Machine Learning

Assignment 2

April 21, 2020

Deadline: Mai 12, 2020 at 23:55h.
Submission: Upload your report and your implementation to the TeachCenter. Please do not zip your files.

1 Online Bayesian Linear Regression

The goal of this task is to do linear regression in a fully Bayesian setting. "Fully Bayesian" denotes thereby that we are interested in the full predictive distribution instead of only a predicted value. Let us therefore assume we have given a dataset consisting of inputs $\mathbf{x} = \{x_1, \dots, x_N\}$ and corresponding target values $\mathbf{t} = \{t_1, \dots, t_N\}$. The predictive distribution is defined as

$$p(t|x, \mathbf{t}, \mathbf{x}) = \int \underbrace{p(t|x, \mathbf{w})}_{(a)} \underbrace{p(\mathbf{w}|\mathbf{t}, \mathbf{x})}_{(b)} d\mathbf{w}, \tag{1}$$

where the (a) is the likelihood for a new sample t given x and the parameters $\mathbf{w}$ and (b) is the posterior distribution over $\mathbf{w}$ given the dataset. We assume that the target variable

$$t = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(x) + \eta, \tag{2}$$

where η is Gaussian noise and the function $\boldsymbol{\phi}(\cdot)$ are Gaussian Radial Basis Functions (RBFs), which we will define precisely later. Next, we can write the likelihood of a target variable t as

$$p(t|x, \mathbf{w}) = \mathcal{N}(t|\mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(x), \beta^{-1}), \tag{3}$$

where $\beta = \frac{1}{\sigma^2}$ is the precision parameter of the Gaussian. Additionally, we assume that the samples in our dataset are independent and identically distributed and thus we can define the likelihood of our dataset using eq. (1) with

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}) = \prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}) \tag{4}$$

Next, we define the conjugate prior to eq. (1) which is given by the Gaussian

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \alpha^{-1} I), \tag{5}$$

where α is the precision parameter of the distribution. In order to compute (b) of eq. (1) we use eqs. (4) and (5) in Bayes rule and get

$$p(\mathbf{w}|\mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{t})}, \tag{6}$$

where we have dropped the explicit dependence on $\mathbf{x}$ to keep the notation uncluttered. Since both likelihood and prior are Gaussian distributions, we know that also the posterior distribution over $\mathbf{w}$ will be a Gaussian. Thus, we can write

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}, \mathbf{S}), \tag{7}$$

where it can be shown that the mean is given by

$$\mathbf{m} = \beta \mathbf{S} \Phi^{\mathrm{T}} \mathbf{t} \tag{8}$$

and the covariance matrix is given by

$$\mathbf{S} = (\alpha \mathbf{I} + \beta \Phi^{\mathrm{T}} \Phi)^{-1}, \tag{9}$$

where $\mathbf{I} \in \mathbb{R}^{(M+1)\times(M+1)}$ denotes the identity matrix. This brings us back to eq. (1) where it remains to to compute the integral. Again, since (a) and (b) are Gaussian distributions, we know that the result will be a posterior predictive distribution will be a Gaussian as well. We define it as

$$p(t|x, \mathbf{t}, \mathbf{x}) = \mathcal{N}(t|\mathbf{m}^{\mathrm{T}} \boldsymbol{\phi}(x), \sigma^2(x)), \tag{10}$$

where it can be shown that

$$\sigma^2(x) = \frac{1}{\beta} + \boldsymbol{\phi}(x)^{\mathrm{T}} \mathbf{S} \boldsymbol{\phi}(x). \tag{11}$$

Thus, the task in this assignment is to compute eq. (10) for a given dataset. Due to the current circumstances regarding Covid-19, we use the official Covid-19 dataset from Austria[1]. It consists of N data points, where the input $\mathbf{x}$ is represents a day and $\mathbf{t}$ represents the number of positively tested people on the corresponding day.

