Study Guide

Quiz 2: PCA, Clustering, and Regression

DataSci 347: Machine Learning 1

How to Use This Guide

This study guide covers all concepts needed for Quiz 2. Work through each section carefully, complete the practice problems, and verify your understanding. If you can answer all the checkpoint questions correctly, you should be well-prepared for the quiz.

Contents

1 Principal Component Analysis (PCA)

1.1 What is PCA?

Principal Component Analysis is a dimensionality reduction technique that transforms correlated variables into a set of uncorrelated variables called principal components (PCs). The key goals are:

- Reduce dimensionality while retaining maximum variance
- Create uncorrelated features from correlated ones
- Identify patterns in high-dimensional data

1.2 Standardization: When and Why

CRITICAL CONCEPT: Before running PCA, you must understand whether to standardize (center and scale) your data.

1.2.1 Centering

Centering means subtracting the mean from each variable:

$$x_{centered} = x - \bar{x} \tag{1}$$

1.2.2 Scaling

Scaling means dividing by the standard deviation after centering:

$$x_{scaled} = \frac{x - \bar{x}}{s_x} \tag{2}$$

When to standardize:

- Variables are measured in different units (e.g., weight in kg, height in cm)
- Variables have vastly different variances
- You want each variable to contribute equally to the analysis

Python Implementation:

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

# Standardize the data
scaler = StandardScaler()  # This centers AND scales
X_scaled = scaler.fit_transform(X)

# Then run PCA
pca = PCA()
pca.fit(X_scaled)
```

1.3 Interpreting Principal Components

1.3.1 Principal Component Loadings

The **loadings** (also called rotation matrix or components) tell you how each original variable contributes to each PC.

```
# Get the loadings
loadings = pca.components_.T  # Transpose to get variables x PCs
print(loadings)
#          PC1    PC2    PC3    PC4
# Var1    0.50  -0.50   0.20  -0.70
# Var2    0.49   0.51  -0.30   0.10
# Var3    0.51   0.48   0.40   0.20
# Var4    0.50  -0.49  -0.30   0.67
```

How to read loadings:

- Each column represents one principal component
- Each row shows how much that original variable contributes
- Positive values: variable increases with PC
- Negative values: variable decreases with PC
- Larger absolute values = stronger contribution

1.3.2 Computing PC Scores

EXAM TIP: You must know how to write the formula for PC scores!
If data is **NOT** standardized, PC1 score for observation i is:

$$PC1_i = w_1 \times x_{1i} + w_2 \times x_{2i} + ... + w_p \times x_{pi} \tag{3}$$

If data **IS** standardized, PC1 score for observation i is:

$$PC1_i = w_1 \times \frac{(x_{1i} - \bar{x}_1)}{s_1} + w_2 \times \frac{(x_{2i} - \bar{x}_2)}{s_2} + ... + w_p \times \frac{(x_{pi} - \bar{x}_p)}{s_p} \tag{4}$$

where w_j are the loadings for PC1, $\bar{x}_j$ are means, and s_j are standard deviations.
Example: Given loadings [0.5, 0.5, 0.5, 0.5] and standardized data, PC1 equals:

$$PC1 = 0.5 \times \frac{(x_1 - \bar{x}_1)}{s_1} + 0.5 \times \frac{(x_2 - \bar{x}_2)}{s_2} + 0.5 \times \frac{(x_3 - \bar{x}_3)}{s_3} + 0.5 \times \frac{(x_4 - \bar{x}_4)}{s_4} \tag{5}$$

1.4 Variance and PCA

KEY FACT: Principal components are ordered by variance!

- PC1 has the **largest variance** of all possible linear combinations
- PC2 has the second largest variance, uncorrelated with PC1
- PC3 has the third largest variance, uncorrelated with PC1 and PC2

- And so on...

