Introduction to Parallel Bayesian Optimization for Machine Learning

Kejia Shi

H_2O.ai Inc.
Columbia University in the City of New York

kejia.shi@columbia.edu

August 16, 2017

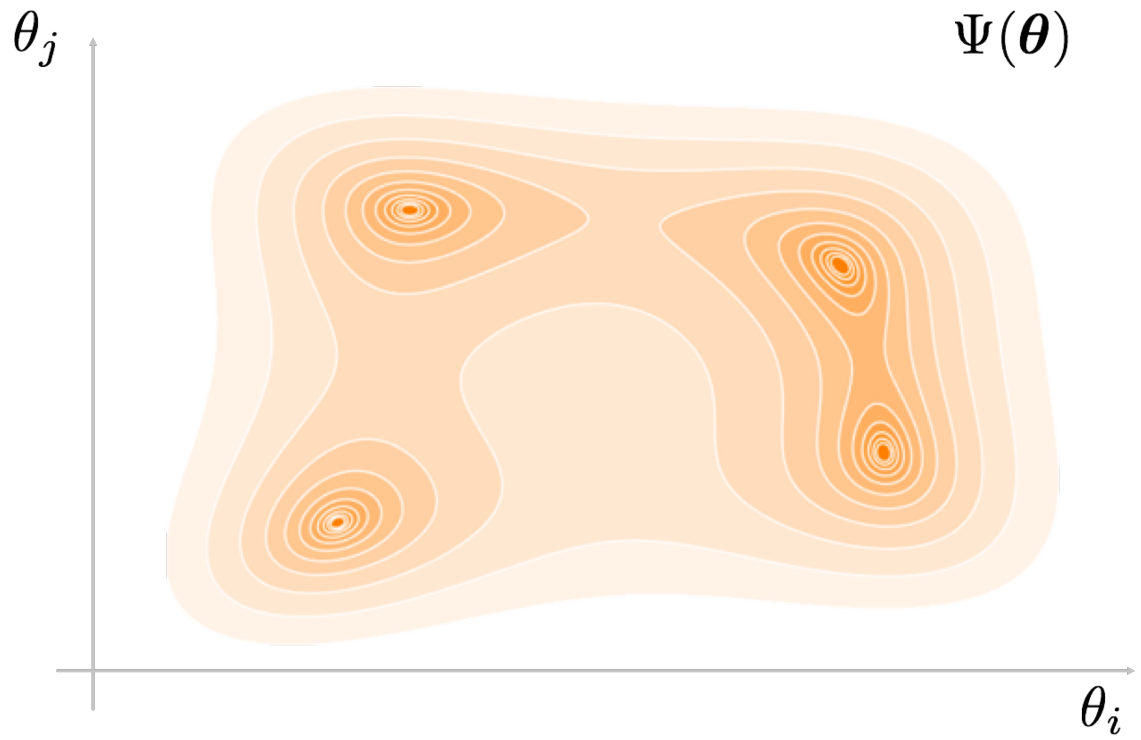
H_2O.ai

Overview

So you did all the data cleaning, feature engineering...
And you fit your first model with hyperparameters...

- Ridge regression: λ...
- Neural network: number of layers, batch size...
- Random forest: number of trees, leaf size...

Now what you are going to do?



- $\theta^* = \arg\min_\theta(\theta)$, for $\theta \in \Theta$
- What is the best set of hyperparameters to optimize the chosen model?

Problem of Hyperparameter Search

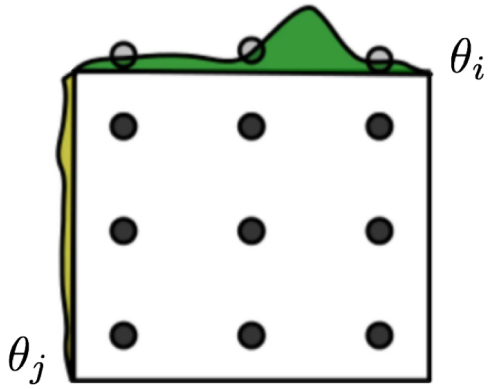
H2O.ai

Hyper-parameter	Variants
Non-linearity	linear, tanh, sigmoid, ReLU, VLReLU, RReLU, PReLU, ELU, maxout, APL, combination
Batch Normalization (BN)	before non-linearity. after non-linearity
BN + non-linearity	linear, tanh, sigmoid, ReLU, VLReLU, RReLU, PReLU, ELU, maxout
Pooling	max, average, stochastic, max+average, strided convolution
Pooling window size	3x3, 2x2, 3x3 with zero-padding
Learning rate decay policy	step, square, square root, linear
Colorspace & Pre-processing	RGB, HSV, YCrCb, grayscale, learned, CLAHE, histogram equalized
Classifier design	pooling-FC-FC-clf, SPP-FC-FC-clf, pooling-conv-conv-clf-avepool, pooling-conv-conv-avepool-clf
Network width	$1/4, 1/2\sqrt{2}, 1/2, 1/\sqrt{2}, 1, \sqrt{2}, 2, 2\sqrt{2}, 4, 4\sqrt{2}$
Input image size	64, 96, 128, 180, 224
Dataset size	200K, 400K, 600K, 800K, 1200K(full)
Batch size	1, 32, 64, 128, 256, 512, 1024
Percentage of noisy data	0, 5%, 10%, 15%, 32%
Using bias	yes/no

