Partial Least Squares Regression

J.R. Powers-Luhn

***Abstract*—A partial least squares (PLS) regression technique was applied to a dataset of body composition measurements to assess its usefulness in generating models. Comparisons between PLS and principal component regression were performed examining the different meaning of the intermediate transformations applied to the data. A final model was generated that predicted body fat percentage with a root mean squared error of 4.32.**

I. Introduction

In principal component regression, an operator matrix is developed that transformed the inputs into a new space where the basis set for that space was orthogonal and ordered by the variance. The new transformed vectors are then regressed against the model output variable. This allows for the removal of low-variance portions of the inputs, theoretically representing noise in the measurements.

This approach did not account for the fact that low variance portions of the input may still be highly correlated with the outputs. In order to address this, partial least squares regression transforms the inputs but does so with regards to covariance with the outputs rather than the variance of the inputs. In the same way that successive principal component loadings capture less of the input variance than the previous loading, partial least squares latent variables capture successively less covariance between the inputs and outputs. This results in the model performance rapidly improving with the first few latent variables, then leveling out. As a consequence of this, more stable models (since the regression inputs have fewer components) can be used with smaller trade offs in model performance.

II. Methodology

A dataset consisting of 247 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.

A partial least squares model was generated to predict body fat percentage from the other fourteen measurements. Multiple algorithms exist for calculating partial least squares regression[2][3]. For the purposes of this paper the nonlinear iterative partial least squares (NIPALS) was used[4].

In order to calculate the transformation matrix, the NIPALS algorithm (algorithm 1) was used. For the kth latent variable, this assigned (normalized) X^Ty to a vector $\vec{w}$ and (also normalized) XX^Ty to $\vec{t}$. Then these were recalculated iteratively until $\vec{t}$ and $\vec{p}$ converge to stability. This was repeated until k latent variables were calculated.

The regression parameters β were then calculated from equation 1. It was then possible to predict new values of y using equation 2.

Algorithm 1 Nonlinear Iterative Partial Least Squares

```
1:  procedure NIPALS
2:      for Number of latent variables, $k$ do
3:          $\vec{u_k} \leftarrow \vec{y}$
4:          $w_k \leftarrow \frac{X^T u_k}{||X^T u_k||}$
5:          $t_k \leftarrow X w_k$
6:          $q_k \leftarrow \frac{u_k^T t_k}{||u_k^T t_k||}$
7:          $u_k \leftarrow y q_k$
8:          while $||t_{old} - t_{new}|| > \epsilon$ do
9:              $p_{k,old} \leftarrow \frac{X^T t_k}{||t_k^T t_k||}$
10:             $p_{k,new} \leftarrow \frac{p_{k,old}}{||p_{k,old}||}$
11:             $t_{k,new} \leftarrow t_{k,old}||p_{k,old}||$
12:             $w_{k,new} \leftarrow w_{k,old}||p_{k,old}||$
13:         $X_{k+1} \leftarrow X_k - t_k p_k^T$
14:         $y_{k+1} \leftarrow y_k - u_k q_k^T$
```

$$U = T\beta \tag{1}$$

$$Y = UQ^T = T\beta Q^T = XP\beta Q^T \tag{2}$$

Cross validation was performed by generating a random array of indices of the same length as the number of inputs. This array was shuffled into a pseudo-random order and divided into train (70% of the samples), test (15% of the samples), and validation (15% of the samples) sets. Any remainders resulting from integer division were placed in the training set. The rows containing the maximum and minimum value for each input variable were then copied (if necessary) to the training set to ensure that the model would be interpolating in every case.

The number of latent variables was determined by cross validation. The algorithm was performed for increasing values of k and the root mean squared error between the predictions and the true values in the test set was calculated. Unlike principal component regression, where more loadings does not necessarily improve the training set accuracy, partial least squares training accuracy always improves. Cross validation becomes more important since it is the minimum error on the test set that determines the correct number of latent variables to employ.