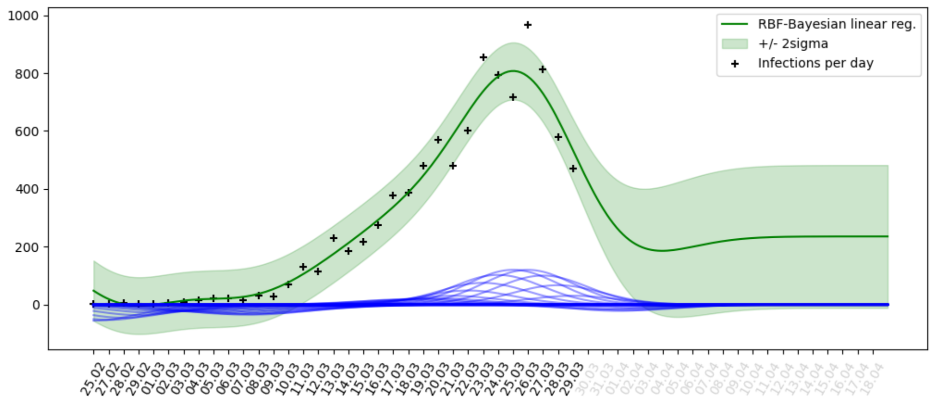

Figure 1: Predictive distribution in Bayesian linear regression for the given Covid-19 dataset. The figure shows the first $K = 33$ data points and the prediction for $P = 20$ days in the future.

As already announced above, we consider a linear model based on Gaussian RBFs. A Gaussian RBF is defined as

$$\phi_m(x) = \exp\left(-\frac{(x - c_m)^2}{2\sigma_r^2}\right), \tag{12}$$

where m is the index, c_m is the center of the m-th basis function and σ_r^2 is the variance of the basis functions. In a multivariate setting we can compactly write this as

$$\mathbf{y} = \mathbf{\Phi} \mathbf{w}, \tag{13}$$

$$\mathbf{\Phi} = \begin{pmatrix} \gamma & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_M(x_1) \\ \gamma & \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_M(x_2) \\ \vdots & \vdots & \vdots & & \vdots \\ \gamma & \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_M(x_N) \end{pmatrix} \in \mathbb{R}^{N\times M+1}, \qquad \mathbf{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_M \end{pmatrix} \in \mathbb{R}^{M+1},$$

where M is the number of RBF kernels, N is the number of data samples and $\gamma = 10^6$ is a large constant.

[1] https://www.data.gv.at/covid-19/

Tasks

We assume that the data-points, *i.e.* the new positive Covid-19 numbers, come in sequentially. This can be modeled in code by simply iterating over the dataset and adding new information in every iterate.

1. Analytically compute eq. (4) by inserting eq. (3) and explicitly compute the log-likelihood $\ln p(\mathbf{t}|\mathbf{w})$.
2. Analytically compute the log-prior of eq. (5), *i.e.* $\ln p(\mathbf{w})$.
3. Using Bayes rule, analytically show how you can compute the log-posterior distribution $\ln p(\mathbf{w}|\mathbf{t})$. Observe that the result can be split into terms independent of $\mathbf{w}$ and terms depending on $\mathbf{w}$.
4. Show that the optimal parameters $\mathbf{w}_{\text{MAP}}$ can be computed in closed form by maximizing the log-posterior given by

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \ \ln p(\mathbf{w}|\mathbf{t}) = \left(\Phi^{\mathrm{T}}\Phi + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} \Phi^{\mathrm{T}}\mathbf{t}. \tag{14}$$

   Show that the optimal solution is given by the mean $\boldsymbol{m}$ of the predictive distribution given in eq. (8).

5. Implement Bayesian Linear regression with RBFs and apply it sequentially to the Covid-19 dataset. Set the number of RBF kernels to $M = N + 2P$ where P denotes the prediction horizon.
6. Make predictions where you look $P = 20$ days into the future, *i.e.* given the data points $\{x_1, \dots, x_K\}$ and $\{t_1, \dots, t_K\}$ make predictions for $\{t_{K+1}, \dots, t_{K+P}\}$ where $K \leq N$. What can you observe?
7. Plot the complete predictive distribution (eq. (10)) together with MAP-solution for $K = \{10, 25, N\}$ data points similar to the plot in fig. 1. Also similar to this figure plot/highlight the area within $\boldsymbol{m}^{\mathrm{T}}\boldsymbol{\phi}(x) \pm 2\sigma$ for each given x. Experiment with the hyper-parameters α, β and σ_r^2 and report your chosen values. How do these parameters influence your model?
8. Add the M weighted RBF kernels to your plot as it is shown in fig. 1.

Implementation details Implement the tasks with Python and name the file `BLR.py`. The data is given in the files `covid_x.npy` and `covid_t.npy`. You can load the data using `np.load()`. Additionally you can use `covid_x_ext.npy` for your prediction plots. Please do not use any other libraries than `numpy` and `matplotlib`.

