Coursework 3
Mathematics for Machine Learning (CO-496)

Instructions

This coursework has both writing and coding components. The coding part must be done in python. The code will be run on the standard CSG installation and will be tested on the labTS [1] system. On starting the course you will be given a gitlab repository. Every commit you make to this repository can be marked automatically.

You are not permitted to use any symbolic manipulation libraries (e.g. `sympy`) or automatic differentiation tools (e.g. `tensorflow`, `pytorch`) for your submitted code (though, of course, you may find these useful for checking your answers). You should not need to import anything other than `numpy` for the submitted code for this assignment.

The writing assignment requires plots, which you can create using any method of your choice. You should not submit the code used to create these plots.

No aspect of your submission may be hand-drawn. You are strongly encouraged to use LaTeXto create the written component.

You are required to submit the following:

- A file `write_up.pdf` for your written answers, submitted through CATe
- A file `answers.py` which implements all the methods for the coding exercises, submitted through gitlab.

[1] https://teaching.doc.ic.ac.uk/labts

1 Bayesian Linear Regression

We use the model

$$p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x}, \alpha, \beta) = \left(\prod_{i=1}^{N} \mathcal{N}(y_i|\boldsymbol{w}^T\boldsymbol{\phi}_i, \beta)\right) \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha\boldsymbol{I}) \tag{1}$$

and again the data set as defined in the instructions.

The marginal likelihood is $\log \int p(\boldsymbol{y}, \boldsymbol{w}|\boldsymbol{x}, \alpha, \beta) d\boldsymbol{w}$

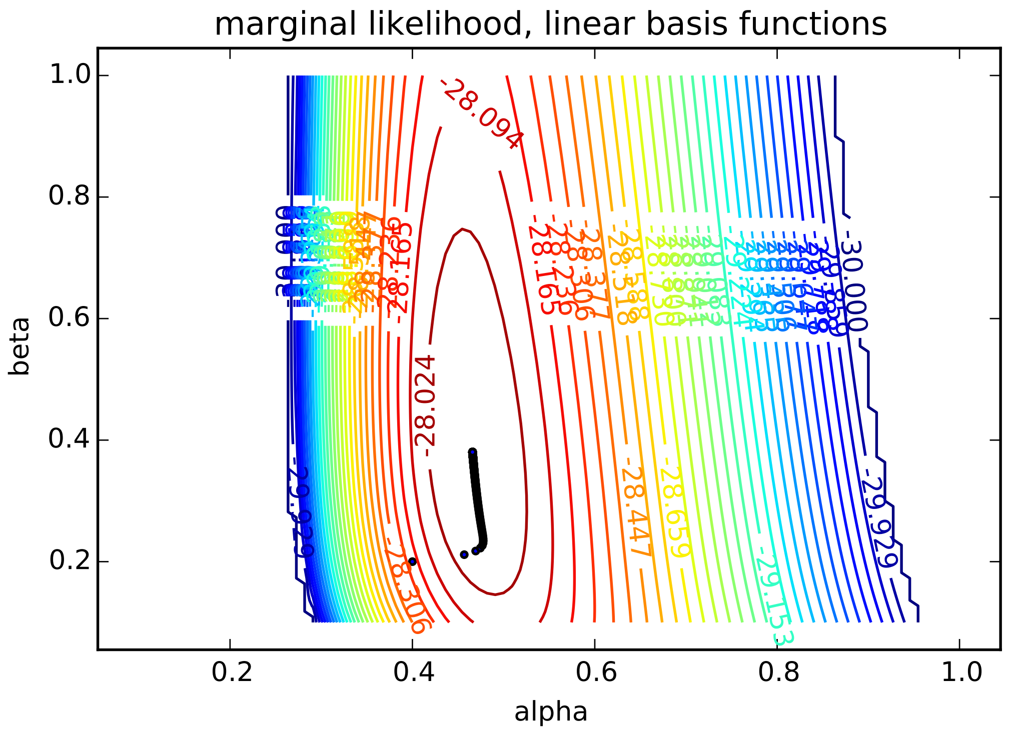
a) Write a python function that `lml(alpha, beta, Phi, Y)` that returns the log marginal likelihood, and also a function `grad_lml(alpha, beta, Phi, Y)` that returns the gradient of the log marginal likelihood with respect to the vector (α, β). [2]. The function should return a numpy vector with the gradient with respect to α in the first component and gradient with respect to β in the second.