Figure: Typical List of a Convolutional Neural Network Training Hyperparameters

Problem of Hyperparameter Search

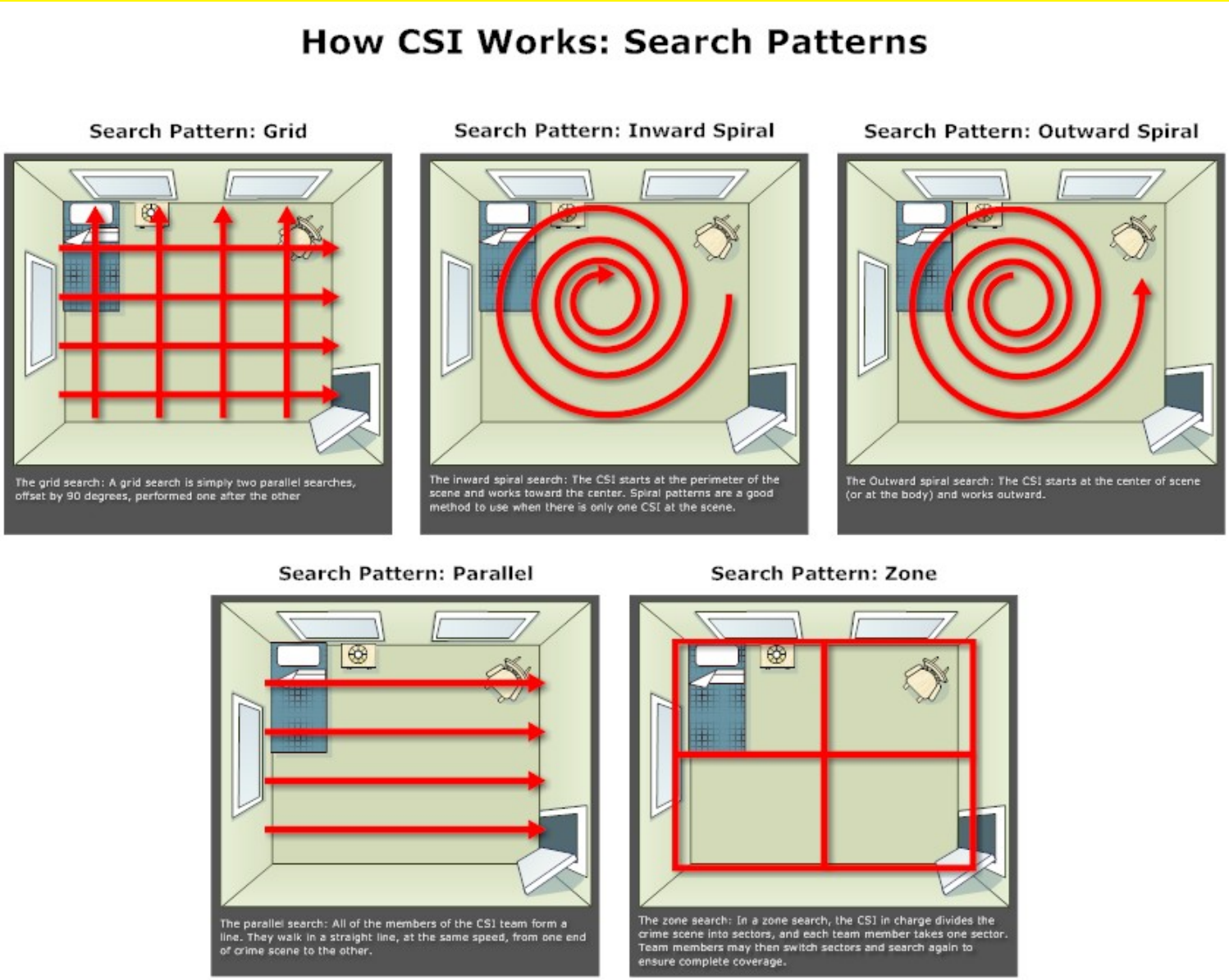
Automated Hyperparameter Search: Grid Search



