The Future of Work:

Machine Learning and Employment

Mu Chen

March 22, 2016

Abstract

Acknowledgement

Contents

1 Introduction **1**

2 Gaussian Process **2**

2.1 Gaussian Process Regression 4

2.1.1 MLE and MAP for Setting Hyperparameters 5

2.1.2 Predictions 6

2.1.3 Computational Issues 8

2.2 Gaussian Process Classification 9

2.2.1 The 'Squashing' Function 9

2.2.2 The Laplace Approximation 10

2.3 Implementation 10

2.3.1 Optimising the latent variable 10

2.3.2 Maximising the Objective Function 11

2.3.3 Predictive Probability 14

3 Gaussian Process for Employment Prediction **16**

Chapter 1

Introduction

The question of how technology could influence employment has been a popular issue for quite a long time [1]. Since 1990s, it was believed that technological changes have favoured the more skilled workers while reduced the demand for less skilled workers [2]. Nowadays, with much faster development of technologies especially in automation technology, it is very reasonable to say that less human worker is required. If it is true, could it be the major cause for lower employment rate? Will the development of technology make more people jobless? Or is it just a temporary phenomenon? This report presents how these questions are answered using machine learning techniques...

(More to go depending on future experiments)

Chapter 2

Gaussian Process

Gaussian Process(GP) is a class of distributions in which each input space follows a Gaussian distribution and any finite combination of input spaces also has a joint Gaussian distribution. Each input represents an input space and therefore the distribution of Gaussian Process can be seen as the infinite-dimensional generalization of multivariate normal distributions.

Gaussian Process has several advantages over other models. First, they are non-parametric, meaning no prior model is required for learning. Second, Gaussian Process is particularly useful for small data sets because the prescence of hyper-parameters gives it characteristics you want (such as smoothness and periodicity), at the same time keeping a fine fit to the data.

By definition, a Gaussian Process $f(x)$ is specified by a mean function and a covariance function, denoted as

$$f(x) \sim GP(\mu(x), K(x, x))$$

where

$$\mu(x) = \mathbb{E}(f(x)) \tag{2.1}$$

$$k(x, x\prime) = \mathbb{E}((f(x) - \mu(x)(f(x\prime) - \mu(x\prime)) \tag{2.2}$$

Both the mean and covariance function are the choice of user and can have crucial effects on the final Gaussian

process distribution.

To describe how all the input points are related together we use the covariance matrix defined as

$$K(x, x\prime) = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{2.3}$$

Mean Function

The prior mean function $\mu(x)$ describes the beliefs in output y(x) before any observations are made. The most popular choices of μ includes 0 and other constants. More complicated mean functions can be defined by using certain parameters based on domain knowledge. However, one should always be careful in selecting the mean function since it is what our inference is extrapolated from.

Covariance Function

Many characteristics such as smoothness, stationarity, and periodicity can be integrated in covariance function. It describes how individual observations are related to each other. In reverse, learning in Gaussian Process also defines the properties of covariance function, giving us the model that can be used for upcoming predictions.

One common choice is the squared exponential covariance function

$$k(x, x\prime) = \sigma_f^2 \exp\left[-\frac{(x - x\prime)^2}{2l^2}\right]$$

where l is the lengthscale representing how far a certain training point could affect the predictions. It is suitable for smooth data. One thing to notice is that the $k(x, x\prime)$ is actually the covariance between y and $y\prime$, corresponding to the output of x and $x\prime$ respectively. Other covariance functions such as rational quadratic function can be useful for data that is smooth over a range of length scales.

Matérn class of covariance functions is another important branch of covariance functions. They take the form of

$$K_{Matrn}(x, x\prime) = \lambda^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d(x, x\prime)}{l}\right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} d(x, x\prime)}{l}\right)$$

where $d(x, x\prime)$ is the distance between x and $x\prime$

They are used for functions of varying smoothness.

Periodicity can be added by using the periodic covariance of the form

$$K_P(x, x\prime) = \lambda^2 \exp\left(-\frac{2sin^2\left(\pi d(x, x\prime)/\rho\right)}{\omega}\right)$$

where ρ determines the period and ω gives the roughness.

