Exam2021_June

June 18, 2021

1 Re-exam 14th of June 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.

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

2 Problem 1 [8p]

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. Implement a pseudo number generator that produces random numbers from the uniform distribution $[0, 1]$.
2. Use that to construct samples from

$$F[x] = \begin{cases} 0, & x \leq 0 \\ \sin(x), & 0 < x < \pi/2 \\ 1, & x \geq \pi/2 \end{cases}$$

3. Consider a random variable $X \sim F$ sampled from distribution F, and use your sampler to estimate $E[X]$ using 1000 samples.
4. Use the bootstrap method to produce a bootstrap confidence interval for $E[X]$ of 95% from your 1000 samples above. How do you produce the bootstrap samples from your uniform sampler in step 1?

```
[ ]: def makeEDFHidden(myDataList, offset=0):
         '''Make an empirical distribution function from a data list.
```

```
    Param myDataList, list of data to make ecdf from.
    Param offset is an offset to adjust the edf by, used for doing confidence␣
↪bands.
    Return list of tuples comprising (data value, cumulative relative␣
↪frequency) ordered by data value.'''

    sortedUniqueValues = sorted(list(set(myDataList)))
    freqs = [myDataList.count(i) for i in sortedUniqueValues]
    from pylab import cumsum
    cumFreqs = list(cumsum(freqs)) #
    cumRelFreqs = [ZZ(i)/len(myDataList) for i in cumFreqs] # get cumulative␣
↪relative frequencies as rationals
    if offset > 0: # an upper band
        cumRelFreqs = [min(i ,1) for i in cumRelFreqs] # use a list␣
↪comprehension
    if offset < 0: # a lower band
        cumRelFreqs = [max(i, 0) for i in cumFreqs] # use a list comprehension
    return list(zip(sortedUniqueValues, cumRelFreqs))

def ecdfPlot(samples):
    '''Returns an empirical probability mass function plot from samples data.'''
    ecdf_pairs = makeEDFHidden(samples)
    ecdf = point(ecdf_pairs, rgbcolor = "red", faceted = false, pointsize="20")
    for k in range(len(ecdf_pairs)):
        x, kheight = ecdf_pairs[k]      # unpack tuple
        previous_x = 0
        previous_height = 0
        if k > 0:
            previous_x, previous_height = ecdf_pairs[k-1] # unpack previous␣
↪tuple
        ecdf += line([(previous_x, previous_height),(x, previous_height)],␣
↪rgbcolor="grey")
        ecdf += points((x, previous_height),rgbcolor = "white", faceted = true,␣
↪pointsize="20")
        ecdf += line([(x, previous_height),(x, kheight)], rgbcolor="grey",␣
↪linestyle=":")
    # padding
    ecdf += line([(ecdf_pairs[0][0]-0.2, 0),(ecdf_pairs[0][0], 0)],␣
↪rgbcolor="grey")
    max_index = len(ecdf_pairs)-1
    ecdf += line([(ecdf_pairs[max_index][0],␣
↪ecdf_pairs[max_index][1]),(ecdf_pairs[max_index][0]+0.2,␣
↪ecdf_pairs[max_index][1])],rgbcolor="grey")
    return ecdf
```

2.1 Problem 1.1

Run the above code cell to have the tools available to plot empirical distributions. Then fill in the code needed to solve the problem in the two functions * `uniform_pseudo_random(n_samples)` to generate a list of length `n_samples` of random numbers in the interval $(0, 1)$. You can test your result using the ecdfPlot.

```
[ ]: def uniform_pseudo_random(n_samples):
         # Implement a pseudo number generator, you are free to choose any
         # as long as you implement it yourself and dont use the packages.
         return XXX # A list of numbers of length n_samples
```

2.2 Problem 1.2

After having implemented the above you can use that to produce samples from F in the description of the problem. Again compare the empirical distribution with the true distribution as a sanity check.

```
[ ]: def sampler_problem_1(n_samples):
         # Fill this function with whatever you need to sample from this function
         return XXX # A list of numbers of length n_samples
```

2.3 Problem 1.3

In the below, put the code in to compute an estimate of $E[X]$ using `sampler_problem_1` using 1000 samples.

```
[ ]: # 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]
     E_X = ???
```

2.4 Problem 1.4

As you might have noticed, I like the bootstrap. Now its time for you to produce a bootstrap confidence interval around $E[X]$ above, put the code in the cell below. You are basically free to write whatever code you like to solve the problem, one hint would be to use the code from the lectures, but it needs to be modified to use your sampler from problem 1.1.

```
[ ]:
```

3 Problem 2 [8p]

Let $X_1, \ldots, X_n \sim N(\theta, 1)$. Define

$$Y_i = \begin{cases} 1, & X_i > 0 \\ 0, & X_i \leq 0 \end{cases}$$

Let $\psi = \mathbb{P}(Y_1 = 1)$. * Find the maximum likelihood estimator $\hat{\psi}$ of ψ. * Find an appropriate 95% confidence interval for ψ. * Repeat the above two steps with the variance unknown!