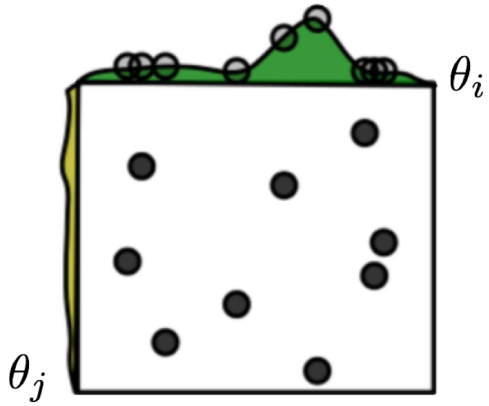

```
hyper_parameters = {
    "learn_rate": [0.01, 0.02, 0.03],
    "max_depth": [5, 10, 50],
    "ntrees": [20, 50, 100]
}

search_criteria = {
    "strategy": "RandomDiscrete", # {Cartesian, RandomDiscrete}
    "stopping_metric": "AUTO", # {AUTO, misclassification, logloss, auc}
    "stopping_tolerance": 0.001,
    "stopping_rounds": 3,

}
```

```
from h2o.estimators.gbm import H2OGradientBoostingEstimator
from h2o.grid.grid_search import H2OGridSearch
gbm_search = H2OGridSearch(H2OGradientBoostingEstimator(nfolds=5),
                           hyper_params=hyper_parameters,
                           search_criteria=search_criteria,
                           grid_id="gbm_search")
gbm_search.train(x=predictors, y=response, training_frame=flights_hex)
```

Automated Hyperparameter Search: Grid Search



Automated Hyperparameter Search: Random Search



Journal of Machine Learning Research 13 (2012) 281-305 Submitted 3/11; Revised 9/11; Published 2/12

Random Search for Hyper-Parameter Optimization

James Bergstra JAMES.BERGSTRA@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Département d'Informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC, H3C 3J7, Canada

Fundamentals

Fundamentals: Review of Probability

Suppose we have two events, A and B that may or may not be correlated,

- A = It's raining
- B = The ground is wet

We can talk about probabilities of these events,

- $P(A)$ = Probability it is raining
- $P(B)$ = Probability the ground is wet

We can also talk about their other probabilities,
Conditional probabilities

- $P(B|A)$ = Probability the ground is wet given that it is raining

Joint probabilities

- $P(A, B)$ = Probability it is raining and the ground is wet

Fundamentals: Calculus of Probability

There are simple rules from moving from one probability to another

- $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$
- $P(A) = \Sigma_b P(A, B = b)$
- and similarly $P(B) = \Sigma_a P(A = a, B)$

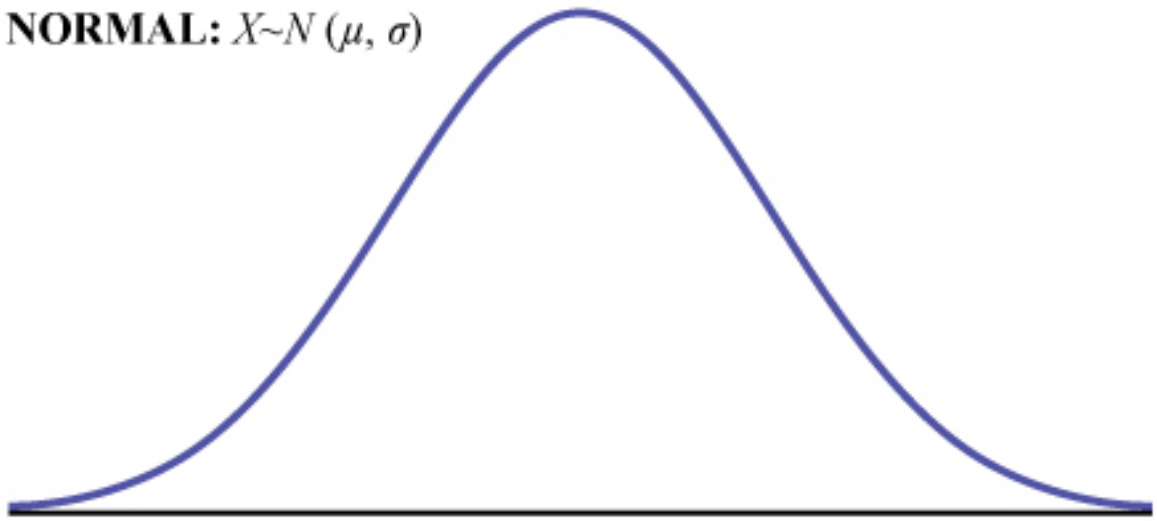
Fundamentals: Bayes Rule

- We want to say something about the probability of B given that A happened, Bayes rule says that the probability of B after knowing A is:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

- Notice: $P(B|A)$ – "posterior", $P(A|B)$ – "likelihood", $P(B)$ – "prior" and $P(A)$ – "marginal" or "evidence".

