Bayesian Optimization for Machine Learning

Math 693 A – Project

by

Pradeep Singh

Outline

- Machine Learning & Hyperparameters
- Hyperparameter Optimization
  - Grid Search
  - Random Search
  - Manual Search / Hand Tuning
- Problem with current methods?
- Bayesian Optimization
  - Surrogate Models
  - Acquistion Functions
- Algorithm
- Tuning XGBoost Hyperparameters
- Results
- Conclusion

What is Machine Learning?

In general, machine learning is study of algorithms and models that learns from data.

Mathematically, a machine learning model is a function that maps inputs to the outputs with several (hyper)parameters that need to be learned from the data by fitting a model to the data.

Hyperparameters & Tuning

- Special kind of parameters that cannot be directly learned from the regular training process.
- Express “higher-level” properties of the model such as its complexity or how fast it should learn.
- Fixed before the actual training process begins.
- Examples: number of neural network layers, learning rate, depth of a tree, number of clusters in k-means, etc.

Can we frame this as an Optimization Problem?

- Given a learner *M*, with parameters *x* and a loss function *f*, find *x* such that *f* is maximized, or minimized, by evaluating *f* for sampled values of *x*.
- In short, find the optimal hyperparameter values of a learning algorithm that produce the best model.

- Current Methods:
    1. Grid Search
    2. Random Search (Current best)
    3. Manual Search (Graduate student search)

Grid Search

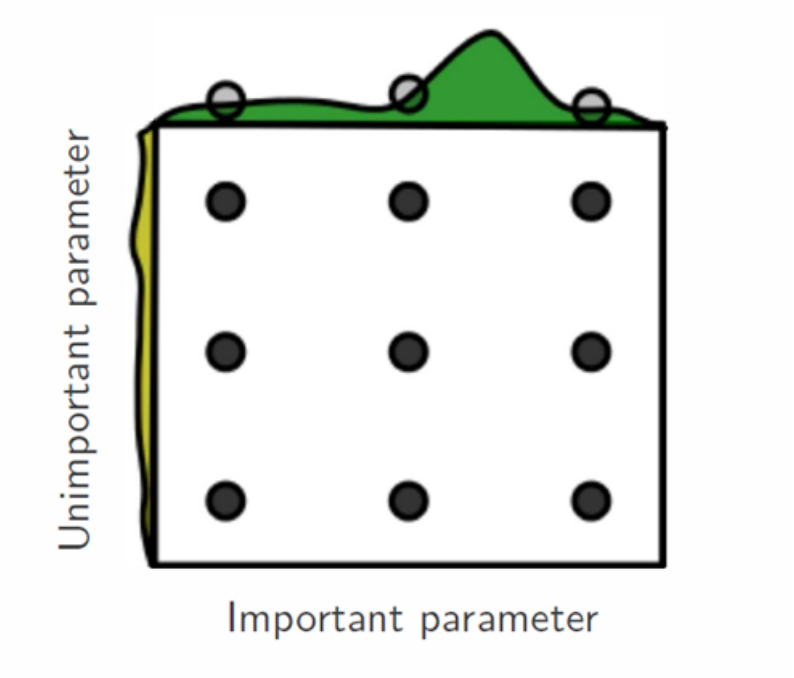


Photo by [Bergstra, 2012]

