Regression Using Gaussian Processes

Rajiv Sambasivan[1]

[1]Chennai Mathematical Institute

Parametric Vs Non-Parametric Regression

- Simple approaches to regression like linear or polynomial regression fail are often inadequate for many real life datasets.
- We may not be able to characterize the regression function in terms of simple parametric forms. Non parametric methods for regression are useful in these cases.

Data does not always take elementary functional forms

- What is the functional form for this curve - do we know it?

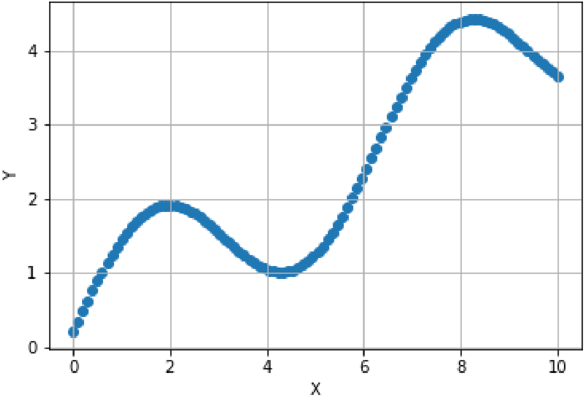

Figure: What is the functional form?

Many Choices for Non-Parametric Regression

- There are many choices for non-parametric methods - splines, kernel regression, localized polynomial regression etc.
- Gaussian Process regression is one such parametric method.

Gaussian Process Regression - Advantages

- Most non-parametric methods require hyper-parameters.
- Optimal selection for most datasets is still black-magic.
- With GP regression, you can determine this using optimization.
- See [Ghahramani, 2011] for a list of advantages of Gaussian Processes over other non-parametric methods.

Gaussian Process Regression - Advantages, contd.

- Gaussian Process regression models are probabilistic
- Confidence intervals are available with estimates.
- Questions like the probability associated with a estimate value are possible to estimate.
- In many practical applications, we want more than just the estimate - confidence intervals are often of interest.

Gaussian Process Regression - Motivation.

- There are two approaches to motivating Gaussian Process regression. These go by the name of **function space view** and **weight space view**
- The weight space view is, arguably, easier to begin with.

Bayesian View Point - A detour.

- The Frequentist view and the Bayesian view are two different approaches to viewing model parameters.
- The Frequentist view treats model parameters as having as non-random quantities while the Bayesian view point treats them as random.
- In $y = ax + b$ with a and b being model parameters, the frequentist views a and b having unqiue values for a particular problem, whereas the Bayesian view point treats them as random variables.

Bayesian Machinery

- In a Bayesian approach, we encode our beliefs about random variables by using a prior distribution.
- When new data is observed, we update our beliefs using Bayes rule. For example if we think of θ as a random variable with a prior distribution of $P(\theta)$. When new data D, about θ, is available, then we update our knowledge of θ given D as:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

- Here $P(D|\theta)$ is called the likelihood and $P(D)$, called the evidence (used to normalize the numerator in the above equation)

Bayesian Linear Regression

- Consider linear regression in a Bayesian setting.

$$y = \sum_{i} \theta_i . x_i$$

- We believe that θ is a random variable that can be updated using the mechanism discussed earlier

Kernels for Non-Linear Behavior

- If the data is such that a linear model is not a good model for the data, then we can consider kernels.
- Kernels or rather kernel regression uses a kernel $K_i(x_i)$ to transform the input x to a space where a linear model is a good fit for the data.
- In other words...

$$y = \sum_i \theta_i . K_i(x_i)$$

- The Bayesian component of modeling is identical to our earlier example.

Gaussian Process Regression

- Gaussian Process Regression is a particular case of the above set up
- In particular, we use a specific prior on θ called the Gaussian process prior.
- The estimates for a test point X_*, are standard results from theory (see [Rasmussen, 2004] for example)
- The expected value of the estimate at a given x is given by

$$\overline{y_*}(x) = E(y_*|X, Y) = K(X_*, X).[K(X, X) + \sigma_n^2.\mathbf{I}]^{-1}.\mathbf{y}$$

- The variance of the estimate at a given x is given by

$$cov(y_*) = K(X_*, X_*) - K(X_*, X).[K(X, X) + \sigma_n^2.\mathbf{I}]^{-1}.K(X, X_*)$$

Computational Difficulties

- Computing the regression estimate at a point requires inverting a matrix
- Inverting a matrix of size N is associated with $\mathbf{O}(N^3)$ time complexity. Storage associated with the computation has $\mathbf{O}(N^2)$ complexity.
- We encounter computational hurdles for large N.

