Please provide complete and well-written solutions to the following exercises.

Due October 17, 4PM PST, to be uploaded as a single PDF document to brightspace.

Homework 5

Exercise 1. This exercise deals with sunspot data from the following files (the same data appears in different formats)

txt file csv file

These files are taken from http://www.sidc.be/silso/datafiles#total

To work with this data, e.g. with pandas in Python you can use the command

```
df = pd.read_csv('SN_d_tot_V2.0.csv')
```

to import the .csv file.

The format of the data is as follows.

- Columns 1-3: Gregorian calendar date (Year, Month, then Day)
- Column 4: Date in fraction of year
- Column 5: Daily total number of sunspots observed on the sun. A value of -1 indicates that no number is available for that day (missing value).
- Column 6: Daily standard deviation of the input sunspot numbers from individual stations.
- Column 7: Number of observations used to compute the daily value.
- Column 8: Definitive/provisional indicator. A blank indicates that the value is definitive. A '*' symbol indicates that the value is still provisional and is subject to a possible revision (Usually the last 3 to 6 months)

It is known that the number of sunspots on the sun follows an approximately 11-year sinusoidal pattern. So, if we plot the number of sunspots over several years, the distance between the highest observed numbers of sunspots should be around 11 years.

Let U_t be the number of sunspots at time t, where t is measured in years. We model U_t as

$$U_t = m_t + a\cos(2\pi\theta t) + b\sin(2\pi\omega t)) + Y_t, \qquad \forall\, t \in \mathbf{R},$$

where $a, b, \theta, \omega \in \mathbf{R}$ are unknown (deterministic) parameters, m_t is an unknown deterministic function of t that is assumed to be a "slowly varying" function of t, and $\{Y_t\}_{t\in\mathbf{R}}$ are i.i.d. mean zero random variables. The quantity m_t is called the **trend** and the quantity $s_t := a\cos(2\pi\theta t) + b\sin(2\pi\omega t))$ is called the **seasonal component** of the time series $\{U_t\}_{t\in\mathbf{R}}$.

Since the 11-year sinusoidal pattern is known, we assume for now that $\theta = \omega = 1/11$. Note that

$$\sum_{s=t,t+1/365,t+2/365,\ldots,t+11} \cos(2\pi\theta s) \approx 0, \qquad \sum_{s=t,t+1/365,t+2/365,\ldots,t+11} \cos(2\pi\theta s) \approx 0, \qquad \forall\, t \in \mathbf{R}.$$

So, if m_t is slowly varying in the sense that $m_t \approx \frac{1}{11\cdot 365.25}\sum_{s=t-5.5,t-5.5+1/365,t+2/365,\ldots,t+5.5} m_s$, an unbiased estimator for m_t is

$$M_t := \frac{1}{11\cdot 365.25}\sum_{s=t-5.5,t-5.5+1/365,t+2/365,\ldots,t+5.5} U_s.$$

M_t defined in this way is called a **moving average**.

• Since -1 denotes a missing data value, we should first consider how to fill in missing data values. Let's first use the `ffill` option of `reindex` to fill in these missing values. (Since `ffill` works best when the index consists of increasing integers, you should either convert the first three column entries of a row to a single integer, or you could take the fourth column entry and multiply it by 1000 to get an integer.)

• Plot M_t versus t. Do you observe any fluctuations in M_t or does it seem to be roughly constant? If so, what is this constant?

Once we have the estimate M_t, we can then use the approximation

$$U_t - M_t \approx a\cos(2\pi\theta t) + b\sin(2\pi\omega t)) + Y_t, \qquad \forall\, t \in \mathbf{R},$$

and then try to estimate a, b. A general way to estimate $s_t := a\cos(2\pi\theta t) + b\sin(2\pi\omega t))$ is to use a (smaller) moving average such as

$$S_t := \frac{1}{11}\sum_{s=t-5/365,t-4/365,\ldots,t+5/365} [U_s - M_s].$$

Note that S_t is unbiased.

• Plot S_t versus t. Does it look like a sinusoidal curve? Note that S_t removed the trend from the time series.

Another way to estimate s_t is to estimate the constants a and b directly. By the double angle formula, note that

$$\sum_{s=t,t+1/365,t+2/365,\ldots,t+11} \cos(2\pi\theta s)\sin(2\pi\theta s) = \sum_{s=t,t+1/365,t+2/365,\ldots,t+11} \frac{1}{2}\sin(4\pi\theta s) \approx 0.$$

Also,

$$\frac{1}{365.25}\sum_{s=t,t+1/365,t+2/365,\ldots,t+11} \cos^2(2\pi s/11) \approx \int_0^{11} \cos^2(2\pi x/11)dx \approx 11/2.$$

So, an unbiased estimator for a is

$$A_t := \frac{2}{11\cdot 365.25}\sum_{s=t,t+1/365,t+2/365,\ldots,t+11} (U_s - M_s)\cos(2\pi\theta s), \qquad \forall\, t \in \mathbf{R}.$$

Similarly, an unbiased estimator for b is

$$B_t := \frac{2}{11\cdot 365.25}\sum_{s=t,t+1/365,t+2/365,\ldots,t+11}(U_s-M_s)\sin(2\pi\theta s),\qquad \forall\, t\in\mathbf{R}.$$

• Plot A_t versus t. Plot B_t versus t. Are they close to being constant in t?

