General constraints for code submissions Please adhere to these rules to make our and your life easier! We will deduct points if your solution does not fulfill the following:

- If not stated otherwise, we will use exclusively Python 3.5.
- If not stated otherwise, we expect a Python script, which we will invoke exactly as stated on the exercise sheet.
- Your solution exactly returns the required output (neither less nor more) – you can implement a `--verbose` option to increase the verbosity level for developing.
- Add comments and docstrings, so we can understand your solution.
- (If applicable) The `README` describes how to install requirements or provides addition information.
- (If applicable) Add required additional packages to `requirements.txt`. Explain in your `README` what this package does, why you use that package and provide a link to it's documentation or GitHub page.
- (If applicable) All prepared unittests have to pass.
- (If applicable) You can (and sometimes have to) reuse code from previous exercises.

Now that you have learned about the theory behind GPs, you will have to use that theory to implement GPs yourself.

1. **Gaussian Processes** [6 points]

   The exercise is mostly concerned with equations 2.11 and 2.12 in chapter 2 of "Gaussian Processes for Machine Learning[1]"

   (a) Equation 2.11 gives the predictive distribution using an explicit feature space formulation as: [2pt.]

   $$f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\frac{1}{\sigma_n^2}\phi(\boldsymbol{x}_\star)^\intercal A^{-1}\Phi\boldsymbol{y}, \phi(\boldsymbol{x}_\star)^\intercal A^{-1}\phi(\boldsymbol{x}_\star))$$

   where $\Phi = \phi(\boldsymbol{X})$ and $A = \sigma_n^{-2}\Phi\Phi^\intercal + \Sigma_p^{-1}$.
   Equation 2.12 is an alternative formulation that requires only an inversion of an $n \times n$ matrix instead of an $N \times N$ one, where n is the number of data points and N the number of features. Equation 2.12 is given as:

   $$f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\phi_\star^\intercal\Sigma_p\Phi(\Phi^\intercal\Sigma_p\Phi + \sigma_n^2 I)^{-1}\boldsymbol{y}, \phi_\star^\intercal\Sigma_p\phi_\star - \phi_\star^\intercal\Sigma_p\Phi(\Phi^\intercal\Sigma_p\Phi + \sigma_n^2 I)^{-1}\Phi^\intercal\Sigma_p\phi_\star)$$

   Show the equivalence of 2.11 and 2.12.
   (*It suffices to show that* $\frac{1}{\sigma_n^2}\phi(\boldsymbol{x}_\star)^\intercal A^{-1}\Phi\boldsymbol{y} = \phi_\star^\intercal\Sigma_p\Phi(\Phi^\intercal\Sigma_p\Phi + \sigma_n^2 I)^{-1}\boldsymbol{y}$).
   Try to apply the following rules wherever possible and clearly state at each step what you did.

Rules	
(1)	$(AB)^{-1} = B^{-1}A^{-1}$
(2)	$A(BC) = (AB)C$
(3)	$I = BB^{-1} = \Sigma_p\Sigma_p^{-1} = (\Phi\Phi^\intercal)(\Phi\Phi^\intercal)^{-1}$

   With I the identity matrix. *Please tex your solution.*

   (b) Your second task is to implement 2.11 to predict the mean and variance given some observations. [2pt.]
   We provide a function prototype that takes the observations, a function ϕ, and an array of points, $[\mathbf{x}_1, \mathbf{x}_2, \dots]$, as arguments. Compute the mean μ and the variance σ^2 at all $\mathbf{x}$. For matrix inversion you can use `numpy.linalg.inv`. Use the data provided in your source folder to create a plot that shows the mean and the 2σ confidence interval around it[2] for a feature space of size $N = 2$. Use $\sigma_n = 1$ and $\Sigma_p = I$ and

   $$\phi(x) = (1, x)^T$$

   which corresponds to Bayesian *linear* regression for one dimensional input.

[1] `http://www.gaussianprocess.org/gpml/chapters/RW2.pdf`
[2] You can use `matplotlib.pyplot.fill_between` to generate the confidence interval.

(c) Finally, implement a second function based on 2.12. Convince yourself, that both yield the same values, by checking the output for the given input[3]. Compare the time it takes for both implementations to compute the output for the given data, and points $\mathbf{x}_i$, for a growing number of features. Again use $\sigma_n = 1$ and $\Sigma_p = I$ and [2pt.]

$$\phi_n(x) = (1, x, x^2, \ldots, x^{n-1})^T \tag{1}$$

to plot the computing time for $n = 2, 4, 8, \ldots, 2048, 4096$.

2. **Feedback** [Bonus: 0.5 points]

For each question in this assignment, state:

- How long you worked on it.
- What you learned.
- Anything you would improve in this question if you were teaching the course.

This assignment is due on 07.06.19 (10:00). Submit your solution for the tasks by uploading a PDF to your groups BitBucket repository. The PDF has to include the name of the submitter(s).

[3]By the nature of numerical calculations, the results will not be identical, but the difference will be very small. Use `numpy.allclose` to test for equality.