Deadline: December 9, 2021

1 Theoretical

1.1 Multiple Outputs

Up until now we only talked about predicting one number for each input using linear regression. However, in many tasks we may want to predict more than one response from the inputs. In this section we will show how to do this and show that this is equivalent to using multiple single output regression problems.

Observe the following generalization of linear regression to multiple outputs:

$$y(x) = \Theta^T h(x) + \eta \tag{1.1}$$

where $\eta \sim \mathcal{N}\left(0, I\sigma^2\right)$ as usual, the weights are given in the matrix $\Theta \in \mathbb{R}^{p \times k}$ and $h : \mathbb{R}^d \to \mathbb{R}^p$ are our p basis functions. In this notation, $y \in \mathbb{R}^k$ is a vector, *not* a scalar output. The log-likelihood of the problem is given by:

$$\log p(y \mid \Theta, \sigma) = \sum_{i=1}^N \log \mathcal{N}\left(y_i \mid \Theta^T h(x_i), I\sigma^2\right) \tag{1.2}$$

1. Define $Y \in \mathbb{R}^{N \times k}$ as the matrix of all of the outputs for the N training points. Find the MLE solution for Θ as well as σ^2

2. Show that the MLE solution can be decoupled into k different regression problems. In other words, show that the solution $\hat{\Theta}_{\text{ML}} \in \mathbb{R}^{p \times k}$ is equivalent to stacking k estimators of the form $\hat{\theta}_{\text{ML}}^{(i)} \in \mathbb{R}^p$

This solution also generalizes to Bayesian linear regression, when the priors for each output are independent (otherwise the result may be a bit different, but that involves using priors over *matrices*, which we will not be doing here). So if we want to predict multiple outputs, and are willing to believe that each dimension in the predictions is independent of the other dimensions, we can either use the matrix form above or fit k different regressors, and get the same result.

1.2 Product of Kernels

In this part of the exercise, we will prove one of the clauses you have seen in the recitation regarding kernels. Specifically, if $k_1(x, y)$ and $k_2(x, y)$ are valid kernels, then:

$$k_\times(x, y) = k_1(x, y) \cdot k_2(x, y) \tag{1.3}$$

is also a valid kernel. To prove this we will use the fact that valid kernels are positive semi-definite.

You may find the following identities helpful (but don't have to use them):

$$x^T A y = \text{trace}\left[x^T A y\right] = \text{trace}\left[yx^T A\right] \tag{1.4}$$

$$\text{trace}\left[AB\right] = \sum_i [AB]_{ii} = \sum_i \sum_j A_{ij} B_{ji} \tag{1.5}$$

where x and y are vectors while A and B are matrices.

3. Let $k(x, y)$ be a valid kernel and suppose that K is the kernel's Gram matrix over some finite set of points $\{x_i\}_{i=1}^N$, such that $K_{ij} = k(x_i, x_j)$. Show that for any finite set of N points, there exists some function $f : \mathcal{X} \to \mathbb{R}^N$ such that:

$$k(x_i, x_j) = f^T(x_i) f(x_j) \tag{1.6}$$

where $\mathcal{X}$ is the space of the points x_i. Using this fact, show that:

$$k_1(x, y) \cdot k_2(x, y) = \sum_i \sum_j g_i(x) f_j(x) f_j(y) g_i(y) \tag{1.7}$$

where $k_1(\cdot, \cdot)$ and $k_2(\cdot, \cdot)$ are valid kernels, and some functions $f, g : \mathcal{X} \to \mathbb{R}^N$, where $f_i(x)$ denotes the ith index of the output of $f(x)$

4. Conclude that $k_\times(x, y) = k_1(x, y) \cdot k_2(x, y) = h^T(x) h(y)$ for some function $h(\cdot)$, thereby proving that $k_\times(\cdot, \cdot)$ is a valid kernel

1.3 Kernel Functions

For each of the following functions, prove whether it is a valid kernel or not:

5. $k(x, y) = e^{\|x - y\|^2}$

6. $k(x, y) = k_1(x, y) - k_2(x, y)$ for any two valid kernels $k_1(\cdot, \cdot)$ and $k_2(\cdot, \cdot)$

7. $k(x, y) = k_a(x_a, y_a) + k_b(x_b, y_b)$ for any two valid kernels $k_a(\cdot, \cdot)$ and $k_b(\cdot, \cdot)$, where $x = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$ and $y = \begin{pmatrix} y_a \\ y_b \end{pmatrix}$

1.4 Evidence in the Dual Space

Recall that, given a kernel function $k(x, x')$, the dual problem has the equivalent form:

$$f_\alpha(x) = \boldsymbol{k}^T(x) \alpha + \eta \tag{1.8}$$

where $\eta \sim \mathcal{N}(0, \sigma^2)$, $\boldsymbol{k}(x) = \begin{pmatrix} k(x, x_1) \\ \vdots \\ k(x, x_N) \end{pmatrix} \in \mathbb{R}^N$ and the points $\{x_i\}_{i=1}^N$ are the training points. Additionally:

$$\alpha \sim \mathcal{N}(0, K^{-1}) \tag{1.9}$$

where K is the Gram matrix of the kernel $k(\cdot, \cdot)$, defined as:

$$K_{ij} = k(x_i, x_j) \tag{1.10}$$

8. Show that the evidence of the dual problem is given by:

$$p(y|k(\cdot, \cdot), \sigma^2) = \mathcal{N}(y \,|\, 0,\, K + I\sigma^2) \tag{1.11}$$

2 Practical

In this exercise, we will try to choose which basis function and sample noise to use in our models by calculating the evidence function. Because of numerical stability, what we will actually want to do is to calculate the log-evidence score, which is just the log of the evidence score.

