Computational Statistics and Data Analysis (MVComp2)

Solutions to exercise 12

Lecturer Tristan Bereau

Semester Wi23/24
Due Feb. 1, 2024, 23:59

1 Second principal component (5 points)

Recall the reconstruction loss for the first principal component

$$J(\boldsymbol{v}_1, \boldsymbol{z}_1) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1)^\top(\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1),$$

which can be used to optimally solve for the principal component, $\boldsymbol{z}_1$.

(a) Extend the loss to the *second* principal component, $J(\boldsymbol{v}_2, \boldsymbol{z}_2; \boldsymbol{v}_1, \boldsymbol{z}_1)$, and show that the solution yields $z_{i2} = \boldsymbol{v}_2^\top \boldsymbol{x}_i$.

(b) Show that the value of $\boldsymbol{v}_2$ that minimizes

$$\tilde{J}(\boldsymbol{v}_2) = -\boldsymbol{v}_2^\top \mathbf{C}\boldsymbol{v}_2 + \lambda_2(\boldsymbol{v}_2^\top\boldsymbol{v}_2 - 1) + \lambda_{12}(\boldsymbol{v}_2^\top\boldsymbol{v}_1 - 0)$$

is given by the eigenvector of $\mathbf{C}$ with the second largest eigenvalue.

Hint: recall that $\mathbf{C}\boldsymbol{v}_1 = \lambda_1\boldsymbol{v}_1$ and $\frac{\partial \boldsymbol{x}^\top \mathbf{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = (\mathbf{A} + \mathbf{A}^\top)\boldsymbol{x}$.

1.1 Solution

(a) We first extend the reconstruction loss

$$J(\boldsymbol{v}_2, \boldsymbol{z}_2; \boldsymbol{v}_1, \boldsymbol{z}_1) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1 - z_{i2}\boldsymbol{v}_2)^\top(\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1 - z_{i2}\boldsymbol{v}_2)$$

and set $\frac{\partial J}{\partial \boldsymbol{z}_2} = 0$. Let's do it for a specific data point, i. First, expand the product

$$\begin{aligned} q(z_{i2}) =& (\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1 - z_{i2}\boldsymbol{v}_2)^\top(\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1 - z_{i2}\boldsymbol{v}_2) \\ =& (\boldsymbol{x}_i^\top - z_{i1}\boldsymbol{v}_1^\top - z_{i2}\boldsymbol{v}_2^\top)(\boldsymbol{x}_i - z_{i1}\boldsymbol{v}_1 - z_{i2}\boldsymbol{v}_2) \\ =& \boldsymbol{x}_i^\top x_i - z_{i1}\boldsymbol{x}_i^\top\boldsymbol{v}_1 - z_{i2}\boldsymbol{x}_i^\top\boldsymbol{v}_2 - z_{i1}\boldsymbol{v}_1^\top\boldsymbol{x}_i + z_{i1}^2\boldsymbol{v}_1^\top\boldsymbol{v}_1 \\ & + z_{i1}z_{i2}\boldsymbol{v}_1^\top\boldsymbol{v}_2 - z_{i2}\boldsymbol{v}_2^\top\boldsymbol{x}_i + z_{i1}z_{i2}\boldsymbol{v}_2^\top\boldsymbol{v}_1 + z_{i2}^2\boldsymbol{v}_2^\top\boldsymbol{v}_2 \end{aligned}$$

and differentiate wrt z_{i2}

$$\frac{\partial q}{\partial z_{i2}} = -\boldsymbol{x}_i^\top \boldsymbol{v}_2 + z_{i1}\boldsymbol{v}_1^\top \boldsymbol{v}_2 - \boldsymbol{v}_2^\top \boldsymbol{x}_i + z_{i1}\boldsymbol{v}_2^\top \boldsymbol{v}_1 + 2z_{i2}\boldsymbol{v}_2^\top \boldsymbol{v}_2$$

The vectors $\boldsymbol{v}$ are orthonormal, which simplifies the expression to

$$\frac{\partial q}{\partial z_{i2}} = -2\boldsymbol{v}_2^\top \boldsymbol{x}_i + 2z_{i2} = 0$$

which yields the expected solution $z_{i2} = \boldsymbol{v}_2^\top \boldsymbol{x}_i$.

