CS 391L HW5: Gaussian Process

Mit Shah (UT EID: mks3226)
University of Texas at Austin
Austin, TX 78712
mkshah@utexas.edu

Abstract

This assignment is aimed at learning modeling Gaussian Processes to the given data. Gaussian Process is a particular kind of statistical model every point in some continuous input space is associated with a normally distributed random variable. Also, every finite linear combination of those random variables has a multivariate normal distribution. The distribution of a Gaussian process is the joint distribution of all those random variables and so it is a distribution over functions with a continuous domain. Here we will learn on, given data and a selected Covariance matrix, how to optimize the hyper-parameters to fit the Gaussian Process to data. We will also implement the algorithm and check it working for three different configurations of the given data set.

1. Introduction

Classical supervised machine learning algorithms generally tries to fit a parameterized function to a set of training data in order to minimize an error function. Then it is used to generalize previously unseen data. Main differences between these methods have been the set of model functions that the algorithm can use to represent the data. But if we do not know anything about the underlying process that generated the data then choosing an appropriate model is often a trial-and-error process.

One of the many solutions to this problem is to let the optimization process search over different models as well as the parameters of the model. To do this, we need to generalize the idea of a probability distribution to something that we can optimize over. This is called a stochastic process.

In general, dealing with stochastic processes is very difficult because combining the random variables is generally hard. However, if we restrict the process in such a way that all of the random variables have a Gaussian distribution, and the joint distribution over any (finite) subset of the variables is also Gaussian, then this Gaussian process (GP) is much easier to deal with.

Modeling with a Gaussian process includes a probability distribution over the space of functions and sample from that. One sample would consist of a specification of this vector. However, because everything is Gaussian, just as we specify a Gaussian distribution with the mean and covariance matrix, we can specify a Gaussian process by the mean function and a covariance function.

As a machine-learning algorithm, it uses a lazy learning and a measure of the similarity between points to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information, it is a one-dimensional Gaussian distribution.

Gaussian processes can be seen as an infinite-dimensional generalization of multivariate normal distributions. Gaussian processes are useful in statistical modeling and many such other applications.

2. Method

Gaussian Process is specified by the mean and covariance functions. Generally we subtract off the mean first, so that the mean function is identically zero. GP is completely described as a function G(k(x, x0)) that can model some underlying function, where covariance function gives us the expected covariance matrix between the values of f at x and x0. The random variables that define the GP are used to provide an estimate of f(x) for each input x. Let's first look at the kernel function we are going to use and its hyper parameters. Then we will focus on how to learn the hyper parameters.

2.1. Squared Exponential Covariance Matrix

Throughout the assignment we will be following the Squared Exponential Covariance Matrix which is as follows:

$$k(x, x') = {\sigma_f}^2 \exp \frac{-1}{2l^2} |x - x'|^2 \quad (1)$$

There are two parameters in this covariance function: σ_f and l.

But all processes are not noise free. One of the ways to add noise into it is to assume that it is independent, identically distributed Gaussian noise and so we can model it by including an extra parameter into the covariance matrix, so that instead of using K we use $K + {\sigma_n}^2$ I, where I is the N N identity matrix. Noise is only added to the covariance for the training data.

All the parameters of the kernel including noise one - $\sigma_f, \sigma_l, \sigma_n$ are known as hyper parameters.

2.2. Learning Hyper Parameters

The squared exponential covariance matrix has three hyper parameters ($\sigma_f, \sigma_n, \sigma_l$), which has a significant effect on the shape of the resulting output curve. So it is necessary that we optimize them.

The ideal solution to this problem is to set up prior distribution over the hyper parameters and then integrate them out in order to maximize the probability of the output targets.

$$P(t*|x,t,x*) = \int P(t*|x,t,x*,\theta)P(\theta|x,t)d\theta \quad (2)$$

Calculating this value is highly difficult so we can calculate the posterior probability of it. The log of the marginal likelihood is:

$$logP(t|x,\theta) = -\frac{1}{2}t^T(K+{\sigma^2}_nI)^{-1}t - \frac{1}{2}log|K+{\sigma^2}_nI| - \frac{N}{2}log2\Pi \quad (3)$$

Also, the product of two Gaussians is also Gaussian (up to normalisation) and we can write out the equation of a multivariate Gaussian and take the logarithm. So, now we have to minimize this log likelihood, which we can do by using any gradient descent algorithm. For that we need to compute the gradient of it with respect to each of the hyper parameters.

$$\frac{\partial Q^{-1}}{\partial\theta} = -Q^{-1}\frac{\partial Q}{\partial\theta}Q^{-1} \quad (4)$$

$$\frac{\partial log|Q|}{\partial\theta} = trace(Q^{-1}\frac{\partial Q}{\partial\theta}) \quad (5)$$

$$\frac{\partial}{\partial\theta}logP(t|x,\theta) = -\frac{1}{2}t^TQ^{-1}\frac{\partial Q\partial\theta Q^{-1}}{t} - \frac{1}{2}trace(Q^{-1}\frac{\partial Q}{\partial\theta}) \quad (6)$$

We need to now actually perform the computations of the derivatives of the covariance with respect to each hyper parameter, and then optimize the log likelihood using the conjugate gradient solver. All of the hyper parameters are positive numbers. We can also use exponential of each of them, and since the derivative of an exponential is just the exponential, which can make things easier.