For this exercise, you will also need to fit Bayesian linear regression models to data. You can either use your implementation from the previous exercise, or use the implementation supplied in `ex3_utils.py`. The utils script also contains an example of how to use the supplied implementation to predict and to find the standard deviation of the MMSE prediction.

1. Implement a function that receives the prior and sample noise of a Bayesian linear regression model and returns the log-evidence score for the given model, defined as[1]:

$$\log p\left(y \mid \mu_\theta, \Sigma_\theta, \sigma^2\right) = \frac{1}{2}\log\frac{\left|\Sigma_{\theta|\mathcal{D}}\right|}{\left|\Sigma_\theta\right|} - \frac{1}{2}\left[\left(\mu_{\theta|\mathcal{D}} - \mu_\theta\right)^T \Sigma_\theta^{-1}\left(\mu_{\theta|\mathcal{D}} - \mu_\theta\right) + \frac{1}{\sigma^2}\|y - H\mu_{\theta|\mathcal{D}}\|^2 + N\log\sigma^2\right] - \frac{p}{2}\log 2\pi \tag{2.1}$$

where N is the number of points and p is the number of basis functions

2.1 Evidence for Artificial Functions

In this section we will look at the behavior of the evidence function on artificial data. The data we will use will be functions with some added noise. We will then evaluate various possible basis functions using the evidence function.

The functions we will use are:

- $f_1(x) = x^2 - 1$

- $f_2(x) = x^3 - 3x$

- $f_3(x) = x^6 - 15x^4 + 45x^2 - 15$

- $f_4(x) = 5\frac{e^{3x}}{1+e^{3x}}$

- $f_5(x) = 2\left(\sin\left(2.5x\right) - |x|\right)$

The noise we will add to these functions will be $\eta \sim \mathcal{N}\left(0, I\sigma^2\right)$ with a variance of $\sigma^2 = 0.25$. For simplicity, the prior we will use in each of the models will be $\theta_R \sim \mathcal{N}\left(0, I\alpha\right)$ with $\alpha = 5$. For each of the functions $f_1(\cdot)$, $f_2(\cdot)$, $f_3(\cdot)$, $f_4(\cdot)$ and $f_5(\cdot)$:

2. Sample 500 points equally spaced in the range $x \in [-3, 3]$, calculate the function on these points and add random noise sampled from η

3. Calculate the log-evidence for 10 models with polynomial basis functions with degrees $d = [2, \cdots, 10]$ and the matching prior θ_R. Plot the log-evidence for each of these models as a function of the degree of the polynomial basis functions. Which model has the highest evidence?

4. Fit Bayesian linear regression to the best and worst model, according to the evidence function, and plot them (together with the confidence intervals). Does the evidence match your choice?

2.2 Estimating the Sample Noise for Temperature Prediction

In this part of the exercise, we will try to find the sample noise of the temperatures data set from the previous exercise. To do this, we will calculate the evidence of out model under different sample noises. In the file `temp_prior.npy` you are supplied with the prior θ_7 for a 7th order polynomial fitted to the temperatures data set from the previous exercise. An example of how to load this can be found in the supplied utilities script `ex3_utils.py`.

5. Calculate and plot the log-evidence score for 100 different models using sample noises equally spaced in the range $\sigma^2 \in [0.05, 2]$ on the temperatures of the first half of November 16 2020, which can be found in the file `nov162020.npy`

6. Plot the log-evidence score for each of the models as a function of the sample noise. Which sample noise has the highest evidence?

7. Does this mean that this is the true sample noise of the original measurements? Explain your answer

---

[1] This is simply the log of the definition we saw in class and in recitation 5

3   Submission Guidelines

Submit a single zip file named "ex3_<YOUR ID>.zip". This file should contain your code, along with an "ex3.pdf" file in which you should write your answers to the theoretical part and add the figures/text for the practical part. Please write readable code, as the code will also be checked manually (and you may find it useful in the following exercises). In the submitted code, please make sure that you write a basic main function in a file named "ex3.py" that will run (without errors) and produce all of the results that you showed in the pdf of answers that you submitted. The only packages you should use are numpy, scipy and matplotlib. You may also reuse code from your previous exercise in order to answer the questions in this exercise, if needed.

In general, it is better if you type your homework, but if you prefer handwriting your answers, please make sure that the text is readable when you scan it.

Part of your assignment will be graded by submitting your answers through Moodle, at this link. In each of the questions, write the answer to the corresponding question for grading. These answers will be graded automatically, so write only numeric values where needed.

4   Supplementary Code

In the file ex3utils.py you can find an example of how to load the supplied data as well as a few helper functions. You can use this code as you see fit, and change any part of it that you want, just be sure to submit it as well if you change it. Finally, we have also supplied an outline code which you can use to get started in ex3.py. You don't have to use the format we outlined, but your code must run without errors and you must submit the plots required in the exercise description.

Good luck!