The first four latent variable loadings were examined to determine if they contained information about the relationships between the input variables and the body fat measurements. The loadings were compared to the correlation values between each variable and the output. In order to further examine the difference between PCR and PLS, the loadings of the first four principal components and the first four latent variables were compared graphically.

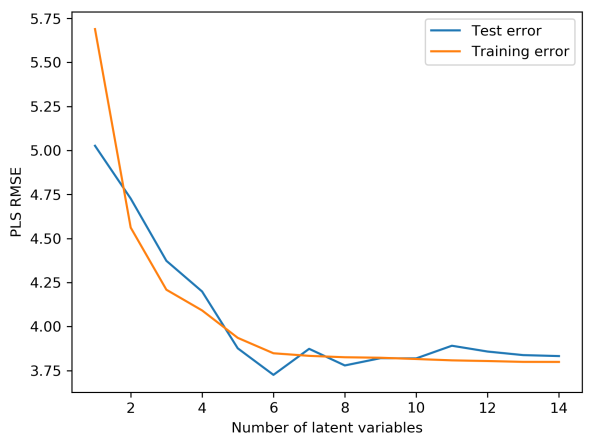

Fig. 1. Performance of the partial least squares regression as a function of the number of latent variables. Training error always goes down since each additional component necessarily fits the residual data not explained by earlier components. Testing error reaches and then oscillates around a minimum value.

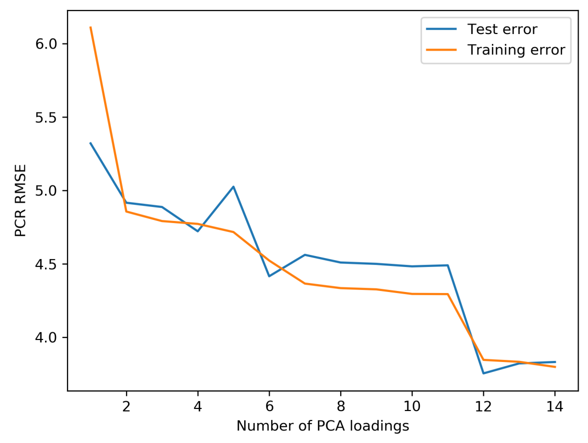

Fig. 2. Performance of the partial least squares regression as a function of the number of latent variables. Training error always goes down since each additional component necessarily fits the residual data not explained by earlier components. Testing error reaches and then oscillates around a minimum value.

III. Results

The training and testing error for a partial least squares regression are shown in figure 1. This is contrasted with the equivalent plot for principal component regression (figure 2) where training error improve or grow with the addition of components (those components may fit variance in the inputs that are immaterial to the outputs).

The difference between PLS latent variable weights and PCR is related to the supervised nature of PLS and the unsupervised nature of PCR. PCR loadings only capture the variance of the input data and are unaware of the output. The components are constructed to be purposefully orthogonal, unlike the PLS loadings. PLS latent variables are designed