Covariance functions can be combined to give multi-character correlations. For example, if we know that it requires both periodicity and smoothness in the distribution, the multiplication of squared exponential function and periodic function can be used. If the prior information is that either periodicity or smoothness exists, the addition of the above two functions suits better. Similar techniques can be applied to distributions that are known to be the sum of independent functions or the product of independent functions.

One thing to notice is that most of the covariance functions involves computing the distance between input points. This distance could either simply be the euclidean distance $d_E(x, x\prime) = \sqrt{\sum_{i=1}^{n}(x_i - x_i\prime)^2}$ or the weighted distance defined by $d_w(x, x\prime) = \sqrt{\sum_{i=1}^{n}\frac{(x_i - x_i\prime)^2}{w_i^2}}$

2.1 Gaussian Process Regression

GP Regression by its name is the process of finding a model to fit a data set using Gaussian Process. In practice, it is often the case that the output data we collected are noise corrupted. eg. $y = f(x) + \epsilon$. If we assume the noise is Gaussian with variance σ_n^2 and independent in each observation, covariance between observations then becomes

$$\mathrm{cov}(y, y\prime) = k(x, x\prime) + \sigma_n^2\delta \tag{2.4}$$

where δ is the Kronecker delta function which becomes 1 when $x = x\prime$ and 0 otherwise.

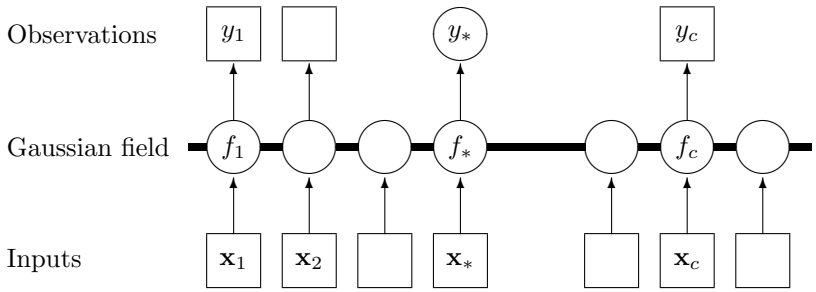


Figure 2.1: The relation between input (x), observational output (y), and the latent variable (f). Figure from [3]

2.1.1 MLE and MAP for Setting Hyperparameters

Although Gaussian Process is non-parametric, we still need to find the hyperparameters which define the mean and covariance functions. In fact, the objective of finding hyperparameters other than the parameters for model itself gives us more freedom in defining the distribution function.

According to Baye's rule,

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

if θ is the collection of all hyperparameters

$$p(\theta|X, y) = \frac{p(y|X, \theta)p(\theta)}{p(y|X)}$$

where the marginal likelihood $p(y|X)$ is given by

$$p(y|X) = \int p(y|X, \theta)p(\theta)d\theta \tag{2.5}$$

The MLE and MAP estimates the integrals by approximating the likelihood $p(y|X, \theta)$ and posterior $p(\theta|X, y)$ respectively as delta function of θ. Indeed, the differences between the approximation and the real function value may have an negative impact on our final results. However, as the approximations are on hyperparameters, which only affects mean and covariance function but not the function parameters (function values), the negative impact arisen from MLE or MAP is reduced as they propagate to the actual function values.

Note that in equation (2.5), θ is marginalised out therefore $p(y|X)$ is independent of the values of theta. The maximum a posterior estimate of θ happens when $p(\theta|X, y)$ is maximised. Since $p(\theta|X, y)$ is proportional to

$p(y|X,\theta)$, if we have little knowledge about the prior information of θ, finding the maximum of the posterior is equivalent to obtaining the value of θ that maximises $p(y|X,\theta)$. $p(y|X,\theta)$ can be written as the integration of latent variable f

$$p(y|X,\theta) = \int p(y|f,X,\theta)p(f|X,\theta)df \tag{2.6}$$

In Gaussian Process model we assume that $p(f|X,\theta)$ is a Gaussian with mean 0 and variance K. A multivariate Gaussian probability density function can be written as

$$\mathcal{N}(\mathbf{x},\mu,\mathbf{\Sigma}) = \frac{1}{\sqrt{|2\pi\mathbf{\Sigma}|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right)$$

therefore we have

$$\log p(f|X,\theta) = -\frac{1}{2}f^T K^{-1} f - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \tag{2.7}$$

also we know that $y|f \sim \mathcal{N}\left(f,\sigma_n^2 I\right)$, combining it with the above equation we get

$$\log p(y|X,\theta) = -\frac{1}{2}y^T(K+\sigma_n^2 I)^{-1}y - \frac{1}{2}\log\left|K+\sigma_n^2 I\right| - \frac{n}{2}\log 2\pi \tag{2.8}$$

This is the final objective function to be maximised. It is also consistent with the covariance of y we derived earlier in equation (2.4) where a noise term is added to the diagonal of K.