(b) Take the derivative and use the hint

$$\partial \tilde{J} \partial \boldsymbol{v}_2 = -(\mathbf{C} + \mathbf{C}^\top)\boldsymbol{v}_2 + 2\lambda_2 \boldsymbol{v}_2 + \lambda_{12}\boldsymbol{v}_1$$

but C is a symmetric matrix, such that we have

$$-2\mathbf{C}\boldsymbol{v}_2 + 2\lambda_2 \boldsymbol{v}_2 + \lambda_{12}\boldsymbol{v}_1 = 0$$

$$\mathbf{C}\boldsymbol{v}_2 = \lambda_2 \boldsymbol{v}_2 + \frac{1}{2}\lambda_{12}\boldsymbol{v}_1$$

But $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are orthogonal, and $\boldsymbol{v}_1$ is the eigenvector associated with the largest eigenvalue λ_1. It follows that $\boldsymbol{v}_2$ must be in the space orthogonal to $\boldsymbol{v}_1$, meaning that λ_{12} must be zero. So the equation simplifies to

$$\mathbf{C}\boldsymbol{v}_2 = \lambda_2 \boldsymbol{v}_2$$

2 Unsupervised exploration of nutrient database (5 points)

Download the following dataset from the USDA national nutrient database: nutrient.csv. Each record is for 100 grams. The are many features, most of which have a naming convention of the type `name_unit`, where unit may be `kcal` for an energy, (`g`, `mg`, `mcg`) for a weight in (gram, milligram, microgram), respectively.

(a) Do you find any redundant features? If so, remove them. Either way, justify your answer.

(b) Scale all your features to 0 mean and unit norm. Run principal component analysis (PCA) on your data. Plot the cumulative explained variance. Use it to identify how many of the first principal components (PCs) you need to explain $1 - \frac{1}{\mathrm{e}}$ of the data.

(c) Check that the first few PCs you've identified in (b) are all orthogonal to each other.

(d) Let's interpret the PCs. Analyze the components of your first three PCs to extract from each one their nutritional composition. Provide the top and bottom three nutrients to help you describe each PC.

(e) For the top three PCs, we wish to identify the food groups that best represent high values of each PC. In each case, focus on the top 500 entries with highest PC. What are the five food groups most represented?

Hint: In Python, you may find the following classes and functions useful:

- `sklearn.preprocessing.StandardScaler`
- `sklearn.decomposition.PCA`

2.1 Solution

(a) Load the model, remove the non-numeric features, and look for highly correlated features (≥ 0.99, to allow for numerical precision)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("data_12_nutrition.csv")
# Keep FoodGroup for later
df_fg = df["FoodGroup"]
# Remove columns of data that is non-numeric
df = df.drop(["ID", "FoodGroup", "ShortDescrip", "Descrip", "CommonName", "MfgName", "ScientificN

mask = np.triu(np.ones_like(df.corr(), dtype=bool))
upper_tri_corr = df.corr().where(mask)
corr_elements = upper_tri_corr.unstack()
[x for x in corr_elements[corr_elements >= 0.99].keys() if x[0] != x[1]]
```

```
[('VitA_USRDA', 'VitA_mcg'),
 ('VitB6_USRDA', 'VitB6_mg'),
 ('VitB12_USRDA', 'VitB12_mcg'),
 ('VitC_USRDA', 'VitC_mg'),
 ('VitE_USRDA', 'VitE_mg'),
 ('Folate_USRDA', 'Folate_mcg'),
 ('Niacin_USRDA', 'Niacin_mg'),
 ('Riboflavin_USRDA', 'Riboflavin_mg'),
 ('Thiamin_USRDA', 'Thiamin_mg'),
 ('Calcium_USRDA', 'Calcium_mg'),
 ('Copper_USRDA', 'Copper_mcg'),
 ('Magnesium_USRDA', 'Magnesium_mg'),
 ('Phosphorus_USRDA', 'Phosphorus_mg'),
 ('Selenium_USRDA', 'Selenium_mcg'),
 ('Zinc_USRDA', 'Zinc_mg')]
```

So the "_USRDA" features are redundant with the ones given as weight—remove them.