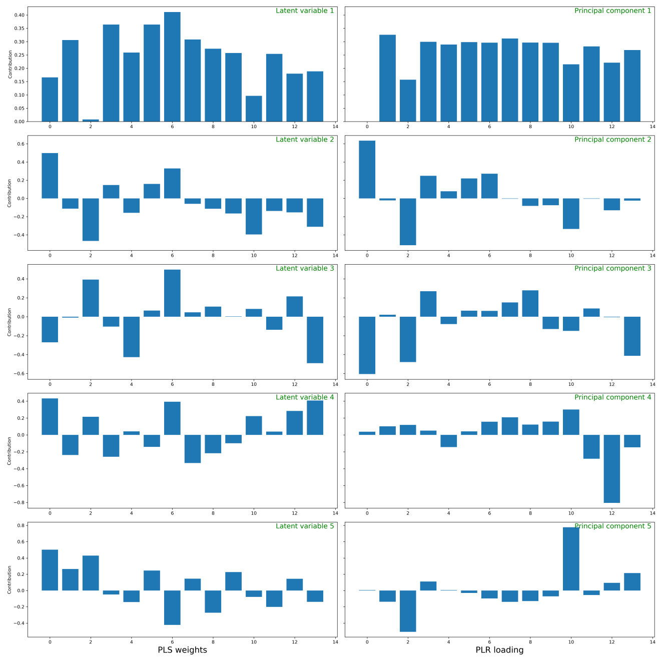

Fig. 3. Latent variable weights (left column) and PCR principal components (right column). The PLS weights differ from the principal components due to the supervised nature of PLS regression. PCR creates loadings based on variance in the input data while PLS creates weights based on covariance with output data.

to capture the maximum possible covariance between X_k and y_k, where X_k, y_k are the residuals from the $(k-1)$th step. PLS latent variables are not necessarily orthogonal (though some algorithms do have this result, NIPALS does not).

The first four PLS weights and PCR components are shown in figure 3. The first weight vector and component share some commonality (all values are positive, for example). Some similarity is to be expected as the first PCR loading was correlated with body fat percentage with a value of 0.64. The second weight/component for each algorithm deviated more since the correlation of PC 2 was only 0.47. Later PCs had a correlation with body fat of 0.1 or less; as expected, these showed no apparent relationship to the equivalent PLS weight vectors.

The weights of the first latent variable were compared to the correlation between the input and output variables (figure 4). The two vectors were nearly identical, differing only in scale and the value corresponding to the height input variable. This is as expected–the first latent variable (LV 1) should be in the direction of maximum covariance with the output variable. Later latent variables (figure 5) are designed to extend in the direction of the maximum covariance of the residual values of $\vec{y}$ once the contributions of earlier LV's are removed.

The first set of residuals of $\vec{y}$ (top right in figure 5) appear the be primarily explained by the difference between age and height. The next largest contribution to LV 2 is ankle circumference. These three variables were the least correlated with body fat and were therefore the smallest contributors to LV 1. As such it makes sense that remaining correlations not captured by LV 1 are primarily in those variables.

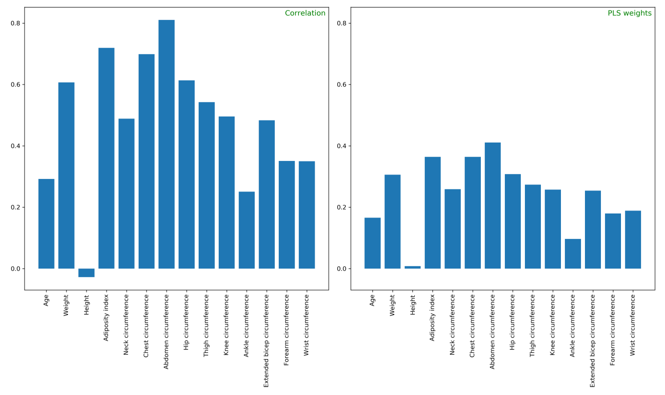

Fig. 4. The first latent variable weights (right) and correlation between the input matrix columns and outputs. While the scale is different, the two graphs clearly follow each other in form.

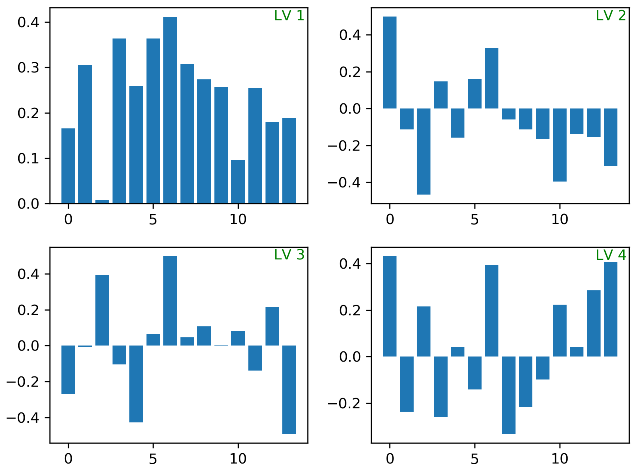

Fig. 5. The weights for the first four latent variables. Each weight is designed to translate the inputs into the direction corresponding to the maximum covariance with the residuals of the output.

