SF1935 Probability Theory and Statistics with Application to Machine Learning

Project in Bayesian inference for regression

Pawel Herman

You will present your group lab effort both orally and in writing. The oral presentation in the lab will be evaluated by one of teaching assistants. Prepare a few slides introducing the project and illustrating your key findings plus your observations and reflections that you find worth sharing as part of the discussion of your results. Importantly, please avoid showing your code, implementation etc. and focus instead on communicating the fruits of your intellectual effort. After your presentation, be prepared to answer questions and try to get all group members equally involved in the entire presentation and discussion.

The written part of project/lab work consists in a short report (one per group) not exceeding 7-8 A4 pages. Your report should include a very brief introduction, a short section on methods and the main part should be devoted to presenting key results and discussing your findings (again, sharing your observations, reflections). The introduction in essence amounts to stating the purpose and aims of the project (ca.0.5-1 page). The method section covers some fundamental aspects of your implementation, e.g. which programming/scripting language and accompanying libraries you have used (0.5-1 page). The results section should be written as a narrative story where you first remind the purpose of your simulation, then a short description of your experimental setup (e.g. how you use data, evaluate, find parameters) followed by the actual results that are accompanied by figures and/or tables (altogether 4-5 pages). Finally, the discussion section should mainly account for your reflections, observations, lessons learnt and conclusions (1 page). You should have a draft (preliminary version) of your project report ready for the actual oral presentation (just bring it along). Most importantly though, the final version should be submitted after the presentation through Canvas.

The reason why this mini-project is evaluated (P/F) based on both oral and written account is to help you develop competence in communicating your work in a concise manner. Please keep in mind that an ability to communicate your results and conclusions is a key aspect of any data science / machine learning practitioner. It is up to you as an author to make sure that the report and presentation clearly shows what you have done, how you prioritise your findings and what observations you have made (and consider worth discussing).

As far as the implementation is concerned, please use any programming/scripting language/environment that you feel comfortable with. I can just recommend Python due to its popularity among data scientists. Matlab, R are other good options. It is assumed that you know how to represent matrices and vectors in the programming/scripting language of your choice. Make sure also how you can visualise 2D matrix, contour plot among others is helpful.

1 Background - theory

1.1 Introduction to probabilistic regression

Regression is the task of estimating a continuous target variable $\mathbf{t}$ from an observed variate $\mathbf{X}$. The target and the observed variates are related to each other through a mapping,

$$f : \mathbf{X} \rightarrow \mathbf{t}, \tag{1}$$

where f indicates the mapping. Given input output pairs $\{(\mathbf{x}_i, t_i)\}_1^N$ our task is to estimate the mapping f such that we can infer the associated t_i from previously unseen $\mathbf{x}_i$. In this task we will work with real vectorial data such that $\mathbf{x}_i \in \mathbf{X}$ where $\mathbf{x}_i \in \mathbb{R}^q$ and $t_i \in \mathbb{R}^1$ (in Fig.1 q=1 so $\mathbf{x}$=x).

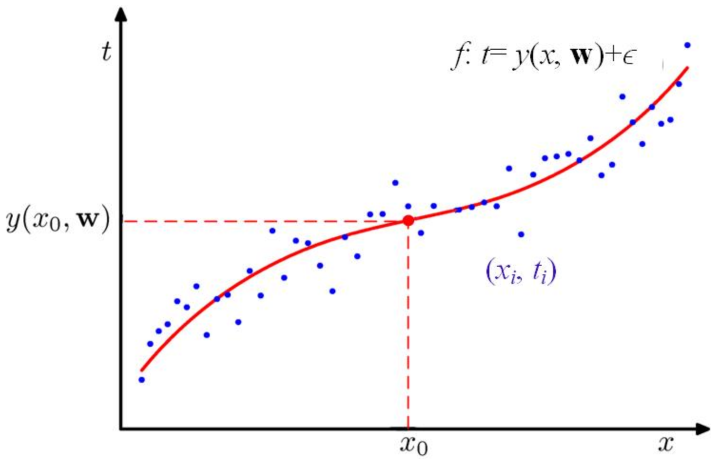

Figure 1: Regression task: function y (in red) should model the mapping f underlying the observations (x_i, t_i) (in blue).