```
df_f = df.drop([c for c in df.columns if "_USRDA" in c], axis=1)
display(df_f)
```

	Energy_kcal	Protein_g	Fat_g	Carb_g	Sugar_g	Fiber_g	VitA_mcg	VitB6_mg	VitB12_mcg	V
0	717.0	0.85	81.11	0.06	0.06	0.0	684.0	0.003	0.17	0.
1	717.0	0.85	81.11	0.06	0.06	0.0	684.0	0.003	0.13	0.
2	876.0	0.28	99.48	0.00	0.00	0.0	840.0	0.001	0.01	0.
3	353.0	21.40	28.74	2.34	0.50	0.0	198.0	0.166	1.22	0.
4	371.0	23.24	29.68	2.79	0.51	0.0	292.0	0.065	1.26	0.
...	...	...	...	...	...	...	...	...	...	..
8613	305.0	18.50	25.10	0.00	0.00	0.0	47.0	0.410	12.00	0.
8614	111.0	20.54	0.84	5.41	0.00	0.0	2.0	0.112	2.15	0.
8615	269.0	0.00	0.00	73.14	73.20	0.0	0.0	0.000	0.00	0.
8616	90.0	16.10	1.40	2.00	0.00	0.0	30.0	0.130	0.50	0.

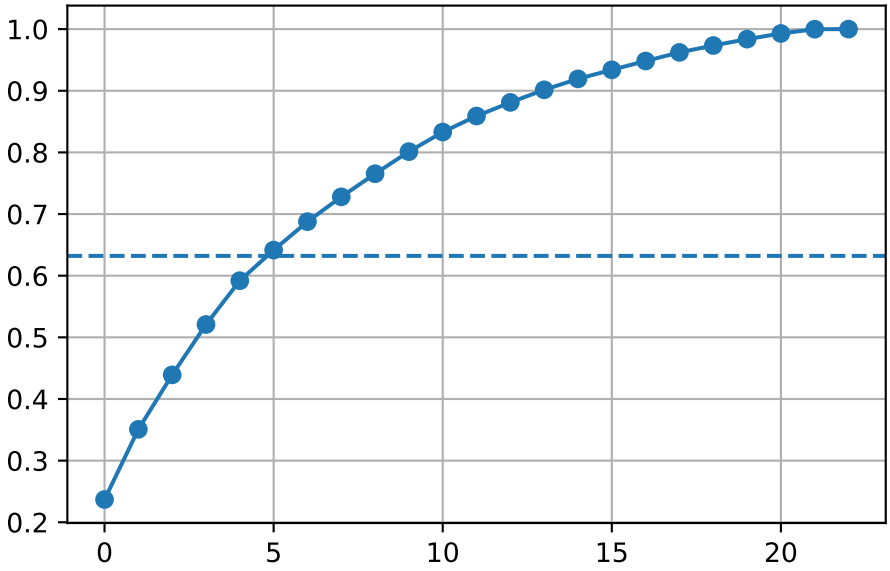
	Energy_kcal	Protein_g	Fat_g	Carb_g	Sugar_g	Fiber_g	VitA_mcg	VitB6_mg	VitB12_mcg	V
8617	89.0	19.80	0.50	0.00	0.00	0.0	30.0	0.120	1.00	0.

(b) Scale the data and run PCA

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df_tf = StandardScaler().fit_transform(df_f)
print(f"mean: {np.round(df_tf.mean(), 2)}, std: {np.round(df_tf.std(), 2)}")
fit = PCA()
pca = fit.fit_transform(df_tf)
plt.plot(fit.explained_variance_ratio_.cumsum(), "o-")
plt.axhline(1. - 1. / np.exp(1.), linestyle="--")
plt.grid();
```

```
mean: 0.0, std: 1.0
```



The first 6 eigenvectors account for more than 1-1/e = 63% of the data

```
num_pcs = 6
fit.explained_variance_ratio_[:num_pcs].cumsum()
```

```
array([0.23692547, 0.35077148, 0.43911486, 0.52081622, 0.59193237,
       0.6415137 ])
```

(c) Compute the correlation matrix between PCs

```
df_pca = pd.DataFrame(pca[:, :num_pcs], index=df.index)
np.round(df_pca.corr(), 5)
```

	0	1	2	3	4	5
0	1.0	-0.0	0.0	-0.0	-0.0	0.0
1	-0.0	1.0	0.0	-0.0	-0.0	0.0
2	0.0	0.0	1.0	-0.0	-0.0	-0.0
3	-0.0	-0.0	-0.0	1.0	0.0	-0.0
4	-0.0	-0.0	-0.0	0.0	1.0	-0.0
5	0.0	0.0	-0.0	-0.0	-0.0	1.0

