Assignment 2

Machine Learning 2

Lea Bogensperger, `lea.bogensperger@icg.tugraz.at`
Benedikt Kantz, `benedikt.kantz@student.tugraz.at`

April 30, 2024

Deadline: May 21, 2024 at 23:55h.
Submission: Upload your report (`report.pdf`), your implementation (`features_and_kernels.py`) and your figure file (`figures.pdf`) to the TeachCenter. Please do not zip your files. Please use the provided framework-file for your implementation.

1 Gaussian Kernel Approximation (8P)

In machine learning, we can make use of feature transform $\varphi(\mathbf{x})$ to transform our D-dimensional data $\mathbf{x}_i \in \mathbb{R}^D$. To deal with infinite-dimensional representations, we can instead directly work with kernels $k(\mathbf{x}, \mathbf{x}')$ which can be constructed by

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}}.$$

In [1] it was proposed to tackle this inner product by using approximations $z : \mathbb{R}^D \to \mathbb{R}^R$ which establishes the following:

$$k(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle_{\mathcal{V}} \approx z(\mathbf{x})^T z(\mathbf{x}'). \tag{1}$$

1.1 Random Fourier Features

The authors in [1] show that random Fourier feature transforms correspond to a Gaussian kernel. For realizations $\{\boldsymbol{\omega}_r\}_{r=1}^R$ with $\boldsymbol{\omega}_r \sim \mathcal{N}^D(\mathbf{0}, \mathbf{I})$ and $\{b_r\}_{r=1}^R$ with $b_r \sim \mathcal{U}_{[0,2\pi]}$ the feature transform is constructed using

$$z_{\boldsymbol{\omega}_r}(\mathbf{x}) = \sqrt{\tfrac{2}{R}} \cos(\boldsymbol{\omega}_r^T \mathbf{x} + b_r). \tag{2}$$

Then, the corresponding Gauss kernel is given by

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2}),$$

which will be shown in the lecture. The goal of this task is to show this qualitatively.

Tasks

1. Explicitly state the dimensions of the components $\boldsymbol{\omega}_r$ and b_r that constitute the random Fourier transform.
2. You are given $N = 1000$ data points $\mathbf{x}_i \in \mathbb{R}^D$ with $D = 2$. Implement the left and right side of (1), where you compare different choices for $R \in \{1, 10, 100, 1000\}$ for the right side. Compute the kernel matrix $\mathbf{K}$ where each element $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j')$ for all $i = 1, \ldots, N$ and $j = 1, \ldots, N$ such that you can generate a 2D plot using `matplotlib.imshow` of your kernel. Create the same plot for your feature approximations for each R.
3. Discuss your results.

1.2 Random Gauss Features

Interestingly, also random Gauss features $\varphi_{\mathbf{t}}(\mathbf{x})$ for $\mathbf{t} \sim \mathcal{U}^D_{[a,b]}$ correspond to a Gauss kernel $k(\mathbf{x}, \mathbf{x}')$. The feature transform is

$$\varphi_{\mathbf{t}}(\mathbf{x}) = \exp\big(-\frac{\|\mathbf{x}-\mathbf{t}\|^2}{2\sigma^2}\big),$$

which can be discretized to

$$z_{\mathbf{t}}(\mathbf{x}) = \sqrt{\tfrac{1}{R}}\exp\big(-\frac{\|\mathbf{x}-\mathbf{t}\|^2}{2\sigma^2}\big). \tag{3}$$

The corresponding kernel is then given by

$$k(\mathbf{x}, \mathbf{x}') = c_1 c_2 \exp\big(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{4\sigma^2}\big), \tag{4}$$

where c_1 is a constant that is related to $\mathbf{t} \sim \mathcal{U}^D_{[a,b]}$. Moreover, c_2 is a constant that appears from the normalization constant when taking the integral over an unnormalized Gaussian distribution. In general, the starting point to compute the kernel from the feature transform is given by

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{t}}\big[\langle\varphi_{\mathbf{t}}(\mathbf{x}), \varphi_{\mathbf{t}}(\mathbf{x}')\rangle\big] = \int_{-\infty}^{\infty} p(\mathbf{t})\ \langle\varphi_{\mathbf{t}}(\mathbf{x}), \varphi_{\mathbf{t}}(\mathbf{x}')\rangle\ d\mathbf{t}. \tag{5}$$

Tasks

1. Compute the kernel in (5). Start by assuming that since $p(\mathbf{t})$ is a uniform distribution, it can be merged into a constant term c_1 and then combine the two exponential terms. Then, re-formulate the resulting expression within the integral into the form of $g(\mathbf{x}, \mathbf{x}')f(\mathbf{t}, \mathbf{x}, \mathbf{x}')$ by bringing $f(\mathbf{t})$ into the form of a Gaussian distribution in $\mathbf{t}$. *Hint:* use a quadratic expansion to obtain these expressions.
   Then, the constant c_2 is the normalization constant of that Gaussian $f(\mathbf{t}, \mathbf{x}, \mathbf{x}')$.

2. Again, implement the computed kernel for the left side and the feature transform with $R \in \{1, 10, 100, 1000\}$ for the right side of (1) for the given data. You can set $c_1 = c_2 = 1$ and $\sigma = 1$. Note that although we use the assumption $\mathbf{t} \sim \mathcal{U}^D_{[a,b]}$, the approximation is better if you sample $\mathbf{t}$ from the data samples $\mathbf{x}_i$ as they are distributed sparsely.

3. Discuss your results.

Implement both subtasks in `task1` of `features_and_kernels.py`.