The selected model had six latent variables, corresponding to the minimum test set RMSE. The validation error for this model was 4.35. This was lower than the error of the best principal component regression model (4.72) and the "best guess" linear regression model (4.79), but still not better than a linear regression using all inputs in the data set (4.19). Still, the improvement is significant and to be expected based on the supervised nature of the regression. Furthermore, the PLS model converged more quickly on the minimum error value with only 5 to 6 latent variables required vice the ≈ 12 required from PCR. PCR also required additional analysis if non-consecutive loadings were used in the regression based on their correlation with the output. Since PLS latent variables are inherently ordered by covariance with the output model selection can be performed by a simple plot of test set error vs number of latent variables. Further, the PLS model outperformed the PCR model in spite of not having any polynomial or inverse terms added to it.

The possibility of using a nonlinear method to fit the U and T matrices was considered. Scatter plots of the X- and y-scores for the latent variables were generated (the first of these, for LV 1, is shown in figure 6). Since no apparent nonlinear relationship exists between the two scores, it was determined that a nonlinear PLS was not warranted in this instance. It did appear that one or more outliers might be affecting the linear fit. Future models may be implemented in such a way that an outlier rejection step is included before the regression step.

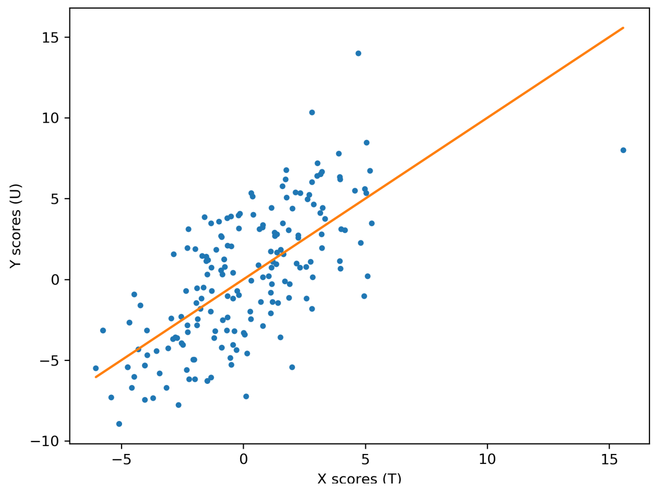

Fig. 6. Regression between the X scores and Y scores for the first latent variable. No evidence of a nonlinear relationship was evident.

IV. Conclusions

Partial least squares was used to generate a model that could predict body fat percentage with a root mean squared error of 4.35. This outperformed models previously generated using linear regression and principal component regression even without the generation of polynomial, inverse, or interaction terms. PLS latent variables were compared to PCR principal components and determined to be fundamentally different in spite of the visual similarity. The smaller number of components required to minimize test set error was shown to be a natural result of selecting latent variables in order of maximum input/output covariance. This provides real-world advantages by simplifying model input and increasing the stability of models.

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.

[2] Ildiko E. Frank and Jerome H. Friedman. "A Statistical View of Some Chemometrics Regression Tools". In: *Technometrics* 35.2 (1993), pp. 109–135. ISSN: 00401706. URL: http://www.jstor.org/stable/1269656.

[3] Sijmen de Jong. "SIMPLS: An alternative approach to partial least squares regression". In: *Chemometrics and Intelligent Laboratory Systems* 18.3 (1993), pp. 251–263. ISSN: 0169-7439. DOI: https://doi.org/10.1016/0169-7439(93)85002-X. URL: http://www.sciencedirect.com/science/article/pii/016974399385002X.

