# Automated Model Selection via Gaussian Processes

**Rui Shu** RSHU15@STANFORD.EDU
**Stephen Koo** SCKOO@STANFORD.EDU
AA 228, Stanford University

## Abstract

Lorem ipsum.

## 1. Introduction

Talk about the difficulty of model selection. There are a lot of techniques out there for variable selection. This doesn't address the question of selecting the model family. There has been recent interest in the use of deep neural networks, but these often require large amounts of data to be trained properly, are computationally costly, are difficult to interpret and intuit, and require careful tuning of hyperparameters to train successfully. There are also methods for hyperparameter tuning, but we still need to decide which family of model! Ultimately, these approaches simply abstracts human-assisted model selection to a higher level: that of determining which of these approaches to even use in the first place.

There are general rules of thumb. Always start with a simple model and gradually work our way to harder models. The question then becomes whether we can automate this process of model selection in its entirety. We propose to do so by formulating the question as a POMDP that trades-off between model complexity and model accuracy.

## 2. Hyperparameter Selection

Many machine learning models requires the selection of hyperparameters *a priori*. These hyperparameters reflect assumptions the model makes about the data distribution prior to training the model on the existing data. The most common approach to hyerparameter tuning involves cross-validation: using a given set of hyperparameters, a model is trained on training data and evaluated on a test set. The optimal set of hyperparameters is thus the hyerparameters that maximizes the model's performance on the test set. It is thus possible to consider the existence of a latent function $f$ that maps a given set of hyperparameters $\mathbf{x} \in \mathbb{R}^k$ to a performance metric. For a classification problem, this metric may, for example, be the accuracy of the model,

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left\{y_i = \hat{y}_i \mid \mathbf{x}\right\}, \tag{1}$$

where $y_i$ corresponds to the true label of the $i^{th}$ sample in the test set and $\hat{y}_i$ corresponds to label predicted by the trained model given hyerparameters $\mathbf{x}$. It is often the case that $f(\mathbf{x})$ is observed with noise (resulting from any randomness associated with the training of the model), thus making it appropriate to reason about the expected value of $f$ instead.

### 2.1. Gaussian Processes

Since the goal to to find $\arg\max_{\mathbf{x}} \mathbb{E}\left[f(\mathbf{x})\right]$, the task of hyperparameter selection has often been formulated as a function optimization procedure performed using Gaussian process regression. Gaussian process regression is a non-parametric Bayesian modeling tool that imposes a prior distribution over $f$ and updates its belief about the distribution upon the observation of data. Formally, a Gaussian process is parameterized by a mean function $m(\mathbf{x})$ and a kernel $k(\mathbf{x}, \mathbf{x}')$, which we define as,

$$m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] \tag{2}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\right], \tag{3}$$

which can be rewritten as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. The exact choice of the functions $m$ and $k$ are priors required by the Gaussian process and the proper selection of such priors is describes in the next section. Crucially, a Gaussian process simply describes a collection of random variables $\{f(\mathbf{x}_i)\}_{1:n}$ drawn from a multivariate Gaussian distribution, and can thus be rewritten as,

$$\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix}, \begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix}\right), \tag{4}$$

where the dependency on $\mathbf{x}_{1:n}$ ($f_i = f(\mathbf{x}_i)$, $m_i = m(\mathbf{x}_i)$ and $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$) is suppressed for notational simplic-

*Proceedings of the $32^{nd}$ International Conference on Machine Learning*, Lille, France, 2015. JMLR: W&CP volume 37. 

ity. For further simplicity, we write,

$$f_{1:n} \sim \mathcal{N}\left(m_{1:n}, k_{(1:n)\times(1:n)}\right). \tag{5}$$

When seen as a multivariable Gaussian distributtion, it is easy to see how updating the Gaussian process involves computing a conditional Gaussian distribution. Supposing that the first $n-1$ points $f_{1:n-1}$ were observed, we compute the conditional distribution as,

$$\begin{bmatrix} f_{1:n-1} \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_{1:n-1} \\ m_n \end{bmatrix}, \begin{bmatrix} k_\alpha & k_\beta \\ k_\beta^\top & k_\gamma \end{bmatrix}\right), \tag{6}$$

$$f_n \mid f_{1:n-1} \sim \mathcal{N}(m_{\mathcal{D}}(x_n), k_{\mathcal{D}}(x_n, x_n)), \tag{7}$$

$$m_{\mathcal{D}} = m_n + k_\beta^\top k_\alpha^{-1}(f_{1:n-1} - m_{1:n-1}), \tag{8}$$

