School of electronic and Computer Engineering

Modelling Consumption of an Electric Grid using Regression Methods

Aggelos Aggelidakis 2012039026

April 10, 2014

1 Intro

Given a dataset that contains the price of electricity and the consumption of electricity, I am modelling the consumption of the electric grid given the price per kilowatt-hour. I consider the price of electricity as an input value and consumption as a target value. For the purpose of Machine Learning's course project, I used both Gaussian process Regression and Bayesian Linear Regression to build my model.

2 Bayesian Linear Regression

I turn to a Bayesian treatment of linear regression, which will avoid the over-fitting problem of maximum likelihood, and which will also lead to automatic methods of determining model complexity using the training data alone. Firstly, I begin by introducing a prior probability distribution over the model parameters w.

The likelihood function $p(t|w)$ is the exponential of a quadratic function of w. The corresponding conjugate prior is therefore given by a Gaussian distribution of the form

$$p(w) = N(w|m_0, S_0),\ where \tag{2.1}$$

m_0 is the mean and S_0 the covariance.

Next I compute the posterior distribution, which is proportional to the product of the likelihood function and the prior. Due to the choice of a conjugate Gaussian prior distribution, the posterior will also be Gaussian. We can evaluate this distribution by the usual procedure of completing the square in the exponential, and then finding the normalization coefficient using the standard result for a normalized Gaussian. Hence, the posterior distribution is

$$p(w|t) = N(w|m_N, S_N),\ where \tag{2.2}$$

$$m_N = S_N(S_0^{-1}m_0 + \beta\Phi^T t) \tag{2.3}$$

$$S_N^{-1} = S_0^{-1} + \beta\Phi^T\Phi \tag{2.4}$$

I consider a zero-mean isotropic Gaussian governed by a single precision parameter α so that

$$p(w|\alpha) = N(w|0, \alpha^{-1}I) \tag{2.5}$$

and the corresponding posterior distribution over w is given by

$$m_N = \beta S_N \Phi^T t \tag{2.6}$$

$$S_N^{-1} = \alpha^{-1}I + \beta\Phi^T\Phi \tag{2.7}$$

2.1 Predictive distribution

In practice, we are not usually interested in the value of w itself but rather in making predictions of t for new values of x. This requires that we evaluate the predictive distribution defined by

$$p(t|t, \alpha, \beta) = \int p(t|w, \beta)p(w|t, \alpha, \beta)dw \tag{2.8}$$

in which t is the vector of target values from the training set, and we have omitted the corresponding input vectors from the right-hand side of the conditioning statements to simplify the notation. The predictive distribution takes the form

$$p(t|x, t, \alpha, \beta) = N(t|m_N^T\phi(x), \sigma_N^2(x)) \tag{2.9}$$

$$\sigma_N^2(x) = \frac{1}{\beta} + \phi(x)^T S_N \phi(x) \tag{2.10}$$

The first term represents the noise on the data whereas the second term reflects the uncertainty associated with the parameters w. Because the noise process and the distribution of w are independent Gaussians, their variances are additive.

2.2 Evidence Approximation

In a fully Bayesian treatment of the linear basis function model, we would introduce prior distributions over the hyperparameters α and β and make predictions by marginalizing with respect to these hyperparameters as well as with respect to the parameters w. However, although we can integrate analytically over either w or over the hyperparameters, the complete marginalization over all of these variables is analytically intractable. An approximation in which we set the hyperparameters to specific values determined by maximizing the marginal likelihood function obtained by first integrating over the parameters w. This approximation is known as the evidence approximation. I used this approximation in my model to calculate the optimal values of the hyperparameters. I used the algorithm 1 in order to find the optimal hyperparameters of the model.

```
$give\ input\ value\ X,\ target\ Y,\ mean\ m,\ arbitrary\ values\ to\ \alpha\ and\ \beta,\ Gram\ Matrix\ \phi;$
$give\ loglikelihood_{old} = \sum(log(normpdf(Y - Xm, 0, sqrt(\frac{1}{\beta}))));$
$give\ N := number\ of\ Gram\ Matrix\ columns,\ M :=$
$number\ of\ Gram\ Matrix\ rows;$
repeat
    $Compute\ \gamma\ and\ m_N\ given\ \alpha\ and\ \beta;$
    $m_N = \beta S_N \Phi^T t;$
    $\lambda = eig(\beta \Phi^T \Phi);$
    $\gamma = \sum \frac{\lambda}{\alpha+\lambda};$
    $Compute\ \alpha\ and\ \beta\ given\ \gamma\ and\ m_N;$
    $\alpha = \frac{\gamma}{m_N^T m_N};$
    $\frac{1}{\beta} = \frac{1}{M-\gamma} \sum_{n=1}^{N} (t_n - m_N^T \phi(x_n))^2;$
    $covariance = (\alpha\ eye(N)\ +\ \beta\ \phi^T\ \phi)^{-1};$
    $loglikelihood_{new} = (N/2)\ log(\alpha)\ +\ (M/2)\ log(\beta)\ -\ (\beta/2)\ norm(Y\ -$
    $\phi\ m_N)^2\ +\ (\alpha/2)\ m'_N\ m_N\ -\ (1/2)\ log(det(covariance))\ -\ (N/2)\ log(2\pi);$
    $terminate\ if\ loglikelihood_{old} \simeq loglikelihood_{new};$
until ;
```