[4] Herman Wold. "Soft Modelling by Latent Variables: The Non-Linear Iterative Partial Least Squares (NIPALS) Approach". eng. In: *Journal of Applied Probability* 12.S1 (1975), pp. 117–142. ISSN: 0021-9002.

V. Appendix

Python code used to perform calculations and generate graphics.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cross_decomposition import PLSRegression
from sklearn.pipeline import Pipeline
from utilities import load_matlab_data, train_test_val_split, rmse
from sklearn.linear_model import LinearRegression
from sklearn.decomposition import PCA


# Set seed to match earlier homeworks
np.random.seed(3)

# Load data from matlab file
x, y = load_matlab_data("data/hwkdataNEW.mat")

colnames = [
    'Age',
    'Weight',
    'Height',
    'Adiposity␣index',
    'Neck␣circumference',
    'Chest␣circumference',
    'Abdomen␣circumference',
    'Hip␣circumference',
    'Thigh␣circumference',
    'Knee␣circumference',
    'Ankle␣circumference',
    'Extended␣bicep␣circumference',
    'Forearm␣circumference',
    'Wrist␣circumference'
]

# Split into training, testing, validation
xtr, ytr, xts, yts, xv, yv = train_test_val_split(x, y)

# The PLSRegression object scales the data by default
plsr = PLSRegression()
plsr.fit(xtr, ytr)

e = rmse(yts, plsr.predict(xts))
print(f"Error␣with␣default␣parameters:␣{e}")

# Vary number of components
for i in range(1, x.shape[1]):
    p = PLSRegression(n_components=i)
    p.fit(xtr, ytr)
    e = rmse(yts, p.predict(xts))
    print(f"Error␣for␣{i}␣components:␣{e}")

pls_models = [PLSRegression(n_components=i).fit(xtr, ytr) for i in range(1, x.shape[1]+1)]
# map(lambda m: m.fit(xtr, ytr), pls_models)

ets = [rmse(yts, m.predict(xts)) for m in pls_models]
etr = [rmse(ytr, m.predict(xtr)) for m in pls_models]
```

```
f = plt.figure()
plt.plot(range(1, x.shape[1]+1), ets, label="Test error")
plt.plot(range(1, x.shape[1]+1), etr, label="Training error")
plt.xlabel("Number of latent variables")
plt.ylabel("PLS RMSE")
plt.legend(loc="upper right")
plt.savefig("images/rmse_vs_number_components.png", dpi=300)
plt.gcf().clear()

def PCR(n_features):
    pcr = Pipeline([
        ('scale', StandardScaler()),
        ('pca', PCA(n_components=n_features)),
        ('linear', LinearRegression())
    ])
    return pcr

pcr_models = [PCR(i).fit(xtr, ytr) for i in range(1, x.shape[1]+1)]
ets = [rmse(yts, m.predict(xts)) for m in pcr_models]
etr = [rmse(ytr, m.predict(xtr)) for m in pcr_models]

f = plt.figure()
plt.plot(range(1, x.shape[1]+1), ets, label="Test error")
plt.plot(range(1, x.shape[1]+1), etr, label="Training error")
plt.xlabel("Number of PCA loadings")
plt.ylabel("PCR RMSE")
plt.legend(loc="upper right")
plt.savefig("images/rmse_vs_pca_loadings.png", dpi=300)
plt.gcf().clear()

f, axes = plt.subplots(nrows=5, ncols=2, sharey='row', figsize=(20, 20))

tmp = pls_models[4]
tmp2 = pcr_models[4]

for i in range(5):
    l = tmp.x_weights_.T[i]
    ax = axes[i, 0]
    ax.bar(x=range(l.shape[0]), height=l)
    ax.set_ylabel("Contribution")
    ax.text(0.99, 0.99, f'Latent variable {i+1}',
        verticalalignment='top', horizontalalignment='right',
        transform=ax.transAxes,
        color='green', fontsize=15)
    if i == 4:
        ax.set_xlabel("PLS weights", fontsize=18)