2 Logistic Regression

The goal of this task is to perform classification with logistic regression[2]. In a classification task we want to classify data points $\mathbf{x}$ into a pre-defined set of discrete classes t. In this task we want to classify the data into two classes $\{-1, +1\}$, which is called a binary classification problem. The probability that a data point $\mathbf{x}$ corresponds to class $t = +1$ is thereby defined as

$$p(t = +1|\mathbf{x}) = \sigma(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}) \tag{15}$$

and since we have a binary classification problem, the probability that the corresponding class $t = -1$ can simply be computed with

$$p(t = -1|\mathbf{x}) = 1 - p(t = +1|\mathbf{x}), \tag{16}$$

where

$$\tilde{\mathbf{x}} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \tag{17}$$

[2]Note that although this model is called logistic *regression*, it is actually performing classification

Algorithm 1: Nesterov Accelerated Gradient Method

```
Input: Choose $\widetilde{\mathbf{w}}^{-1} = \widetilde{\mathbf{w}}^0 \in \mathbb{R}^N$, Compute $L$
Result: $\widetilde{\mathbf{w}}^*$
for $k = 1, 2, \dots$ do
    $\beta^k = \frac{k-1}{k+1}$ // Compute extrapolation factor
    $\bar{\mathbf{w}}^k = \widetilde{\mathbf{w}}^k + \beta^k(\widetilde{\mathbf{w}}^k - \widetilde{\mathbf{w}}^{k-1})$ // Extrapolation step
    $\widetilde{\mathbf{w}}^{k+1} = \bar{\mathbf{w}}^k - \frac{1}{L}\nabla E(\bar{\mathbf{w}}^k)$ // Gradient step
end
```

is the $D+1$-dimensional homogeneous input vector,

$$\widetilde{\mathbf{w}} = \begin{pmatrix} b \\ \mathbf{w} \end{pmatrix} \tag{18}$$

is the the $D+1$-dimensional vector capturing the bias parameter b and the weights $\mathbf{w}$. σ is the sigmoid function defined as

$$\sigma(a) = \frac{1}{1+e^{-a}}, \tag{19}$$

which squashes a scalar input a to the interval $[0,1]$ and thus allows to interpret the output as a probability. Given a data set $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ with corresponding class labels $\mathbf{t} = \{t_1, \dots, t_N\}$ with $t_n \in \{-1, +1\}$, we can define the likelihood function of this dataset as

$$p(\mathbf{t}|\mathbf{x}, \widetilde{\mathbf{w}}) = \prod_{n=1}^{N} \sigma(t_n \widetilde{\mathbf{w}}^T \widetilde{\mathbf{x}}_n). \tag{20}$$

Additionally, we assume a zero-mean Gaussian prior on the parameter vector $\mathbf{w}$ defined as

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, s^2\mathbf{I}), \tag{21}$$

where $\mathbf{I}$ is the identity matrix with size $D \times D$ and s^2 is a hyper-parameter. Note that the prior eq. (21) is only defined on the weights $\mathbf{w}$ and not on the bias parameter b. This is important, because the bias parameter defines the distance of the hyperplane $\widetilde{\mathbf{w}}$ to the origin which should not be penalized. Combining eq. (20) and eq. (21), we get

$$p(\widetilde{\mathbf{w}}|\mathbf{t}, \mathbf{x}) = p(\mathbf{w}) \prod_{n=1}^{N} \sigma(t_n \widetilde{\mathbf{w}}^T \widetilde{\mathbf{x}}_n) \tag{22}$$

denoting logistic regression with a Gaussian prior. Before we can use eq. (22) to perform classification we need to compute optimal parameters $\widetilde{\mathbf{w}}^*$ for our model. This can be done by computing

$$\widetilde{\mathbf{w}}^* \in \arg\max_{\widetilde{\mathbf{w}}} p(\widetilde{\mathbf{w}}|\mathbf{t}, \mathbf{x}). \tag{23}$$

However, in difference to linear regression, we cannot compute $\widetilde{\mathbf{w}}^*$ in closed form, but we need to compute $\widetilde{\mathbf{w}}^*$ iteratively with algorithm 1. In order to avoid numerical problems, we perform the optimization on the negative log-likelihood yielding the optimization problem

$$\widetilde{\mathbf{w}}^* \in \arg\min_{\widetilde{\mathbf{w}}} E(\widetilde{\mathbf{w}}) := -\ln p(\widetilde{\mathbf{w}}|\mathbf{t}, \mathbf{x}), \tag{24}$$

where E is called energy. If we have found optimal parameters $\widetilde{\mathbf{w}}^*$ we can use them to make predictions for previously unseen data using eq. (15). The classification estimate is then given by

$$\hat{t} = \begin{cases} -1 & \text{if } p(t = +1 \mid \mathbf{x}) < 0.5 \\ +1 & \text{else.} \end{cases} \tag{25}$$

