Reinforcement Learning Hyperparameter tuning using Bayesian Optimization

Hari Guhan Thillairajan
hariguha@buffalo.edu

Shathesh Kumar
shathesh@buffalo.edu

Vinay Kempanahalli Nagaraju
vinaykem@buffalo.edu

1. Introduction

Reinforcement learning has gained wide traction in recent times in the areas of game theory, robotics, multi-agent systems and many more. The learning involves training an agent in an environment and maximising the reward the environment has to offer. Easy availability of computing power has led to the exploration of bigger environments, however the time and resources needed grow exponentially with the problem. Algorithms such as Q-Learning and neural network based architectures such as Deep Q networks are being widely used to improve the agent's learning and time taken for convergence. These algorithms internally use hyperparameters which directly influence the way agent's learn. It is vital to tune the hyperparameters to obtain the best rewards.

Existing hyperparameter tuning methods such as grid search and random are exhaustive and make little or no use of prior information. However, bayesian optimization offers powerful methods to specify prior information and make informed future decisions. The core objective of our project is to replace the naive grid search algorithm for hyperparameter tuning with a much more efficient bayesian optimization algorithm.

Keywords : *Reinforcement Learning, Q-Learning, Deep Q-Learning, Hyperparameters Optimization, Bayesian Optimization, Acquisition Functions.*

2. Reinforcement Learning

The environments explored in this project are Taxi and Lunar Lander. RL agents use algorithms which drive them to their best reward. Throughout the report, we will looking into following two algorithms which are well suited for discrete action spaces:

- **Q-Learning -** The agent uses a Q-table to store the environment states. For every state, the table will have all the actions the agent can possibly take at that state and the long term rewards it will get by performing the action. Initially this table is populated with zeros and as the agent interacts with the environment, the table is updated using the formula.

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha\,(r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

- **Deep Q learning** - This uses a deep neural network which is trained during episodic learning. The trained model is used to predict the next best action based on the current state.

The following hyperparameters are chosen for tuning using bayesian optimisation:

- **Learning Rate -** The learning rate in the above equation influences the importance given to the current and future rewards. In the context of Deep Q networks, it is used to

influence the gradient computation to find the best parameters which predict the next best actions for the agent.

- **Discount Factor (Gamma)** - The gamma parameter of the Bellman equation for Q-learning influences the Q values' updates for every state-action transition. A value close to 0 makes the agent only consider current rewards and a value towards 1 will make the agent learn long-term rewards.

3. Support Vector Machines

SVM is a supervised learning method that can work with non-linear and multi-dimensional data. The method's nonlinearity with multiple maxima to optimize over makes it ideal to explain how acquisition functions query the next point, and also how we can control the exploration/exploitation of the algorithms. We tune the SVM hyperparameter C (regularization parameter), optimizing to get the max on test_accuracy. Sklearns's example breast cancer classification dataset is used for the task.

We next talk about bayesian optimisation and further present evaluations on tuning the above parameters using bayesian optimization.

4. Bayesian Optimization (BO)

Bayesian optimization is a powerful strategy for finding the extrema of objective functions that are expensive to evaluate. It allows for an elegant means by which informative priors can describe attributes of the objective function, such as smoothness or the most likely locations of the maximum, even when the function itself is not known. The search for the locations is guided by acquisition functions. Following describes the common steps:

4.1 Algorithm:

for t in 1, 2, 3 …

Find x_t by optimizing the acquisition function over the GP: $x_t = argmax_x\ u(x|D_{1:t-1})$

Sample the objective function: y_t = $f(x_t) + \varepsilon$

Augment the data $D_{1:t} = \{D_{1:t-1}, (x_t, y_t)\}$

4.2 Acquisition Functions

Acquisition functions are defined such that a high acquisition corresponds to a potentially high value of the objective function whether because the predictive mean is high or variance is great or both. Maximizing the acquisition function is used to select the next point at which to evaluate the function. That is, we wish to sample f at $argmax_x$ u(x|D), where u(·) is the generic symbol for an acquisition function. Following acquisition functions are used throughout the report: Thompson Sampling, probability of improvement and expected improvement.

5. Observations

The 3 acquisition functions mentioned above are applied to tuning RL agent hyperparameters for Taxi, Lunar Lander environments and support vector machines(SVM).

Each acquisition function is discussed separately explaining their advantages and disadvantages on hyperparameter tuning. We conclude by comparing the results between the various functions.

5.1 Thomson Sampling

- Initially, randomly sample the first data point from the state space, observe its output and fit a GP model over it.
- From the updated GP model, randomly sample a function and choose the maxima of this sample as the next data point.
- Repeat the previous step until the GP model has zero variance all over or until a pre-set number of iterations

The intuition behind this acquisition function is that when we randomly sample from the gaussian posterior, the upper bound is either maximum mean or maximum variance which are potential areas for the global maxima. Also at every iteration, our model will learn better and once enough data is seen, the model will have a complete picture of the function and hence the global maxima as well.