Random Search

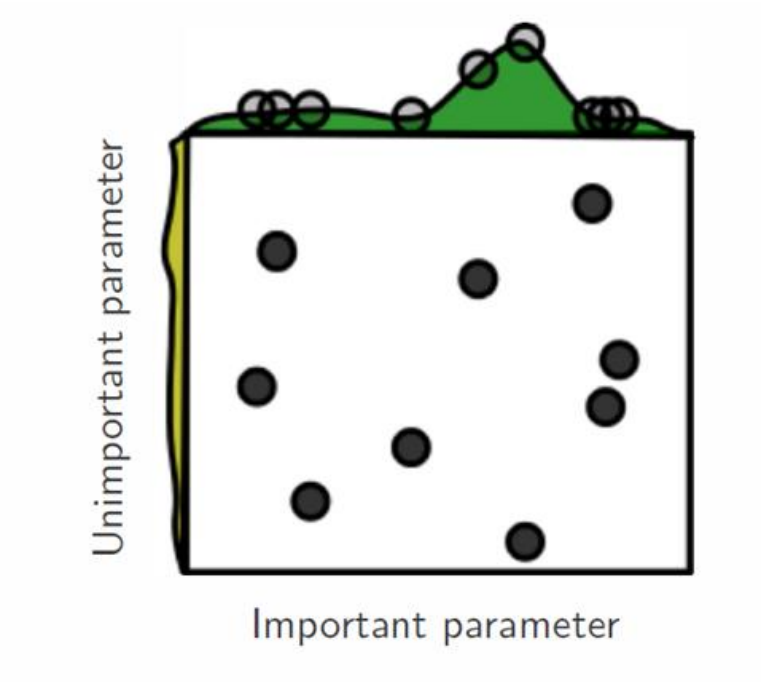


Photo by [Bergstra, 2012]

Manual Search (Grad student search)

- Humans manually pick and choose different values for different parameters and select the best one.
- Humans have ability to learn from previous experience and mistake, which they use to evaluate their choices for parameters when tuning their models.
- Least productive methods.

Problem?

- Let's assume, we need to tune a model with 5 hyperparameters:

Hyperparameters	5 (5-dimensional problem)
Number of values	10 per parameter
Number of evaluations	100,000 evaluations
Training Time	10 mins (on avg.) per evaluation
Time for tuning model	100,000 * 10 mins = almost 2 years

- This is crazy.
- **NOTE:** Typically, network trains much longer and we need to tune more hyperparameters, which means that it can take forever to run grid search, manual search or even random search for typical neural network.

Problems & What do we need?

- Time consuming.
- Resource intensive.
- Use previous evaluations to select next sample to evaluate for. Sample wisely.
- How certain or uncertain are we of our model? Account for uncertainty.
- Move in right direction. If you can't be correct, try to be less wrong.

- And, that's what Bayesian methods do:
  a) They use prior belief to find next samples.
  b) They account for uncertainty.

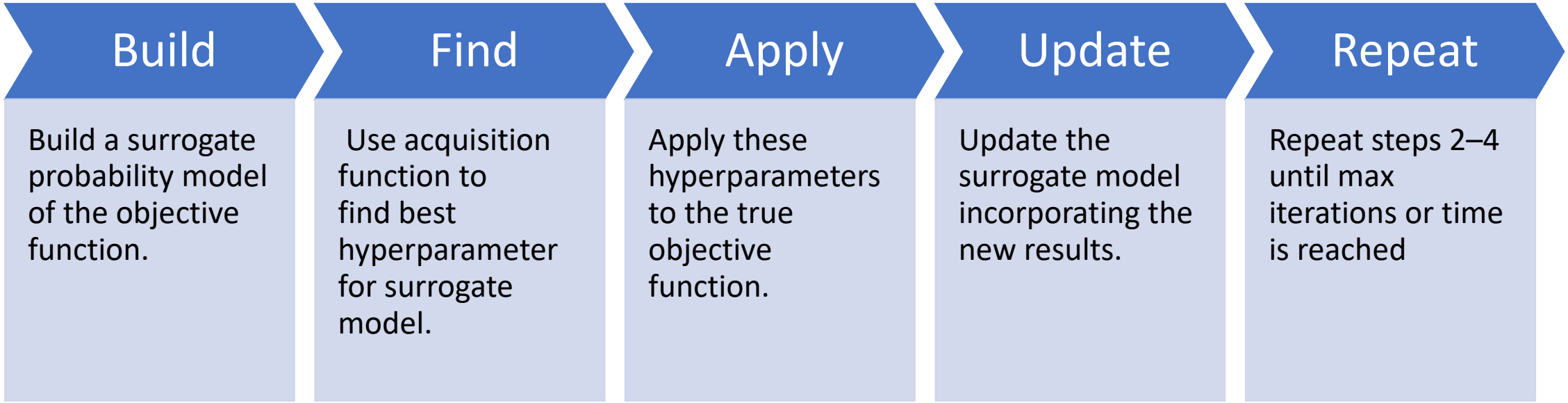
Bayesian Optimization

- Bayesian optimization is a derivative-free optimization method.
- Falls in a class of optimization algorithms called *Sequential model-based optimization (SMBO)* algorithms.