2.1.2 Predictions

After finding the optimised hyperparameters, predictions can be made by substituting the new inputs into the Gaussian Process model defined by mean and covariance function with the optimised parameters. The joint distribution of training data with no noise can be written as

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mu(X), \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{2.9}$$

Where $\mathbf{f}_*$ denotes the predictions in f domain. Similarly, distribution of noisy training data can be expressed as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mu(X), \begin{bmatrix} K(X,X)+\sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{2.10}$$

Then the distribution of $\mathbf{f}_*$ conditioned on X, X_* and $\mathbf{f}$ follows a Gaussian with mean and variance as below

$$\bar{\mathbf{f}}_* = \mu(X_*) + K(X_*,X)K(X,X)^{-1}(\mathbf{f}-\mu(X)) \tag{2.11}$$

$$\text{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*) \tag{2.12}$$

and the noisy version,

$$\bar{\mathbf{f}}_* = \mu(X_*) + K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}(\mathbf{y}-\mu(X)) \tag{2.13}$$

$$\text{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*) \tag{2.14}$$

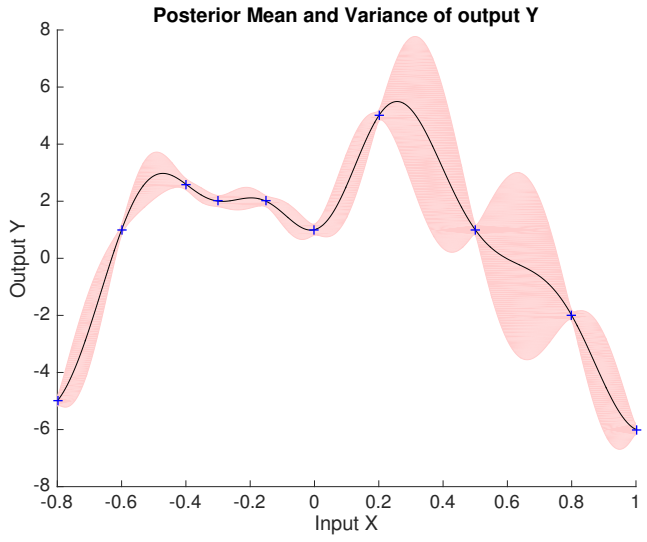


Figure 2.2: Example plot of Regression with 1D input and output. Blue points represent the training set. Red region represents the variance of prediction. Note that the variance at the training points is almost reduced to zero as there is little uncertainty at these points (they are actually zero in noise free cases). The further way from training points, the more uncertain predictions become.

2.1.3 Computational Issues

Conditioning on K

Because of the requirement for K to be invertible, K has to be full rank. However, when the input data is so closed to each other compared to the length scale used, adjacent rows in K might become the same under limited significant figures, making the matrix not full rank anymore.

This problem is avoided in regression problems by adding noise. While in classification problems that we are going to talk about later, artificial noise, also called jitters, has to be added. The amount of jitters needed differs by data, and user is required to adjust its size until K becomes positive definite.

Cholesky Decomposition

For large data sets, Gaussian Process can be very slow. The computation on large covariance matrix makes it highly time consuming. The most expensive computation is the inversion of K. Cholesky decomposition is one way to tackle this problem. It decomposes a positive definite matrix into product of two triangular matrices

$$K = LL^T = U^T U \tag{2.15}$$

where L is a lower triangular matrix and U is an upper triangular matrix. The inverse of K can simply be computed by

$$K^{-1} = L^{-T} L^{-1} = U^{-1} U^{-T} \tag{2.16}$$

It is much faster to compute the inverse of a triangular matrix, giving a reduction in computation complexity from $\mathcal{O}(n)$ to $\mathcal{O}(\frac{1}{2}n)$ [4]. In addition, matrix inversion based on cholesky decomposition is more numerically stable.