Being probabilistic means that we need to consider the uncertainty in both the observations as well as the relationship between the variates. Starting with the relationship between two *single* points $\mathbf{x}_i$ and t_i we can assume the following form of the likelihood (noise model for the outputs),

$$p(t_i|f, \mathbf{x}_i) = \mathcal{N}(f(\mathbf{x}_i), \beta^{-1}). \tag{2}$$

Assuming that each output point is conditionally independent given the input and the mapping we can write the likelihood of the data as follows:

$$p(\mathbf{t}|f, \mathbf{X}) = \prod_{i=1}^{N} p(t_i|f, \mathbf{x}_i). \tag{3}$$

Bear in mind that $\mathbf{t}$ is a vector of target values of all samples, i.e. $\mathbf{t} = [t_1, t_2, ..., t_N]^{\mathrm{T}}$, and $\mathbf{X}$ is the corresponding matrix of inputs, i.e. $\mathbf{X}^{\mathrm{T}} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$ (q-by-N).

The task of regression means that we wish to infer t_i from its corresponding variate $\mathbf{x}_i$. These two variates are related to each other by the mapping f so from a probabilistic viewpoint we wish to find the mapping from the observed data. More specifically, taking uncertainty into account, what we wish to reach is the posterior distribution over the mapping given the observations,

$$p(f|\mathbf{X}, \mathbf{t}). \tag{4}$$

There are nonparametric methods to figure out this posterior but we are going to simplify the task and take a parametric approach where the underlying model is linear with respect to parameters. Then for this linear model (see Eq. 6), we wish to identify the posterior over the parameters $\mathbf{w}$, i.e. $p(\mathbf{w}|\mathbf{X}, \mathbf{t})$. Once we have the posterior, we can make predictions in the Bayesian spirit for any new input $\mathbf{x}$ using predictive distribution ($\mathcal{D}_{trn}$ describes the training data $(\mathbf{X}, \mathbf{t})$) - see section 1.5:

$$p(t|\mathbf{x}, \mathcal{D}_{trn}) = \int p(t|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathcal{D}_{trn}) d\mathbf{w} \tag{5}$$

1.2 Linear regression - problem setting

Let's make an assumption about the mapping and model the relationship between the variates in terms of a linear basis function model [1]. Next, let's assume that the structure of the noise in the observations follows additive Gaussian distribution ($i = 1, .., N$):

$$t_i = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i) + \epsilon, \tag{6}$$

where $\epsilon \sim \mathcal{N}(0, \beta^{-1})$ with β being called precision (inverse of variance). From this we can formulate the likelihood of the data (cf. Eq. 3),

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} p(t_i|\mathbf{x}_i, \mathbf{w}) \tag{7}$$

If we look at Eq. 6, we can deduce that $t_i \sim \mathcal{N}(\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1})$, which we can write as

$$p(t_i|\mathbf{x}_i, \mathbf{w}, \beta) = \mathcal{N}(t_i|\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1}). \tag{8}$$

So the likelihood becomes

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(t_i|\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1}), \tag{9}$$

where

$$\mathcal{N}(t_i|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{t_i - \mu}{\sigma}\right)^2\right) \tag{10}$$

with the mean $\mu = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i)$ and the variance $\sigma^2 = \beta^{-1}$.
In fact, the likelihood is a multivariate normal distribution (even though we calculate this in practice as the aforementioned product of univariate normal distributions due to t_i independence):

$$\boxed{p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}\left(t_i|\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1}\right) = \mathcal{MN}\left(\mathbf{t}|\boldsymbol{\Phi}(\mathbf{X})\mathbf{w}, \beta^{-1}\mathbf{I}\right),} \tag{11}$$

where

$$\mathcal{MN}(\mathbf{t}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{Z} \exp\left(-\frac{1}{2}(\mathbf{t} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{t} - \boldsymbol{\mu})\right), \tag{12}$$

where $Z = (2\pi)^{\frac{N}{2}} det(\boldsymbol{\Sigma})^{\frac{1}{2}}$, the vector mean $\boldsymbol{\mu} = \boldsymbol{\Phi}(\mathbf{X})\mathbf{w}$ and the covariance matrix $\boldsymbol{\Sigma} = \beta^{-1}\mathbf{I}$ [2].
Let us be pragmatic and stick to product of univariate distributions in any calculations later on.
We have also used a so-called design matrix, $\boldsymbol{\Phi}$ (dimensions NxM), with information about the values of M basis functions $\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x})]$[3] for N input samples, $\mathbf{x}_i, i = 1, .., N$, collected in the data matrix $\mathbf{X}$:

$$\boldsymbol{\Phi}(\mathbf{X}) = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix} \tag{13}$$

