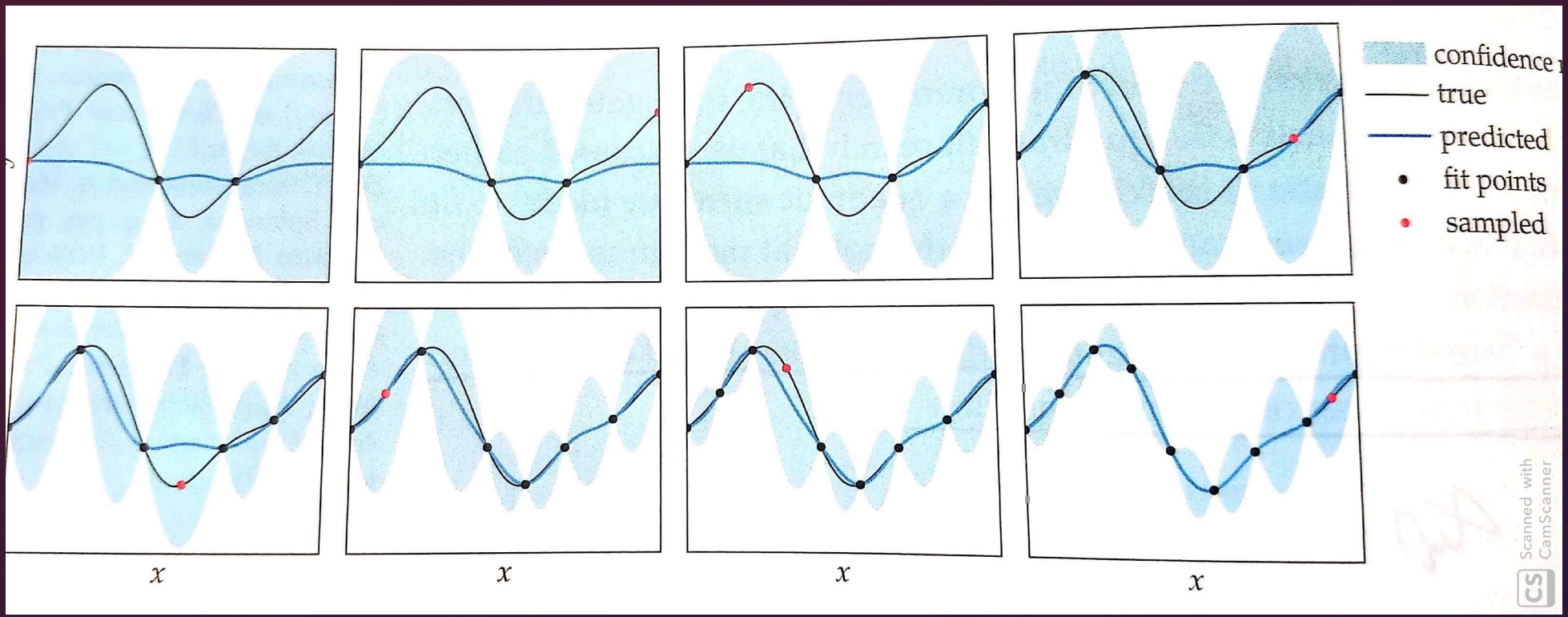


Image from ‘Algorithms for Optimization’

GPs: Exploration

MATH ALERT

- Prediction-based exploration – Seek to find the minimizer of the mean function of the GP. (Greedy minimization)
- Does not take uncertainty into account and new samples can be close to existing ones
- Error-based exploration seeks to find sample points with the maximum uncertainty given by the GP's variance
- Lower Confidence Bound minimization exploration seeks to find a balance between the two