Partial Derivatives of Hyperparameters

To optimise the hyperparameters using Newton's method, we need to compute the gradients of log likelihood of θ wrt. the hyperparameters θ_i, given by

$$\begin{aligned}\frac{\partial}{\partial\theta_i}\log p(y|X,\theta) &= \frac{1}{2}y^T K^{-1}\frac{\partial K}{\partial\theta_i}K^{-1}y \;-\; \frac{1}{2}tr(K^{-1}\frac{\partial K}{\partial\theta_i}) \\ &= \frac{1}{2}tr\big((\alpha\alpha^T - K^{-1})\frac{\partial K}{\partial\theta_i}\big) \qquad \text{where} \quad \alpha = K^{-1}y\end{aligned} \tag{2.17}$$

One thing to notice is that many of the parameters, such as lengthscale and frequency, are always positive. Therefore it is more convenient to optimise over their logrithms other than themselves. e.g $\theta = \log\theta$

2.2 Gaussian Process Classification

Gaussian Process classification can be simply derived from GP regression except for the fact that outputs are discrete numbers representing class labels instead of continuous numbers in regression. Typically class labels are 1 and -1 for binary class problems. However, we do get a continuous intermediate function before the final label is decided. This is the latent variable on which regression is applied. Then the result of regression will be 'squashed' into range [0,1] to give the probability of a certain class.

2.2.1 The 'Squashing' Function

The 'squashing' function can be any sigmoid function. Two typical sigmoid functions are logistic function $\lambda(f) = 1/(1+\exp(-y_i f_i))$ and cumulative Gaussian function $\Phi(f)$. Inference is hence divided into two steps. First is to compute the distribution of the prediction latent variable f_* in terms of previous observations

$$p(\boldsymbol{f}_*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{x}_*) = \int p(\boldsymbol{f}_*|\boldsymbol{X},\boldsymbol{x}_*,\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})\;d\boldsymbol{f} \tag{2.18}$$

where $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X})/p(\boldsymbol{y}|\boldsymbol{X})$ is the posterior of latent variables $\boldsymbol{f}$, which is estimated by MAP with respect to hyperpamaters(l and σ_f).

Then the probabilistic prediction is estimated by substitute the latent f into sigmoid function and average over

$$\bar{\pi}_* \triangleq p(y_* = +1|X,y,x_*) = \int \sigma(f_*)p(f_*|X,y,x_*)df_* \tag{2.19}$$

2.2.2 The Laplace Approximation

In the case of regression (equation 2.6), we assumed that both the likelihood $p(y|f)$ and the posterior over latent variable $p(f|X,\theta)$ are Gaussian. The integral for finding $p(y|X,\theta)$ and the prediction can be computed analytically. However, in classification, the non-Gaussian likelihood and the posterior of latent function make the integral (equation (2.18)) not analytically tractable. Laplace's approximation can be used to approximate these two terms by doing a second order Taylor expansion around its maximum point.

The Gaussian approximation of the posterior of f can be obtained by:

$$q(f|X,y) = \mathcal{N}(f|\hat{f}, A^{-1}) \propto \exp(-\frac{1}{2}(f-\hat{f})^T A(f-\hat{f})) \sim p(f|X,y) \tag{2.20}$$

where $\hat{f} = \mathrm{argmax_f}\, p(f|X,y)$ and $A = - \bigtriangledown \bigtriangledown \log p(f|X,y)|_{f=\hat{f}}$ is the hessian of negative log posterior.

The reason we choose Laplace's approximation here is that it is more likely to give a better approximation than MLE or MAP [5]. Although it may still be inappropriate for the true shape of the data: the peak can be much sharper or flatter than what the approximation described.