Algorithm 1: Evidence approximation

2.3 Cross-Validation

I used a polynomial basis function for bayesian linear regression. However, I had to decide the degree of the polynomial basis function. Hence, I used random sub-sampling and I separated my data into 80% training data and 20% testing data. For the separation I found the MSEs for 60%-90% and chose the percentage with minimum MSE. During the cross-validation process I used degrees of polynomial from two to ten. Cross validation gave me the results shown in the table **??**. The degree of the polynomial with the minimum average mean square error is D=3. The figure 2.1 presents indicatively some predictive models and their fitting. Finally figure **??** presents fitting of the predictive model with the optimal degree choice (D=3) while **??** presents five sampled regression functions by drawing their parameters w from posterior distribution $p(w|t, \alpha, \beta)$

Degree of Polynomial	MSE
1	0.012617
2	0.007474
3	0.003761
4	0.004039
5	0.004260
6	0.004426
7	0.004549
8	0.004638
9	0.004703
10	0.004751
11	0.004787
12	0.004814

Table 2.1: Mean Square Error bayesian linear regression

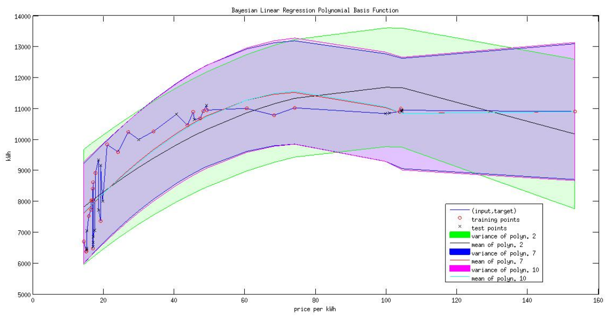

Figure 2.1: Sampling Posterior Bayesian Linear Regression

3 Gaussian Process Regression

Here I use the role of kernels as probabilistic discriminative models and define a prior probability distribution over functions directly. For a finite training set I only need to consider the values of the function at the discrete set of input values x_n corresponding to the training set and test set data points.

In order to apply Gaussian process models to the problem of regression, we need to take account of the noise on the observed target values, which are given by $t_n = y_n + \epsilon_n$, where $y_n = y(x_n)$, and n is a random noise variable whose value is chosen independently for each observation n. I considered noise processes that have a Gaussian distribution, so that $p(t_n|y_n) = N(t_n|y_n, \beta^{-1})$, where β is a hyperparameter representing the precision of the noise. Because the noise is independent for each data point, the joint distribution of the target values $t = (t)1, \dots, t_N)^T$ conditioned on the values of $y = (y_1, \dots, y_N)^T$ is given by an isotropic Gaussian of the form

$$p(t|y) = N(t|y, \beta^{-1} I_N) \tag{3.1}$$

where I_N denotes the $N \times N$ unit matrix. From the definition of a Gaussian process, the marginal distribution p(y) is given by a Gaussian whose mean is zero and whose covariance is defined by a Gram matrix K so that $p(y) = N(y|0, K)$. The kernel function that determines K is typically chosen to express the property that, for points x_n and x_m that are similar, the corresponding values $y(x_n)$ and $y(x_m)$ will be more strongly correlated than for dissimilar points. Here the notion of similarity will depend on the application. In order to find the marginal distribution p(t), conditioned on the input values $x_1, \ldots, x_N$, we need to integrate over y. The marginal distribution of t is given by

$$p(t) = \int p(t|y)p(y)dy = N(t|0, C) \tag{3.2}$$

where the covariance matrix C has elements $C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm}$
This result reflects the fact that the two Gaussian sources of randomness, namely that associated with y(x) and that associated with ϵ, are independent and so their covariances simply add.
The kernel functions that I used for Gaussian process regression are given by a polynomial form and gaussian form respectively.