[1] Basis functions ϕ can be nonlinear, which makes the mapping between the variates nonlinear. At the same time, we still call it a linear model since it is a linear function of the parameters $\mathbf{w}$, which we are trying to figure out.
[2] $\mathbf{I}$ is the identity matrix with the dimensionality corresponding to the number of samples N.
In the formula for the multivariate normal distribution the focus here is on the part that depends on $\mathbf{t}$.
[3] Typically, $\phi_0 = 1$ to accommodate for a free term (intersection at $\mathbf{x} = 0$) in a regression model.

The basis functions could be, for example, polynomials so for the univariate case ($\mathbf{x}$ is reduced to x) we could write $\boldsymbol{\phi}(x) = [1, x, x^2, ..., x^{M-1}]$. Another simplifying example would be a linear model with respect to both parameters $\mathbf{w}$ (weights) and the inputs, $\mathbf{x}$ (multivariate case so back to the vector notation) with $\boldsymbol{\phi}_0 = \mathbf{1}$ and $\boldsymbol{\phi}_1(\mathbf{x}) = \mathbf{x}$ (in other words, $\boldsymbol{\phi}(\mathbf{x}) = [\mathbf{1}, \mathbf{x}]$), which implies that

$$t_i = w_0 + w_1 x_1 + ... w_q x_q + \epsilon \tag{14}$$

For the compact notation, we often introduce the notion of extended input vector $\mathbf{x}_{ext} = [1, \mathbf{x}]^{\mathrm{T}}$ so that w_0 corresponds to 1 in the input allowing us to write [4]

$$t_i = \mathbf{w}^{\mathrm{T}} \mathbf{x}_i + \epsilon \tag{15}$$

The design matrix in such case gets reduced to the data matrix $\mathbf{X}$ with the extra column of 1s corresponding to the weight w_0 (as above). Analogously, we use the extended notation for all input data $\mathbf{X}_{\text{ext}}$ with the dimensionality of N by $q + 1$

$$\boldsymbol{\Phi} = [\mathbf{1}, \mathbf{X}] = \mathbf{X}_{\text{ext}} \tag{16}$$

Then the likelihood for the Gaussian noise model in Eq. 10 becomes (cf. Eq. 11)

$$\boxed{p(\mathbf{t}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(t_i | \mathbf{w}^{\mathrm{T}} \mathbf{x}_i, \beta^{-1}).} \tag{17}$$

1.3 Maximum likelihood approach to parameter estimation in regression

Assuming that we have data, i.e. a set of inputs $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]^{\mathrm{T}}$ with the corresponding target values $\mathbf{t} = [t_1, t_2, ..., t_N]^{\mathrm{T}}$, we can in the spirit of the frequentist approach find the parameters $\mathbf{w}$ of the underlying linear model in Eq. 6. For that purpose we should find parameters that maximise the likelihood in Eq. 9. For computational purposes the log-likelihood is typically maximised, so the equation is set to find parameters $\mathbf{w}$ for which the gradient of the log-likelihood is 0 (covex case for Gaussian distributions). By taking derivative of the log normal, we get the gradient first:

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N} \{t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n)\} \tag{18}$$

and then solve the equation with the gradient above equal to 0. This way we obtain the fixed values of parameters $\mathbf{w}$ from the max likelihood principle (frequentist approach):

$$\boxed{\mathbf{w}_{\text{ML}} = (\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}} \tag{19}$$

Then we can also estimate β_{ML} (though the focus in the frequentist approach is on $\mathbf{w}_{\text{ML}}$):

$$\frac{1}{\beta_{\text{ML}}} = \frac{1}{N} \sum_{n=1}^{N} \{t_n - \mathbf{w}_{\text{ML}}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n)\}^2 \tag{20}$$