2.3 Implementation

2.3.1 Optimising the latent variable

By Baye's rule

$$p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X})/p(\boldsymbol{y}|\boldsymbol{X})$$

We need to find the $\hat{f}$ that maximises $p(f|X,y)$. Also, $p(y|X)$ is independent of f, only the numerator need to be considered. Take the logrithm of $p(f|X,y)$ we get

$$\begin{aligned}\Psi(f) &= \log p(f|X,y) \\ &= \log p(y|f) + \log p(f|X) \\ &= \log p(y|f) + \frac{1}{2} f^T K^{-1} f - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi\end{aligned} \tag{2.21}$$

and its derivatives w.r.t. f:

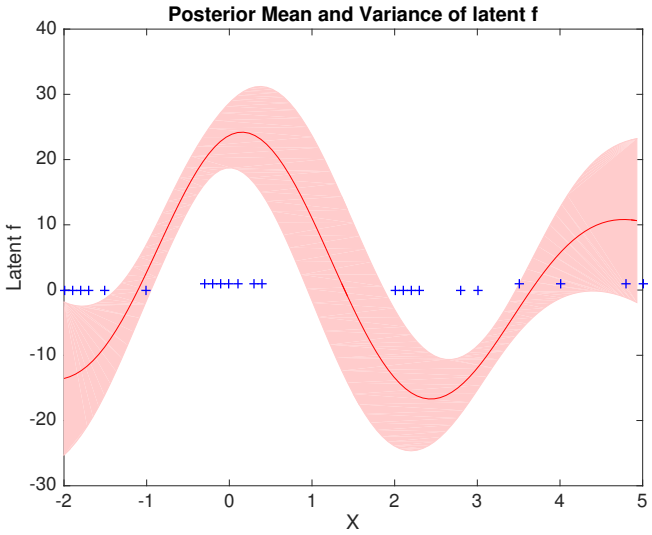


Figure 2.3: Example plot of latent variable. Blue crosses are training points and red region is its variance. The variance now is never going to be zero because the latent function value is not told directly. While in noise free regression, the exact value of output at training points is told.

$$\nabla\Psi(f) = \nabla \log p(y|f) - K^{-1}f \tag{2.22}$$

$$\nabla\nabla\Psi(f) = \nabla\nabla \log p(y|f) - K^{-1} = -W - K^{-1} \tag{2.23}$$

where W is the second derivative of negative log likelihood of f. Since we know that y_i depends on f_i only, W is a diagonal matrix. The best latent f can then be found at $\nabla\Psi = 0$, which gives

$$\hat{f} = K(\nabla \log p(y|\hat{f})) \tag{2.24}$$

This is a non-linear function therefore can be solved by Newton's method. Commonly used convergence criteria include the difference between successive values of $\Psi(f)$, the magnitude of gradient vector $\nabla\Psi(f)$ or the changes between values of f. In practice, the convergence of objective function is assured by checking that each iteration gives an increase in $\Psi(f)$. If not, a smaller step change in f should be used.

2.3.2 Maximising the Objective Function

The objective function here is the log posterior $p(y|X,\theta) = \int p(y|f)p(f|X)df = \int \exp(\Psi(f))df$ which can be obtained by using Laplace's approximation of $p(f|X,y)$ (Taylor expansion around $\hat{f}$)

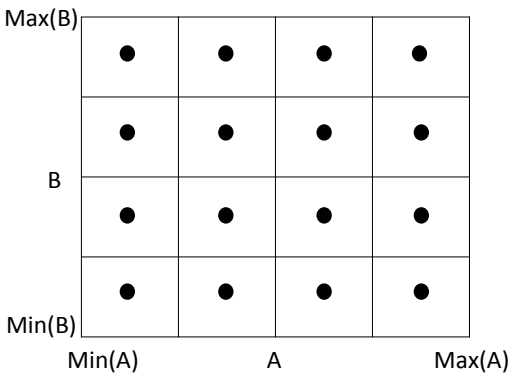


Figure 2.4: A simple example of 2-dimensional uniform sampling with 4 sampling intervals in each dimension.

$$p(y|X,\theta) \simeq q(y|X,\theta) = exp(\Psi(\hat{f})) \int \exp(-\frac{1}{2}(f-\hat{f})A(f-f\hat{f}))df \tag{2.25}$$

thus

$$\log q(y|X,\theta) = -\frac{1}{2}\hat{f}^T K^{-1}\hat{f} + \log p(y|\hat{f}) - \frac{1}{2}\log|B| \tag{2.26}$$