(d)

```
vects = fit.components_[:num_pcs]
pd.Series(vects[0], index=df_f.columns).sort_values(ascending=False)
```

```
Riboflavin_mg    0.341325
Niacin_mg        0.337779
VitB6_mg         0.315663
Iron_mg          0.299857
Folate_mcg       0.284102
Thiamin_mg       0.272453
Zinc_mg          0.243551
Magnesium_mg     0.241348
Phosphorus_mg    0.199403
Fiber_g          0.181570
Copper_mcg       0.180806
VitB12_mcg       0.177985
Carb_g           0.169685
Calcium_mg       0.168112
Energy_kcal      0.157814
Protein_g        0.140620
VitE_mg          0.137122
VitA_mcg         0.133519
Manganese_mg     0.093567
Selenium_mcg     0.092319
VitC_mg          0.087639
Sugar_g          0.076323
Fat_g            0.033008
dtype: float64
```

Component 1 Foods that are high in riboflavin, niacin, vitamin B6; Low on Fat, sugar, Vitamin C.

```
vects = fit.components_[:num_pcs]
pd.Series(vects[1], index=df_f.columns).sort_values(ascending=False)
```

```
VitB12_mcg       0.355045
```

```
Protein_g        0.343397
Selenium_mcg     0.239322
VitA_mcg         0.236470
Copper_mcg       0.212669
Zinc_mg          0.177798
Manganese_mg     0.088783
Phosphorus_mg    0.087448
Niacin_mg        0.084801
Riboflavin_mg    0.073471
VitB6_mg         0.021129
VitC_mg         -0.038525
Thiamin_mg      -0.075150
Iron_mg         -0.093812
Folate_mcg      -0.097093
Magnesium_mg    -0.103361
Calcium_mg      -0.105173
VitE_mg         -0.106372
Fat_g           -0.111670
Fiber_g         -0.257733
Energy_kcal     -0.273449
Sugar_g         -0.358769
Carb_g          -0.443416
dtype: float64
```

Component 2 Foods that are high in Vitamin B12, protein, selenium; Low on sugar, carb, fiber.

```
vects = fit.components_[:num_pcs]
pd.Series(vects[2], index=df_f.columns).sort_values(ascending=False)
```

```
Folate_mcg       0.230985
Riboflavin_mg    0.192098
Thiamin_mg       0.184351
VitB6_mg         0.174648
Niacin_mg        0.164885
VitC_mg          0.162303
Iron_mg          0.087109
Sugar_g          0.055247
Carb_g           0.049822
VitB12_mcg      -0.012760
VitA_mcg        -0.021929
Zinc_mg         -0.038639
Fiber_g         -0.040397
Manganese_mg    -0.072630
Calcium_mg      -0.128139
Copper_mcg      -0.152263
Selenium_mcg    -0.163361
Magnesium_mg    -0.201225
VitE_mg         -0.207331
Protein_g       -0.213567
```

```
Phosphorus_mg   -0.274814
Energy_kcal     -0.462006
Fat_g           -0.534051
dtype: float64
```

Component 3 Foods that are high in folate, riboflavin, thiamin; Low on phosphorus, fat, protein.

(e) Combine the PCA dataframe with information about food groups

```
df_pca_fg = df_pca.join(df_fg)
df_pca_fg.sort_values(by=0, ascending=False)["FoodGroup"][:500].value_counts().head(5)
```

```
FoodGroup
Breakfast Cereals              229
Nut and Seed Products           48
Legumes and Legume Products     36
Spices and Herbs                33
Baby Foods                      27
Name: count, dtype: int64
```

```
df_pca_fg.sort_values(by=1, ascending=False)["FoodGroup"][:500].value_counts().head(5)
```

```
FoodGroup
Beef Products                         243
Finfish and Shellfish Products         68
Lamb, Veal, and Game Products          64
Poultry Products                       52
American Indian/Alaska Native Foods    30
Name: count, dtype: int64
```

```
df_pca_fg.sort_values(by=2, ascending=False)["FoodGroup"][:500].value_counts().head(5)
```

```
FoodGroup
Breakfast Cereals                   190
Vegetables and Vegetable Products   122
Beverages                            56
Fruits and Fruit Juices              47
Baby Foods                           37
Name: count, dtype: int64
```