For the squared exponential kernel:

$$k(x,x') = \exp(\sigma_f)\exp(\frac{-1}{2}\exp(\sigma_l)|x-x'|^2) + \exp(\sigma_n)I \quad (7)$$

$$k(x,x') = k' + \exp(\sigma_n) \quad (8)$$

$$\frac{\partial k}{\partial\sigma_f} = k' \quad (9)$$

$$\frac{\partial k}{\partial\sigma_l} = k'x(-0.5\exp(\sigma_l)|x-x'|^2) \quad (10)$$

$$\frac{\partial k}{\partial\sigma_n} = \exp(\sigma_n)I \quad (11)$$

Using these three gradients, now we can actually perform the Gradient Descent algorithm and optimize the hyper parameters to fit the data points.

3. Experiments and Results

Let's first look at details of the data set we are going to use and then we will check the results of how optimized hyper parameters fit the data curves.

3.1. Dataset

Dataset mainly consists of values of a curve sketched by people in virtual reality. These values are measured by sensors attached to people's body and their coordinate values in 3 dimensions - x,y and z. There are 12 people, each sketching the curve for 5 times. So, there are total 60 different trials. For each trial, 50 sensors are used. That C-labeled data means whether the marker position is valid or not. If the motion capture system can not get the marker positions, then the corresponding C-label value is negative.

So, what we will do is, pick up a sensor (from 50 available) and one of the 3 coordinates, and try to fit a Gaussian Process to all the 60 trials for it. We will do these runs for 3 different (sensor, coordinate) configurations to understand hyper parameters learning in Gaussian Process.

3.2. Results

As we discussed in the methodology section now that we have the data and we have decided upon the covariance matrix, we need to optimize the hyper-parameters $\sigma_f, \sigma_l, \sigma_n$. By using the methods defined in the previous section, we try to fit Gaussian Process on the following three configurations:

1) Sensor 15, x coordinate

2) Sensor 25, y coordinate

3) Sensor 35, z coordinate

In Figure 1, we can see that Gaussian Process has nicely fit the actual data points. For Figure 3, it is also the same case. For 2, it has fit the data points, but not as nicely as the other two.

Main point here to notice is that every time these graphs are generated, we have to multiply a random vector in the calculation. So every time graph generated are a bit different. And some of them must be nicely fitting than the other ones.

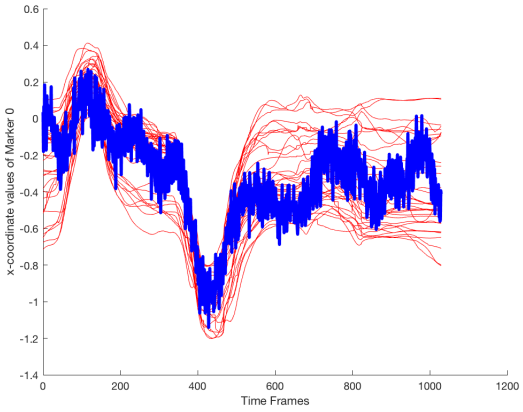

Figure 1. Sensor 15, X Coordinate

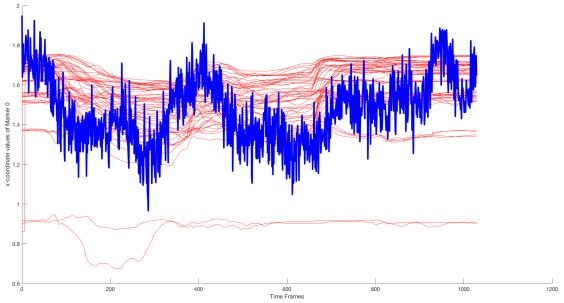

Figure 2. Sensor 25, Y Coordinate

The optimized hyper-parameter values $\sigma_f, \sigma_l, \sigma_n$ for the three configurations are (0.214, 0.943, 0.714), (0.378, 0.481, 0.337), (0.719, 0.316, 0.149) respectively.

4. Conclusion

In this paper, we implemented hyper parameter learning for Gaussian Process. We had 60 trials, where each trial had values recorded from 50 sensors along 3 dimensions. We discussed the Squared Exponential Covariance matrix and its hyper parameters. We used it for our task and discussed on how to optimize these parameters for the correct

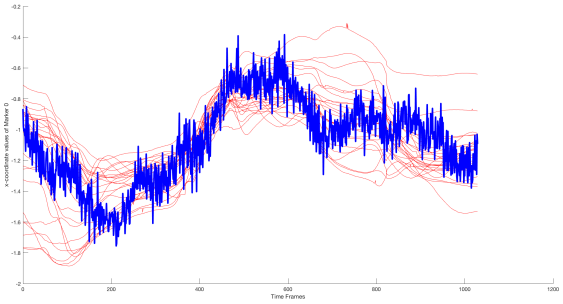

Figure 3. Sensor 35, Z Coordinate

Gaussian Process fit. We used this algorithm for 3 configurations of our data and showed that it actually fits the data curves.