   code

   4 marks for the marginal likelihood correct and 8 for the gradients (4 per component)

b) For the given data set and the linear basis functions (i.e. polynomial of order 1), maximize the log marginal likelihood with respect to α and β using gradient descent. Show your steps on a contour plot as you did in previous questions. It is up to you where you start, but be careful that the log marginal likelihood varies over several orders of magnitude so you may have to start fairly close. You may have to clip your contours to show anything interesting on the plot. Don't use a log scale for α and β (though this would be sensible). Report your results for the maximum.

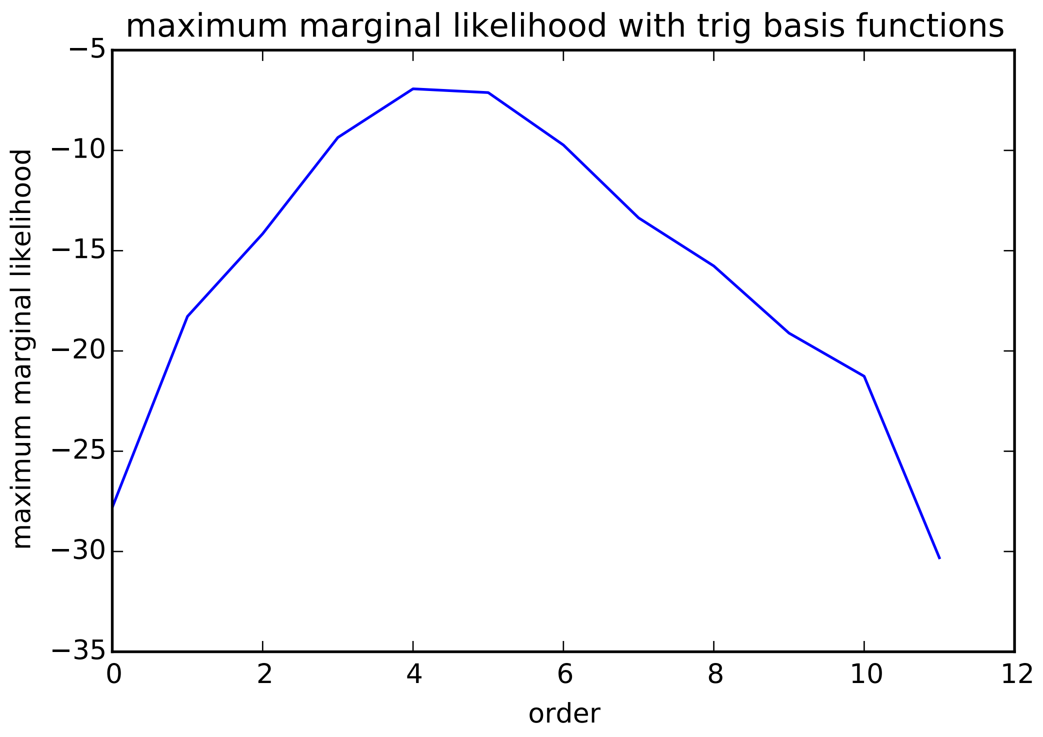
[2]It is more straightforward if you do this in $N \times N$ form. That is, write the likelihood as $\mathcal{N}(\boldsymbol{y}|\boldsymbol{\Phi w}, \beta\boldsymbol{I})$ and use the standard results for Gaussians. Leaving the matrices in this form is very inefficient for large N, but if you like you can use the Woodbury identity to rewrite it in a way that only requires the determinant and inverse of an $M \times M$ matrix, where M is the dimension of the basis functions. Alternatively you can complete the square and get the result out directly, following e.g. Bishop PRML p167



$\alpha = 0.42454258$ $\beta = 0.44923188$

- Correct value for α and β
- Contour with sensible scales showing the maximum clearly
- Gradient descent steps shown, with sensible starting position and step size indicated

c) In the case of trigonometric basis function, compute the maximum of the log marginal likelihood for orders 0 to 11 inclusive using gradient descent (make sure you choose good starting values and a small step size with plenty of iterations). Plot these values on a graph against the order of the basis functions. Compare your answer to your cross validation graph from the first coursework (1c) and describe briefly the merits of the two approaches