```python
# Get variance explained by each PC
var_explained = pca.explained_variance_
print(var_explained)
# [3.2, 0.5, 0.2, 0.1]  # PC1 has highest variance

# Total variance
total_var = np.sum(var_explained)
print(total_var)  # Sum of all PC variances = total variance
```

1.5 Proportion of Variance Explained (PVE)

PVE tells you what percentage of total variance each PC captures.

$$PVE_j = \frac{\text{Variance of } PC_j}{\text{Total Variance}} = \frac{\text{Variance of } PC_j}{\sum_{i=1}^{p} \text{Variance of } PC_i} \tag{6}$$

```python
# Calculate PVE
pve = pca.explained_variance_ratio_
print(pve)
# [0.80, 0.125, 0.05, 0.025]

# This means:
# PC1 explains 80% of total variance
# PC2 explains 12.5% of total variance
# PC3 explains 5% of total variance
# PC4 explains 2.5% of total variance

# They sum to 1 (100%)
print(np.sum(pve))  # 1.0
```

Interpreting PVE Plots:

- X-axis: Principal component index (1, 2, 3, ...)
- Y-axis: Proportion of variance explained
- Look for "elbow" to determine how many PCs to keep
- First PC always has highest PVE

1.6 Practice Problems: PCA

Problem 1: You run PCA on 5 variables after centering and scaling. The loadings for PC1 are [0.45, 0.45, 0.44, 0.46, 0.45]. Write the formula for computing PC1 scores. Include the means and standard deviations in your formula.

Problem 2: Given PVE values [0.65, 0.20, 0.10, 0.05], what percentage of variance does PC1 capture? What about PC1 and PC2 combined?

Problem 3: True or False: If you run PCA on standardized data and all variables have equal weight (similar loadings) in PC1, then PC1 is approximately an average of the standardized variables.

Problem 4: You have 4 PCs with variances [16, 3, 0.8, 0.2]. Calculate the PVE for each PC. Which PC explains the most variance?

2 K-Means Clustering

2.1 What is K-Means?

K-means is an unsupervised learning algorithm that partitions data into k clusters. Each observation belongs to the cluster with the nearest mean (centroid).

2.2 How K-Means Works

1. Choose number of clusters k
2. Randomly initialize k cluster centroids
3. Assign each point to nearest centroid
4. Recalculate centroids based on assigned points
5. Repeat steps 3-4 until convergence

2.3 Python Implementation

```python
from sklearn.cluster import KMeans

# Fit k-means with 2 clusters
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)

# Get cluster assignments (0 or 1 for 2 clusters)
labels = kmeans.labels_
print(labels)  # [0, 1, 0, 1, 1, 0, ...]

# Count observations in each cluster
unique, counts = np.unique(labels, return_counts=True)
print(counts)  # [120, 80] means 120 in cluster 0, 80 in cluster 1

# Get cluster centers
centers = kmeans.cluster_centers_
print(centers)
```

2.4 Important Notes

EXAM TIP: Pay attention to:

- The `random_state` parameter - affects initialization
- Cluster labels are arbitrary (0, 1, 2, ...) - no inherent ordering
- Cluster sizes are NOT necessarily equal
- Results depend on initialization and can vary

2.5 Interpreting Cluster Sizes

```
# If you get output like:
unique, counts = np.unique(labels, return_counts=True)
print(counts)
# [116, 84]

# This means:
# - There are 2 clusters total (length of array)
# - Cluster 0 has 116 observations
# - Cluster 1 has 84 observations
# - Total: 116 + 84 = 200 observations
```

Common Question Format: "There are X subjects in cluster 1 and Y in cluster 2"

- Make sure X + Y = total number of observations
- Check which number corresponds to which cluster
- Labels can be 0-indexed or 1-indexed depending on context

2.6 Practice Problems: Clustering

Problem 1: You run k-means with k=3 on 300 observations. The output shows counts [100, 125, 75]. How many observations are in each cluster? Do clusters have equal sizes?

Problem 2: True or False: K-means always produces clusters of equal size.

Problem 3: If you run k-means twice with different random seeds, will you get the same cluster sizes? Why or why not?

3 Linear Regression

3.1 Simple Linear Regression

Simple linear regression models the relationship between one predictor X and response Y:

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{7}$$

where:

- β_0 = intercept
- β_1 = slope (coefficient)
- ϵ = random error

3.2 Interpreting Coefficients

The Slope (β_1):
"On average, for every 1 unit increase in X, Y changes by β_1 units."
CRITICAL DISTINCTION:

- **Average effect:** On average, Y decreases by β_1 when X increases by 1
- **Individual predictions:** For specific observations, actual values can vary around the predicted mean

Example:

```
# Output:
# Intercept: 35.82
# Coefficient: -0.044

# This means:
# Predicted MPG_Hwy = 35.82 - 0.044 * Horsepower
```

Interpretation:

- On average, MPG_Hwy decreases by 0.044 for each 1 unit increase in Horsepower
- A car with Horsepower=200 has predicted MPG = 35.82 - 0.044(200) = 27.02
- A car with Horsepower=201 has predicted MPG = 35.82 - 0.044(201) = 26.976

EXAM TIP: Watch out for tricky questions!
TRUE Statement: "On average, MPG decreases by 0.044 when Horsepower increases by 1."
FALSE Statement: "For any two specific cars where one has Horsepower=200 and another has Horsepower=201, the first car is GUARANTEED to have higher MPG."

Why is the second false? Because individual observations have variability around the regression line. The model predicts the **mean**, not individual values exactly.

3.3 Multiple Linear Regression

Multiple regression includes multiple predictors:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{8}$$

CRITICAL CONCEPT: Interpretation of coefficients in multiple regression

β_1 = "The average change in Y for a 1-unit increase in X_1, **holding all other variables constant**."

Example:

```
# Output:
# Intercept: 46.99
# Horsepower: -0.028
# Weight: -4.01

# Model: MPG_Hwy = 46.99 - 0.028*Horsepower - 4.01*Weight
```

How to interpret -0.028 for Horsepower:

- "On average, for each 1 unit increase in Horsepower, MPG_Hwy decreases by 0.028, **when Weight is held constant**."
- You cannot say "1 unit increase in Horsepower always decreases MPG by 0.028" without the "holding Weight constant" qualifier

3.4 Making Predictions

To predict: Plug values into the equation

```
# Given: MPG_Hwy = 46.99 - 0.028*Horsepower - 4.01*Weight
# Predict for: Horsepower=240, Weight=3.5

predicted_MPG = 46.99 - 0.028*240 - 4.01*3.5
predicted_MPG = 46.99 - 6.72 - 14.035
predicted_MPG = 26.235
```

IMPORTANT: You can only use variables that are in the model!

- If Seats and Length are not in the model, you don't need their values to predict
- Having extra information doesn't hurt - just ignore variables not in the model
- You cannot make predictions if required variables are missing

3.5 Python Implementation

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Simple regression: Y ~ X1
X = car_data[['Horsepower']]  # Need double brackets for DataFrame
y = car_data['MPG_Hwy']

model = LinearRegression()
```

```python
model.fit(X, y)

print(f"Intercept: {model.intercept_}")
print(f"Coefficient: {model.coef_[0]}")

# Multiple regression: Y ~ X1 + X2
X_multi = car_data[['Horsepower', 'Weight']]
y = car_data['MPG_Hwy']

model2 = LinearRegression()
model2.fit(X_multi, y)

print(f"Intercept: {model2.intercept_}")
print(f"Coefficients: {model2.coef_}")
# model2.coef_[0] is coefficient for Horsepower
# model2.coef_[1] is coefficient for Weight

# Make predictions
new_data = np.array([[240, 3.5]])  # Horsepower=240, Weight=3.5
prediction = model2.predict(new_data)
print(f"Predicted MPG: {prediction[0]}")
```

3.6 Model Quality: R^2

R^2 **(R-squared)** measures the proportion of variance in Y explained by the model.

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{9}$$

Interpretation:

- $R^2 = 0.465$ means "The model explains 46.5% of the variance in Y"
- Range: 0 to 1 (0% to 100%)
- Higher R^2 = better fit
- R^2 always increases when adding more variables

Adjusted R^2: Penalizes adding unhelpful variables

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{10}$$

where n = number of observations, p = number of predictors

```python
from sklearn.metrics import r2_score

# Calculate R-squared
y_pred = model.predict(X)
r2 = r2_score(y, y_pred)
print(f"R-squared: {r2}")

# Or use model's score method
r2 = model.score(X, y)
print(f"R-squared: {r2}")
```

3.7 F-Test for Overall Model Significance

The F-test tests whether **at least one** predictor is useful:

$$H_0 : \beta_1 = \beta_2 = ... = \beta_p = 0 \text{ (no predictors are useful)} \tag{11}$$

$$H_A : \text{At least one } \beta_j \neq 0 \text{ (at least one predictor is useful)} \tag{12}$$