3.1 Problem 2.1

Compute the maximum likelihood of ψ and implement the function below

[ ]:
```
def mle_psi_problem_2(data):
    # Do whatever you need here to produce the MLE of the data
    return XXX #The MLE of psi above
```

3.2 Problem 2.2

Use the below sample data to compute the MLE of ψ and compute the corresponding 95% confidence interval for ψ.

[ ]:
```
prob2_samples = [ 0.88, -0.75, -0.46, -0.13,  0.96,  0.17,  1.24, -1.03, -1.  ,␣
↪1.7 ,  0.34,  1.01,  0.75,  0.58,  0.5 , -1.2 , -1.45,  1.59, 1.79, -1.32]
```

[ ]:
```
mle_psi = mle_psi_problem_2(prob2_samples)
# The left edge of the confidence interval
low_edge_confidence = ??
# The right edge of the condidence interval
high_edge_confidence = ??
```

3.3 Problem 2.3

Compute the maximum likelihood of ψ when the variance is unknown.

[ ]:
```
def mle_psi_problem_2_uv(data):
    # Do whatever you need here to produce the MLE of the data
    return XXX #The MLE of psi above
```

Use the samples from the above data to compute the MLE of ψ and compute the corresponding 95% confidence interval for ψ.

[ ]:
```
mle_psi_uv = mle_psi_problem_2_uv(prob2_samples)
# The left edge of the confidence interval
low_edge_confidence_uv = ??
```

```
# The right edge of the condidence interval
high_edge_confidence_uv = ??
```

4 Problem 3 [8p]

The Gaussian Annulus theorem tells us that the spherical Gaussian distribution in d dimensions concentrates in a spherical shell of radius $\sqrt{d}$. 1. Generate 1000 samples of a spherical Gaussian for dimensions $d = 10, 20, ..., 100$ and for each d estimate

$$P(\sqrt{d} - 1 < |X| < \sqrt{d} + 1)$$

show your result as a plot with the x axis being d and the y axis being the probability. Did you get what you expected? 2. In the statement of the Gaussian annulus theorem (Theorem 2.9 in the book) there is an unknown constant c, based on the above simulation what would your guess for the value c be? 3. Generate 1000 pairs of vectors from a spherical Gaussian for dimensions $d = 2$ and $d = 20$, $d = 200$, $d = 2000$ and for each d compute the angle between the pairs, using these angles, produce a histogram of the angles for each d. Hint. It is enough use the proper dot-product and `arccos`. What conclusion can you make about the angles as a function of dimension? 4. Revisiting step 1, can you, using simulation in the case $d = 100$ find β such that

$$P(\sqrt{d} - \beta < |x| < \sqrt{d} + \beta) \approx 0.99?$$

Hint: is it possible to bootstrap?

4.1 Problem 3.1

Put your code in the box below for part 1.

[ ]:
```

```

4.2 Problem 3.2

Write the answer and a motivation in the cell below

[ ]:
```
# The reason I guess the value of c below is...
c = ??
```

4.3 Problem 3.3

Put your code in the box below for part 3

[ ]:
```

```

4.4 Problem 3.4

Put your code below for part 4

```
[ ]: ###



beta = ??
```

5 Problem 4 [8p]

You now get another chance to solve this fun problem.

