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

Describe the formulation in a single-model setting, where the objective is simply to find the best hyper using Gaussian process. Formalize this with math about test set performance.

2.1. Gaussian Processes

The task of hyperparameter selection has often been formulated as a function optimization procedure performed that involves 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,

$$m(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x})\right] \tag{1}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\right], \tag{2}$$

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,

$$\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_1 \\ \vdots \\ m_n \end{bmatrix}, \begin{bmatrix} k_{11} & \dots & k_{1n} \\ \vdots & \ddots & \vdots \\ k_{n1} & \dots & k_{nn} \end{bmatrix} \right), \tag{3}$$

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 simplicity. For further simplicity, we write,

$$f_{1:n} \sim \mathcal{N}\left(m_{1:n}, k_{(1:n)\times(1:n)}\right). \tag{4}$$

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,

$$\begin{bmatrix} f_{1:n-1} \\ f_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_{1:n-1} \\ m_n \end{bmatrix}, \begin{bmatrix} k_\alpha & k_\beta \\ k_\beta^\top & k_\gamma \end{bmatrix} \right), \tag{5}$$
$$f_n \mid f_{1:n-1} \sim \mathcal{N}(m_{\mathcal{D}}(x_n), k_{\mathcal{D}}(x_n, x_n)), \tag{6}$$
$$m_{\mathcal{D}} = m_n + k_\beta^\top k_\alpha^{-1}(f_{1:n-1} - m_{1:n-1}), \tag{7}$$
$$k_{\mathcal{D}} = k_\gamma - k_\beta^\top k_\alpha^{-1} k_\beta, \tag{8}$$

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

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

realizations of the random variables $f_{1:n-1}$ are observed with some σ-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,

$$k(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp\left(-\frac{1}{2}\gamma\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{9}$$

where $\gamma\left(\mathbf{x}, \mathbf{y}\right) = \sum_{i=1}^{k} \frac{\left(x_i - x_i'\right)^2}{\theta_i^2}$. 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,

$$\begin{aligned} &\log p\left(f_{1:n} | \mathbf{x}_{1:n}, \theta_{0:k}\right) \propto \\ &-\left(f_{1:n} - m_{1:n}\right)^\top k_\alpha^{-1} (f_{1:n} - m_{1:n}) - \log\det k_\alpha, \end{aligned} \tag{10}$$

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_α is also a function of the noise parameter σ, it is also possible to learn the value of σ via gradient ascent as well. However, introducing σ as a learnable paramater often produces many local optima that reflects a different interpretation of the data; a large σ is often accompanied by the learning of larger length scales, reflecting a low-variance high-bias interpolation, whereas a small σ 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 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 for hyperparameter selection, defined as,

$$m_D(\mathbf{x}) + 2\sqrt{k_D(\mathbf{x}, \mathbf{x})}. \tag{11}$$

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

3.2. Sequential Model Selection

4. Experiments

4.1. Experimental Details

We test our approach v. random sampling in a limit time duration.

4.2. Results

5. Discussion

Interesting stuff regarding how to leverage information across datasets. Limitations regarding acquisition function optimization.

Because action and state space are both continuous.