2 Least Squares Regression (12P)

The goal in this subtask is to now use the random feature transforms from the previous task to solve a linear regression problem in a least squares setting. Assume we again have input data $\mathbf{x} = \{\mathbf{x}_i\}$, with $i = 1, \dots, N$ and $\mathbf{x}_i \in \mathbb{R}^D$ D-dimensional samples. Moreover, you are provided associated targets $\mathbf{y} = \{y_i\}$ with $y_i \in \mathbb{R}$. The goal is to model the underlying relation between training input data $\mathbf{x}$ and targets $\mathbf{y}$ and apply this to new, unseen test data by minimizing the regularized least squares error function

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^R} \frac{1}{2}\sum_{i=1}^{N}\Big(\boldsymbol{\theta}^T z(\mathbf{x}_i) - y_i\Big)^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2. \tag{6}$$

Again, the non-linear feature transform $z(\cdot)$ lifts the input data $\mathbf{x}_i$ to a higher-dimensional space with R features. We are using both the random Fourier features from (2) and random Gauss features from (3). The additional regularization term is balanced by $\lambda \in \mathbb{R}^+$ and penalizes large parameters $\boldsymbol{\theta}$. Evaluation metrics for predictions $\hat{\mathbf{y}}$ are computed using the mean squared error:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{7}$$

Tasks

1. Rewrite eq. (6) in pure matrix/vector notation, such that there are no sums left in the final expression. Use $\Phi = z(\mathbf{x})$ for the feature transform which can be computed prior to the optimization. Additionally, state the matrix/vector dimensions of all occurring variables.
2. Analytically derive the optimal parameters $\boldsymbol{\theta}^*$ from eq. (6).
3. Give an analytic expression to compute predictions $\hat{\mathbf{y}}$ given $\boldsymbol{\theta}^*$.
4. A training set with $N = 200$ data samples and a test set with $N_t = 100$ data samples with $D = 5$ is already provided for you. Implement the computation of $\boldsymbol{\theta}^*$ from the training data.
5. Carefully choose the hyperparameter $\lambda \in \mathbb{R}^+$ and explain your choice. What can you say about its influence?
is taking 0 for R really valid?
6. For both feature transforms: run the experiment for a number of feature vectors $R = \{0, 1, 2, ..., 100\}$ and save the training and test loss in each run. Further, compute the mean squared error denoted in eq. (7) for both the training and test data based on the optimal parameters $\boldsymbol{\theta}^*$. Repeat each experiment 5 times to obtain an averaged score on your metrics.
7. For both feature transforms: plot both the averaged (over the 5 runs) train and test errors depending on the number of feature vectors R in the same plot. Include the standard deviation of each setting in addition to the mean loss. Give an interpretation of your results and discuss the performance of both feature transforms.

Implementation details

- If you want to plot the standard deviation $\pm\sigma$ in addition to an averaged curve use `matplotlib.pyplot.fill_between`, where the parameters `y1` and `y2` denote $\mu - \sigma$ and $\mu + \sigma$, respectively. You can set an `alpha` value for blending.

Implement the least squares regression in `task2` of `features_and_kernels.py`.

3 Dual Representation (5P)

The linear least squares problem from Task 2 can be reformulated in its dual representation, where an equivalent solution can be obtained. Thus, the corresponding dual problem is given by

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}\in\mathbb{R}^N} \frac{1}{2}\boldsymbol{a}^T\mathbf{K}\boldsymbol{a} + \frac{\lambda}{2}\|\boldsymbol{a} + \boldsymbol{y}\|_2^2, \tag{8}$$

using the kernel matrix $\mathbf{K} = \Phi\Phi^T \in \mathbb{R}^{N\times N}$. Having knowledge on either the feature transform $z(\mathbf{x})$ or the corresponding kernel $k(\mathbf{x}, \mathbf{x}') \approx z(\mathbf{x})^T z(\mathbf{x}')$ allows us to operate very flexible in either the primal or the dual domain. The corresponding kernels to the random Fourier and Gauss features were already computed in Task 1.

Hence, similar to Task 2 the dual solution can be obtained in closed-form and can be subsequently used to make predictions for unseen test data $\mathbf{x}$. The relation between the primal solutions $\boldsymbol{\theta}$ required for making new predictions and the dual variable $\mathbf{a}$ is as follows:

$$\boldsymbol{\theta} = -\frac{1}{\lambda}\Phi^T\boldsymbol{a}. \tag{9}$$

Tasks

1. Analytically compute the optimal parameters $\boldsymbol{a}^*$ from (8). State the dimension of the resulting matrix that has to be inverted in the process and compare them those required in Task 2. When is it favourable to use the primal and when the dual solution?
2. Give an analytic expression to compute predictions $\hat{\mathbf{y}}$ given $\boldsymbol{a}^*$ using eq. (9), such that you only rely on $\mathbf{K}$ and do not need to compute the features Φ explicitely.
3. For the train data $\mathbf{x}$ implement both kernel matrices (corresponding to each the random Fourier and Gauss features). Repeat the same process for the test data, ensuring that the resulting kernel matrices are of dimensionality $\mathbb{R}^{N\times N}$ and $\mathbb{R}^{N_t\times N}$, respectively.

4. Implement the computation of $\boldsymbol{a}^*$ and report the mean squared error on the train and test data, using the same $\lambda \in \mathbb{R}^+$ that you have chosen in the previous task. Compare train and test errors obtained with the primal solution for each setting of R with the dual solution.

Implement the dual problem also in `task2` of `features_and_kernels.py`.

References

[1] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.