for i in range(5):
    l = tmp2.named_steps.pca.components_[i]
    ax = axes[i, 1]
    ax.bar(x=range(len(l)), height=l)
    ax.text(0.99, 0.99, f'Principal component {i+1}',
        verticalalignment='top', horizontalalignment='right',
        transform=ax.transAxes,
        color='green', fontsize=15)
    if i == 4:
        ax.set_xlabel("PLR loading", fontsize=18)
```

```
plt.tight_layout()
plt.savefig("images/pls_vs_pcr_loadings.png", dpi=300)
plt.gcf().clear()

for i in range(4):
    p = pls_models[-1]
    p = p.x_weights_.T[i]

f = plt.figure(figsize=(15, 9))

c = np.corrcoef(x, y, rowvar=False)[-1]
p = pls_models[-1].x_weights_.T[0]

ax1 = f.add_subplot(121)
ax1.bar(x=colnames, height=c[:-1])
plt.xticks(rotation=90)
ax1.text(0.99, 0.99, 'Correlation',
         verticalalignment='top',
         horizontalalignment='right',
         transform=ax1.transAxes,
         color='green', fontsize=12)

ax2 = f.add_subplot(122, sharey=ax1)
ax2.bar(x=colnames, height=p)
plt.xticks(rotation=90)
ax2.text(0.99, 0.99, 'PLS␣weights',
         verticalalignment='top',
         horizontalalignment='right',
         transform=ax2.transAxes,
         color='green', fontsize=12)

plt.tight_layout()
plt.savefig("images/pls_weights_and_correlation.png", dpi=300)
plt.gcf().clear()

p = pls_models[-1]
w1 = p.x_weights_.T[0]
w2 = p.x_weights_.T[1]
w3 = p.x_weights_.T[2]
w4 = p.x_weights_.T[3]

f = plt.figure()

ax1 = f.add_subplot(221)
ax1.text(0.99, 0.99, "LV␣1",
         verticalalignment='top',
         horizontalalignment='right',
         transform=ax1.transAxes,
         color='green')
ax1.bar(x=range(14), height=w1)

ax2 = f.add_subplot(222)
ax2.text(0.99, 0.99, "LV␣2",
         verticalalignment='top',
         horizontalalignment='right',
         transform=ax2.transAxes,
         color='green')
```

```
ax2.bar(x=range(14), height=w2)

ax3 = f.add_subplot(223)
ax3.text(0.99, 0.99, "LV 3",
         verticalalignment='top',
         horizontalalignment='right',
         transform=ax3.transAxes,
         color='green')
#ax3.bar(x=colnames, height=w3)
ax3.bar(x=range(14), height=w3)
#plt.xticks(rotation=90)

ax4 = f.add_subplot(224)
ax4.text(0.99, 0.99, "LV 4",
         verticalalignment='top',
         horizontalalignment='right',
         transform=ax4.transAxes,
         color='green')
#ax4.bar(x=colnames, height=w4)
ax4.bar(x=range(14), height=w4)
#plt.xticks(rotation=90)

plt.tight_layout()
plt.savefig("images/first_four_latent_variables.png", dpi=300)
plt.gcf().clear()

p = pls_models[0]
lr = LinearRegression()
lr.fit(p.x_scores_, p.y_scores_)
plt.plot(p.x_scores_, p.y_scores_, '.')
plt.plot(p.x_scores_, lr.predict(p.x_scores_))
plt.xlabel("X scores (T)")
plt.ylabel("Y scores (U)")
plt.savefig("images/pls_first_scores_regression.png", dpi=300)
plt.gcf().clear()

perf = rmse(yv, pls_models[5].predict(xv))
print(f"Final validation performance: {perf}")
```