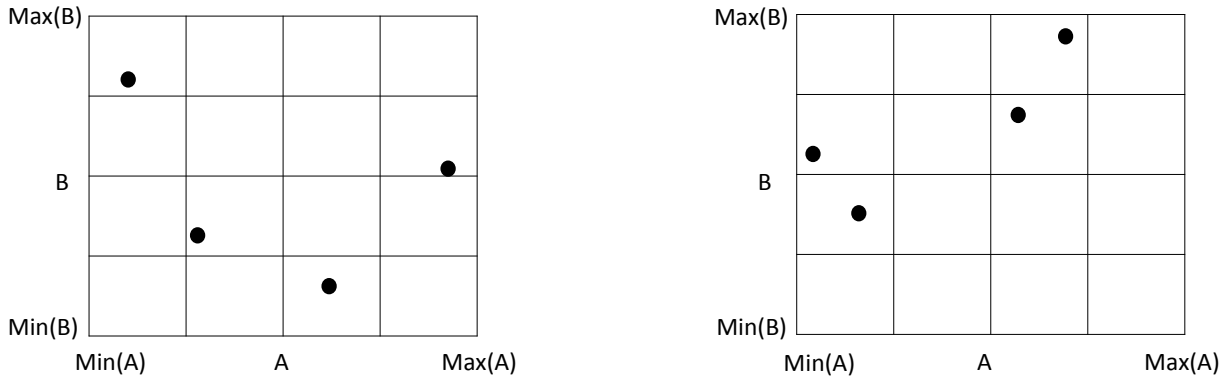
where $|B| = |I_n + W^{\frac{1}{2}}KW^{\frac{1}{2}}|$ and K is covariance defined by θ. This is the final function we need to run optimisation on.

Sampling Methods for Hyperparameters

In most of the cases, the objective function is non-linear, which means that more than one set of initial hyperparameters has to be assigned as the starting points for optimisation. Following are two methods used for low number of hyperparameters (usually 1 or 2) and higher number of hyperparameters.

Uniform sampling is one of the most common and easily obtainable sampling method. Samples are simply drawn from sample space by dividing the sample space with the number of required samples and taking one in each subspaces (or more strictly, with equal distance between samples). This is usually an idea sampling method for low dimensional problems where the number of dimensions will not largely increase the number of samples if we want to achieve the same number of sampling intervals in each dimension.

Latin Hypercube sampling is commonly used for higher dimensional problems. If we want N samples from a one-dimensional variable, we can either uniformly take N points with equal distant from one another, or we can divide the sampling space into N equal intervals and take one sample from each interval. Latin Hypercube



(a) Latin Hypercube Sampling

(b) Random Sampling

Figure 2.5: Comparison between Latin Hypercube sapling and random sampling. Although Latin hypercube sampling may not represent the most variable overall (which can be improved by Orthogonal sampling), it gives a pretty good variability in each dimension. Random sampling does not secure any variability.

Sampling (LHS) is a simple extension from uniform sampling. When taking N samples from a M-dimensional variable, the sampling space of each dimension is divided into N equal intervals. Each interval is sampled once with the order of sampling being random. Then all the samples from M dimensions are combined to give N samples each has M dimensions. Each of the possible combinations of intervals forms a multi-dimensional region that sample could draw from, called a Latin square for two-dimensional sampling space and a Latin hypercube for higher dimensions.

One advantage of LHS over random sampling is its uniformity. It ensures that samples are well-distributed in each dimension independently. While in random sampling, samples are generated without taking into account the previous samples, they results may be squeezed together, resulting in repeated optimisation results while leaving the other area blank. Non-linear function optimisation means that samples has to be well distributed. The more uniform the samples are, the more likely the global maximum could be found. Another important feature is that LHS does not require more samples for larger dimensions. In uniform sampling, a M-dimensional variable with N samples for each dimension means that N$\times$M samples will be drawn in order to cover the whole sampling space. Obviously for higher dimensional problems this is too time-consuming. For Gaussian process, the hyperparameters involved in optimisation include those used to define mean and covariance functions. When each weighting computing the distance between inputs are treated as hyperparameters, much higher dimensional samples needs to be drawn, leading to a strong preference in Latin Hypercube Sampling.

Derivatives of Hyperparameters

Again, as in regression, we need to find the gradient of likelihood w.r.t. all the hyperparameters we are trying to optimise. The case is slightly more difficult than regression as we introduced a latent variable f and used Laplace's Approximation for likelihood itself.