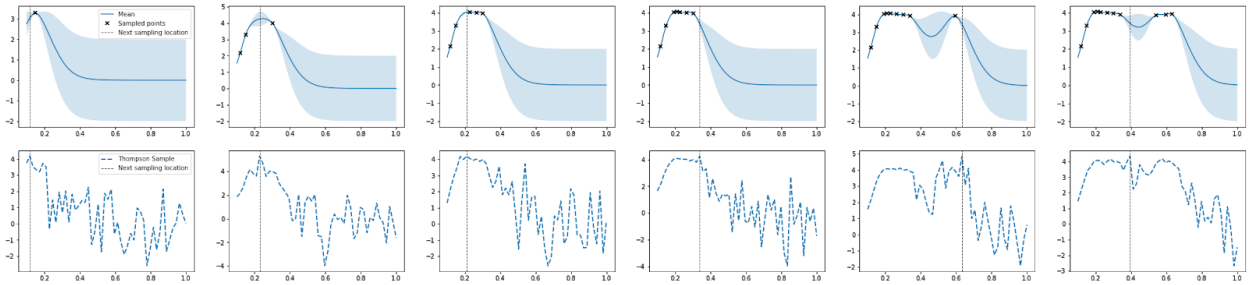

Figure 1: Optimize C of SVM. The model converged to C = -0.21 where accuracy = 0.81

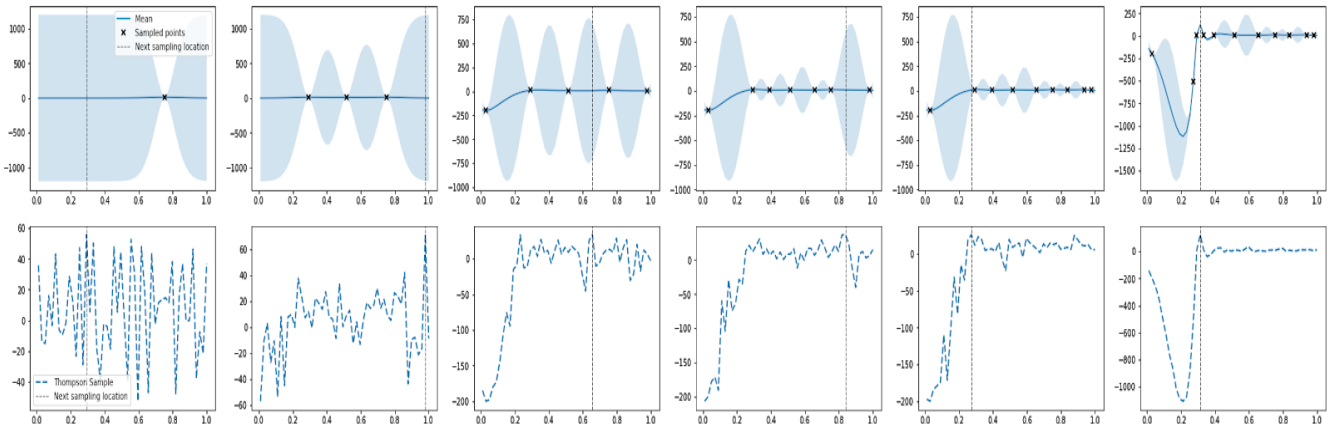

Figure 2:Optimize gamma in Q-Learning Algorithm on Taxi environment. The model converged to gamma 0.29 where reward is 12

We could see in Figure 1 and Figure 2 that the acquisition function is similar to the GP posterior. Since here, the acquisition function is only a random sample from the GP posterior,

the only way we can control the exploration exploitation trade off is by varying the variance in the kernel.

5.2 Probability of Improvement

It is an improvement-based acquisition function. It works by maximizing the probability of improvement over the existing $f(x^+)$ where $x^+ = argmax_{x_i \in x_{1:t}} f(x_i)$ so that,

$$PI(x) = P(f(x) \geq f(x^+) + \xi)$$
$$= \Phi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right)$$

Where $\Phi(.)$ is the normal cumulative distribution function. The trade-off parameter $\xi \geq 0$ is added to influence the function to select points that offer larger gains but less uncertainty. over points that have a high probability of being infinitesimally greater than $f(x^+)$ over.

Below are some of the results we have shown which helped us understand the acquisition function better. First, we will start with a simple multiple optima function.

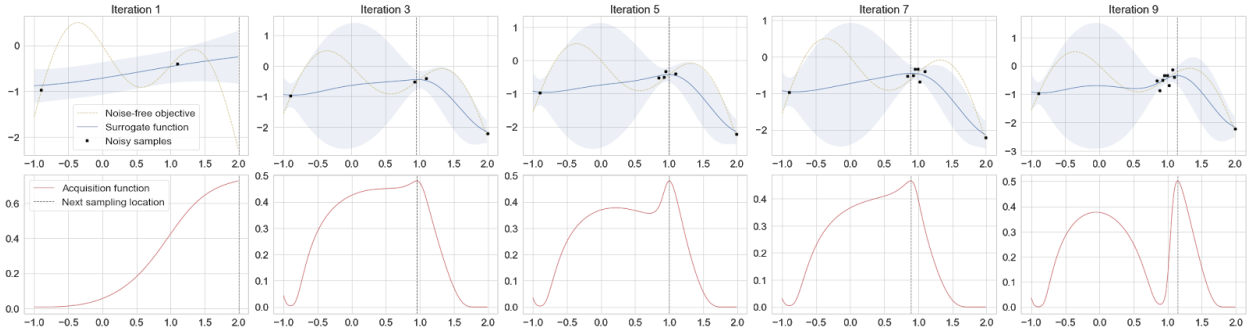

Figure 3: Simple function with multiple optima, ξ=0.01. The model converged to x = 1.0 where y = 0.4 (CSE610 class example)