$$k_{\mathcal{D}} = k_\gamma - k_\beta^\top k_\alpha^{-1} k_\beta, \tag{9}$$

where $k_\alpha = k_{(1:n-1)\times(1:n-1)}$, $k_\beta = k_{(1:n-1)\times n}$, and where $k_\gamma = k_{nn}$. Here, $m_{\mathcal{D}}$ and $k_{\mathcal{D}}$ denote the posterior mean and covariance functions upon observation of the data $\mathcal{D} = f_{1:n-1}$. Since it is often the case that realizations of the random variables $f_{1:n-1}$ are observed with some $\sigma$-Gaussian noise, it is common to use $k_\alpha = k_{(1:n-1)\times(1:n-1)} + \sigma^2 I_n$. Crucially, performing Gaussian process regression returns not only a posterior mean function, but also a posterior covariance function that properly captures the uncertainty of the interpolation. It is this ability of a Bayesian modeling tool to reason about the uncertainty of the system makes Gaussian processes for the online reinforcement learning task of function optimization.

#### 2.1.1. Selection of Mean

Despite being non-parameteric, a Gaussian process is not completely free-form and does require the imposition of certain priors, namely the mean and kernel functions. Since it is often the case that the regression task is performed within the unit hypercube, a zero-mean Gaussian process where $m(\mathbf{x}) = 0$ will often suffice and its impact on the posterior mean diminishes quickly with the introduction of more data. It is sometimes useful, however, to choose a quadratic prior.

#### 2.1.2. Selection of Kernel

Perhaps of greater importance to the behavior of the Gaussian process is the kernel function. The choice of kernel functions can greatly impact the posterior mean and covariance of the Gaussian process. A plethora of kernel functions exist, ranging from periodic kernels to the Matérn and squared exponential kernel. Extensive research has been done on the performance of composite kernels generated from the linear combination of a set of basis kernels. Within the context of test-set performance function, however, if one is reasonably confident that the performance function varies smoothly over the hyperparameter space, it is often enough to impose the squared exponential kernel,

$$k(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp\left(-\frac{1}{2}\gamma(\mathbf{x}, \mathbf{x}')\right), \tag{10}$$

where $\gamma(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{k} \frac{\left(x_i - x_i'\right)^2}{\theta_i^2}$. It is still necessary to learn the parameters $\theta_{0:k}$. Of especial importance are the length scale parameters $\theta_{1:k}$, which reflect the extend to which observed points can influence the interpolation in its neighborhood. A large length scale reflects a greater sphere of influence, and vice versa. To learn these parameters, we choose the parameters $\theta_{0:k}$ that maximize the log marginal likelihood,

$$\begin{aligned} &\log p(f_{1:n}|\mathbf{x}_{1:n}, \theta_{0:k}) \propto \\ &-(f_{1:n} - m_{1:n})^\top k_\alpha^{-1}(f_{1:n} - m_{1:n}) - \log\det k_\alpha, \end{aligned} \tag{11}$$

which can be performed via any gradient ascent algorithm. The term involving $\log\det k_\alpha$ can be interpreted as a regularization parameter that encourages longer length scales (i.e. a low variance model) as long as such simplicity does not greatly hinder the model's ability to fit the data, which is captured by the quadratic form $f^\top k_\alpha^{-1} f$. Since $k_\alpha$ is also a function of the noise parameter $\sigma$, it is also possible to learn the value of $\sigma$ via gradient ascent as well. However, introducing $\sigma$ as a learnable paramater often produces many local optima that reflects a different interpretation of the data; a large $\sigma$ is often accompanied by the learning of larger length scales, reflecting a low-variance high-bias interpolation, whereas a small $\sigma$ is accompanied by small length scales, reflecting a high-variance low-bias interpolation. Within the context of hyperparameter selection, the noise parameter is chosen beforehand and noted in the Experimental section.

### 2.2. Sequential Hyperparameter Selection

The use of a Gaussian process provides the Bayesian approach to modeling the uncertainty of the system, namely the confidence of our interpolated performance function. In an online function optimization framework, however, a sequential series of hyperparamters must be chosen, with the hope that each choice in the series allows us to make a more informed decision about the identity of the hyperparameters. To do so, it is important for the online system to properly manage exploration and exploitation. It is possible to formulate this task as a Belief-State Markov Decision Process (MDPs) where the belief-state is our belief about the latent performance function as capture by the Guassian process, the action-space is the set of all possible hyperparameters to choose from, and the observation-space is the set of all possible performance function evaluations at a particular hyperparameter. However, the continuous

nature of the action and observation space makes most online policies intractable without discretization. It is thus common within the hyperparameter optimization literature to adopt the use of an acquisition function—a deterministic function of the posterior mean and covariance from the Gaussian process that allows for a simple albeit ad-hoc trade-off between exploration and exploitation.