1. Take the string `prideAndPrejudiceFirstChapter` and split it by `' '` into a list of "words" and put this in `words`.
2. Consider the list of words as a list of states, precisely as in the `wet - dry` Markov chain that we studied. We model this list of states using a Markov chain, as such there is an associated transition matrix P. The first four words are `['it', 'is', 'a', 'truth']`, if we think of this as a Markov chain we will have transitions from `'it'` to `'is'` for instance, as such there is a $p_{\text{'it','is'}}$, i.e. a transition probability from `'it'` to `'is'`.
3. Your goal is to find the maximum likelihood estimate of P, recall from notebook 13 that for two states we have
$$\widehat{p}_{0,0} = \frac{n_{0,0}}{n_{0,0}+n_{0,1}} \quad \text{and} \quad \widehat{p}_{1,1} = \frac{n_{1,1}}{n_{1,0}+n_{1,1}}$$
there is nothing special about two states, so for arbitrary number of states $i = 1, \ldots, N$ we have
$$\widehat{p}_{i,j} = \frac{n_{i,j}}{\sum_{k=1}^{N} n_{i,k}}$$
4. The order of the indices should be the same as the list `unique_words` i.e. the first word in that list corresponds to $i = 0$, the second $i = 1$ etc.

```
[ ]: # REQUIRED-CELL
     # DO NOT MODIFY this cell
     # Evaluate this cell before trying this PROBLEM so that the required functions␣
      ↪and variables are loaded
     def makeFreqDict(myDataList):
         '''Make a frequency mapping out of a list of data.

         Param myDataList, a list of data.
         Return a dictionary mapping each unique data value to its frequency count.
      ↪'''

         freqDict = {} # start with an empty dictionary

         for res in myDataList:
```

```
        if res in freqDict: # the data value already exists as a key
                freqDict[res] = freqDict[res] + 1 # add 1 to the count using␣
↪sage integers
        else: # the data value does not exist as a key value
            freqDict[res] = 1 # add a new key-value pair for this new data␣
↪value, frequency 1

    return freqDict # return the dictionary created

# end of makeFreqDict(...)

prideAndPrejudiceFirstChapter = '''It is a truth universally acknowledged, that␣
↪a single man in
      possession of a good fortune, must be in want of a wife.

      However little known the feelings or views of such a man may be
      on his first entering a neighbourhood, this truth is so well
      fixed in the minds of the surrounding families, that he is
      considered the rightful property of some one or other of their
      daughters.

      "My dear Mr. Bennet," said his lady to him one day, "have you
      heard that Netherfield Park is let at last?"

      Mr. Bennet replied that he had not.

      "But it is," returned she; "for Mrs. Long has just been here, and
      she told me all about it."

      Mr. Bennet made no answer.

      "Do you not want to know who has taken it?" cried his wife
      impatiently.

      "_You_ want to tell me, and I have no objection to hearing it."

      This was invitation enough.

      "Why, my dear, you must know, Mrs. Long says that Netherfield is
      taken by a young man of large fortune from the north of England;
      that he came down on Monday in a chaise and four to see the
      place, and was so much delighted with it, that he agreed with Mr.
      Morris immediately; that he is to take possession before
      Michaelmas, and some of his servants are to be in the house by
      the end of next week."

      "What is his name?"
```

```
"Bingley."

"Is he married or single?"

"Oh! Single, my dear, to be sure! A single man of large fortune;
four or five thousand a year. What a fine thing for our girls!"

"How so? How can it affect them?"

"My dear Mr. Bennet," replied his wife, "how can you be so
tiresome! You must know that I am thinking of his marrying one of
them."

"Is that his design in settling here?"

"Design! Nonsense, how can you talk so! But it is very likely
that he _may_ fall in love with one of them, and therefore you
must visit him as soon as he comes."

"I see no occasion for that. You and the girls may go, or you may
send them by themselves, which perhaps will be still better, for
as you are as handsome as any of them, Mr. Bingley may like you
the best of the party."

"My dear, you flatter me. I certainly _have_ had my share of
beauty, but I do not pretend to be anything extraordinary now.
When a woman has five grown-up daughters, she ought to give over
thinking of her own beauty."

"In such cases, a woman has not often much beauty to think of."

"But, my dear, you must indeed go and see Mr. Bingley when he
comes into the neighbourhood."

"It is more than I engage for, I assure you."

"But consider your daughters. Only think what an establishment it
would be for one of them. Sir William and Lady Lucas are
determined to go, merely on that account, for in general, you
know, they visit no newcomers. Indeed you must go, for it will be
impossible for _us_ to visit him if you do not."

"You are over-scrupulous, surely. I dare say Mr. Bingley will be
very glad to see you; and I will send a few lines by you to
assure him of my hearty consent to his marrying whichever he
chooses of the girls; though I must throw in a good word for my
```

```
      little Lizzy."

      "I desire you will do no such thing. Lizzy is not a bit better
      than the others; and I am sure she is not half so handsome as
      Jane, nor half so good-humoured as Lydia. But you are always
      giving _her_ the preference."

      "They have none of them much to recommend them," replied he;
      "they are all silly and ignorant like other girls; but Lizzy has
      something more of quickness than her sisters."

      "Mr. Bennet, how can you abuse your own children in such a way?
      You take delight in vexing me. You have no compassion for my poor
      nerves."

      "You mistake me, my dear. I have a high respect for your nerves.
      They are my old friends. I have heard you mention them with
      consideration these last twenty years at least."

      "Ah, you do not know what I suffer."

      "But I hope you will get over it, and live to see many young men
      of four thousand a year come into the neighbourhood."

      "It will be no use to us, if twenty such should come, since you
      will not visit them."

      "Depend upon it, my dear, that when there are twenty, I will
      visit them all."

      Mr. Bennet was so odd a mixture of quick parts, sarcastic humour,
      reserve, and caprice, that the experience of three-and-twenty
      years had been insufficient to make his wife understand his
      character. _Her_ mind was less difficult to develop. She was a
      woman of mean understanding, little information, and uncertain
      temper. When she was discontented, she fancied herself nervous.
      The business of her life was to get her daughters married; its
      solace was visiting and news.'''.lower()

import re
subs = '''_;.,""?!'''
for sub in subs:
    prideAndPrejudiceFirstChapter = prideAndPrejudiceFirstChapter.replace(sub,'␣
 ↪')
prideAndPrejudiceFirstChapter = re.sub('\\s+', '␣
 ↪',prideAndPrejudiceFirstChapter).strip()
```

```
[ ]: # Part 1, find the words by splitting prideAndPrejudiceFirstChapter on ' ',
     # Make sure you ran the cell above before you try this
     words = XXX
     unique_words = sorted(set(words)) # The unique words
     n_words = len(unique_words) # The number of unique words
```

```
[ ]: # Part 2, count the different transitions
     transitions = XXX # A list containing tuples ex: ('it','is') of all transitions␣
      ↪in the text
     transition_counts = XXX # A dictionary that counts the number of each␣
      ↪transition
     # ex: ('it','is'):4
     indexToWord = XXX # A dictionary that maps the n-1 number to the n:th␣
      ↪unique_word,
     # ex: 0:'a'
     wordToIndex = XXX # The inverse function of indexToWord,
     # ex: 'a':0
```

```
[ ]: # Part 3, finding the maximum likelihood estimate of the transition matrix
     import numpy as np

     transition_matrix = XXX # a numpy array of size (n_words,n_words)

     # The transition matrix should be ordered in such a way that
     # p_{'it','is'} = transition_matrix[wordToIndex['it'],wordToIndex['is']]

     # Make sure that the transition_matrix does not contain np.nan from division by␣
      ↪zero for instance
```