The function is only looking at the maximum mean. We will have have the ξ parameter to make the function select higher variances, Once we increase the ξ, we need new mean to be greater than $\mu_{opt} + \xi$ for positive acquisition. Else, the value will be negative and since variance is in denominator higher variance will give higher acquisition. Figure 4 observes the high ξ parameter.

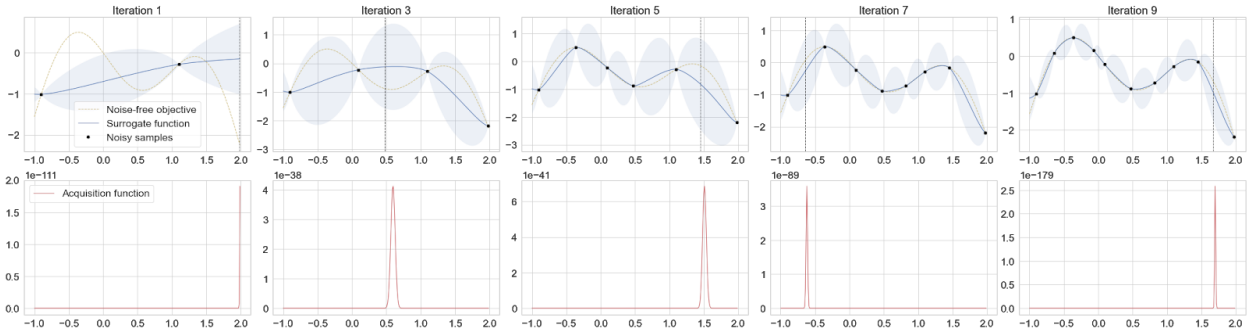

Figure 4: Simple function with multiple optima, ξ=10. The model converged to x = 0.36 where y = 0.49 (CSE610 class example)

We will go on to show how the acquisition function expected improvement does a much better job on the multiple optima function without controlling ξ in the later part of the report. But probability of improvement works better when we know the optimum value.

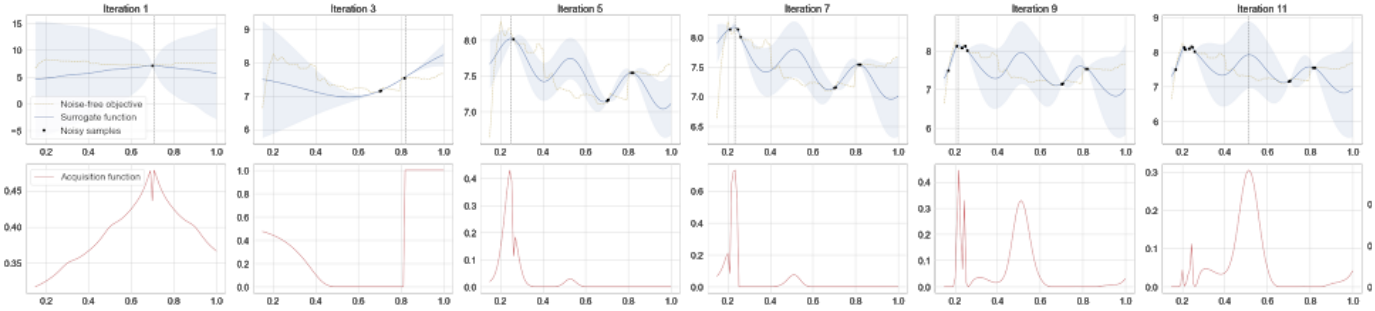

Figure 5: Tuning gamma in SVM, ξ = 0.01. The model converged to gamma - 0.21 where accuracy = 0.81

Figure 5 presents observations for ξ = 0.01. The model is trying to learn as much near the regions where it has seen the maximum mean and only afterwards, it's moving to the regions with high variances. This is key in converging to optima in this function.