$$LB(x) = \mu(x) - \alpha\sigma(x)$$

Probability of Improvement

MATH ALERT

- The improvement of a function *(f(x) written as y*) is given as

$$I(y) = y_{min} - y \, , y < y_{min}$$

$$= 0 \, , \qquad\qquad y > y_{min}$$

- The probability of improvement is given as

$$\mathrm{P}(\mathrm{y} < y_{min}) = \int p(y|x)dy = \int N(y|\mu, \sigma)dy$$

where the integral goes from negative infinity to y_min

Probability of Improvement

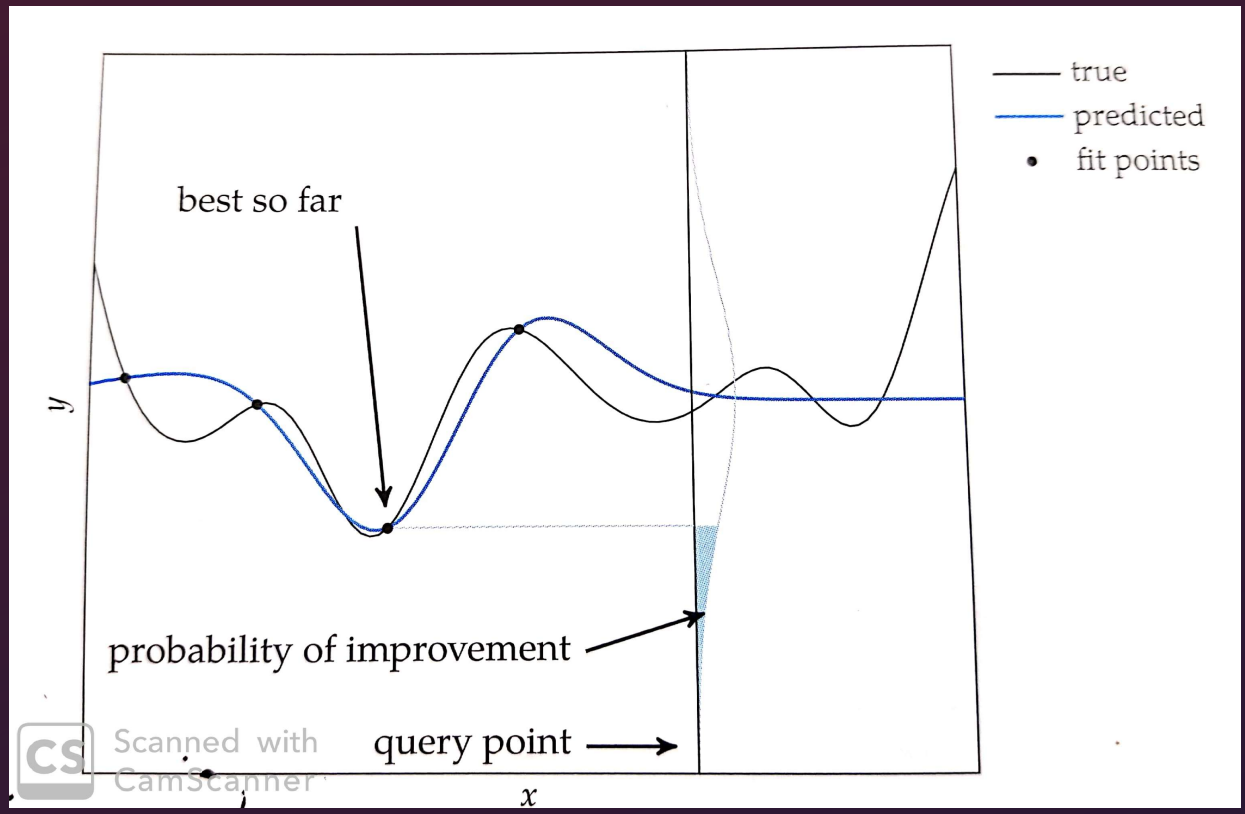


Image from 'Algorithms for Optimization'

Probability of Improvement

MATH ALERT

- For Improvement of a function: the new point has to be better than the current best point ($y < y_{min}$)
- For Probability of improvement: the new candidate point should have sufficient variance such that the probability obtained by integrating the probability density from negative infinity to y_{min} is large enough.
- For this the mean should be low enough and the variance should be high enough.

