# Re-exam 20th of August 2021 for the course 1MS041 (Introduction to Data Science)

1. Fill in your anonymous exam code in the cell below.
2. Complete the Problems by following instructions. COMMENT your code to explain every detail.
3. When done, submit this file with your solutions saved, as instructed on Studium.

Any questions regarding the exam can be asked by sending a message through Studium to the teacher.

In [0]:
```
# Enter your anonymous exam id by replacing XXXX in this cell below
# do NOT delete this cell
MyAnonymousExamID = 'XXX'
total_points = 0.
```

# Problem 1 [10p]

In many areas of data science and machine learning we need to produce random samples in different ways. This can be done to compute difficult integrals or validate algorithms. In this problem you will be asked to basically

1. [2p] Implement a Linear Congruential Generator that produces pseudo random numbers from the uniform distribution $[0, 1]$.
2. [2p] Use that to construct samples from the uniform distribution over the unit ball in $100$ dimensions.
3. [2p] Estimate $V[X]$ using $1000$ samples.
4. [4p] Use the bootstrap method to produce a bootstrap confidence interval for $V[X]$ of $95\%$ from your $1000$ samples above. Implement your own bootstrap sampler using your implemented sampler from 1. (Max score 1.5 if you dont use your own sampler)

## Problem 1.1

In [0]:
```
def uniform_pseudo_random(n_samples):
    # Implement a linear congruential generator to sample from the uniform [0,1] distri
    return XXX # A list of numbers of length n_samples
```

In [0]:
```
total_points += 0
```

## Problem 1.2

In [0]:
```
def sampler_problem_1(n_samples):
    # Fill this function with whatever you need to sample from the distribution given i
    # problem statement
    return XXX # A list of numbers of length n_samples
```

In [0]:
```
total_points += 0
```

## Problem 1.3

In the below, put the code in to compute an estimate of $V[X]$ using $1000$ samples.

In [0]:
```
# Replace ??? below with the code necessary for the problem

# Since you are going to use the samples again, you can store them in
# the samples variable
samples = ???

# Use the samples variable and compute an estimate of E[X]
V_X = ???
```

In [0]:
```
total_points += 0
```

## Problem 1.4

As you might have noticed, I like the bootstrap. Now its time for you to produce a bootstrap confidence interval around $V[X]$ above, put the code in the cell below. Implement bootstrap from scratch. Use the sampler you produced in Problem 1.1 for everything random.

In [0]:
```
total_points += 0
```

# Problem 2 [10p]

Sometimes it is important to regress on time to event data, that is $Y_i$ corresponds to the time to an event of something, taking values $[0, \infty)$. A reasonable distribution for time to event data is the Exponential distribution, that is we could consider

$$Y_i \mid X_i \sim \text{Exponential}(\lambda(X_i)), \text{ where } \lambda(X_i) = G(\beta_0 + \beta_1 X_i)$$

where $G(x) = e^x$. Recall that a random variable $X \sim Exponential(\lambda)$ if its probability density function is:

$$f(x; \lambda) = \lambda \exp(-\lambda x), \quad \lambda > 0, x \geq 0$$

1. [4p] Implement the `ExponentialRegression` class below by
   A. Writing down the conditional likelihood as the loss
   B. Writing the code to perform multidimensional optimization of the loss to find the parameters of the model.
2. [2p] Load the data from the file `regression.csv`, the header contains two columns, X and Y.
3. [2p] Use the loaded data to fit the `ExponentialRegression` that you just implemented.
4. [2p] Did it converge? If not, then why? If it did, how small did the loss get?

## Problem 2.1

Put your code in the box below for part 1.

In [0]:
```
class PoissonRegression(object):
    def __init__(self,lam=0,max_iter=10000):
        self.coeffs = None
        self.result = None
        self.lam = lam # The value for lambda in the penalization
        self.max_iter = max_iter

    def fit(self,X,Y):
```

```
        import numpy as np
        from scipy import optimize

        # define the objective/cost/loss function we want to minimise
        def f(x):
            return XXX+self.lam*np.sum(x**2)

        #Use the f above together with an optimization method from scipy
        #to find the coefficients of the model

        self.coeffs = XXX



    def predict(self,X):
        #Use the trained model to predict Y
        if (self.coeffs is not None):
            return XXX

    def loss(self,X,Y):
        # Return the value of the loss of the trained model based on X and Y
        return XXX
```

In [0]:
```
total_points += 0
```

## Problem 2.2

Put your code in the box below for part 2.

In [0]:
```

```

In [0]:
```
total_points += 0
```

## Problem 2.3

Put your code in the box below for part 3

In [0]:
```

```

In [0]:
```
total_points += 0
```

## Problem 2.4

Put your code and explanation below for part 4

In [0]:
```
#-------------- Put your explanation between these lines------

#-------------------------------------------------------------
# Put the necessary code below this line

beta = ??
```

In [0]:
```
total_points += 0
```

# Problem 3 [10p]

You now get to solve this fun problem.

1. [2p] Take the file a_sequence.txt and load it as a list of integers. Use bash or something to figure out how to parse the file.
2. [2p] Define a Markov chain from this list of integers
   A. What are the states?
3. [4p] Estimate the transition matrix using maximum likelihood estimator
4. [2p] Find the steady state distribution.

## Problem 3.1

Read the file a_sequence.txt and load it as a list of integers

In [0]:
```
numbers = XXX
```

In [0]:
```
total_points += 0
```

## Problem 3.2

Construct a Markov chain of this list of integers, that is. EXPLAIN in text below what this means, explain what the states are and what the transition probabilities mean.

In [0]:
```
#---------Put your explanation between the lines-------------

#------------------------------------------------------------
```

In [0]:
```
# number of states
n_states = XXX

# states, stored as a sorted set of states
states = XXX
```

In [0]:
```
total_points += 0
```

## Problem 3.3

Estimate the transition matrix, which should be of the form `n_state` x `n_states`.

In [0]:
```
# The transition matrix
P = XXX
```

In [0]:
```
total_points += 0
```

## Problem 3.4

What is the steady state, i.e. what is the fixed point vector when you perform

$$P^k v$$

for a large number $k$? What is the interpretation of this?

In [0]:
```
#---------Put your explanation between the lines-------------

#------------------------------------------------------------
```

In [0]:
```
fixed_point = XXX
```

In [0]:
```
total_points += 0
```

# Problem 4 [10p]

In the exam assignment, there is a csv file called `data.csv`. The first line is the header.

1. [2p] Download the file and load it such that it is stored in a numpy array. It should be a numpy array of shape 400x4096.
2. [2p] First task is to implement the power method to compute the first singular value and singular vector of this dataset.
3. [2p] Compute the second singular value and component using the power method.
4. [2p] Perform full singular value decomposition using for instance numpy.
5. [2p] Compare the results from the power method and numpy.

## Problem 4.1

Put your code in the cell below

In [0]:
```

```

In [0]:
```
total_points += 0
```

## Problem 4.2

Put your code in the cell below

In [0]:
```

```

In [0]:
```
total_points += 0
```

## Problem 4.3

Put your code in the cell below

In [0]:
```

```

In [0]:
```
total_points += 0
```

## Problem 4.4

Put your code in the cell below

In [0]:
```
```

In [0]:
```
total_points += 0
```

## Problem 4.5

Put your code in the cell below

In [0]:
```
```

In [0]:
```
total_points += 0
```

# Total score

In [0]:
```
print("Your total score is: %.2f" % total_points)
```