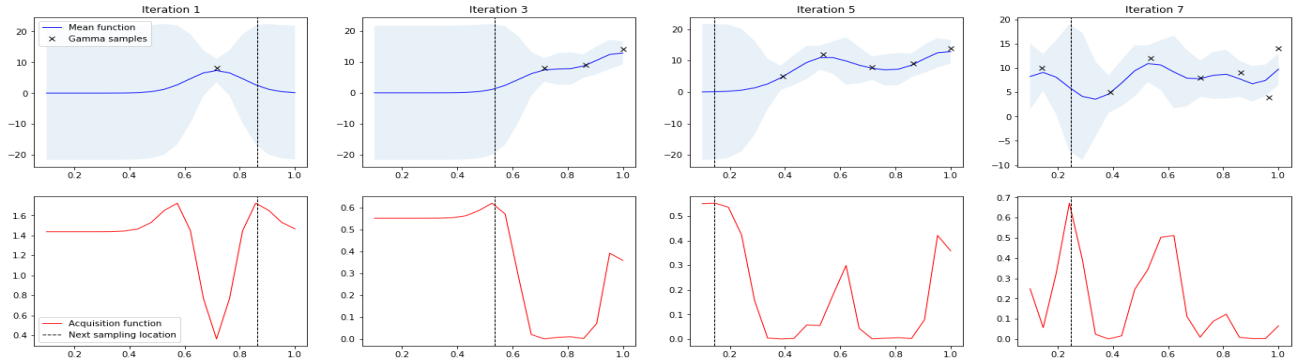

Figure 6: Tuning gamma in Q-Learning - Taxi environment. The model converged to gamma = 1 where reward =14

In figure 6, the rewards are in a non-linear relationship with gamma peaks at gamma=1. Since we set the initial mean to 0, the first point chosen is higher than any point in the mean vector. That is why variance is accounted for and query points are sampled from where variance is higher.

We can see a similar trend of the 1D example for 2D example in Figure 7. The model is looking into regions with higher mean. But here the state space is much bigger than 1 dimensional and hence the convergence is not as good as expected.

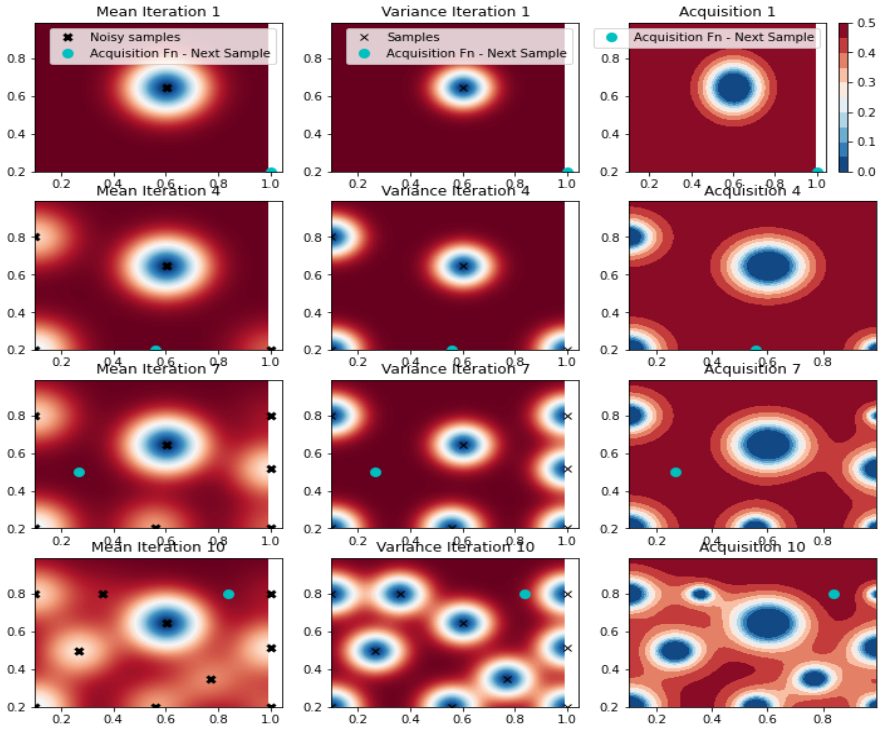

Figure 7: Tuning learning rate and gamma in DQN - Lunar Lander Environment - ξ=0.01. The model converged to learning rate = 0.39, gamma = 0.8 where reward = 5

5.3 Expected Improvement

Probability of improvement acquisition function only looked at how likely there is an improvement over the domain space, but did not consider how much we can improve. The key idea here is we sample the next query point as the one with the highest expected improvement. The acquisition function also has a ξ which we can tune to trade off exploration and exploitation.

Without tuning the ξ, we can see the model has explored much better than probability of improvement in figure 8.

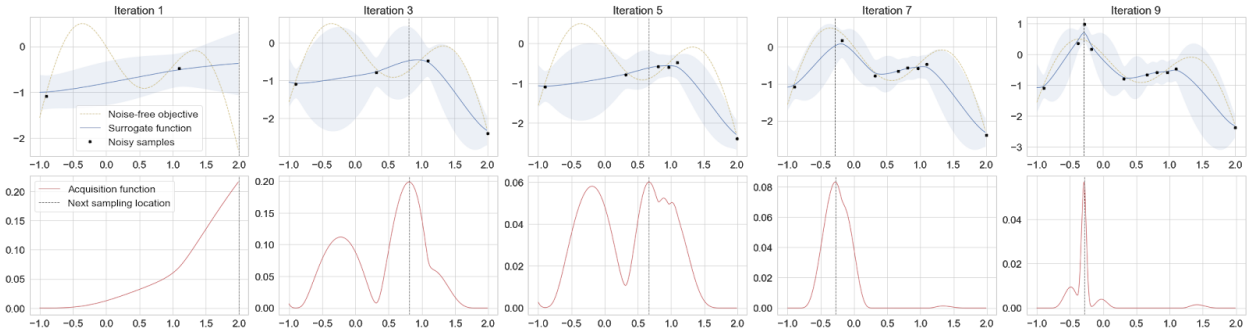

Figure 8: Expected Improvement - Comparison with probability of improvement (Figure 3)