Tasks

1. Show analytically that eqs. (15) and (16) can be written compactly as

$$p(t|\mathbf{x}) = \sigma(t\widetilde{\mathbf{w}}^T\widetilde{\mathbf{x}}). \tag{26}$$

2. Analytically compute the negative log-likelihood of eq. (22) which we will need in eq. (24).

3. Analytically compute the gradient $\nabla E(\widetilde{\mathbf{w}})$ of the energy(=negative log-likelihood) defined in eq. (24) by deriving the function with respect to $\widetilde{\mathbf{w}}$.

4. Implement algorithm 1 to find the optimal parameters $\widetilde{\mathbf{w}}^*$. You can compute the Lipschitz constant L used in algorithm 1 with

$$L = \frac{1}{4}\sigma_{\max}\left(\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T\right) + \frac{1}{s^2}, \tag{27}$$

where $\sigma_{\max}(A)$ is the largest singular value of A.

5. 2D point dataset

   a) Generate a linearly separable $2D$ data set with $N = 1000$ samples drawn from Gaussian distributions with parameters $\mu_1 = \left(\begin{smallmatrix}6.5\\2\end{smallmatrix}\right)$, $\mu_2 = \left(\begin{smallmatrix}0.5\\2\end{smallmatrix}\right)$, $\Sigma_1 = \left(\begin{smallmatrix}0.8 & 0\\0 & 0.7\end{smallmatrix}\right)$, $\Sigma_2 = \left(\begin{smallmatrix}1 & 0\\0 & 0.2\end{smallmatrix}\right)$ with corresponding class labels $t_n \in \{-1, +1\}$. You can draw samples from a multivariate normal distribution by using `numpy.random.multivariate_normal`.

   b) Make a scatterplot of the generated $2D$ points. Use a different colour for both classes and include a legend in your plot.

   c) Implement the gradient $\nabla E(\widetilde{\mathbf{w}})$ of the energy and compute the result by applying it on the generated data.

   d) Numerically check your computed gradient by using `scipy.optimize.approx_fprime` as shown in the exemplary file `grad_check.py`.

   e) Train your model for the 2D data set you generated with your implementation of algorithm 1. Include the resulting decision boundary in your previous plot and describe your findings.

   f) Use your trained model to generate a surface plot showing the probabilities $p(t = +1|\mathbf{x})$. This can be done by densely sampling the area of your plot and computing $p(t = +1|\mathbf{x})$ for all sampled points. You can use the function `numpy.meshgrid` to densely sample the points. Plot the result together with the data points and the decision boundary using `matplotlib.pyplot.contour`. Explain what you can you see in this plot.

6. Real world dataset for spam-classification

   a) Import the spambase training and validation data from the TeachCenter. The spambase data[3] contains a collection of spam e-mails, where each e-mail is analyzed based on 57 continuous attributes and 1 nominal class label. The files `spam_train.npy` and `spam_val.npy` thus contain a number of N rows representing the number of e-mails and 58 columns, where the first $D = 57$ columns hold the attributes. The class label can be found in the last column and denotes whether the e-mail was considered spam, *i.e.* $t = +1$ or no spam, *i.e.* $t = -1$. Use the provided `.npy` files for training and validation data, since they have already been preprocessed.

   b) Train the logistic regression model using algorithm 1 on the spambase training data set. Use the computed $\widetilde{\mathbf{w}}$ to classify the spam validation data and calculate the accuracy according to

$$\mathcal{A} = \tfrac{1}{N}\sum_{n=1}^{N}\delta(\sigma(\widetilde{\mathbf{w}}^T\widetilde{\mathbf{x}}_n), t_n), \tag{28}$$

   where δ is the indicator function defined as follows

$$\delta(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{else.} \end{cases}$$

[3]http://archive.ics.uci.edu/ml/datasets/spambase

Report the achieved accuracies on the training set and on the validation set in your report. How does the hyper-parameter s^2 influence the results?

Implementation details Implement the tasks with Python and name the file `logistic_regression.py`. Please do not use any other libraries than `numpy` and `matplotlib`. You can read in the spambase training and validation data using `numpy.load('spam_train.npy')`.