So, for the case $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}_{ext}$ we obtain:

$$\boxed{\mathbf{w}_{\text{ML}} = (\mathbf{X}_{\text{ext}}^{\mathrm{T}} \mathbf{X}_{\text{ext}})^{-1} \mathbf{X}_{\text{ext}}^{\mathrm{T}} \mathbf{t}} \tag{21}$$

This $\mathbf{w}_{\text{ML}}$ can now be used to make (max likelihood) predictions with the linear model in Eq. 15, corresponding to the predicitve mean, i.e. $t(\mathbf{x}) = \mathbf{w}_{ML}^{\mathrm{T}} \mathbf{x}$.

[4]Here we imply that $\mathbf{x}_i$ is in the extended form (we drop the subscript *ext*) for individual vectors $\mathbf{x}$.

1.4 The posterior and prior - towards Bayesian regression

Let us discuss now a different approach to regression than the maximum likelihood estimation in the previous subsection. In the Bayesian spirit, the posterior (likelihood x prior) is the object that integrates our prior beliefs with the data

$$p(\mathbf{w}|\mathbf{X},\mathbf{t}) = \frac{1}{Z} p(\mathbf{t}|\mathbf{X},\mathbf{w})p(\mathbf{w}), \tag{22}$$

where Z is the normalization constant that does not depend on $\mathbf{w}$ and ensures that the posterior integrates to value 1 (probabilities should add up to 1). In the above equation we can see that we need to formulate our belief of the model parameters $\mathbf{w}$ in a prior $p(\mathbf{w})$. We can make many different choices of priors, but we pick the Gaussian prior (i.e. with multivariate normal distribution) over the independent parameters with the same precision α (inverse of variance)[5]:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) \tag{23}$$

Now if we aim for calculating the posterior from Eq. 22 we exploit the fact that the likelihood and prior are both Gaussian distributions (we then say that our prior is a conjugate Gaussian prior), which implies that the posterior is also Gaussian:

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N) \tag{24}$$

where

$$\boxed{\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \mathbf{\Phi}^{\mathrm{T}} \mathbf{t} && (25) \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} && (26) \end{aligned}}$$

So, for the case $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}_{ext}$ we get

$$\boxed{\begin{aligned} \mathbf{m}_N &= \beta \mathbf{S}_N \mathbf{X}_{\mathrm{ext}}^{\mathrm{T}} \mathbf{t} && (27) \\ \mathbf{S}_N^{-1} &= \alpha \mathbf{I} + \beta \mathbf{X}_{\mathrm{ext}}^{\mathrm{T}} \mathbf{X}_{\mathrm{ext}} && (28) \end{aligned}}$$

Bear in mind what we have done here - instead of point estimating values of our parameters $\mathbf{w}$, which maximise the likelihood, as in Eq. 21, we have computed their posterior distribution with the mean $\mathbf{m}_N$ and covariance $\mathbf{S}_N$.

1.5 The predictive distribution in Bayesian regression

Once we have the posterior (Eq. 22), obtained from the likelihood and prior, and its concrete estimate in Eqs. 27 and 28 given the assumption of Gaussian distributions, we can make predictions (for new values of $\mathbf{x}$). In the lecture we talked about the central role of predictions in machine learning. If we do not follow Bayesian philosophy, i.e. we care about the estimated value of fixed parameters $\mathbf{w}$ in line with the frequentist mindset (maximum likelihood estimate in sec. 1.3), then we can directly apply these point estimates to Eq. 6 or its simplified version without basis functions $t_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i + \epsilon$ (as suggested at the end of section 1.3). However, if we adopt a Bayesian mindset we consider the posterior as the probabilistic description of parameters that can be used to make predictions (i.e. without knowing the exact values of these weights). The predicitve distribution we can produce then by "averaging out" (or in statistical terms - marginalisation of) the weight parameters (hence we integrate out over $d\mathbf{w}$) [6]:

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta) p(\mathbf{w}|\mathbf{t}, \alpha, \beta) d\mathbf{w} \tag{29}$$

