Principal Component Analysis of Body Composition Measurements

J.R. Powers-Luhn

***Abstract*—The generation of physical models suffers from the presence of noise in the measured quantities. Simple models such as linear regression assume that the input parameters are noiseless and further assume that they are linearly independent. Real-life data rarely satisfies the latter assumption and never satisfies the former. A technique for compensating for this, principal component analysis, is examined and applied to body composition measurements in an attempt to predict body fat percentage. The dataset is transformed into a principal component space of orthogonal components and these are used to generate regression models of the data. This method underperformed naive linear regressions with a root mean squared error of 5.10%.**

I. Introduction

With the advent of automated data collection systems, datasets have become both deeper (with more granular data recorded at the second or sub-second level thanks to digital storage) and wider (with hundreds or thousands of sensors querying a system at once). These sensors are still subject to noise and may measure correlated quantities. Choosing which combination of sensors to include as inputs in a model scales in proportion to the number of sensors squared, making the creation of useful models grow rapidly more complex. This can be addressed by dimensionality reduction techniques such as principal component analysis (PCA).

Principal component analysis is a linear transformation of a dataset. The measured input variables are used to generate an orthogonal spanning set of principal components that covers the input space. Since the transformation is linear it can be applied to future additions to the dataset without updating the parameters. The columns of the transformation matrix are referred to as principal components; the products of the input measurements and the principal component transformation matrix are referred to as loading vectors or loadings. Principal component analysis has the additional design benefit of ordering each component based on the variance in the input data. This allows for dimensionality reduction with little loss of variance. Further, the rejection of low-variance components in the transformed space allows for more stability in the model by reducing the condition number of the matrix of input measurements, $\frac{\lambda_{max}}{\lambda_{min}}$.

A final advantage of PCA is apparent if the assumption is made that the variance in the measured data exceeds the variance introduced by the noise in the measurements. This means that be rejecting loadings that correspond to low-variance principal components, the effect of noise on regression models can be reduced.

II. Methodology

A dataset consisting of NUMBER samples with fifteen measurements recorded for each participant was obtained [1]. The first fourteen measurements were age, weight, adiposity index, and various circumference measurements. The last measurement was body fat percentage. Since many of the circumference measurements showed a high correlation (e.g. wrist and forearm circumference) principal component analysis was performed in order to isolate the varying components. The data set was split into training, testing, and validation sets by randomly sampling rows such that 70% of the data was placed in the training set and 15% each placed in the testing and validation set. The random number generator used to sample the rows was saved for reproduceability.

Principal component analysis consists of a linear transformation from the measured units (in this case, a fourteen-dimensional space spanned by non-orthogonal measurements) into a new space where the unit vectors are orthogonal to each other. This process has the additional property that the spanning set that is so derived is ordered from highest to lowest variance from the original data. If the noise in a measurement can be assumed to be smaller in variance than the measured quantities, this means that the noise will be isolated to the later vectors.

In order to derive the new spanning set, the covariance matrix of the input measurements is taken. This is then decomposed into equation 1:

$$\mathbf{C} \propto \mathbf{X}^T\mathbf{X} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T \tag{1}$$

where $\mathbf{X}$ is the original sample (where each column represents a measurement and each row represents a sample), $\mathbf{\Lambda}$ represents a diagonal matrix of the eigenvalues, and $\mathbf{W}$ represents the matrix which transforms the rows from $\mathbf{X}$ into the new space. This is performed in such a fashion that the eigenvalue/eigenvector pair selected is the one that also captures the maximum amount of variance from the original data. This gives the first principal component vector. Subsequent loadings must also satisfy the condition that they be orthogonal to the loadings calculated before them. This is satisfied by first removing the already-calculated loadings from $\mathbf{X}$, then repeating the process on the new measured value (as in equations 2 and 3). This can be repeated up to the minimum of the number of columns or the number of rows in $\mathbf{X}$. These vectors can be assembled into the matrix $\mathbf{W}$ which can transform measurement vectors from $\mathbf{X}$ into the loadings. The eigenvalues are also proportional to the amount of variance captured by the corresponding component. These eigenvalues

can be normalized (equation 4) to determine the fraction of the total variance carried by each principal component.