Since covariance matrix K is a function of hyperparameters, therefore $\hat{f}$ and W are implicit functions of hyperparameters. The derivative of equation (2.20) is

$$\frac{\partial \log q(y|X,\theta)}{\partial \theta_i} = \frac{\partial \log q(y|X,\theta)}{\partial \theta_i}\bigg|_{explicit} + \sum_{i=1}^{n} \frac{\partial \log q(y|X,\theta)}{\partial \hat{f}_i}\frac{\partial \hat{f}_i}{\partial \theta_i} \tag{2.27}$$

where

$$\frac{\partial \log q(y|X,\theta)}{\partial \theta_i}\bigg|_{\text{explicit}} = \frac{1}{2}\hat{f}^T K^{-1}\frac{\partial K}{\partial \theta_i}K^{-1}\hat{f} - \frac{1}{2}tr\Big((W^{-1}+K)^{-1}\frac{\partial K}{\partial \theta_i}\Big) \tag{2.28}$$

$$\frac{\partial \log q(y|X,\theta)}{\partial \hat{f}_i} = -\frac{1}{2}\big[(K^{-1}+W)^{-1}\big]_{ii}\frac{\partial^3}{\partial f_i^3}\log p(y|\hat{f}) \tag{2.29}$$

2.3.3 Predictive Probability

Predictive mean can be decided under Laplace's approximation by combining the prediction in regression eq. (2.11) with eq. (2.24)

$$\mathbb{E}[f_*|X,y,x_*] = K(X_*,X)K(X,X)^{-1}\hat{f} = K(X_*,X)\nabla \log p(y|\hat{f}) \tag{2.30}$$

Predictive variance consists of two terms: one from f_* given f, the other from our estimation of f

$$\mathbb{V}[f_*|X,y,x_*] = \mathbb{E}[(f_* - \mathbb{E}[f_*|X,y,x_*])^2] + \mathbb{E}[(\mathbb{E}[f_*|X,x_*,f] - \mathbb{E}[f_*|X,y,x_*])^2] \tag{2.31}$$

using the matrix inversion lemma [6] we get

$$\mathbb{V}[f_*|X,y,x_*] = K(x_*,x_*) - K(x_*,X)(K(X,X)+W^{-1})^{-1}K(X,x_*) \tag{2.32}$$

Predictive probability the new output belong to class 1 is then computed by eq. (2.19) except that the distribution

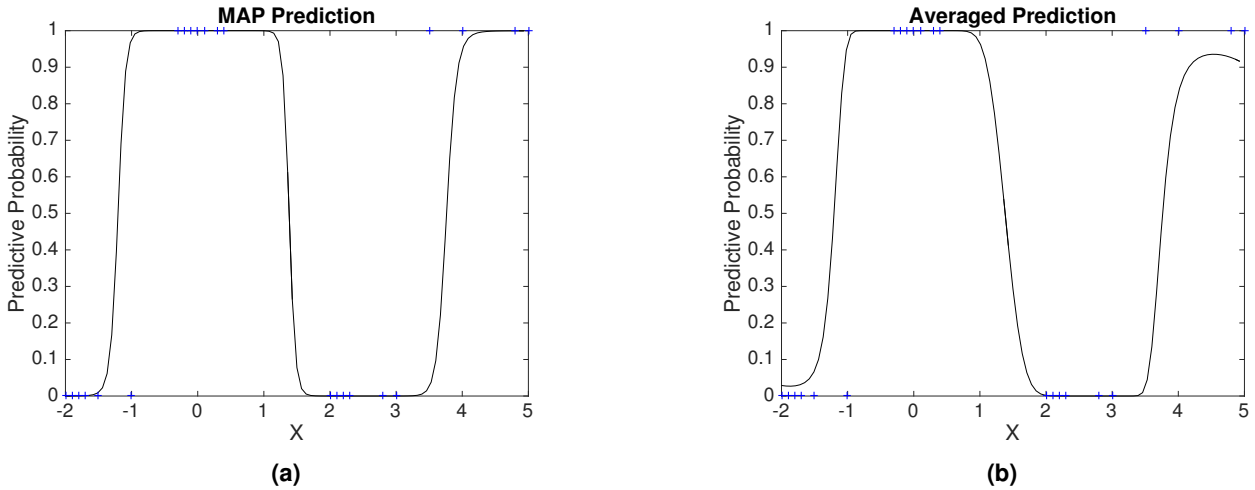


Figure 2.6: (a) the MAP prediction. (b) the averaged probability prediction. Blue crosses are training points (class label 1 and -1 but marked as 1 and 0 for better display). The MAP prediction goes to the extremes quicker while the averaged probability is more moderate and tend to be affected by adjacent points

of f_* is now the result of Laplace's approximation with mean and variance as stated above.