• Plot $U_t-[M_t+A_t\cos(2\pi t/11)+B_t\sin(2\pi t/11))]$ versus t. This is the time series with the trend and seasonal components removed. Does this plot look like random fluctuations?

• Our modeling assumptions used a period of 11 for the seasonal component of the time series. Does the data reflect this assumption? For example, would it be more accurate to have $\theta=\omega=1/(10.9)$ in our modeling assumption?

Exercise 2. This exercise will use the following code.

```
data = {
    "state": ["Ohio", "Ohio", "Ohio", "Nevada", "Nevada", "Nevada"],
    "year": [2000, 2001, 2002, 2001, 2002, 2003],
    "pop": [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]
}
frame = pd.DataFrame(data)

populations = {
    "year": {0: 2000, 1: 2002, 3: 2004, 4: 2006},
    "pop": {0: 4, 2: 6, 3: 8, 4: 10}
}
frame2 = pd.DataFrame(populations)

ser = pd.Series([3, 6, 8, 9])

def f1(x):
    return x**2 + 1
```

- Using the `add` function, add `frame` and `frame2` together (e.g. use `df.add(df2)`), and fill in any resulting NaN values to zeros.
- Apply the function `f1` to `frame`. (The syntax is `frame.map(f1)` .)
- For both NBA and WNBA players, answer the following question: Let x denote a player's highest single season 2 point field goal percentage. Let y denote a player's highest single season 3 point field goal percentage. Who has the highest value of $x+y$ (among those listed on both leaderboards)? (The percentage for a single player can be used across two different seasons.) To answer this question, you can find data from the following sites:
  WNBA Leaders
  NBA Leaders

Exercise 3. In class, we examined the MovieLens 1M Dataset available at https://grouplens.org/datasets/movielens/

We then loaded the files with

```
unames = ["user_id", "gender", "age", "occupation", "zip"]
# users format:   UserID::Gender::Age::Occupation::Zip-code
users = pd.read_table(
    "users.dat",
    sep = "::",
    header = None,
    names = unames,
    engine = "python",
    encoding = "latin-1"
)

rnames = ["user_id", "movie_id", "rating", "timestamp"]
# ratings format:    UserID::MovieID::Rating::Timestamp
ratings = pd.read_table(
    "ratings.dat",
    sep="::",
    header = None,
    names = rnames,
    engine = "python",
    encoding = "latin-1"
)

mnames = ["movie_id", "title", "genres"]
# movies format:   MovieID::Title::Genres
movies = pd.read_table(
    "movies.dat",
    sep="::",
    header = None,
    names = mnames,
    engine = "python",
    encoding = "latin-1"
)

data = pd.merge(pd.merge(ratings, users), movies)
```

One main motivating question was: can we predict the user-specified gender using only their movie ratings. We used the following code to answer this question:

```
track_sum = 0
from tqdm import tqdm
for i in tqdm(range(len(users))):
    current_gender = users["gender"][i]
    current_id = i + 1
    current_data = data[data["user_id"] == current_id].set_index("title")
    #current_sorted = current_data.reindex(index = sorted_by_diff.index)
    current_nan = current_data.isna()
    current_data = current_data[~current_data.isna()]
    current_data.dropna(inplace = True)
    #current_data now has all ratings for user i, indexed by title
    f_score = (current_data["rating"] - mean_ratings["F"][current_data.index]) ** 2
    m_score = (current_data["rating"] - mean_ratings["M"][current_data.index]) ** 2
    f_avg = np.mean(f_score)
    m_avg = np.mean(m_score)
    if f_avg < m_avg:
        # then predict F
        gender_predict = "F"
    else:
        gender_predict = "M"

    if current_gender == gender_predict:
        # then prediction was correct
        track_sum += 1

print("Percentage prediction correct:", track_sum / len(users))
```

The output is approximately 75.24...%.

Your task is to improve on this performance. Find a way to modify the above program to get a higher percentage of correct predictions.

If you want you can use an entirely different program or classification procedure, but that should not be necessary to get a higher percentage correct.

(Optional: Repeat the above for a larger MovieLens dataset such as the 10M or 20M datasets.)

Exercise 4. In class, we described a procedure (known as Alternating Least Squares or ALS) for filling in missing movie ratings in the MovieLens 1M dataset. However, we did not discuss how well this procedure works. Using the MovieLens 1M dataframe `data` defined in the previous exercise, recall that we created a matrix of ratings using the commands

```
rows = data["user_id"]
cols = data["movie_id"]
entries = data["rating"]
rating_matrix = np.empty([6040, 3952], dtype = "uint8")
rating_matrix[rows.values - 1, cols.values - 1] = entries.values
rating_matrix = rating_matrix
```

To test how well ALS performs, set the first ten rows of `rating_matrix` equal to zero, perform ALS on the resulting matrix, and see how closely you can recover the first ten rows of the original `rating` matrix. Check the average difference between the original ratings and recovered ratings. Try a few different choices of the parameters `rank`, `iterations` and `reg_param` to try to improve the performance.

(Optional: Repeat the above for a larger MovieLens dataset such as the 10M or 20M datasets.)