- **Basic Idea:** Model objective function using some surrogate model and use previous observations of the objective function, to determine the next (optimal) point to sample function for.

- Two important parts:
    1. Surrogate Models
    2. Acquisition Functions

Basic Idea:


Build
Build a surrogate probability model of the objective function.
Find
Use acquisition function to find best hyperparameter for surrogate model.
Apply
Apply these hyperparameters to the true objective function.
Update
Update the surrogate model incorporating the new results.
Repeat
Repeat steps 2–4 until max iterations or time is reached

Surrogate Model (Gaussian Processes)

- A GP is the generalization of a Gaussian distribution to a distribution over *functions*, instead of random variables and is specified by its mean function $m(x)$, and covariance function $k(x,x')$.
- Think of a GP as a function that, instead of returning a scalar $f(x)$, returns the mean and variance of a normal distribution over the possible values of f at x.

Why GP?

- The GP posterior is cheap to evaluate.
- Posterior distribution over the loss function is analytically tractable.

What point to sample next?

- Given all this useful guessed information, what point should we check next?
- There are two things we care about:
  - Exploration
  - Exploitation

- Use acquisition functions
  - Expected improvement
  - Probability of improvement
  - Entropy search
  - Upper confidence bound

Acquisition functions

- Acquisition functions propose sampling points in the search space using exploitation-exploration tradeoff.
- Exploitation - sampling where the surrogate model predicts a high objective.
- Exploration - sampling at locations where the prediction uncertainty is high.
- Both correspond to high acquisition function values.

Expectation of Improvement (EI)

- Evaluate the next point where the expected improvement is highest.
- So if:
    1. $\mu(x)$ is the guessed value of the function at x.
    2. $\sigma(x)$ is the standard deviation of output at x.
    3. $\Phi(\cdot)$ and $N(\cdot)$ refers to the CDF and PDF of a standard normal distribution.
- Then our expectation of improvement is $A(x)$,

$$\gamma(x) = \frac{f(x^*_{\text{current guessed arg min}}) - \mu(x)}{\sigma(x)}$$

$$A(x) = \sigma(x)\left(\gamma(x)\Phi(\gamma(x)) + \mathcal{N}(\gamma(x))\right)$$