$$\mathbf{X}_k = \mathbf{X} - \sum_{i=0}^{k-1} \mathbf{X}\mathbf{w}_i\mathbf{w}_i^T \tag{2}$$

$$\mathbf{w}_k = \underset{||\mathbf{w}_k||}{\operatorname{argmin}} \frac{\mathbf{w}^T\mathbf{X}_k^T\mathbf{X}_k\mathbf{w}}{\mathbf{w}^T\mathbf{w}} \tag{3}$$

$$|\lambda_i| = \frac{\lambda_i}{\sum_i \lambda_i} \tag{4}$$

Because the loadings are orthogonal to each other, it is possible to exclude individual columns from the transformed. This is equivalent to excluding the input from an individual sensor from a regression but with the added benefit that later loadings correspond to lower variance. If the assumption that the noise is lower variance than the measurements, this means that noise can be excluded simply by eliminating the low-variance vectors. In order to explore this, principal components were excluded based on three different parameters. First components which cumulatively captured at least 90% of the variance (calculated as the first n loadings such that $\underset{n}{\operatorname{argmin}} \sum_{i=0}^{n-1} |\lambda_i| \geq 0.9$. Second, components representing at least 1% of the variance from the original sample ($|\lambda_i| \geq 0.01$) were isolated. Finally, principal components that were determined to be correlated with the output variable (body fat percentage) were isolated and used.

III. Results

The input data was transformed into the PCA space. The first four principal components were visually examined to see of what they consisted (see figure 1). All inputs were included in the first component except age–the variable least correlated with the other inputs. All other variables were included reduced by a factor of approximately 0.2 to 0.3 except height, which was the second lest correlated. Age and height are more represented in PCA 1 and 2 (PCA's were numbered starting with 0).

A regression was performed against the loadings corresponding to the first five PC's. This corresponded to the PC's that carried >90% of the variance from the original sample, as shown in figure 2. The performance of this model was the worst of all the models tested with a root mean squared error (RMSE) on the test set of 5.03.

Next, a regression model was generated that used all principal components with eigenvalues corresponding to >1% of the variance. This included PC's 0 to 8 (figure 3).. This model performed better, with a RMSE of 4.50. It seems that PC's 5 to 8 contain information required to calculate body fat percentage so that including the loadings corresponding to these components improved the performance of the model.

Loadings that were correlated with the output variable were determined by calculating the absolute value of the correlation coefficient between each loading and body fat percentage (see figure 4). Linear regression against these components showed degredation over the previous method attempted, with a test set RMSE of 4.92. When the next most correlated loading

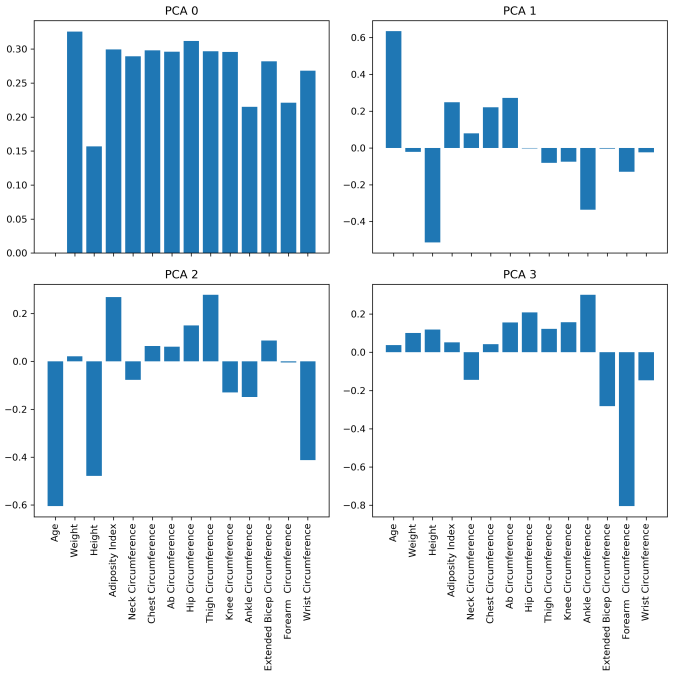

Fig. 1. The first four principal components, capturing approximately 90% of the variance. Age does not contribute to the first principal component, likely because it does not covary with the other measured values.