Approaches to overcoming Computational Difficulties

- Since GP's are attractive computational tools, there is a lot of research in overcoming the above computational hurdle. Many computational approaches, we will mention a few.
- Sparsification - if the kernel matrix is diagonal, then computation is not expensive, however we may not be able to capture the correlation in the feature space.[Titsias, 2009]
- Approximate Inference - rather than compute the exact analytical solution, we can approximate the posterior computation using variational or mcmc techniques.[Hensman et al., 2013]
- Divide and Conquer approaches.[Tresp, 2000]

Our Recent Research - Ensembling

- We reported a method to scale Gaussian Process regression to large datasets recently[Das et al., 2018].
- The approach is based on developing models on samples obtained by simple random sampling from the data.
- We then combine the estimates from each of the models by averaging. Other methods of combining estimates are also possible.

Ensembling - Actual Algorithm

```
input  : A dataset $\mathcal{D}$ of size N, $\delta$, $K$
output: An estimator f that combines the
        estimators fitted from resampling
for $i \leftarrow 1$ to $K$ do
    /* select a sample from $\mathcal{D}$.  Two
       ways of selecting the sample
       size are presented                    */
    $N_s \leftarrow$SampleWithReplacement($\mathcal{D}, \delta$);
    /* A kernel is fit for each
       sample.  Hyper-parameter
       selection is done for each
       sample.  This computation can
       be parallelized.                      */
    $\hat{f}_i \leftarrow$FitGP($N_s$);
end
/* the estimate for a point $x \in \mathcal{D}_{test}$
   (the test dataset) is the average
   of the estimates from the K
   estimators fitted above.                  */
$f_{resampled}(x) \leftarrow \frac{1}{K} \sum_{i=1}^{i=K} \hat{f}_i(x)$
```

Algorithm 1: Gaussian Process Regression Using Resampling

Ensembling Algorithm - Parameters

- The size of the subset and the number of subsets to use are important parameters.
- [Das et al., 2018] reports two methods to pick the appropriate subset.
- [Das et al., 2018] also reports the effect of these parameters on the performance of the algorithm

Selection of GP scaling method

- There are many methods to scale Gaussian Processes
- Choice of what is appropriate depends on the characteristics of the application or context.
- Consider the application needs carefully when selecting a method.

Selection of Algorithm - Practical Guidelines

- In general, sophisticated algorithms require a number of hyper-parameters.
- Picking good values for these hyper-parameters can be critical for good performance of the algorithm.
- Pick algorithms that you are familiar with or those for which configurations are available.

Practical Guidelines, contd.

- Pay careful attention to the needs of your application.
- Use the level of sophistication and flexiblity you need for your problem.
- Do not forget or ignore data quality and feature selection. These can be critical.

Summary

Thanks!

References I

Das, S., Roy, S., and Sambasivan, R. (2018).
Fast gaussian process regression for big data.
Big Data Research, 14:12 – 26.

Ghahramani, Z. (2011).
A tutorial on gaussian processes (or why i dont use svms).
MLSS Workshop talk by Zoubin Ghahramani on Gaussian Processes [Accessed: 2016 07 19].

Hensman, J., Fusi, N., and Lawrence, N. D. (2013).
Gaussian processes for big data.
In *Conference on Uncertainty in Artificial Intellegence*, pages 282–290. auai.org.

References II

Rasmussen, C. E. (2004).
Gaussian processes in machine learning.
In *Advanced lectures on machine learning*, pages 63–71. Springer.

Titsias, M. K. (2009).
Variational learning of inducing variables in sparse gaussian processes.
In *AISTATS*, volume 12, pages 567–574.

Tresp, V. (2000).
A bayesian committee machine.
Neural computation, 12(11):2719–2741.