Fundamentals: Gaussian Distribution



$$x \sim N(\mu, \sigma^2)$$

$$\text{Density: } f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$$

$$\text{Log Density: } \ln f(x|\mu, \sigma^2) = -\frac{(x-\mu)^2}{2\sigma^2} + \text{constant}$$

Fundamentals: Probabilistic Model

- In short, **probabilistic model** is a set of probability distribution $p(x|)$. the distribution family $p(\cdot)$ is selected but we don't know the parameter θ.
- In Gaussian distribution, $p(x|\theta)$, $\theta = \{\mu, \Sigma\}$.

Fundamentals: Maximum Likelihood

- The independent and identically distributed (iid) assumption: $x_i \overset{iid}{\sim} p(x|\theta),\ i = 1, ..., n$.
- With this, the joint density function becomes $p(x_1, ..., x_n|\theta) = \prod_i p(x_i|\theta)$.
- Maximum likelihood approach seeks the value of θ that mximize the likelihood function: $\hat{\theta}_{ML} \triangleq \arg\max_\theta p(x_1, ..., x_n|\theta)$.
- How to solve it? Take gradients!

Fundamentals: Logarithm Trick

- Taking gradients of $\prod_{i=1}^{n} f_i p(x_i|\theta)$ can be complicated. Therefore we use the property of monotonically increasing functions on $\mathbb{R}_+$. Specifically, maximizing $g(y)$ and its monotone transformation $f(g(y))$ is the same.
- Hence we have the equality

$$\ln(\prod_i f_i) = \Sigma_i \ln(f_i).$$

- Notice: *location* of a maximum does not change, while *value* of a maximum does change.

$$\max_y \ln g(y) \neq \max_y g(y)$$

$$\arg\max_y \ln g(y) = \arg\max_y g(y)$$

Bayesian Optimization

Gaussian Process

The linear regression model may be written as:

$$y_i \approx f(x_i;\beta) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_d x_{id} = \beta_0 + \Sigma_{j=1}^{d} \beta_j x_{ij}.$$

We have our training data $(x_1, y_1), ..., (x_n, y_n)$ and we want to use them to learn a set of weights $\beta_0, \beta_1, ..., \beta_d$ such that $y_i \approx f(x_i;\beta)$.
Our *least square* objective function asks us to choose the β's that minimize the sum of squared errors

$$\beta_{LS} = \text{argmin}_\beta \Sigma_{i=1}^{n} (y_i - f(x_i;\beta))^2 \equiv \text{argmin}_\beta L.$$

Gaussian Process: From Linear Regression

H_2O.ai

- Vertical length error. And now $(\beta_0, \beta_1, \beta_2)$ defines this plane. (Share the same solution as *maximum likelihood* approach.)
- **Idea:** Imagine only *one* feature one response, now you are fitting to a **straight line**, which leads to the assumption that errors are **normally distributed**.
- **Extension:**
  - Regularization: l_2, *Ridge*, l_1, *LASSO*
  - Priors: MAP

Probabistic View

- *Maximum Likelihood*
  Likelihood term rising from squared errors $y \sim N(X\beta, \sigma^2 I)$

$$\begin{aligned} \beta_{ML} &= \text{argmax}_\beta \ln p(y|\mu = X\beta, \sigma^2) \\ &= \text{argmax}_\beta -\frac{1}{2\sigma^2}||y - X\beta||^2 \end{aligned} \tag{1}$$

- *Maximum a Posteriori*
  With extra Gaussian prior $\beta \sim N(0, \lambda^{-1} I)$
  Notice $\ln p(y|X)$ is unknown. From Bayes Rule:

$$\begin{aligned} \beta_{MAP} &= \text{argmax}_\beta \ln p(y|\beta, X) + \ln p(\beta) - \ln p(y|X) \\ &= \text{argmax}_\beta -\frac{1}{2\sigma^2}||y - X\beta||^2 - \frac{\lambda}{2}||\beta||^2 \end{aligned} \tag{2}$$

Gaussian Process: In a Bayesian Way

Moving from point estimates to total Bayesian inference:

- How do we take into account the uncertainty associated with the limited data instead of just finding the MAP solution?
- Take off prior from the likelihood! Instead, build a Gaussian prior (over weights), Gaussian likelihood model.
  (easy for computation of the weights)
- The idea is to integrate over **multiple** straight lines.

- Settings:

$$\begin{aligned} \textit{Prior} &: \beta \sim N(0, \lambda^{-1} I) \\ \textit{Likelihood} &: y \sim N(X\beta, \sigma^2 I) \end{aligned} \tag{3}$$

- The weights don't even have to be represented explicitly. Different weights result in different possible possible functions related to your predictions.
- Bayesian linear regression takes into account the imperfect information and makes subjective guesses with insufficient data.

