

# Advanced Machine Learning (GR5242)

Fall 2019

## Homework 1

Due: Wednesday 25 September, at 4:10pm on courseworks/canvas (for both sections of the class)

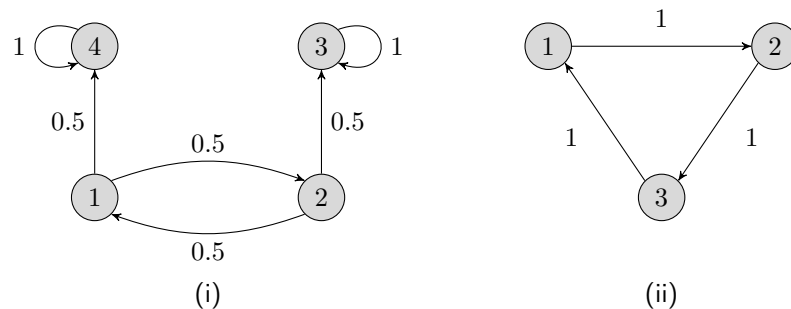
**Homework submission:** Please submit your homework by publishing a notebook that cleanly displays your code, results, and plots to pdf or html.

For conciseness, we assume that you have imported the `numpy` package as `np`, and the `matplotlib.pyplot` package as `plt`.

### Problem 1 (Markov Chains)

*This is a former exam problem. Homeworks will include some problems from exams in previous years to give you an idea of what to expect in the midterm.*

- a) Does either of the following two Markov chains have an invariant distribution? If so, is it unique? Please explain your answers.



- b) The *gambler's fallacy* is the belief among gamblers that, when playing a game of chance for money, an “unlucky streak” will be followed by luck. Suppose that was true and a gambler could be lucky, unlucky or neutral. If the outcomes of  $n$  successive games are variables  $X_1, \dots, X_n$ , does the gambler's fallacy constitute an i.i.d. model, a Markov model, or a hidden Markov model for the sequence  $X_1, \dots, X_n$ ? Please explain your answer.

### Problem 2 (Sampling a $d$ -dimensional Gaussian distribution)

We consider the problem of sampling a multivariate normal (Gaussian) distribution. The `np.random` library provides a function called `randn`, which by default produces pseudo-random samples of an i.i.d normal. We wish to produce samples from a Gaussian  $N(\mu, \Sigma)$  with arbitrary parameters  $\mu$  and  $\Sigma$ , so we have to transform the sample in a suitable manner.

Our approach is based on the eigenvalue structure of symmetric matrices: The eigenvectors of a full-rank symmetric matrix form an orthogonal basis of the underlying vector space. With respect to this basis, the matrix is diagonal, with the eigenvalues as diagonal entries. Denote this diagonal matrix of eigenvalues  $D$  and the matrix describing the change of basis  $V$ . Then

$$\Sigma = VDV^{-1}$$

$V$  is orthogonal (since it describes a change of basis between two orthonormal bases), so  $V^{-1} = V^T$  and  $\Sigma = VDV^T$ . This representation of  $\Sigma$  is called the Schur decomposition. We can produce a sample from a

normal distribution with parameters  $(\mu, \Sigma)$  by drawing a sample vector  $g$  from  $(0, 1)$  using `multivariate_normal`, changing basis, and adding the expectation vector:

$$\tilde{g} = V\sqrt{D}g + \mu$$

As you will recall from linear algebra,  $\sqrt{D} = \text{diag}(\sqrt{D_{ii}})$ .

- a) Implement a function to produce  $n$  draws from a  $d$ -dimensional Gaussian.

---

```
def GSAMPLE(mu, Sigma, n):
    # Your code here
    return(G) # an n by d matrix of d-dim gaussian vectors
```

---

The output should be an `np.ndarray` of size  $(n, d)$ . (The dimension  $d$  is implicitly specified by the size of the arguments `mu` and `Sigma`.)

- b) Verify that your function works as intended by applying the functions `np.mean` and `np.cov` to a sufficiently large sample generated by your program. They should approximately reproduce your input parameters.

- c) Produce 100 samples each in two and three dimensions, using the parameter values  $\mu = (1, 1)^T$ ,  $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$

and  $\mu = (1, 1, 1)^T$ ,  $\Sigma = \begin{pmatrix} 5 & 2 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ , respectively. Plot your results using the `plt.scatter` function.

Hint: For the 3D plot you will find it helpful to include the line

```
from mpl_toolkits.mplot3d import axes3d
```

with the rest of your imports and create your plot axes with the line

```
ax = fig.add_subplot(111, projection='3d').
```

### Problem 3 (Sample size)

This problem visualizes the effects of small sample size. We will draw a number  $n$  of sample points from a one-dimensional Gaussian, sort them into a histogram, and see how stable the result is with respect to different samples. The basic procedure is the following:

- Choose a number  $n$  of sample points and a number  $N_{\text{bins}}$  of histogram bins.
- Use the python function `np.random.randn` to draw  $n$  samples from a one-dimensional Gaussian distribution ( $\mu = 0, \sigma = 1$ ).
- Turn the complete data sample into a histogram using the function `plt.hist`

Please complete the following steps:

- Produce four histograms with  $n = 100$  and  $N_{\text{bins}} = 10$ . Plot the histograms using the `plt.hist` function.
- Repeat the procedure with  $n = 100000$ .
- For  $n = 100000$ , plot one histogram each for  $N_{\text{bins}} \in \{10, 100, 1000\}$ . (matplotlib lets you display multiple plots within a single figure using the `plt.subplot` function.)
- Finally, choose  $n = 100$  and  $N_{\text{bins}} = 1000$ .
- Give a brief discussion of the results. Remember, this is about sample size and reliability of estimates.

Please submit your plots and discussion. Make sure that plots are adequately labeled.