[5]If we do not consider α to be a fixed parameter, we can also write prior as $p(\mathbf{w}|\alpha)$

[6]To simplify notation we just skip obvious dependence of these distributions on $\mathbf{x}$

Since the posterior (second term in the integral) and our underlying (original) linear model (Eq. 8) are Gaussian, their multiplication is also Gaussian, which in the end as a result of rather simple calculus leads to the following form of the predicitve distribution (now for the explicit notation, we bring back the obvious dependence of this distribution on the input $\mathbf{x}$ for which we make a prediction t):

$$p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x})) \tag{30}$$

where the variance σ_N^2 is

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{S}_N\boldsymbol{\phi}(\mathbf{x}) \tag{31}$$

Please recall that $\mathbf{m}_N$ and $\mathbf{S}_N$ are defined in Eq. 25 and Eq. 26, respectively. So, for the simplest case with $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}_{ext}$, the Gaussian (normal) predictive distribution

$$\boxed{p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \mathcal{N}\left(t|\mu_N(\mathbf{x}), \sigma_N^2(\mathbf{x})\right)} \tag{32}$$

has the parameters (see Eq. 28 for $\mathbf{S}_N^{-1}$ and $\mathbf{x}_{ext}$ is the extended version of the specific input $\mathbf{x}$):

$$\boxed{\begin{aligned} \mu_N(\mathbf{x}) &= (\beta\mathbf{S}_N\mathbf{X}_{\mathrm{ext}}^{\mathrm{T}}\mathbf{t})^{\mathrm{T}}\mathbf{x}_{ext} && (33)\\ \sigma_N^2(\mathbf{x}) &= \frac{1}{\beta} + \mathbf{x}_{ext}^{\mathrm{T}}\mathbf{S}_N\mathbf{x}_{ext} && (34) \end{aligned}}$$

1.6 Some summarising remarks

In this project/lab assignment we use only Gaussian (normal) distributions for prior, posterior and likelihood. These are typically multivariate normal distributions except the predictive posterior for the output predictions (unless we work with models with multiple outputs, here our output samples t_i are univariate). Bear in mind that the weight prior and posterior are defined in the weight space, $\mathbf{w}$, whereas the likelihood is in the joint data space over all samples $(\mathbf{X}, \mathbf{t})$ [7]. So, when you use your multivariate (or univariate) normal distribution make sure you formulate it in the right space (data or weight parameter). Accordingly, you need to attend which mean and covariances matrices are used. Also, please do not mix hyperparameters α (precision for the prior) and β (precision for the likelihhod). We fix them in the project rather than learn. In addition, there is σ^2, which does not describe any uncertainty about the model but the noise in the "hidden" data generating mechanism. Finally, please pay attention to the dimensionality of vectors and matrices.

[7] We can still calculate the likelihood of concrete data samples for different weight parameters, $\mathbf{w}$, as in Task 1. We understand it as the probability of observing these concrete data samples as model outputs if we assume a concrete model $\mathbf{w}$.

2 Course assignment on regression

2.1 Warm-up task with a single input variable

Now we will implement a very simple probabilistic linear regression (where the input is one-dimensional so $\mathbf{x}_{\text{ext}} = [1, x]$) with the focus on estimating the posterior over the parameters $\mathbf{w} = [w_0, w_1]$ and then making predictions for new inputs. In essence, the aim here is to:

- familiarise yourself with a probabilistic nature of a simple (1D) linear regression problem - estimating the probabilistic model ($\mathbf{w}$ parameters) based on training samples and applying it to generate predictions for test samples (t for new unseen test x)
- understand where the prior and likelihood come from, what they represent, how they are parameterised (Gaussian distributions with the mean and covariance), and how they weigh in for the posterior over the parameters
- distinguish between the posterior over the parameters and the predictive posterior to make probabilistic predictions (output of the regression)
- learn how to operate with multivariate normal distributions, draw random samples and visualise 2D distributions
- gain insight into the conceptual difference between the maximum likelihood (point estimates of parameters $\mathbf{w}$) and Bayesian regression (posterior over the parameters and predictive posterior for the outputs)