Local Test for PROBLEM 5 Use the cell below to evaluate the feasibility of your answer.

```
[ ]: # Once you have created all your functions, you can make a small test here to␣
      ↪see
     # what would be generated from your model.

     start = np.zeros(shape=(n_words,1))
     start[0,0] = 1

     current_pos = start
     for i in range(100):
         random_word_index = np.random.choice(range(n_words),p=current_pos.
      ↪reshape(-1))
         current_pos = np.zeros_like(start)
         current_pos[random_word_index] = 1
```

```
    print(indexToWord[random_word_index],end=' ')
    current_pos = (current_pos.T@transition_matrix).T
```

6 Problem 5 [8p]

In the exam assignment, there is a csv file called `digits.csv`. 1. Download the file and load it such that the digits are stored in a numpy array and the labels in another. The digits are 8x8 bitmaps so flattened it should be a numpy array of shape 1797x64 2. We wish first to perform PCA on this dataset using 2 components. Your task is to first implement PCA using SVD (you can use numpy for that) and then apply it to this dataset. 3. Compute the explained variance for the first two components 4. Apply a suitable model for predicting the label based on the two pca components. 5. What happens if you perform a random projection of the full dataset into $\mathbb{R}^2$ and then repeat the previous prediction model? Can you explain the result?

```
[ ]: # Do all steps in the cell below, I only provided some skeletons but you have␣
      ↪to do the rest
     def load_digits(file_name):

         return XXX # Returns a tuple of numpy arrays one has shape 1797x64 and the␣
      ↪other has shape (1797,)

     def plot_digit(digit):
         # Takes a numpy array of shape (64,) and outputs plots the image as an 8x8␣
      ↪bitmap
         assert digit.shape == (64,)
         import matplotlib.pyplot as plt
         plt.gray()
         plt.imshow(digit.reshape(8,8))
```

```
[ ]:
```