- The inner product of two vectors $\mathbf{a} = [a_1, a_2, ..., a_n]$ and $\mathbf{b} = [b_1, b_2, ..., b_n]$ is defined as

$$\mathbf{a} \cdot \mathbf{b} = \Sigma_{i=1} n a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_n b_n.$$

- Inner product operation can be viewed as the simplest kernel function.

$$K(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^T \mathbf{x}_m.$$

Gaussian Process: Kernelization

- **Kernel tricks** add more flexibility without changing the algorithm by efficiently letting you use "inner product" to frame all kinds of computation.
- Specifically, suppose we have some inputs that cannot be easily represented as elements of R^d elements, we can still apply a kernelized algorithm to these inputs, as long as we can define an inner product on them.
- This enable us to integrate over more complicated bases, not just straight lines to some finite basis (e.g. a polynomial basis $\{1, x, x^2\}$).

Common Kernels

- Polynomial kernel: $K(\mathbf{x}_n, \mathbf{x}_m) = (a\mathbf{x}_n^T\mathbf{x}_m + c)^d, a > 0, c \geqslant 0, d \in \mathbb{Z}_+$
- Exponential of a quadratic kernel:
  $K(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left(-\frac{\theta_1}{2}(\mathbf{x}_n - \mathbf{x}_m)^T(\mathbf{x}_n - \mathbf{x}_m)\right) + \theta_2 + \theta_3\mathbf{x}_n^T\mathbf{x}_m$
- Squared exponential kernel:
  $K(\mathbf{x}_n, \mathbf{x}_m) = \tau^2 \exp\left(-\frac{1}{2l^2}(\mathbf{x}_n - \mathbf{x}_m)^T(\mathbf{x}_n - \mathbf{x}_m)\right)$
- Matérn 3/2 kernel: $K(\mathbf{x}_n, \mathbf{x}_m) = \sigma_f^2\left(1 + \frac{\sqrt{3}r}{\sigma_l}\right)\exp\left(-\frac{\sqrt{3}r}{\sigma_l}\right)$ where,
  $r = \sqrt{(\mathbf{x}_n - \mathbf{x}_m)^T(\mathbf{x}_n - \mathbf{x}_m)}$.
- Matérn 5/2 kernel: $K(\mathbf{x}_n, \mathbf{x}_m) = \sigma_f^2\left(1 + \frac{\sqrt{5}r}{\sigma_l} + \frac{5r^2}{3\sigma_l^2}\right)\exp\left(-\frac{\sqrt{5}r}{\sigma_l}\right)$

Gaussian Process Model:

- Prior: $\mathbf{f}|\mathbf{X} \sim N(\mathbf{m}, \mathbf{K})$
- Likelihood: $y|f, \sigma^2 \sim N(\mathbf{f}, \sigma^2\mathbf{I})$
- where the elements of the mean vector and kernel/covariance matrix are defined as $m_i \triangleq \mu_0(\mathbf{x}_i)$ and $K_{i,j} \triangleq k(\mathbf{x}_i, \mathbf{x}_j)$.
- Posterior:
- Mean: $\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m})$
- Variance: $\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$
- $\mathbf{k}(\mathbf{x})$ is a vector of covariance terms between $\mathbf{x}$ and $\mathbf{x}_{1:n}$
- These posteriors are used to select the next query point $\mathbf{x}_{n+1}$ as well.

Gaussian Process: Demo and Evaluation

- Linear mean function:
  - Represents an initial guess at the regression function, with the linear model $m(x) = X\beta$ corresponding to a convenient special case, possibly with a hyperprior chosen for the regression coefficients β in this mean function. Posterior concentrates close to the linear model to an extent supported by data
- Covariance function:
  - Controls the smoothness of realizations from the Gaussian process and the degree of shrinkage towards the mean
- Simplicity and flexibility in terms of conditioning and inference!

Gaussian Process: Demo and Evaluation

Advantages

- Can fit a wide range of smooth surfaces while being computationally tractable even for moderate to large numbers of predictors

Disadvantages

- $O(n^3)$ complexity in terms of total number of observations from the inversion of a dense covariance matrix
- This becomes troublesome as the size of the search space grows with the model complexity

Solution

- Computation becomes significantly more accessible and it's possible to train many models in parallel

Gaussian Process v.s. Multivariate Gaussian

In short, Gaussian processes can be considered as *an infinite-dimensional generalization* of multivariate Gaussian distributions.

Acquisition Function

Acquisition Function: Overview

What does *acquisition function* mean?

- An **inexpensive** function $a(x)$ that can be evaluated at a given point x that shows **how desirable** evaluating f at x is expected to be for the minimization problem.

How do we query points to evaluate and update our hyperparameters?

- Still select arbitrarily?
- Utilize our model!

Acquisition Function: Overview

How to utilize the posterior model to create such acquisition functions?