As is the case with kernel selection, many acquisition functions exist. In our experiments, we use the upper-confidence bound (UCB) for hyperparameter selection, defined as,

$$m_D(\mathbf{x}) + 2\sqrt{k_D(\mathbf{x}, \mathbf{x})}. \tag{12}$$

The key benefit of such an acquisition function is that it is easily differentiable with respect to $\mathbf{x}$, enabling the use of gradient ascent to identify where the acquisition function is maximized without the need of evaluating the Gaussian process on a pre-defined grid of hyperparameters.

## 3. Model Selection

While hyperparameter optimization has been extensively studied, there has been few research done on the hyperparameter optimization in a multi-model setting. Research in model selection has largely been focus on variable selection and hyperparameter tuning constrained within a single family of model, such as least-squares regression. As the number of such approaches increases, it is natural to consider the possibility of extending the task of Bayesian optimization to include not only hyperparameter selection, but also model family selection. Here, we describe our formulation of the problem as an multi-armed Gaussian bandit.

### 3.1. Multi-Armed Gaussian Bandit

A multi-armed Gaussian bandit problem addresses the question of optimal decision making where one sequentially selects from one of $N$ gaussian distributions with unknown means and variances. In the bandit problem, the reward of each action is the observed realization from the chosen distribution. The multi-armed Gaussian bandit formulation of the model selection problem can be stated as follows.

Each sequential decision involves selecting both a model type (i.e. logistic regression, radius nearest-neighbors, etc.) and a set hyperparameters to initialize the model. The difficulty of performing model selection stems from the fact that each model type has an associated Gaussian process conditioned on the observations made thus far. However, note that we have predetermined our hyperparameter selection policy to be that of maximizing the UCB acquisition function. By doing so, we reduce our decision of which model down to that of selecting one of $N$ models at particular hyperparameter initializations. By looking at each model at a particular hyperparameter, we reduce the Gaussian process to a single Gaussian distribution for each model, thus avoiding the need to make a decision based on $N$ Gaussian processes.

### 3.2. Sequential Model Selection

As with sequential hyperparameter selection, there are many methods with which sequential model selection can be performed. The reduction of the problem to that of a multi-armed Gaussian bandit further opens up room for the use of various online methods available for Partially Observable Markov Decision Processes (POMDPs). Of particular interest to us is to build a simple system that seeks a dynamic trade-off between exploration and exploitation dependent on the amount of time a user is willing to dedicate to automated model selection. Intuitively, the optimal policy ought to take into account a time limit imposed on the automated selection process; as the time limit gets smaller, the selection algorithm should become increasingly more conservative about querying computationally costly models. To reflect this, we define our reward function as follows,

$$R(a_t) = \begin{cases} o_t^{(f)} & o_t^{(t)} > 0 \\ 0 & o_t^{(t)} \leq 0 \end{cases}, \tag{13}$$

where $a_t$ is the model chosen at time $t$, yielding a 2-tuple observation $o_t = (o_t^{(f)}, o_t^{(t)})$ where $o_t^{(f)}$ is the observed performance of the model at the chosen hyperparameters $\mathbf{x}$, and $o_t^{(t)}$ is the amount of time left on the clock after the $t$ steps.

#### 3.2.1. Lookahead with Approximate Value Function

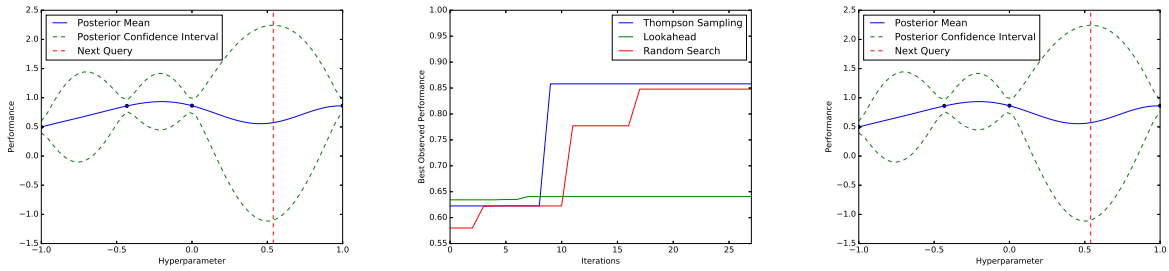
In the multi-armed Gaussian Bandit problem, the continuous obseravtion space still presents a challenge for optimal policy construction. One simple approach for tackling such a POMDP is to a one-step lookahead policy. Given a belief about the $N$ Gaussian distributions, we choose the action that maximizes the following expression,

$$R(b_t, a_t) + \gamma \frac{1}{n} \sum_{i=1}^{n} U(b_{t+1}^{(i)}), \tag{14}$$

where $b_{t+1}^{(i)}$ is the updated belief after simulating a virtual observation $o_t$ from the current belief-action pair. $U$ is the approximate value function which is approximated via a rollout evluation strategy described in (**?**).