$$k(x_n, x_m) = \theta_0 + \theta_1(x^T z) + \theta_2(x^T z)^2 \tag{3.3}$$

$$k(x_n, x_m) = \theta_0 \ \exp\left(-\frac{(x-z)^T(x-z)}{2(\theta_1)^2}\right) \tag{3.4}$$

Our goal in regression, is to make predictions of the target variables for new inputs, given a set of training data. Let us suppose that $t_N = (t_1, \ldots, t_N)T$, corresponding to input values $x_1, \ldots, x_N$, comprise the observed training set, and our goal is to predict the target variable t_{N+1} for a new input vector x_{N+1} . This requires that we evaluate the predictive distribution $p(t_{N+1}|t_N)$. Note that this distribution is conditioned also on the variables $x_1, \ldots, x_N$ and x_{N+1} . To find the conditional distribution $p(t_{N+1}|t)$, we begin by writing down the joint distribution $p(t_{N+1})$, where t_{N+1} denotes the vector $\{t_1, \ldots, t_N, t_{N+1}\}^T$. Hence, the joint distribution over $t_1, \ldots, t_{N+1}$ will be given by

$$p(t_{N+1}) = N(t_{N+1}|0, C_{N+1}) \tag{3.5}$$

where C_{N+1} is an (N + 1) $\times$ (N + 1) covariance matrix. This joint distribution is Gaussian, hence we can find the conditional Gaussian distribution. I partition the covariance matrix as:

$$C_{N+1} = \begin{bmatrix} C_N & k \\ k^T & c \end{bmatrix}$$

where C_N is the N $\times$ N covariance matrix, the vector k has elements $k(x_n, x_{N+1})$ for $n = 1, \dots, N$, and the scalar $c = k(x_{N+1}, x_{N+1}) + \beta^{-1}$. The conditional distribution $p(t_{N+1}|t)$ is a Gaussian distribution with mean and covariance given by

$$m_{x_{N+1}} = k^T C_N^{-1} t \tag{3.6}$$

$$\sigma^2(x_{N+1}) = c - k^T C_N^{-1} k \tag{3.7}$$

These are the key results that define Gaussian process regression. Because the vector k is a function of the test point input value x_{N+1} , we see that the predictive distribution is a Gaussian whose mean and variance both depend on x_{N+1}.

3.1 Learning Hyperparameters – Gradient Descent

The predictions of a Gaussian process model will depend, in part, on the choice of covariance function. In practice, rather than fixing the covariance function, we may prefer to use a parametric family of functions and then infer the parameter values from the data. These parameters govern such things as the length scale of the correlations and the precision of the noise and correspond to the hyperparameters in a standard parametric model. Techniques for learning the hyperparameters are based on the evaluation of the likelihood function $p(t|\theta)$ where θ denotes the hyperparameters of the Gaussian process model. The simplest approach is to make a point estimate of θ by maximizing the log likelihood function. Maximization of the log likelihood can be done using efficient gradient-based optimization algorithms. In my case I used gradient descent (3) and for line search I used backtracking line search (2) .

```
Choose $\mathbf{x}$;
Compute $\bigtriangledown f(\mathbf{x}), f(\mathbf{x})$;
while $\| \bigtriangledown f(\boldsymbol{x}) > tol\|$ do
    $t := 1$;
    while $f(\boldsymbol{x} + t\Delta\boldsymbol{x}) > f(\boldsymbol{x}) + \alpha t \bigtriangledown f(\boldsymbol{x})^T \Delta x$ do
        $t := \beta t$
    Update $x := x - t \bigtriangledown f(x)$;
    Compute $\bigtriangledown f(\mathbf{x}), f(\mathbf{x})$;
```

Algorithm 2: Backtracking line search

```
Choose $\mathbf{x}$;
Compute $\bigtriangledown f(\mathbf{x}), f(\mathbf{x})$;
while $\| \bigtriangledown f(\boldsymbol{x}) > tol\|$ do
    Compute $t_{start} > 0$ that minimizes $f(x - t \bigtriangledown f(x))$;
    Update $x := x - t \bigtriangledown f(x)$;
    Compute $\bigtriangledown f(\mathbf{x}), f(\mathbf{x})$;
```