There are a couple of packages in Python that are really useful (similar functionality can also be found in Matlab, R and other languages):

```python
 import pylab as pb
 import numpy as np
 import matplotlib.pyplot as plt
 from math import pi
 from scipy.stats import multivariate_normal
 from scipy.spatial.distance import cdist
# To draw n samples from multivariate Gaussian distribution with mu and Cov:
 f = np.random.multivariate_normal(mu,Cov,n)
```

The idea is to compute the posterior over the parameters $\mathbf{w} = [w_0, w_1]$ using the Bayesian approach based on the available training data and the prior (again, over the parameters $\mathbf{w} = [w_0, w_1]$). By varying the number of training samples we will study the effect of the likelihood and the prior on the model. To do so we will need to have some data to experiment with. What we want to show is that the methodology that we have learned is capable of recovering the true underlying mapping from the observed data. In other words, we will pretend that we do not know the original values of $\mathbf{w}$ used for data generation and try to recover their distribution (remember, we do not care about their exact values in the Bayesian approach) from the data and the prior assumption. Therefore let's generate some training and testing (unseen) data $(\mathbf{x}, \mathbf{t})$ and then simply forget about the generating parameters (in real-world task you would only have the data and you would not know anything about the mechanism or function that generated them):

$$D_{trn}: \quad t_i = w_0 + w_1 x_i + \epsilon = -1.2 + 0.9 x_i + \epsilon, \qquad \mathbf{x} = [-1, -0.99, \ldots, 0.99, 1] \tag{35}$$

$$D_{tst}: \quad t_i = w_0 + w_1 x_i + \epsilon = -1.2 + 0.9 x_i + \epsilon, \qquad \mathbf{x} = [-1.5, -1.4, \ldots, -1.1, 1.1, \ldots, 1.5] \tag{36}$$

$$\epsilon \sim \mathcal{N}(\mu, \sigma^2) \text{ , where } \mu = 0 \text{ and } \sigma^2 = 0.2 \tag{37}$$

We will run several models for varying subset sizes of the training data. In the first step we will assume a Gaussian prior over $\mathbf{w}$ (weights) and then based on the available observations from the training set (selected subset) we will compute the Gaussian likelihood over (as a function of) the parameters $\mathbf{w} = [w_0, w_1]$. In the next step we will estimate the posterior using the equations derived in sec. 1.4 (Eqs. 27 and 28). Ultimately, we would like to see that the more data we have available for training, the more accurate posterior distribution of the weights can be obtained. We expect that it will eventually be centered around the parameters we have used for generating the data (and then threw away or forgot about) with rather small uncertainty or, we could say, great precision ("tight" covariance of the Gaussian distribution). Let us take a simplistic view and assume ad hoc the precision of the prior α and also fix β[8].

Task 1:

1. *Set the prior distribution over* $\mathbf{w}$ *with* $\alpha = 2$ *(in other simulations you can try also other values, e.g. 0.5 or 4) and visualise it. Although it is a multivariate normal distribution (in 3D over 2D space of the weight parameters), make two-dimensional contour plot over that* $w_0 \times w_1$ *space. You can check out some plotting clues in sec. 3.1). Remember that the actual probability density function in itself is deterministic (describes probability values).*

2. *From the generated training data pick a subset of training data samples (x,t) (pick 3 samples in the first model simulation and then examine larger training data subsets, say 10, 20 and 100 samples). For your training data subset calculate and plot the likelihood across all* $\mathbf{w}$ *in your parameter space (*$w_0 \times w_1$*) - see Eq. 17 (for each* $\mathbf{w}$ *it is a multiplication of probabilities / likelihoods over the available training samples).*

3. *Next, compute the posterior distribution over* $\mathbf{w}$ *(Eqs. 27 and 28) and visualise it in the same way as the prior.*

4. *Now we would like to investigate what model we have learnt so far. For this, let's draw, say, 5 model samples (weight parameters)* $\mathbf{w} = [w_0, w_1]$ *from the posterior we obtained in the previous step and plot the resulting models, i.e. linear functions (for each concrete sample* $\mathbf{w} = [w_0, w_1]$ *that you randomly draw from the posterior over the weight parameter space just plot* $y = w_0 + w_1 x$ *in the data space for* $x \in \mathbf{x}$*) together with both training samples (from the training subsets) and all the test samples.*