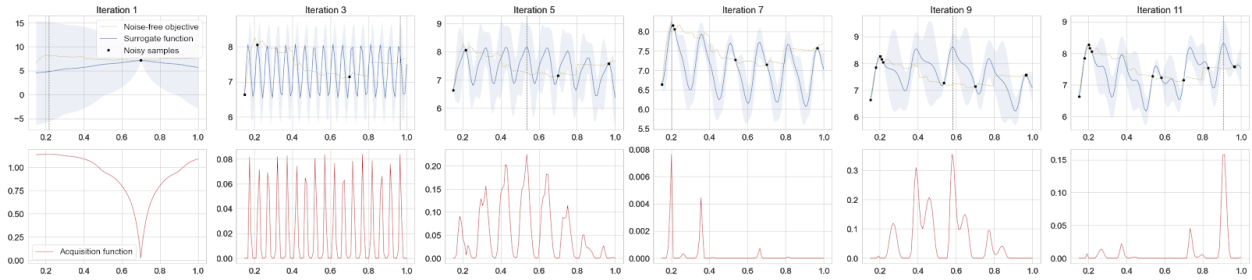

Figure 9: SVM for C, ξ = 1, Matern + sinusoidal kernel(0.2 periodicity). The model converged to c = 0.2 where accuracy =0.82

In figure 9, we have used a combination of periodic and the matern kernel. We wanted to make the kernel accountable to guide the search for optima with spikes in graphs and also the model converged to a global maximum quickly.

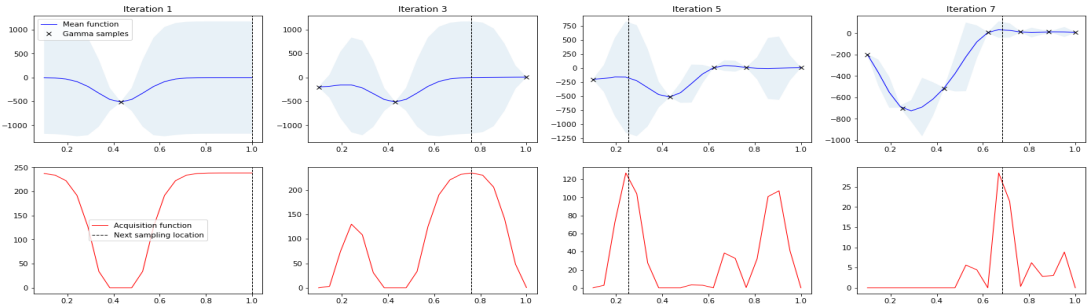

Figure10: Optimizing gamma in Q-Learning, ξ = 0.01, RBF kernel - 600 variance, 0-mean function. The model converged to gamma = 0.89 where reward = 11

In figure 10, we can see that until iteration 7, the model will explore and once the GP is fit with enough data, the variance drops and the model begins to exploit. As a result, we can see that the convergence is much faster than Probability of Improvement, without controlling the ξ factor.

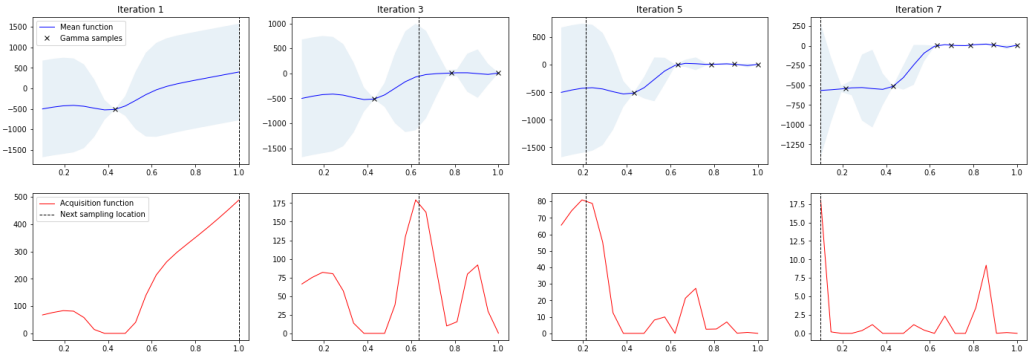

Figure 11: Optimizing gamma in Q-Learning, ξ = 0.01, RBF kernel - 600 variance, linear mean function. The model converged to gamma = 0.89 where reward = 11

For very low gamma, the reward is also low, as we limit the agent from looking more into the future. To overcome this behaviour, we introduce a linear mean function. As a result, we can clearly see in figure 11, the model is searching for maximum in the regions of higher gamma and not much in the regions of lower gamma.