The Bayesian approach

- Does not require expensive loop over all data points
- Does not waste any data
- Leverages our prior assumptions (assuming they are correct)

The cross validation approach

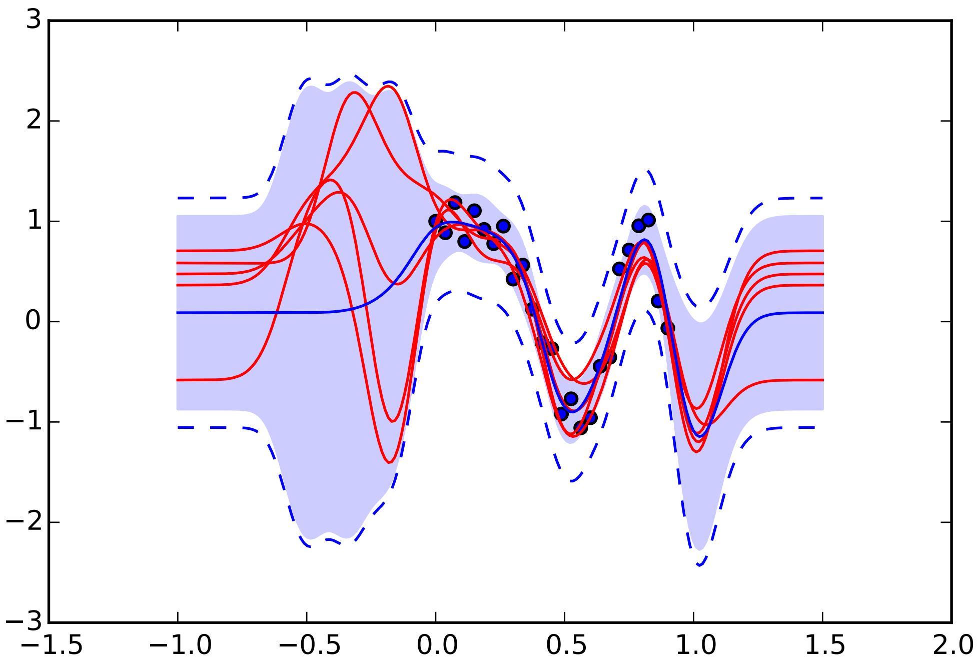
- Does not require prior assumptions
- Is conceptually simpler
- Might give better generalization performance if the the prior assumptions of Bayesian approach are false

They need one point for each. There may be plenty of other value reasons so use your judgment.

3 marks for the correct graph

- A point in favor of the Bayesian approach
- A point in favor of the cross validation approach

d) For $\alpha = 1.$ and $\beta = 0.1$ take 5 samples from the posterior distribution over the weights in the case of 10 Gaussian basis functions equally spaced between -0.5 and 1 (inclusive) with scale 0.1. Use these samples to plot the noise-free predicted function values at the test points (i.e. with y-values $\mathbf{\Phi}^*\boldsymbol{w}$, where Φ^* is the matrix of stacked basis functions evaluated at the test inputs x^*). Plot also the predictive mean and 2 standard deviation error bars as a shaded region. Don't include the noise in your shaded region, but do add also dotted curves indicated two standard deviations including the noise (i.e. dotted for $\boldsymbol{y}^*$ and shaded for $\boldsymbol{\Phi}^*\boldsymbol{w}$). Use test points in the interval -1 to 1.5 to show the behavior away from the data and away from the basis function centers. Plot the samples in a different color and use a low alpha (in the sense of opacity!) for the shaded region. Plot also the data.



- Predictive mean
- Shaded region for noise-free prediction
- Error bars including noise
- 5 samples, clearly shown in different colour
- All correct

e) (Extension: not to be graded). Use a large number of basis functions in a wider interval and experiment with different values of α and β. Use gradient descent to find the best α and β (you will probably have to use a log scale to get this to work effectively), or alternatively use a more sophisticated algorithm like conjugate gradients (you will certainly need to use a log scale for this work). Plot your results.