5. *Finally, you can also make predictions for the testing points* x*. Keep in mind that for the Bayesian approach each prediction is actually a Gaussian distribution (see Eq. 32). For each testing point* x*, plot the corresponding mean (Eq. 33) and standard deviation from Eq. 34 (as an error bar) of the predicted output* t*.*

6. *To your plot you could also add a prediction (deterministic value) made using a maximum likelihood estimator, as described in sec. 1.3 .*

7. *Test the exercise for different values of* σ^2*, e.g. 0.1, 0.4 and 0.8, varying* α *and the increasing number of training samples. How does your model account for data with varying noise levels? What is the effect of noise, the precision of the prior and the number of training samples on the parameter posterior as well as the uncertainty of the model predicitions for test samples?*

8. *What is the difference between the predictions obtained with the maximum likelihood in comparison with Bayesian approach?*

[8]We can set β to the value corresponding to σ^{-2}. In a real-world problem we would not know σ so we would need to guess or assume some β and then iteratively estimate it using methods beyond the scope here.

2.2 Mini-project task with linear regression in multidimensional input space and batch learning

In this mini-project task your aim is to perform a similar regression task. This time however $\mathbf{x}$ is a two-dimensional vector and $\phi(\mathbf{x}) = [1, {x_1}^2, {x_2}^3]^\mathrm{T}$ so that

$$t_i = \mathbf{w}^\mathrm{T}\phi(\mathbf{x}_i) + \epsilon = w_0 + w_1{x_{1i}}^2 + w_2{x_{2i}}^3 + \epsilon \text{ , where } \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{38}$$

The model is still linear with respect to weight parameters $\mathbf{w}$ but thanks to nonlinear basis functions it allows for nonlinear mapping $f : \mathbf{x} \to t$. The overarching aim is similar as before though you should learn now how to compute the data matrix with basis functions (cf. Eq. 13) and use it to estimate the model and its predictions. We will follow a similar plan as before, i.e. we first generate some data for some fixed model parameters, say $\mathbf{w} = [0, 2.5, -0.5]$, then forget these model parameters (throw them away) and use the data to either estimate these parameters (maximum likelihood) or find their posterior (Bayesian approach). Finally, we will generate predictions for unseen test inputs $\mathbf{x}$. As we have access in the test set to the corresponding desirable outputs, we will be able to measure the model's test (generalisation) performance. Bear in mind that with Bayesian mindset we are actually not so interested in the weight parameters themselves (to start with, from the Bayesian point of view, $\mathbf{w}$ is a random variable with probability distribution, not fixed parameters) but the target predictions or rather their distribution.

Task 2:

1. *Generate and plot data points using the model defined in Eq. 38 over the 2D domain of the input space where* $\mathbf{x} = [x_1, x_2] \in [-1, -0.95, \dots, 0.95, 1] \times [-1, -0.95, \dots, 0.95, 1]$. *Please use some fixed value of data noise* σ, *say,* $\sigma = 0.3$ *(try also* $\sigma \in \{0.5, 0.8, 1.2\}$*).*

2. *Choose the samples for which* $|x_1| > 0.3$ *and* $|x_2| > 0.3$ *as your test data and keep the rest as your training subset. Add extra zero-mean noise to the test data* $(t + \epsilon_{extra})$

3. *Fit the model using the maximum likelihood principle on the training data for the value of* σ *used to geberate it in p.1. For the fixed maximum likelihood estimates of the parameters* $\mathbf{w}$, *generate output predictions for test data inputs,* $\mathbf{x}$, *and evaluate the predictive performance of the model by calculating mean square error (maximum likelihood gives you point estimates of the predicted* t *so you can see what error is made relative the corresponding true values in the test data).*

4. *For the given value of* σ^2 [a] *and different values of the uncertainty parameter of the Gaussian prior over the weight parameters* $(\alpha \in \{0.3, 0.7, 2.0\})$, *perform Bayesian linear regression, i.e. compute the predictive posterior for the outputs* t *(the mean and variance of this Gaussian distribution - generalise Eqs. 33 and 34 to the version with* $\mathbf{\Phi}$*).*