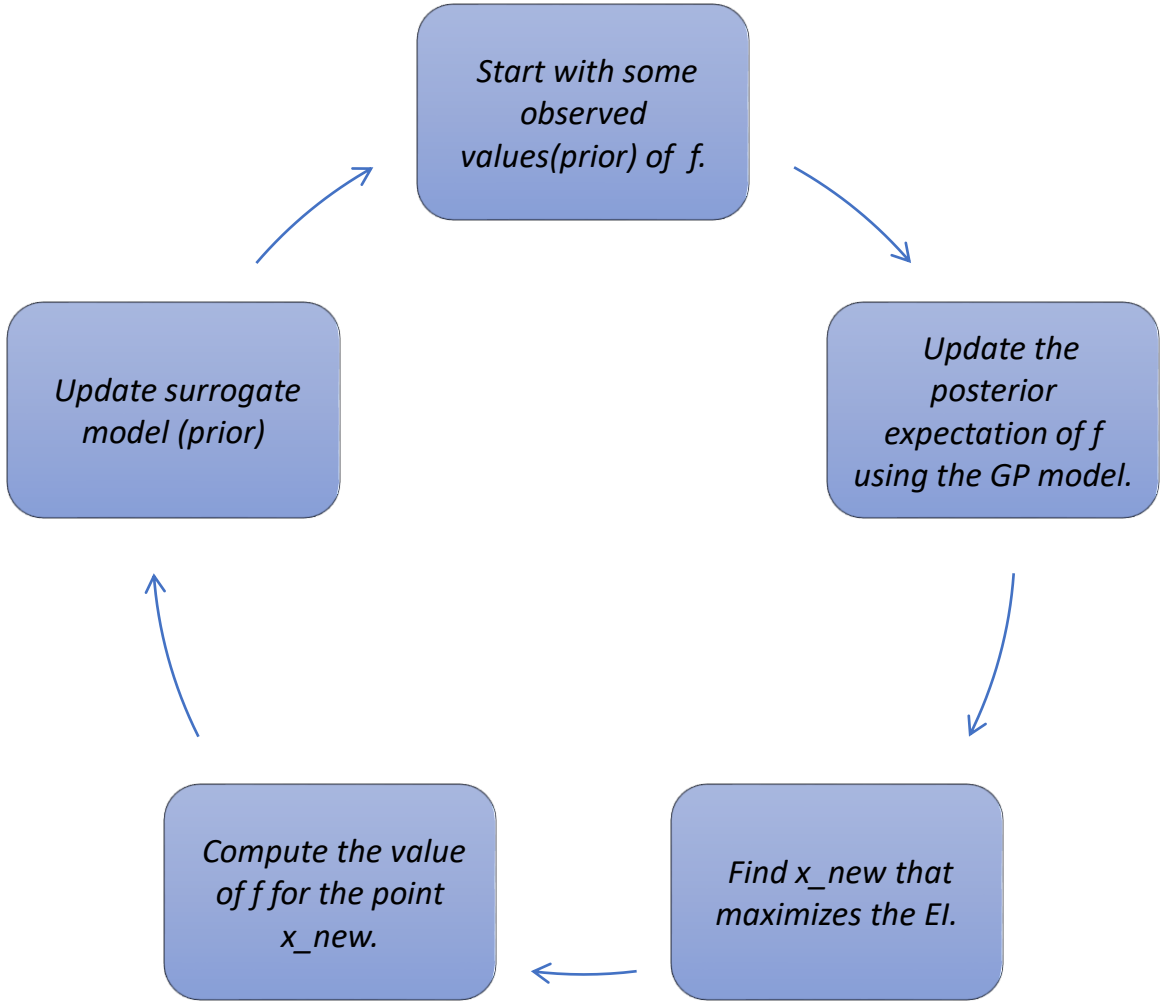
Intution behind EI:

- EI is high when the (posterior) expected value of the loss $\mu(\mathbf{x})$ is higher than the current best value $f(\mathbf{x})$; OR
- EI is high when the uncertainty $\sigma(\mathbf{x})$ around the point $\mathbf{x}$ is high.

Intuitively, this makes sense,

- If we maximize the expected improvement, we will either sample from points for which we expect a higher value of f, or points in a region of f we haven't explored yet ($\sigma(\mathbf{x})$ is high).
- In other words, it trades off exploitation VS exploration.

Algorithm:



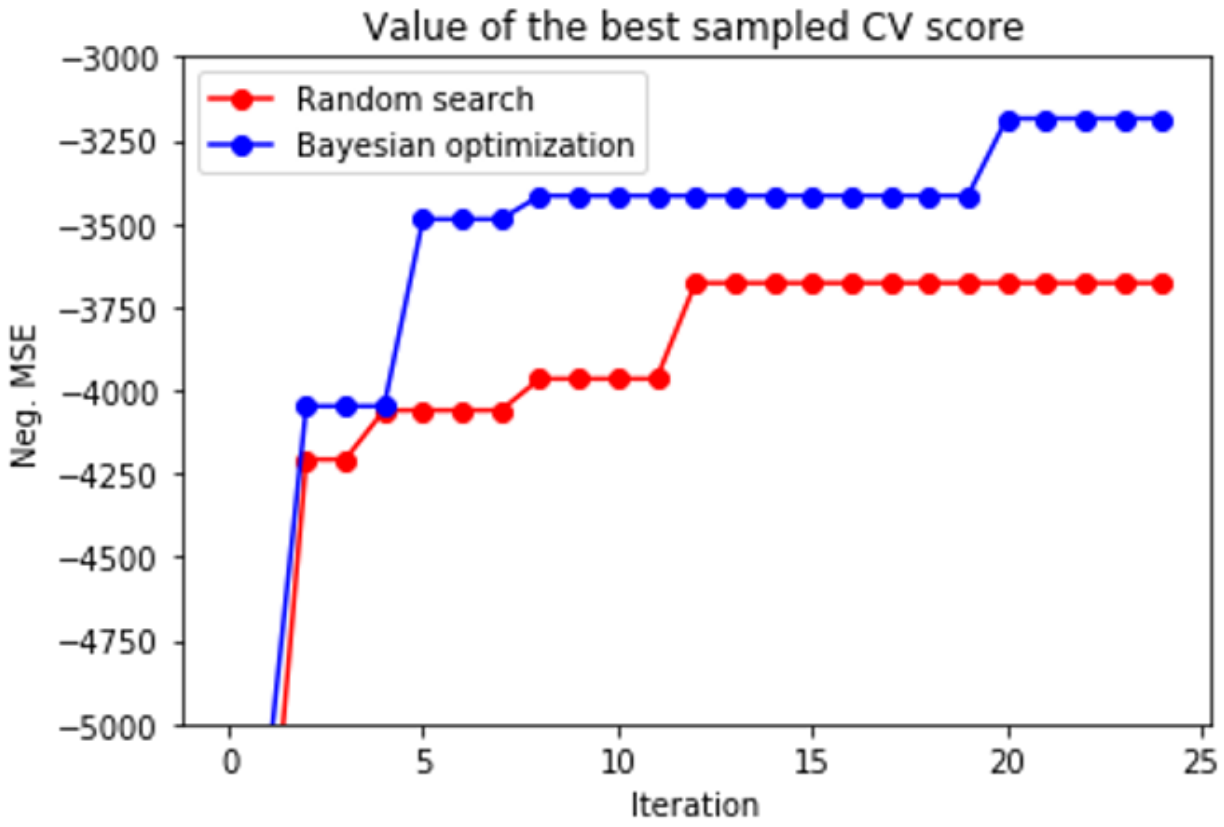
* Repeat for a pre-specified number of iterations, or until convergence.

Bayesian Optimization of XGBoost:

- Using Random search and Bayesian Optimization
- Hyperparameters to be optimzed: (5-dimensional)
    - Learning rate {Uniform (0, 1)}
    - Gamma {Uniform (0, 5)}
    - Max Depth {range(1, 50)}
    - Number of estimators {range(1, 300)}
    - Minimum child weight {range(1, 10)}

- Dataset – A toy (diabetes) dataset with 500 samples

Results


Value of the best sampled CV score
Random search
Bayesian optimization
Neg. MSE
−3000
−3250
−3500
−3750
−4000
−4250
−4500
−4750
−5000
0
5
10
15
20
25
Iteration

Conclusion

- On average, Bayesian optimization finds a better optimium in a smaller number of steps than random search.
- This trend becomes even more prominent in higher-dimensional search spaces.
- Here, the search space is 5-dimensional which is rather low to substantially profit from Bayesian optimization.

References:

1. J. Mockus (2013), Bayesian approach to global optimization: theory and applications. Kluwer Academic.
2. E. Brochu (2010), A Tutorial on Bayesian Optimization of Expensive Cost Functions. arXiv:1012.2599.
3. J. Bergstra (2011), Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems, 2011.
4. J. Bergstra (2012), Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research 13 (2012) 281-305
5. Peter I. Frazier (2018), A Tutorial on Bayesian Optimization. arXiv:1807.02811
6. J. Snoek (2012) Practical Bayesian Optimization of Machine Learning Algorithms. Advances in Neural Information Processing Systems, 2012.
7. B. Letham (2018), Efficient tuning of online systems using Bayesian optimization. Facebook AI Research - Bayesian Analysis 2018.
8. R. Jenatton (2017), Bayesian Optimization with Tree-structured Dependencies. PMLR 70:1655-1664, 2017.

Thanks!