Bayesian Optimization

Motivation

Exploring Bayesian Optimization

HPO problem as a Bayesian Optimization problem

1. **Objective function:** what we want to minimize
2. **Configuration space:** possible values of the hyperparameters.
3. **Surrogate function:** a model for $p(y|x)$. For HPO, y represents the **loss** and x the **configuration**.
4. **Trials:** score, parameter pairs recorded each time we evaluate the objective function.

Sequential Model-Based Optimization

- After each evaluation, the probability model gets updated.
- Next values to try are selected by the algorithm according to a criteria, usually Expected Improvement.
- Finding values that maximize expected improvement is cheaper than evaluating the function itself.
- Having a probabilistic model gives us hope that convergence will take less time.

Sequential Model-Based Optimization (cont.)

There are different choices for building the surrogate model.

- **Gaussian Processes:** $p(y|x) \approx \mathcal{N}(\mu_K, \sigma_K)$
  - K is a *kernel function* that is used to calculate a local mean and variance.
- **Random Forest Regression:** $p(y|x) \approx \mathcal{N}(\mu_B, \sigma_B)$
  - μ_B, σ_B are calculated over the values given by a regression forest.
- **Tree-structured Parzen Estimator**

Tree Parzen Estimator

- Instead of modeling $p(y|x)$, one models $p(x|y)$ and $p(y)$ directly.
- This is achieved by estimating two different processes $\ell(x)$ and $g(x)$, each of which is estimated from quantiles of y.

Implementations

- SMAC (Random forests)
- TPE
- Gaussian Processes: GPyOpt, scikit-optimize.
    - For classification and regression: scikit-learn

References

- Bayesian Optimization Primer
- Hyperopt Jupyter Notebook example, including GBM example.
- Practical Bayesian Optimization of ML algorithms
- Algorithms for hyperparameter optimization