Algorithm 3: Gradient descent method

The log likelihood function for a Gaussian process regression model is easily evaluated using the standard form for a multivariate Gaussian distribution:

$$ln\ p(t|\theta) = -\frac{1}{2}ln|C_N| - \frac{1}{2}t^T C_N^{-1} t - \frac{N}{2}ln(2\pi) \tag{3.8}$$

For non-linear optimization, we also need the gradient of the log likelihood function with respect to the parameter vector θ:

$$\frac{\partial}{\partial\theta_i}\ ln\ p(t|\theta) = -\frac{1}{2}Tr(C_N^{-1}\frac{\partial C_N}{\partial\theta_i}) + \frac{1}{2}t^T C_N^{-1}\frac{\partial C_N}{\partial\theta_i}C_N^{-1}t \tag{3.9}$$

Because $lnp(t|\theta)$ will in general be a non-convex function, it can have multiple maxima. It is straightforward to introduce a prior over θ and to maximize the log posterior using gradient descent algorithm 3. In a fully Bayesian treatment, we need to evaluate marginals over θ weighted by the product of the prior $p(\theta)$ and the likelihood function $p(t|\theta)$.

3.2 Cross-Validation

I used a polynomial and a gaussian kernel. However, I had to decide which one is the best. In fact, the best kernel is denoted as the one with the minimum mean square error. Hence, I used random sub-sampling and I separated my data into 80% training data and 20% testing data. During the cross-validation process I used a polynomial kernel of two degrees. The figure 4.1 presents the fitting of the two predictive models using polynomial and gaussian kernel respectively. It is pretty obvious that the black line which denotes the model with the gaussian kernel fits more accurate to the data. For argument's sake the mean square errors of the GPs with polynomial and gaussian kernel over the test data are 0.008315 and 0.004678 respectively.

MSE polynomial	MSE gaussian
0.0055	0.00531
0.0037	0.0159
0.0070	0.0059
0.0044	0.0042
0.0083	0.0040
0.0073	0.0148
0.0089	0.0036
0.0068	0.0036
0.0063	0.0045
0.0053	0.0052

Table 3.1: Gaussian Process Mean Square Error

4 GP (gaussian, polynomial kernels) vs. Bayesian Linear Regression

Having created both two predictive models, Bayesian Linear Regression and Gaussian Process Regression, it will be more than important to conclude to the best predictive model over the three (GP with gaussian and polynomial kernels and Bayesian Linear Regression). Although I have already compared GP with gaussian and polynomial kernel I present the figure wich presents the fitting of the three predictive models. The figure 4.2 was chosen randomly between a large number of experiments, just to present how in general these three models fit to the test data. The MSE of the specific case is presented to the table 4.1

Regression Models	MSE
Bayesian Linear Regression	0.004468
GP polynomial kernel	0.006359
GP gaussian kernel	0.006698

Table 4.1: Mean Square Error

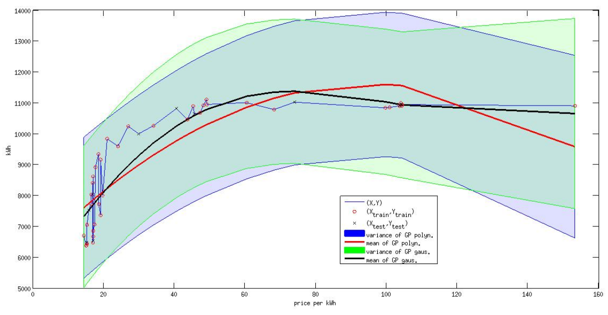

Figure 4.1: Bayesian Linear Regression

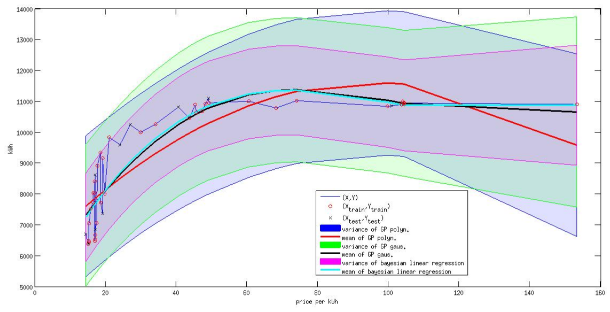

Figure 4.2: Bayesian Linear Regression