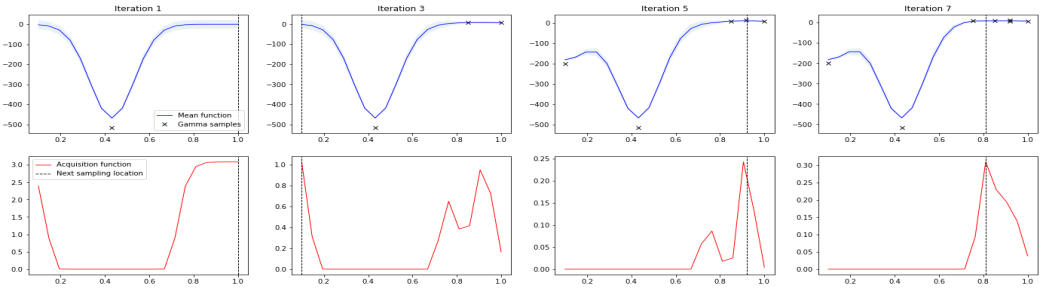

Figure 12: Optimizing gamma in Q-Learning, ξ = 0.01, RBF kernel - 10 variance. The model converged to gamma = 0.81 where reward = 13

One more factor in the exploration/exploitation trade off is the variance we set initially. If the kernel does not allow for much variance, irrespective of the ξ choice, the model will always exploit. This is apparent in figure 12. This is because of lack of variance. This is also exactly how the acquisition function exploits more once GP is fit with many samples. We have shown expected improvement trades off exploitation and exploration much better than probability of improvement.

5.4 Rational Quadratic Kernel ($\alpha, l > 0$)

$$k_{RQ}(r) = \sigma^2 \left(1 + \frac{r^2}{2\alpha l}\right)^{-\alpha}$$

The parametrization for this kernel differs from the RBF kernel as we have an alpha/power/exponent parameter to vary the length scale between each inputs and a special case when $\alpha \to \infty$ This expression will be similar to the squared exponentiation kernel function, hence the kernel is similar to adding squared exponential kernels with different length scales. Power parameter indicates the weighting of scale variations among inputs. As seen from the figure 13, where the $\alpha = 2$ & we observe the plots that variance has reduced in few iterations for the similarity expressed by the kernel.

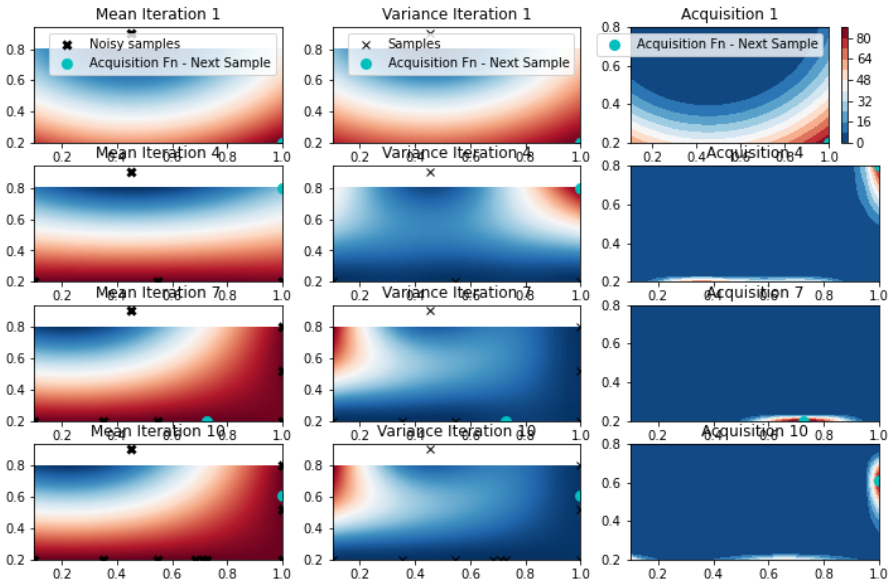

Figure 13: Tuning gamma and learning rate of Q-Learning, ξ = 0.01, RatQuad kernel - power 2. The model converged to learn rate = 0.54 and gamma =0.2 where reward =14