- A setting of the model hyperparameters θ
- An arbitrary query point x and its correponding function value $v = f(\mathbf{x})$
- Map them to a utility function $U = U(\mathbf{x}, v, \theta)$

Now the acquisition function can be defined as an integration over all its parameters, i.e. $v|\mathbf{x}, \theta$ and θ, the selection of the $n+1$ point given its previous data $\mathcal{D}_n$,

$$a(\mathbf{x}; \mathcal{D}_n) = \mathbb{E}_\theta \mathbb{E}_{v|\mathbf{x},\theta}[U(\mathbf{x}, v, \theta)].$$

Acquisition Function: Marginalization

- Deal with $a(\mathbf{x}; \mathcal{D}_n) = \mathbb{E}_\theta \mathbb{E}_{v|\mathbf{x},\theta}[U(\mathbf{x}, v, \theta)]$ is very complicated.
- We start from the outer expectation over unknown hyperparameters θ.
- We wish to marginalize out our uncertainty about θ with $a_n(x) \triangleq \mathbb{E}_{\theta|\mathcal{D}_n}[a(x;\theta)] = \int a(x;\theta) p(\theta|\mathcal{D}_n) \mathrm{d}\theta$.
- Notice: from Bayes Rule, $p(\theta|\mathcal{D}_n) = \frac{p(y|\mathbf{X},\theta)p(\theta)}{p(\mathcal{D}_n)}$.
- Two ways to tackle the marginalized acquisition function:
  - Point estimate $\hat{\theta} = \hat{\theta}_{ML}$ or $\hat{\theta}_{MAP}$, therefore $\hat{a}_n(x) = a(x; \hat{\theta}_n)$.
  - Sampling from posterior distribution $p(\theta|\mathcal{D}_n)$, $\{\theta_n^{(i)}\}_{i=1}^S$, therefore $\mathbb{E}_{\theta|\mathcal{D}_n}[a(x;\theta)] \approx \frac{1}{S}\Sigma_{i=1}^S a(x; \theta_n^{(i)})$
- **Idea:** By far we can ignore the θ dependence!

Acquisition Function: Categories

- **Improvement-based Policies**: how much has been improved from current optimal?
- **Optimistic Policies**: based on our optimistic estimation, what would this give us? (exploitation and exploration)
- **Information-based Policies**: how much by evaluating this point can help us reduce the entropy?

Acquisition Function: Improvement-based Policies

Favors points that are likely to improve upon an *incumbent target* τ!
Notice τ in practice is set to the best observed value, i.e. $\max_{i=1:n} y_i$.

- Probability of improvement(PI):
  - Earliest idea
  - $a_{PI}(\mathbf{x}; \mathcal{D}_n) \triangleq \mathbb{P}(v > \tau) = \Phi(\frac{\mu_n(\mathbf{x})-\tau}{\sigma_n(\mathbf{x})})$
- Expected improvement(EI):
  - Improvement function, $I(\mathbf{x})$: $I(\mathbf{x}, v, \theta) \triangleq (v - \tau)\mathbb{I}(v > \tau)$
  - $a_{EI}(\mathbf{x}; \mathcal{D}_n) \triangleq \mathbb{E}[I(\mathbf{x}, v, \theta)] = (\mu_n(\mathbf{x}) - \tau)\Phi(\frac{\mu_n(\mathbf{x})-\tau}{\sigma_n(\mathbf{x})}) + \sigma_n(\mathbf{x})\phi(\frac{\mu_n(\mathbf{x})-\tau}{\sigma_n(\mathbf{x})})$

Acquisition Function: Optimistic Policies

$H_2O.ai$

- As we mentioned acquisition function balances between exploration of the search space and exploitation of current promising areas, the **upper confidence bound** method is one popular way. This category is optimistic facing uncertainty, as the original method uses the upper confidence bound as its acquisition function. Since the posterior at any arbitrary point is a Gaussian, any quantile of the distribution is computed with its corresponding value of β_n as follows,

$$a_{UCB}(\mathbf{x}; \mathcal{D}_n) \triangleq \mu_n(\mathbf{x}) + \beta_n \sigma_n(\mathbf{x}).$$

- We have extra strategies of choosing hyperparameter β_n.

Acquisition Function: Information-based Policies

- What's different? Consider posterior distribution over the unknown minimizer x^*, denoted $p_*(\mathbf{x}|D_n)$.
- One example is the Entropy Search (ES) technique, which aims to reduce the uncertainty in the location x^* by selecting points that are expected to cause the largest reduction in entropy of distribution $p_*(\mathbf{x}|D_n)$.
- Information gain: $U(\mathbf{x}, y, \theta) = H(\mathbf{x}^*|D_n) - H(\mathbf{x}^*|D_n \cup \{(\mathbf{x}, y)\})$
- $a_{ES}(\mathbf{x}; D_n) = H(\mathbf{x}^*|D_n) - \mathbb{E}_{y|D_n,\mathbf{x}} H(\mathbf{x}^*|D_n \cup \{(\mathbf{x}, y)\})$
- $H(\mathbf{x}^*|D_n)$ is the differential entropy of the posterior distribution.