5. *Make comparison between the two modelling approaches (maximum likelihood vs Bayesian) in terms of their predictive performance on the test data (MSE). Bear in mind that we need to simplify the Bayesian philosophy here as we need to reduce the predictive posterior (each prediction is described by the Gaussian distribution in Eq. 32) to a point estimate by taking its mean (Eq. 33) - this is not really in the Bayesian spirit.*

6. *For the Bayesian approach, generate predictions also for the training data and compare the variance of the predicitions (uncertainty) between the training and test data. You can only do that for the Bayesian framework as it outputs the Gaussian distribution with the mean and variance as a prediction for each input sample. How do the uncertainty (variance) and quality (MSE) of these predictions change with varying* α *and* σ^2*?*

[a] As previously, let's simplistically assume that we know β and set it to the values corresponding to σ^{-2}

3 Some implementation guidelines in Python

3.1 Plotting two-dimensional distributions

To start with, I strongly recommend that you check the following tutorial for plotting with matplotlib: https://www.tutorialspoint.com/matplotlib/index.htm. In the assignment, you are asked to plot probability distributions (prior, likelihood, posterior...). Since we operate in the continuous space we shoud say probability density function, which in itself is a deterministic function. You can read out its values for concrete arguments (inputs) using, e.g. *scipy.stats.multivariate_ normal.pdf*. For a two-dimensional case your input space (data or weight parameters) is 2D so you need first to make a meshgrid over the Cartesian product of two input variables (spanning 2D space), then calculate the value of your function for all points in the meshgrid and, finally, plot them using, e.g. contour.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
w0list = np.linspace(-3.0, 1.0, 200)
w1list = np.linspace(-2.0, 2.0, 200)
W0arr, W1arr = np.meshgrid(w0list, w1list)
pos = np.dstack((W0arr, W1arr))
 # set your mu vector and Cov array
rv = multivariate_normal(mu, Cov)
Wpriorpdf = rv.pdf(pos)
plt.contour(W0arr, W1arr, Wpriorpdf)
plt.show()
```

Alternatively, you could implement your own multivariate Gaussian pdf

```python
def pdf_multivariate_Gauss(x, mu, Cov):
    Z = 1 / ( ((2* np.pi)**(len(mu)/2)) * (np.linalg.det(Cov)**(1/2)) )
    norm_exp = np.exp((-1/2) * ((x-mu).T.dot(np.linalg.inv(Cov))).dot((x-mu)))
    return float(Z * norm_exp)

 # iterate over W0arr, W1arr arrays to calculate PDF array:
 # PDFarr[i,j]=pdf_multivariate_Gauss([W0arr[i,j],W1arr[i,j]], mu, Cov)
plt.contour(W0arr, W1arr, PDFarr)
```

3.2 Matrix operations

In this assignment you need to define and operate (add, dot product, scalar product, element-wise multiplication, transpose) on matrices and vectors. Please check the existing tutorials on the web, e.g.: https://python-course.eu/numerical-programming/matrix-arithmetics-under-numpy-and-python.php or https://www.pythonpool.com/python-vector/ or https://www.oreilly.com/library/view/machine-learning-with/9781491989371/ch01.html

3.3 Common pitfalls or things to pay special attention to

1. One of key issues is to differentiate reading from the probability density function, *pdf* (obtaining the probability or visualising the distribution) from actually drawing a sample from the probability distribution.
   The probability distribution function, *pdf*, itself is a deterministic function one can obtain calling *scipy.stats.multivariate_ normal.pdf* (some more info and examples can be found under https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.multivariate_normal.html). This gives probabilities (readout from deterministic *pdf* as a function of x).
   Sampling from the distribution on the other hand implies drawing a random sample with probability described by the distribution (*numpy.random.multivariate_ normal* for both uni- and multivariate normal). One can draw a few samples at once. So, drawing a sample means getting random x from *pdf* (probability density function), where $x \sim pdf$.

2. Another thing to keep in mind is the sapce (domain) for different probability distributions: weight prior and posterior over the weight parameter space, the likelihood in the data space and the predictive posterior in the output (target variable) domain.

3. Make sure you use the extended format of the input data (with 1s).