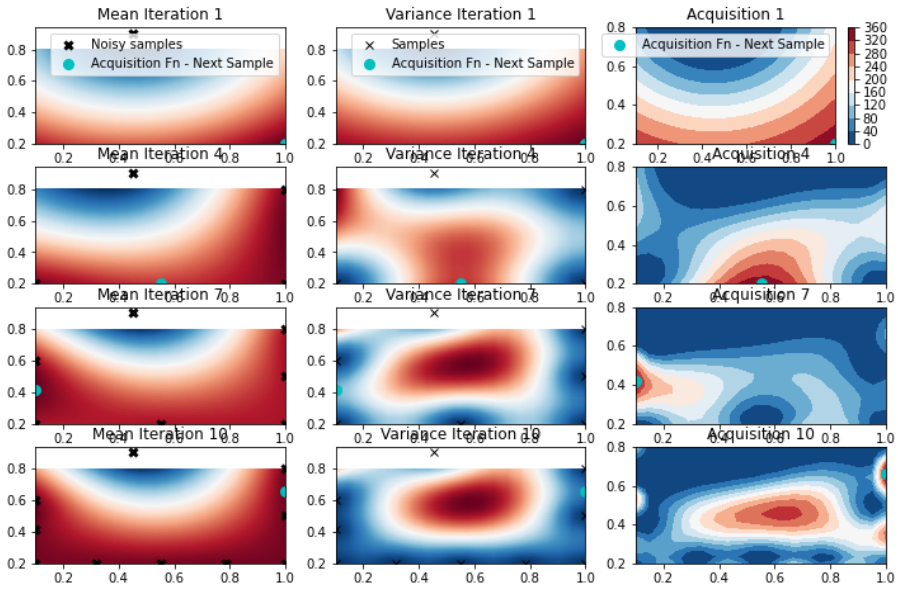

Figure 14: Combination of kernels - RatQuad + Matern52 + RBF. The model converged to Learning rate = 0.68, gamma = 0.35, reward = 12

The combination of kernels observed from figure 14, where the revised GP parameters are not changed much from augmenting queried data. As a result, in every iteration the variance shrinks much less compared to the previous kernels. The impact of kernels is that if the kernel approximates the covariances between points with less number of samples, we will have a good approximation of the function itself and hence in turn converge towards the global maxima.

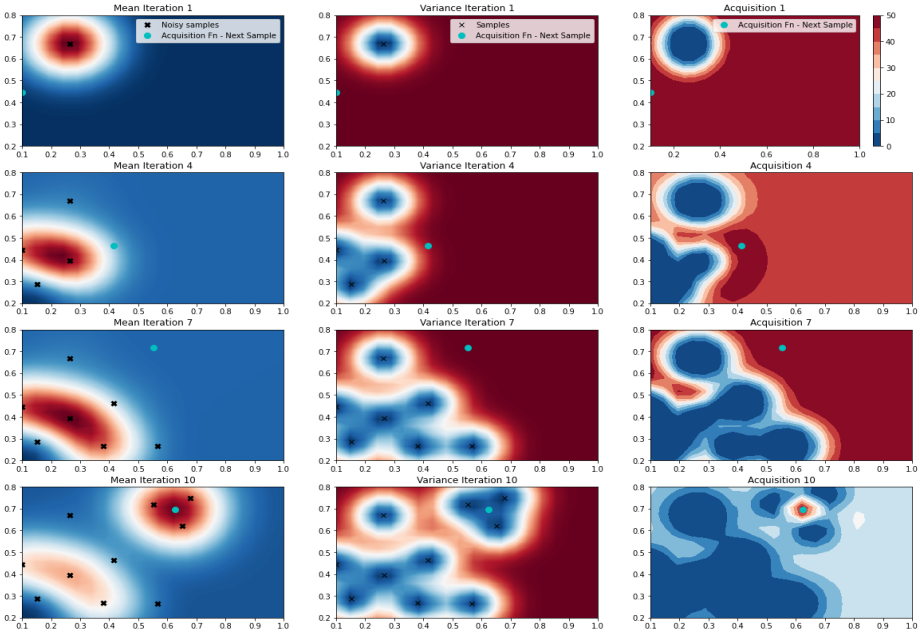

Figure 15: DQN, RBF kernel variance 100, length scale 0.01. The model converged to learning rate = 0.7 and gamma = 0.65 where rewards = 47.68

In figure 15, we can observe the difference from the previous combination of kernels, where the GP could converge to an optima. After exploring with enough samples, it begins to exploit from the acquisition function.