Expected Improvement

- The Expected Improvement can be computed from the two terms as

$$EI(x) = E[I(y)] = \int I(y)\, p(y|x)dy$$
$$= \int (\, y_{min} - y)p(y|x)dy$$

where

$$p(y|x) = N(y|\mu, \sigma)dy$$

and the integral goes from negative infinity to y_min

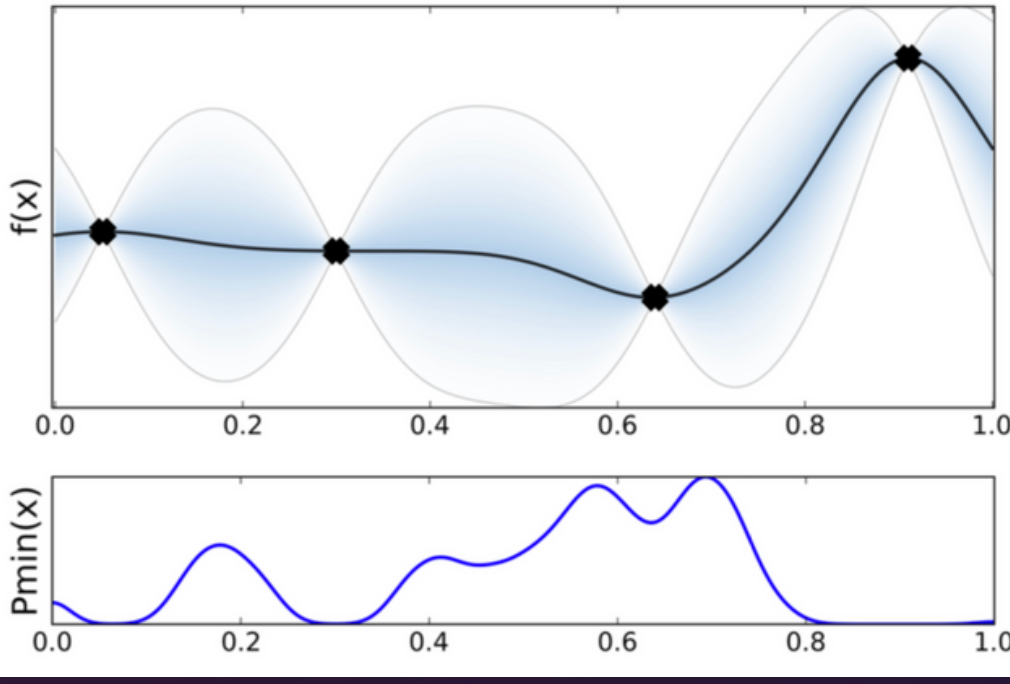
Expected Improvement

MATH ALERT

Acquisition function

$$EI(x) = \int_{-\infty}^{fbest} (f_{best} - f) p(f|x) df$$

Image from Cedric Archchambeau' slides
Image courtesy of Javier Gonzalez



Expected Improvement

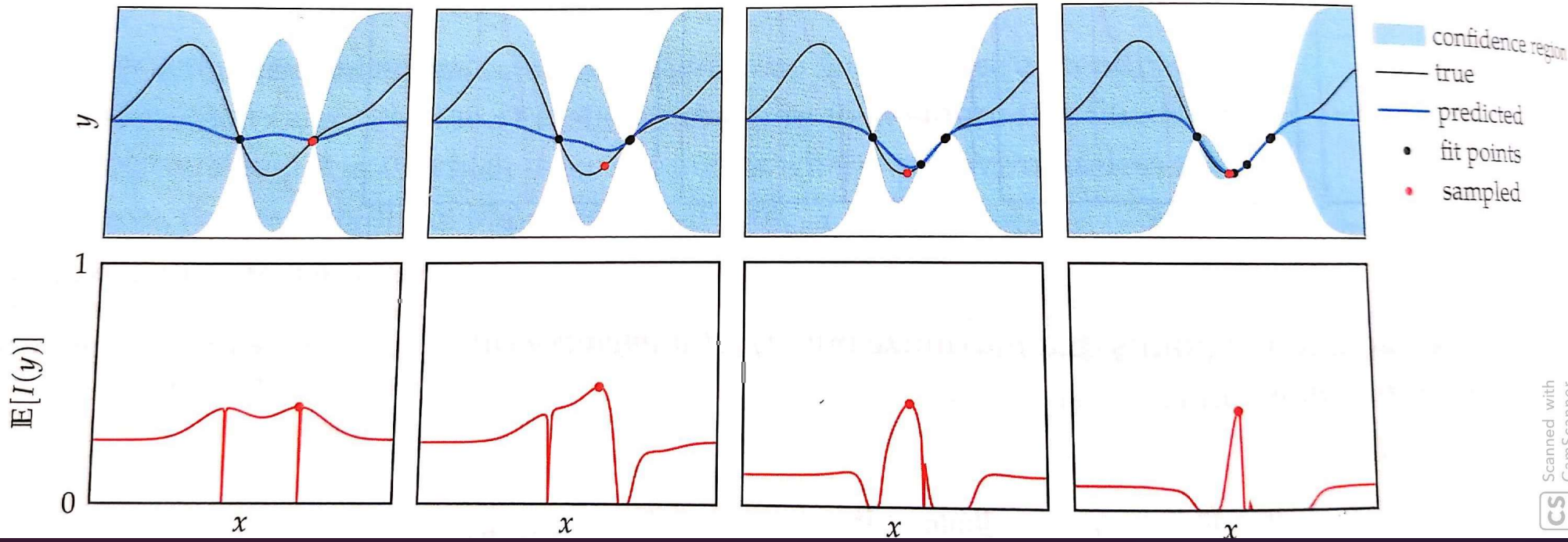


Image from 'Algorithms for Optimization'

SUMMARY

MATH ALERT

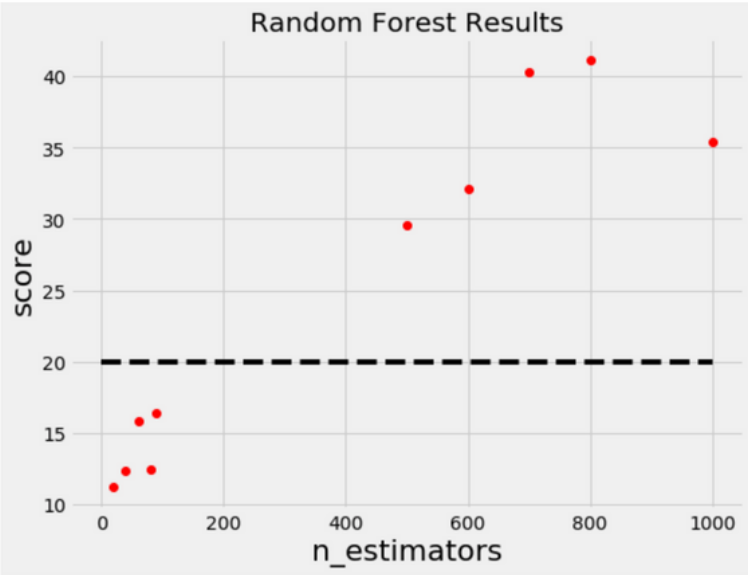
- Select a number of points at random and run the ML model to initialize a surrogate model
- Using the surrogate model and a suitable ‘acquisition function’, optimize this ‘acquisition function’ to get a new point. This is way cheaper than running the real ML model.
- Using the newly identified point, update the surrogate model and repeat the step above.
- Usually runs for a fixed time, before the optimization is terminated and performance of hyperparameter optimization is evaluated.

TREE STRUCTURED PARZEN ESTIMATORS

- Tree of Parzen estimators
- Uses p(configuration|score) instead of p(score|configuration)
- Uses probability densities as a surrogate function
- Sample the response surface to get objective function values (score), split the results into two groups based on a threshold y*
  - $p(config|score) = l(config), score < y^*$
  - $p(config|score) = g(config), score > y^*$
- Two probability densities l(x) and g(x)