$$\bar{\pi}_* \triangleq p(y_* = +1|X, y, x_*) = \int \sigma(f_*) q(f_*|X, y, x_*) df_* \tag{2.33}$$

One may argue that expectation of the prediction could be simply equal to the sigmoidal mean of f_*. These are actually two different things because of the non-linearity of sigmoid function, the first is averaged probability $E[\pi_*|X, y, x_*]$ while the later one is MAP prediction $\sigma(E[f_*|y])$. Although the final class labels assigned are the same for both cases, only the averaged probability gives the statistically correct results.

Chapter 3

Gaussian Process for Employment Prediction

To integrate Gaussian process into the prediction of occupational automatability, we first define the occupational descriptors as the input and the probability of being automated in the future as the output. Hand-labelled training data [7] which has 33 occupations with label '0'(not possible to be automated) and 37 of them with label '1'(highly possible to be automated).

Squared exponential is chosen as the covariance function as the output is not expected to have any periodicity but a general smoothness would be helpful (occupations with similar descriptors should have a similar probability of being automated).

The classifier is first tested by splitting the training set into two groups each with 35 instances, and using one group for training, the other for comparison. The result of the classification on the second group is compared with its true labels by plotting a ROC curve.

ROC (short for Receiver Operating Characteristic) is a popular tool for performance measurements in binary classification. It is the plotting of true positive rate against false positive rate. One particular interpretation of the curve is its Area Under Curve (AUC). In normalised form, AUC is equal to the probability that a classifier will justify a class 1 object more likely to be class 1 than an object from the other class. The higher the value of AUC, the better the classifier.

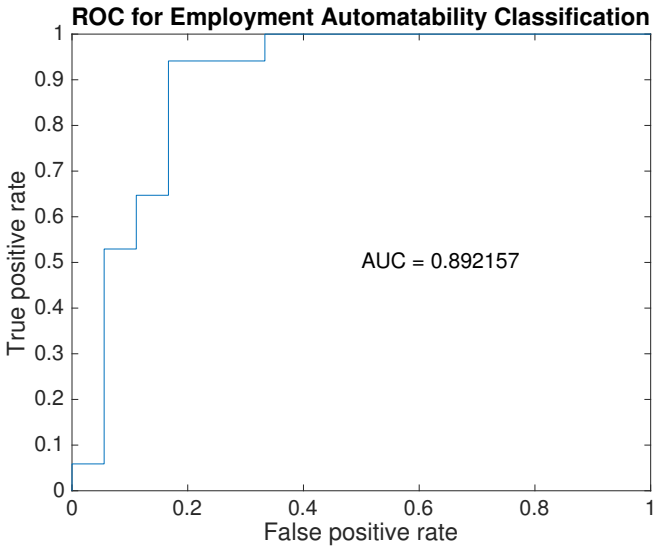


Figure 3.1: A typical ROC curve plotted with 35 training data.

References

[1] John Van Reenen Stephen Machin. Technology and changes in skill structure: Evidence from seven oecd countries. *The Quarterly Journal of Economics*, 113(4):1215–1244, 1998.

[2] John Bound and George E Johnson. Changes in the structure of wages during the 1980's: An evaluation of alternative explanations. Technical report, National Bureau of Economic Research, 1989.

[3] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Process for Machine Learning.* MIT Press, 2006.

[4] Damián Edgardo Marelli and Minyue Fu. Distributed weighted least-squares estimation with fast convergence for large-scale systems. *Automatica*, 51:27–39, 2015.

[5] Adriano Azevedo-Filho and Ross D Shachter. Laplace's method approximations for probabilistic inferencein belief networks with continuous variables. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 28–36. Morgan Kaufmann Publishers Inc., 1994.

[6] Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42:106, 1950.

[7] Carl Benedikt Frey and Michael A Osborne. The future of employment: how susceptible are jobs to computerisation. *Retrieved September*, 7:2013, 2013.