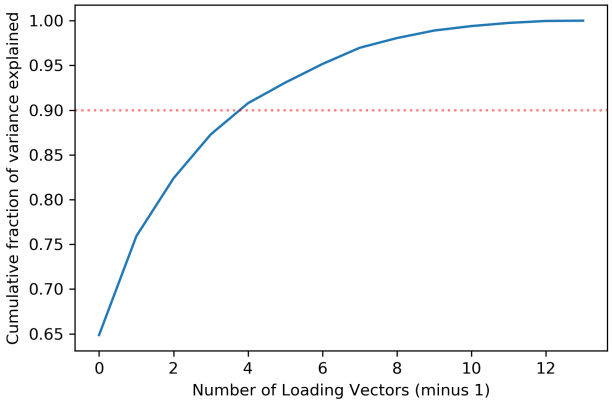

Fig. 2. Cumulative variance captured by the first n principal components. Note that the plot starts at 0.65 and has a negative second derivative. This shows that subsequent principal components capture less and less of the variance.

column was included in the regression model performance improved, reaching a RMSE of 4.24. In each case consideration was given to the stability of the solution by examining the condition number of the input (see figure 5) to verify that it did not significantly exceed the threshold value of 100. An examination of plots of each loading column vs. body fat percentage showed no evidence of non-linear relationships so models incorporating these elements were not attempted.

Finally, a principal component regression of all components was performed to test that performance. This was not expected to perform well, as it was mathematically equivalent to a linear

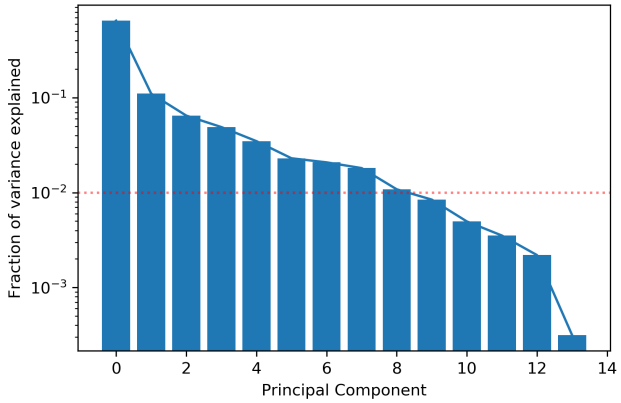

Fig. 3. Fraction of variance captured by each principal component. Each subsequent component captures less variance than its predecessor, necessitating a log scale. The red dotted line represents the 1% variance threshold.

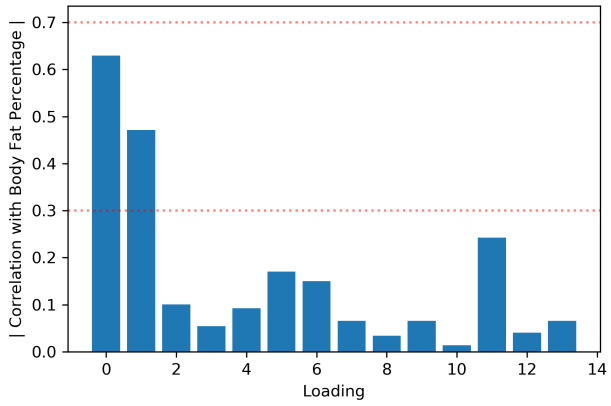

Fig. 4. Correlation between each loading column and body fat percentage. No loading reached the threshold of "strongly correlated" (>0.7) and only the first two were correlated (>0.3).

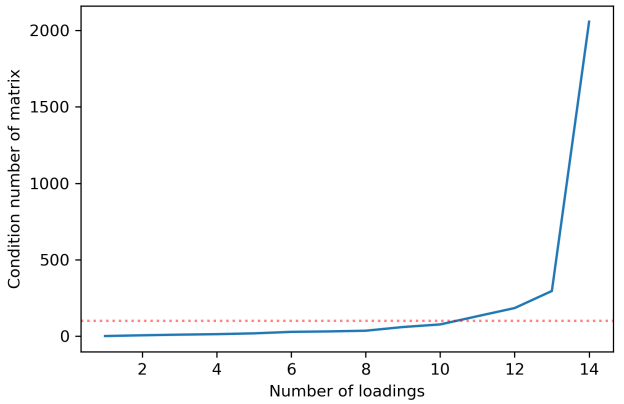

Fig. 5. Condition number of the input matrix vs number of principal components included. The rule of thumb threshold value of 100 is denoted in red.

TABLE I
RESULTS OF PRINCIPLE COMPONENT REGRESSIONS (TEST SET)

Model	RMSE
PCR Capturing 90% of Variance	5.025354
PCR of loadings with >1% of Variance	4.500349
PCR of correlated loadings (>0.3)	4.235354
PCR of correlated loadings (>0.2)	4.916291
PCR of all loadings	3.832659

regression using the original inputs. This model, however, outperformed all of the PCR models on the test set, with a RMSE of 3.83.

Based on these results, a regression of the most correlated loadings against body fat percentage was selected as the best model. Based on the visualization in figure 1, all measured values were present in the first two loading columns, implying that including additional columns introduces the potential to overfit the data.