TREE STRUCTURED PARZEN ESTIMATORS

MATH ALERT



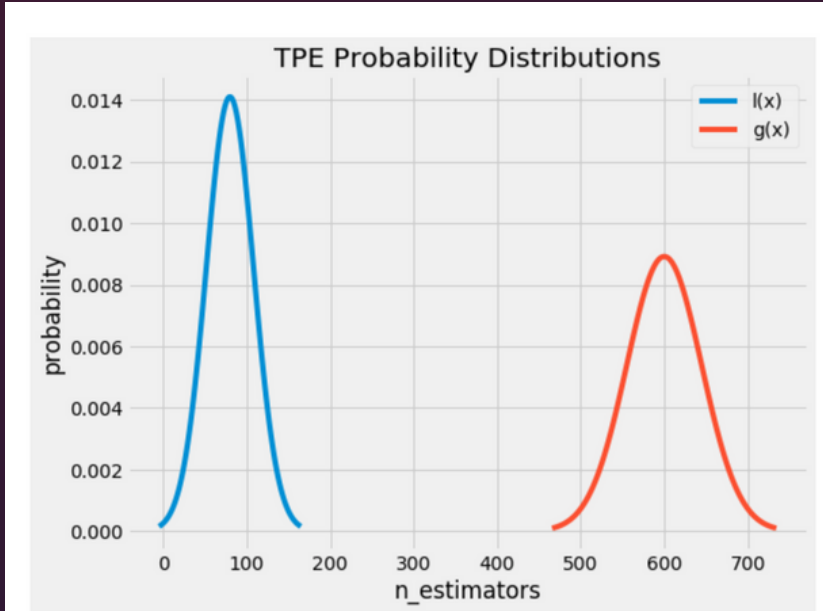


Image from https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f

TREE STRUCTURED PARZEN ESTIMATORS

- Can be shown using Bayes theorem that
  - $EI \propto l(x)/g(x)$, where x is a point in the hyperparameter search space
- Pick the next point as the one that maximizes the EI using the ratio above
- Limitation: cannot optimize hyperparameters independently of each other

THE SMAC ALGORITHM

- Sequential Model-based Algorithm Selection (SMAC) uses Random Forests or approximate GPs to determine viable configurations
- For RFs, the mean at each point is the prediction of the RF while the variance at each point is now the variance of the predictions of the separate trees in the RF
- Based off the Sequential Model-Based Optimization (SMBO) algorithm but can also handle categorical variables with Hamming distance function kernel
- The next point that is selected for evaluation has the highest EI

META-LEARNING

- Used in Auto-sklearn and H2O
- Uses characteristics or attributes of the data, called meta-features, that can be computed efficiently
- They can be number of points, features, classes etc.
- Used to determine what algorithm to use on unseen data
- Provides a warm-start for Bayesian optimization
- Provides coarse-grained information on performance by suggesting good candidates

META-LEARNING

- Use a set of datasets and evaluate their meta-features
- Find the ML model that performs the best on each dataset
- For each new dataset D, calculate its meta-features and find the k closest datasets, using a distance metric on the meta-features, from the set above
- Use the stored ML models corresponding to the above k datasets and use them as starting points for the new dataset

META-LEARNING

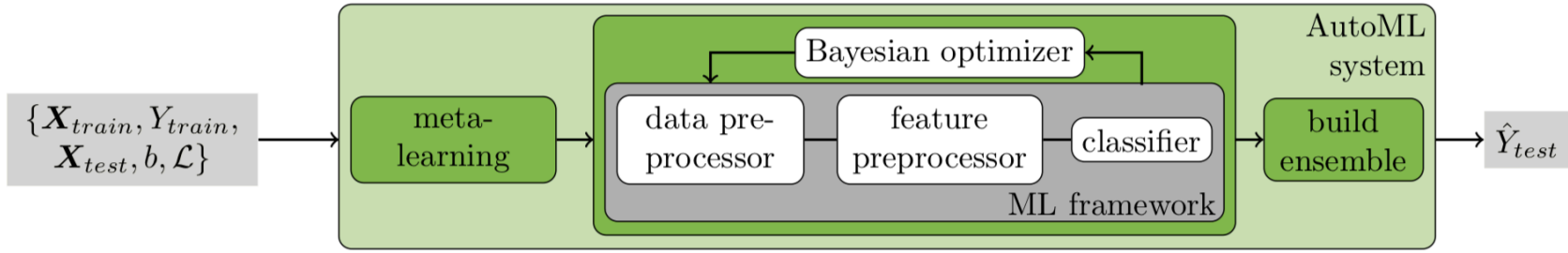


Image from" Efficient and Robust Automated Machine Learning "

NEURAL ARCHITECTURE SEARCH

- Most Neural Network architectures created by people, a very time consuming iterative process
- Neural Network 'architecture engineering' can be automated by Neural Architecture Search (NAS)
- Adjust the number of layers, nodes and the connectivity through an optimization process
- Three key components: search space, search strategy and performance estimation strategy

NEURAL ARCHITECTURE SEARCH

- Very time and resource intensive process
- Deep architectures for Computer Vision problems almost impossible to do without a cluster of GPUs
- Google Cloud AutoML: an end-to-end solution for doing AutoML on the Google Cloud provides this option

Optuna

- Black-box optimization
- No requirements for differentiability
- Consists of a sampler and a pruner
- Sampler decides which hyperparameter to optimize next in the iteration
- Pruner decides when to terminate the evaluation or optimization of a certain hyperparameter

H2O.ai

- Hyperparameter optimization and model selection
- Python and R version
- Runs locally or install on a cloud instance or scalable across multiple nodes
- Available as a service on AWS
- Available with Hadoop
- GPU-enabled version with limited functionality