Acquisition Function: Demo

Parallelization

Parallelization

Package	License	URL	Language	Model
SMAC	Academic non-commercial license.	http://www.cs.ubc.ca/labs/beta/Projects/SMAC	Java	Random forest
Hyperopt	BSD	https://github.com/hyperopt/hyperopt	Python	Tree Parzen estimator
Spearmint	Academic non-commercial license.	https://github.com/HIPS/Spearmint	Python	Gaussian process
Bayesopt	GPL	http://rmcantin.bitbucket.org/html	C++	Gaussian process
PyBO	BSD	https://github.com/mwhoffman/pybo	Python	Gaussian process
MOE	Apache 2.0	https://github.com/Yelp/MOE	Python / C++	Gaussian process

An

more Bayesian optimization packages (scikit-optimize, GPyOpt...)

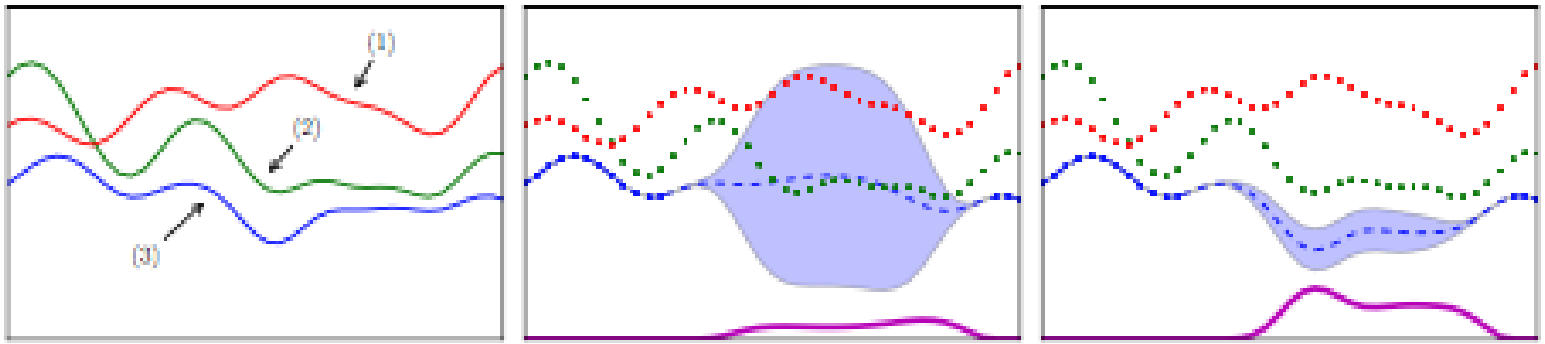
1. On Query Methods (Or Mostly,) Acquisition Function:
Most common: Expected Improvements

- Snoek et al. 2012, Spearmint: Monte Carlo Acquisition Function
- Wang et al. 2016, MOE (Yelp)[12]: q-EI, $argmax x_1, , x_q EI(x_1, ..., x_q)$
  - Construct an unbiased estimator of $\nabla EI(x_1, , x_q)$ using infinitesimal perturbation analysis (IPA)
  - Turn to a gradient estimator problem under certain assumptions

2. Entropy Search

Consider a distribution over the minimum of the function and iteratively evaluating points that will most decrease the entropy of this distribution.