The selected model had an error of 5.10 on the validation set. This was worse than the error on the validation set for a naive linear regression on all columns in the original data set (4.19).

IV. CONCLUSIONS

While principal component analysis and principal component regression are promising tools for future analysis, they are not an automatically superior modeling technique.

REFERENCES

[1] K W Penrose, A G Nelson, and A G Fisher. "Generalized Body Composition Prediction Equation For Men Using Simple Measurement Techniques". In: *Medicine & Science in Sports & Exercise* 17.2 (1985). ISSN: 0195-9131.

V. Appendix

Python code used to perform calculations and generate graphics.

```
# coding: utf-8

# In[1]:


get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import make_scorer


# In[2]:


get_ipython().run_line_magic('run', 'utilities.py')


# # Load and scale the data

# In[3]:


np.random.seed(3)


# In[4]:


x, y = load_matlab_data('hwkdataNEW.mat')


# In[5]:


xtr, ytr, xts, yts, xv, yv = train_test_val_split(x, y)


# In[6]:


column_names = [
    'Age', 'Weight', 'Height', 'Adiposity Index',
    'Neck Circumference', 'Chest Circumference',
    'Ab Circumference', 'Hip Circumference',
    'Thigh Circumference', 'Knee Circumference',
    'Ankle Circumference', 'Extended Bicep Circumference',
    'Forearm  Circumference', 'Wrist Circumference'
```

```
]


# In[7]:


scaler = StandardScaler()
scaler.fit(xtr)


# # Transform the data into PCA space

# In[8]:


xtr_scaled = scaler.transform(xtr)
pca = PCA()
xtr_scaled_pca = pca.fit_transform(xtr_scaled)


# In[9]:


pca_pipeline = Pipeline(
    memory=None,
    steps=[
        ('scale', StandardScaler()),
        ('pca', PCA()),
    ]
)
pca_pipeline.fit(xtr)


# # Analyze the PCA

# ## How much variance does each loading capture?

# In[10]:


evr = pca_pipeline.named_steps['pca'].explained_variance_ratio_
plt.plot(pca_pipeline.named_steps['pca'].explained_variance_ratio_)
plt.bar(range(len(evr)), evr)
plt.axhline(y=0.01, linestyle=':', color='r', alpha=0.5)
plt.xlabel("Principal Component")
plt.ylabel("Fraction of variance explained")
plt.yscale('log')
plt.savefig('images/fractional_variance_explained.png', dpi=300)
plt.show()


# In[11]:


cum_evr = np.cumsum(pca_pipeline.named_steps['pca'].explained_variance_ratio_)
plt.plot(cum_evr)
plt.xlabel("Number of Loading Vectors (minus 1)")
plt.ylabel("Cumulative fraction of variance explained")
```

```
plt.axhline(y=0.9, linestyle=':', color='r', alpha=0.5)
plt.savefig('images/cumulative_variance.png', dpi=300)
plt.show()


# In[12]:


np.where(cum_evr < 0.9)


# In[13]:


cum_evr[4]


# ## How correlated is each PCA to Body Fat Percentage?

# In[14]:


corr_vec = np.corrcoef(x=xtr_scaled_pca, y=ytr, rowvar=False)[-1, :-1]
plt.bar(x=range(len(corr_vec)), height=np.abs(corr_vec))
plt.xlabel('Loading')
plt.ylabel('| Correlation with Body Fat Percentage |')
plt.axhline(y=0.7, linestyle=':', color='r', alpha=0.5)
plt.axhline(y=0.3, linestyle=':', color='r', alpha=0.5)
plt.savefig('images/pca_correlation.png', dpi=300)
plt.show()


# ## Are the correlations between the various loadings linear?

# In[15]:


def plot_loading_scatter(loading_num):
    sns.scatterplot(x=xtr_scaled_pca[:,loading_num], y=ytr.ravel())
    plt.xlabel(f'Loading {loading_num}')
    plt.ylabel('Body Fat Percentage')
    plt.savefig(f'images/pca_{loading_num}_scatter.png', dpi=300)
    plt.show()


# In[16]:


for _ in range(14):
    plot_loading_scatter(_)


# This shows that the first loading is correlated with body fat,
# the second loading less so, and there is no visual correlation
# after that. No apparent non-linear relationships were noted.

# ## What are the first few PCA loadings comprised of?
```

```
# In[17]:


pca_data = pca_pipeline.named_steps['pca']


# In[18]:


f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex='col', figsize=(10,10))