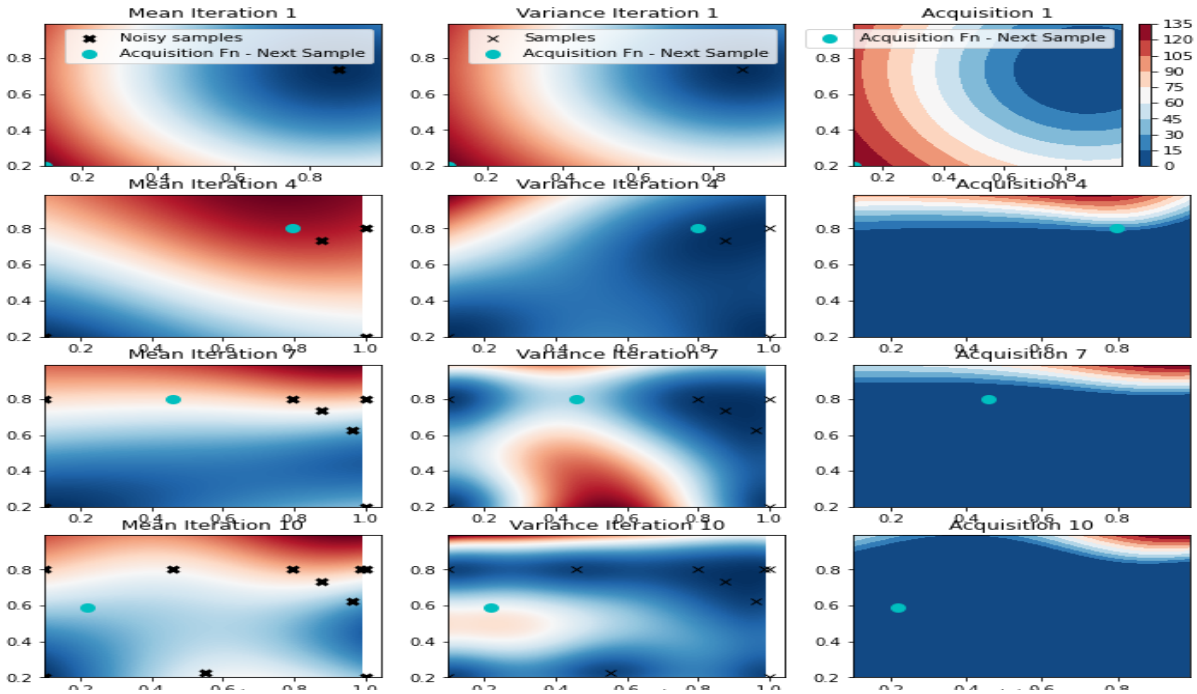

Figure 16: Matern52. The model converged to learning rate = 0.78, gamma = 1 where reward = 83.7

In figure 16, for the Matern32 kernel, we can see that the GP variance is very low in the 10th iteration, which implies the GP has a perfect fit of the function. As a result, the model shows better reward than the RBF kernel used in figure 15 and also converges faster.

6. Results

The results tabulated from figures above. RL agent f_{opt} rewards are calculated from a single episodic run of an agent, convergence within 200 steps for the Q learning problem with the taxi Environment. SVM f_{opt} on test accuracy.

6.1 CSE610-class example

Acquisition Function	ξ	f_{opt}	x_{opt}
Thompson Sampling	N/A	0.49	-0.32
Probability of Improvement	0.01	0.4	1
Probability of Improvement	10	0.49	-0.32
Expected Improvement	0.01	0.49	-0.32

6.2 SVM

Acquisition Function	ξ	f_{opt} : Accuracy	x_{opt} : C
Thompson Sampling	NA	0.81	-0.21
Probability of Improvement	0.01	0.81	0.21

Expected Improvement (Mater52)	0.01	0.76	0.99
Expected Improvement (M52, Sin)	1	0.82	0.2

6.3 Q - Learning

Acquisition Function	ξ	f_{opt} : Reward	x_{opt} : γ Gamma
Thompson Sampling	NA	12	0.29
Probability of Improvement	0.01	14	1
Expected Improvement (RBF kernel: σ-600, $\mu = 0$)	0.01	11	0.89
Expected Improvement (RBF kernel: σ-600, $\mu \neq 0$)	0.01	11	0.89

6.4 Deep Q Networks:

Acquisition Function	ξ	f_{opt} : Reward	$(x_{1opt}, x_{2opt}) : (\gamma, lr)$ Gamma, learning rate
Probability of Improvement	0.01	5	(0.8, 0.39)
Expected Improvement $K_{RQ,}\ \alpha = 2$	0.01	14	(0.2, 0.54)
Expected Improvement $K_{RQ,}\alpha = 2$, Matern52, RBF	0.01	12	(0.35, 0.68)
Expected Improvement (RBF kernel: σ-600, $l \neq 0.01$)	0.01	47.68	(0.65, 0.7)
Expected Improvement (Matern52)	0.01	85	(1,0.78)

7. Conclusion

Bayesian Optimization, with GP models, helps incorporate the prior of sampled parameters from the parameter space. We have observed various ways to control the algorithm for faster convergence to global optima in different acquisition functions, changing GP and acquisition function parameters with a combination of kernels. These influence the model to sample by either exploring/exploiting which is essential in learning large parameter spaces. Among the functions, expected improvement offers the best results. For future work, the acquisition functions can be explored for larger environments.

References:

- **Gaussian Processes for Machine Learning**
  Carl Edward Rasmussen & Christopher K. I. Williams
  MIT Press, 2006. ISBN-13 978-0-262-18253-9.

- A Tutorial on Bayesian Optimization
  Eric Brochu, Vlad M. Cora and Nando de Freitas
  https://arxiv.org/pdf/1012.2599.pdf

- Exploring Bayesian Optimization
  Apoorv Agnihotri, Nipun Batra
  https://doi.org/10.23915/distill.00026

- CSE610: NPML Advanced Notebooks
  Dr. Varun Chandola
  https://github.com/ubdsgroup/mladvanced-notebooks

- Libraries for Project:
  - GPy: https://gpy.readthedocs.io/en/deploy/
  - SciPy: https://docs.scipy.org/doc/scipy/reference/
  - OpenAI Gym: https://gym.openai.com/