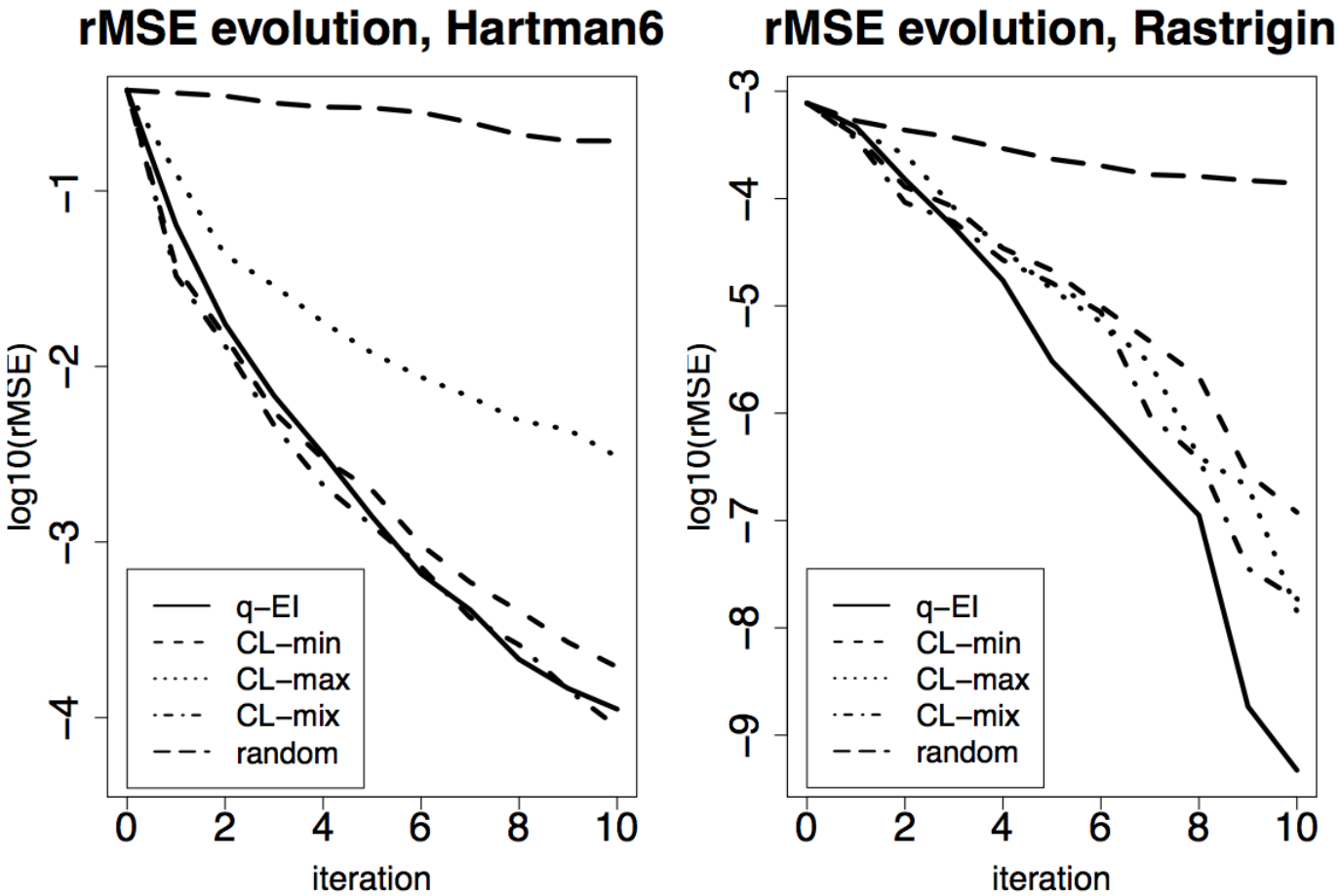
- Swersky et al. 2013, Multi-task:



(a) Multi-task GP sample functions (b) Independent GP predictions (c) Multi-task GP predictions

Motivation: a single model on multiple datasets, perform low-cost exploration of a promising location on the cheaper task before risking an evaluation of the expensive task

Constant Liar: a constant target value is chosen for all pending new evaluations (Chevalier, Ginsbourger 2012)



References

1. Adams, R. P. (2014) A Tutorial on Bayesian Optimization for Machine Learning. Retrieved from https://www.iro.umontreal.ca/ bengioy/.../Ryan$_a$*dams*$_1$40814$_b$*ayesopt*$_n$*cap.pdf*
2. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.
3. Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). Bayesian Data Analysis (Vol. 2). Boca Raton, FL: CRC press.
4. Hennig, P., & Schuler, C. J. (2012). Entropy Search for Information-Efficient Global Optimization. Journal of Machine Learning Research, 13(Jun), 18091837.
5. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. Proceedings of the IEEE, 104(1), 148175. http://doi.org/10.1109/JPROC.2015.2494218

References

6. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2016). Hyperband: A novel bandit-based approach to hyperparameter optimization. arXiv preprint arXiv:1603.06560.
7. scikit-optimize Package Documentation. Retrieved from https://scikit-optimize.github.io
8. Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Advances in neural information processing systems (pp. 2951-2959).
9. Swersky, K., Snoek, J., & Adams, R. P. (2013). Multi-task bayesian optimization. In Advances in neural information processing systems (pp. 2004-2012).

References

Pictures:
1. How CSI Works/Grid Search: https://s-media-cache-ak0.pinimg.com/originals/3b/dd/d2/3bddd27d048504c1bdb556b17f9ae86e.jpg
2. Catalhoyuk, The team excavates in the South Area on buildings in multiple levels to better understand the stratigraphy, Retrieved from (https://www.flickr.com/photos/catalhoyuk/19080999299):
3. A. Honchar, Neural Networks for Algorithmic Trading - Hyperparameters optimization, Retrieved from (https://medium.com/alexrachnog/neural-networks-for-algorithmic-trading-hyperparameters-optimization-cb2b4a29b8ee)
4. Grid Search and Random Search Demo Plots, Retrieved from (http://blog.kaggle.com/2015/07/16/scikit-learn-video-8-efficiently-searching-for-optimal-tuning-parameters/)

The End