plt.setp(ax3.xaxis.get_majorticklabels(), rotation=90)
ax1.bar(x=column_names, height=pca_data.components_[0])
ax1.set_title('PCA 0')
ax2.bar(x=column_names, height=pca_data.components_[1])
ax2.set_title('PCA 1')
ax3.bar(x=column_names, height=pca_data.components_[2])
plt.xticks(rotation=90)
ax3.set_title('PCA 2')
ax4.bar(x=column_names, height=pca_data.components_[3])
plt.xticks(rotation=90)
ax4.set_title('PCA 3')
#plt.suptitle("The First 4 Principal Component Loadings")
plt.tight_layout()
plt.savefig('images/first_four_pca.png', dpi=300)
plt.show()


## Conclusions

The data are approximately 2-3 dimensional based on two loadings
that explain greater than 1% of the variance each. A total of 90%
of the variance is explained by the first 5 loadings. The first
loading consists of all of the measured input variables except age.

# PCA Regression

## PCR capturing 90% of the variation in the input space

# In[19]:


pcr_90_percent = LinearRegression()
pcr_90_percent.fit(xtr_scaled_pca[:,:5], ytr)


`pca_pipeline.transform(x)` applies the same scaling and PCA
transformation determined from the training set to `x` (in
this case, the test data).

# In[20]:


xts_scaled_pca = pca_pipeline.transform(xts)


# In[21]:
```

```
pcr_90_percent_performance = rmse(
    pcr_90_percent.predict(xts_scaled_pca[:,:5]), yts)
print(pcr_90_percent_performance)


# ## PCR consisting of all loadings with >1% of variance

# To pick the correct PC's based on fraction of variance they
# capture, we compare their eigenvalues (normalized to the sum
# of all of the eigenvalues). This is equivalent to the fraction
# of variance they explain.

# In[22]:


pcr_gt10_percent = LinearRegression()
pcr_gt10_percent.fit(xtr_scaled_pca[:,:9], ytr)


# In[23]:


pcr_gt10_percent_performance = rmse(
    pcr_gt10_percent.predict(xts_scaled_pca[:,:9]), yts)
print(pcr_gt10_percent_performance)


# ## PC's most useful for predicting percent body fat

# This was equivalent to including PC's that are correlated with body
# fat (>0.3), and consisted of only the first 2 loadings. To explore
# this further, I will include the next most correlated PC (index 11).

# In[24]:


pcr_correlated = LinearRegression()
pcr_correlated.fit(xtr_scaled_pca[:, [0, 1, 11]], ytr)


# In[25]:


pcr_correlated_performance = rmse(
    pcr_correlated.predict(xts_scaled_pca[:, [0, 1, 11]]), yts)
print(pcr_correlated_performance)


# In[26]:


pcr_correlated2 = LinearRegression()
pcr_correlated2.fit(xtr_scaled_pca[:, [0, 1]], ytr)
pcr_correlated2_performance = rmse(
    pcr_correlated2.predict(xts_scaled_pca[:, [0, 1]]), yts)
print(pcr_correlated2_performance)
```

```
## All the PCA's

In[27]:


pcr_all = LinearRegression()
pcr_all.fit(xtr_scaled_pca, ytr)
pcr_all_performance = rmse(pcr_all.predict(xts_scaled_pca), yts)
print(pcr_all_performance)


## Comparison of PCR performance

In[28]:


models = {
    'Model': [
        'PCR Capturing 90% of Variance',
        'PCR of loadings with >1% of Variance',
        'PCR of correlated loadings (>0.3)',
        'PCR of correlated loadings (>0.2)',
        'PCR of all loadings'
    ],
    'RMSE': [
        pcr_90_percent_performance,
        pcr_gt10_percent_performance,
        pcr_correlated_performance,
        pcr_correlated2_performance,
        pcr_all_performance
    ]
}


In[29]:


test_results = pd.DataFrame(models)
test_results


In[30]:


print(test_results.to_latex(index=False))


## Calculate the error on the validation set

In[31]:


val_scaled_pca = pca_pipeline.transform(xv)
pcr_correlated_val_performance = rmse(
    pcr_correlated.predict(val_scaled_pca[:, [0, 1, 11]]), yv)
print(pcr_correlated_val_performance)
```

```
# In[32]:


naive_linear_model = LinearRegression()
naive_linear_model.fit(scaler.transform(xtr), ytr)
naive_linear_model_performance = rmse(
    naive_linear_model.predict(scaler.transform(xv)), yv)
print(naive_linear_model_performance)
```