F-statistic formula:

$$F = \frac{(R^2/p)}{((1 - R^2)/(n - p - 1))} \tag{13}$$

where p = number of predictors, n = sample size
EXAM TIP: Relationship between F-test and R^2

- Larger R^2 leads to larger F-statistic
- But you need the F-test to determine if R^2 is "large enough" to be statistically significant
- Large R^2 alone doesn't prove significance - you need the p-value from the F-test

How to reject H_0:

- If F-statistic p-value ¡ α (significance level), reject H_0
- Example: p-value = 0.0001, α = 0.001, so 0.0001 ¡ 0.001 → reject H_0
- "p-value < 2e-16" means p-value is extremely small (essentially 0)

```python
import scipy.stats as stats

# Calculate F-statistic
n = len(y)
p = X_multi.shape[1]  # number of predictors

# Get R-squared
y_pred = model2.predict(X_multi)
r2 = r2_score(y, y_pred)

# Calculate F-statistic
f_stat = (r2 / p) / ((1 - r2) / (n - p - 1))

# Get p-value
f_pvalue = 1 - stats.f.cdf(f_stat, p, n - p - 1)

print(f"F-statistic:␣{f_stat}")
print(f"p-value:␣{f_pvalue}")

# Interpretation:
if f_pvalue < 0.001:
    print("Reject␣H0␣at␣alpha=0.001")
    print("At␣least␣one␣predictor␣is␣significant")
```

3.8 Practice Problems: Regression

Problem 1: Given the model: $\hat{Y} = 50 - 0.05X$

- What is the predicted Y when X=100?
- What is the predicted Y when X=101?
- On average, what happens to Y when X increases by 1?
- If person A has X=100 and person B has X=101, is Y guaranteed to be higher for person A?

Problem 2: Given: $\hat{Y} = 30 + 2X_1 - 5X_2$

- Interpret the coefficient of X_1
- Interpret the coefficient of X_2
- Predict Y when $X_1 = 10, X_2 = 3$
- Can you predict Y if you're given $X_1 = 10, X_2 = 3, X_3 = 5$?

Problem 3: A model has $R^2 = 0.65$ and p=2 predictors, n=200 observations. Calculate the F-statistic.

Problem 4: True or False: "If $R^2 = 0.8$, we can automatically reject $H_0 : \beta_1 = \beta_2 = 0$ at $\alpha = 0.001$." Explain.

4 Model Diagnostics

4.1 Residual Plots

Residuals are the differences between observed and predicted values:

$$e_i = y_i - \hat{y}_i \tag{14}$$

4.1.1 Residuals vs. Fitted Values Plot

This plot helps check the **linearity assumption**.

```python
# Create residual plot
y_pred = model.predict(X)
residuals = y - y_pred

plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted')
plt.show()
```

What to look for:

- **Good:** Random scatter around horizontal line at 0
- **Bad:** Clear patterns, curves, or systematic deviations

4.2 Interpreting Residual Patterns

Pattern 1: Underestimation

- If residuals are mostly **positive** (above 0) for certain fitted values
- This means $y_i - \hat{y}_i > 0$, so $y_i > \hat{y}_i$
- The actual values are **higher** than predictions
- We are **underestimating** (predicting too low)

Pattern 2: Overestimation

- If residuals are mostly **negative** (below 0) for certain fitted values
- This means $y_i - \hat{y}_i < 0$, so $y_i < \hat{y}_i$
- The actual values are **lower** than predictions
- We are **overestimating** (predicting too high)

EXAM TIP: Common question pattern
"For cars with smaller MPG_Hwy (left side of plot), residuals are positive. This suggests:"

- Positive residuals = actual > predicted
- We are predicting too low
- We are **underestimating**
- Linearity might be a problem

4.3 Visual Examples

Scenario 1: Left side of plot shows mostly positive residuals

```
Residuals
    |     *  *
  3 |   *  *
    |  *
  0 |----------*----------
    |        *   *
 -3 |          *   *
    |________________
      10    20    30
      Fitted values
```

Interpretation: Small fitted values (left) have positive residuals → underestimating

Scenario 2: Left side of plot shows mostly negative residuals

```
Residuals
    |
  3 |          *   *
    |        *   *
  0 |----------*----------
    |  *
 -3 |   *  *
    | *  *
    |________________
      10    20    30
      Fitted values
```

Interpretation: Small fitted values (left) have negative residuals → overestimating

4.4 Practice Problems: Diagnostics

Problem 1: You fit a model predicting house prices. The residual plot shows that for expensive houses (right side, high fitted values), most residuals are negative. Are you overestimating or underestimating expensive houses?