#### 3.2.2. Thompson Sampling

Given current belief $b_t$ about the multi-armed Gaussian bandit, Thompson sampling samples a observation of each




(a) Squared exponential spectrum without parameter training.

(b) Spectral distribution after gradient ascent on the regularized likelihood.

(c) The true spectral of the underlying squared exponential kernel.

*Figure 1.* A representative Bayesian optimization task is shown in 1(a) for a single-model, single-hyperparameter setting. Performance values were from the L1-logistic regression tested on a binary dataset that was not linearly separable. The UCB acquisition function chooses the next hyerparameter query location based on the location with the largest upper-confidence bound. Following the experimenta procedure in §4.1,

of the $N$ Gaussian distribution based on our current belief and simply performs the action that maximizes the reward. This simple approach is effective despite its simplicity and has often outperforms competing techniques. However, it is known that Thompson sampling is still a myopic selection strategy and is prone to overexploration.

## 4. Experiments

### 4.1. Experimental Details

The evaluate the performance of our online multi-model selection algorithm, we evaluated three models, L1-logistic regression L2-logistic regression, and Radius Neighbors classifier on a synthetic binary classification dataset containing 10000 samples. The samples were generated from a mixture of four Gaussian distributions and labeled according to the XOR labeling scheme. As such, the data set is not linearly separable. Each model contained a single hyperparameter, namely the penalty hyperparameter for the logistic regressions and the radius for the Radius Neighbors classifier.

We ran three different three different model selection algorithms. We used pure random search as our baseline. Despite its simplicity, random search is often used for hyperparameter selection and serves as a minimal baseline for our multi-model selection task. The random search algorithm randoml selects a model and a single set of hyperparameter at each iteration. We then used two flavors of our proposed multi-model selection algorithm. The first implements sequential model selection using lookahead with approximate value function, while the second uses Thompson sampling. Both versions iteratively selects a single model type (according to their respective methods) as well as a single hyperparameter using the UCB acquisition function.

All three selection strategies were tested with a fixed time limit of 60 seconds for model training and testing. The imposition of the time limit for this example problem simulates the situation in reality when model selection and hyperparameter selection are costly operations that make take many hours to run. Under these circumstances, the balance of exploitation and exploration ought to change depending on the amount of time allowed for the optimization task as well as the differential computation cost of each family of models.

### 4.2. Results

## 5. Discussion

Interesting stuff regarding how to leverage information across datasets. Limitations regarding acquisition function optimization. (Snoek et al., 2015) (Snoek et al., 2012) (Rasmussen & Wiliams, 2006) (Rasmussen, 2006) (Duvenaud, 2014) (Garnett, 2015) (Wang et al., 2005) (Kanter & Veeramachaneni, 2015) (Kochenderfer, 2015)

Because action and state space are both continuous.

## References

Duvenaud, David. Automatic model construction with gaussian processes. University of Cambridge, 2014.

Garnett, Roman. Bayesian optimization. In *CSE 515T Lecture Notes*. 2015. URL www.cse.wustl.edu/~garnett/cse515t/files/lecture_notes/12.pdf.

Kanter, James and Veeramachaneni, Kalyan. Deep feature synthesis: Towards automating data science endeavors. In *IEEE*. 2015.

Kochenderfer, Mykel. State uncertainty. In *Decision Making under Uncertainty*, pp. 150. The MIT Press, 2015.

Rasmussen, C and Wiliams, C. Gaussian processes for machine learning. In *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Rasmussen, Carl. Gaussian processes in machine learning. 2006. URL http://www.cs.ubc.ca/~hutter/earg/papers05/rasmussen_gps_in_ml.pdf.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In *NIPS*. 2012.

Snoek, Jasper, Rippel, Oren, Swersky, Kevin, Kiros, Ryan, Satish, Nadathur, Sundaram, Narayanan, Patwary, Mostofa, Prabhat, and Adams, Ryan. Scalable bayesian optimization using deep neural networks. In *JMLR*. 2015.

Wang, Tao, Lizotte, Daniel, Bowling, Michael, and Shuurmans, Dale. Bayesian sparse sampling for on-line reward optimization. In *ICML*. 2005.