Problem 2: For a model predicting test scores, you observe positive residuals for students with low predicted scores. What does this suggest about the model's predictions for low-performing students?

Problem 3: True or False: "If all residuals on the left side of the plot are above 0, the model is overestimating for small values."

5 Common Pitfalls and Tips

5.1 PCA Pitfalls

1. **Forgetting to mention standardization** when writing PC score formulas
2. **Confusing loadings with PC scores** - loadings are weights, scores are the transformed data
3. **Misreading PVE plots** - make sure you understand what the y-axis represents
4. **Thinking PCs are in any order other than by variance** - PC1 ALWAYS has highest variance

5.2 Clustering Pitfalls

1. **Assuming equal cluster sizes** - K-means does NOT force equal sizes
2. **Forgetting clusters depend on initialization** - random seed matters
3. **Confusing cluster labels** - the numbers (0, 1, 2) are arbitrary

5.3 Regression Pitfalls

1. **Confusing "on average" with "always"** - regression predicts means, not guarantees
2. **Forgetting "holding other variables constant"** in multiple regression interpretation
3. **Thinking high R^2 alone proves significance** - need F-test p-value
4. **Mixing up over- and underestimation** - positive residuals = underestimating
5. **Trying to use variables not in the model** for prediction

5.4 Exam Strategy

1. **Read carefully:** Look for words like "always," "guaranteed," "on average"
2. **Check your assumptions:** Is data standardized? What variables are in the model?
3. **Show your work:** For calculations, write out the formula first
4. **Think about interpretation:** Don't just memorize - understand what statistics mean
5. **Watch for trick answers:** Some will be technically correct but misleading

6 Quick Reference Formulas

6.1 PCA

$$\text{Standardized variable: } \frac{x - \bar{x}}{s} \tag{15}$$

$$\text{PC score (unstandardized): } PC = w_1 x_1 + w_2 x_2 + ... + w_p x_p \tag{16}$$

$$\text{PC score (standardized): } PC = w_1 \frac{(x_1 - \bar{x}_1)}{s_1} + ... + w_p \frac{(x_p - \bar{x}_p)}{s_p} \tag{17}$$

$$\text{PVE: } \frac{\text{Var}(PC_j)}{\sum_{i=1}^{p} \text{Var}(PC_i)} \tag{18}$$

6.2 Regression

$$\text{Simple: } Y = \beta_0 + \beta_1 X + \epsilon \tag{19}$$

$$\text{Multiple: } Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \epsilon \tag{20}$$

$$\text{Residual: } e_i = y_i - \hat{y}_i \tag{21}$$

$$R^2 : 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \tag{22}$$

$$\text{F-statistic: } \frac{(R^2/p)}{((1 - R^2)/(n - p - 1))} \tag{23}$$

6.3 Key Python Commands

```
# PCA
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA()
pca.fit(X_scaled)
loadings = pca.components_.T
pve = pca.explained_variance_ratio_

# K-Means
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=k, random_state=seed)
kmeans.fit(X)
labels = kmeans.labels_
counts = np.unique(labels, return_counts=True)[1]

# Regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X, y)
intercept = model.intercept_
coefficients = model.coef_
predictions = model.predict(X_new)
r2 = model.score(X, y)
```

Final Checklist

Before the quiz, make sure you can:

- ☐ Write the formula for PC scores with and without standardization
- ☐ Explain what PVE means and how to interpret a PVE plot
- ☐ Identify which PC has the largest variance
- ☐ Interpret cluster sizes from k-means output
- ☐ Distinguish between ”on average” and ”always” in regression
- ☐ Interpret regression coefficients in multiple regression
- ☐ Make predictions using regression equations
- ☐ Calculate R^2 and understand what it means
- ☐ Explain when to use the F-test and how to interpret p-values
- ☐ Identify over- vs. underestimation from residual plots
- ☐ Know when holding other variables constant matters

Study smart, think carefully, and you’ve got this!