Regularization and Optimization
Scenario-Based Questions

Module B, Week 2, Assignment 2

1 Multiple Choice Questions (MCQs)

1. **Predicting Crop Yield with Regularization**
   Scenario: A farmer collects data on soil quality and sunlight hours to predict crop yield. Concerned about overfitting due to a complex polynomial model, she applies Ridge regression using the following Python code:

```python
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score

X = np.array([[1, 2], [2, 3], [3, 4], [4, 5], [5,
    6]])
y = np.array([10, 15, 20, 25, 30])

ridge = Ridge(alpha=2.0)
ridge.fit(X, y)

y_pred = ridge.predict(X)
r2 = r2_score(y, y_pred)
print(f"R^2 Score: {r2:.4f}")
```

   Output: R^2 `Score: 0.9917`
   Question: What is the R^2 score for the Ridge regression model, and what does it suggest about the model's fit on the training data?

   A) 0.99, indicating a poor fit

B) 0.99, indicating a strong fit

C) 0.95, indicating moderate overfitting

D) 0.95, indicating underfitting

Answer: **B) 0.99, indicating a strong fit**
Explanation: The output shows an R^2 score of 0.9917, which rounds to 0.99. This high value indicates that the Ridge regression model explains nearly all the variance in the training data, suggesting a strong fit. Ridge regression applies L2 regularization by adding a penalty ($\alpha \sum \beta_i^2$) to reduce coefficient magnitudes, helping prevent overfitting compared to a standard polynomial model. However, a high training R^2 alone doesn't guarantee generalization, and the farmer should evaluate the model on unseen data to confirm robustness.

2. **Lasso Feature Selection**
Scenario: A data analyst models energy consumption based on appliance usage and room temperature. To simplify the model, she uses Lasso regression, which can zero out coefficients. She runs this code while rounding off the output to nearest whole number:

```python
import numpy as np
from sklearn.linear_model import Lasso

X = np.array([[1, 10], [2, 20], [3, 30], [4, 40],
    [5, 50]])
y = np.array([100, 200, 300, 400, 500])

lasso = Lasso(alpha=5.0)
lasso.fit(X, y)

print(f"Coefficients: {lasso.coef_}")
```

Output: `Coefficients:  [ 0.  10.  ]`
Question: What does the Lasso model's output suggest about the features' impact on energy consumption?

A) Appliance usage is irrelevant

B) Temperature has no impact

C) Both features are equally important

D) Temperature dominates the prediction

Answer: **B) Temperature has no impact**
Explanation: The coefficients are [0.0, 10.0], corresponding to appliance usage and temperature, respectively. Lasso regression applies L1 regularization ($\alpha \sum |\beta_i|$), which can shrink coefficients to exactly zero, effectively removing features. Here, the zero coefficient for appliance usage indicates it was deemed irrelevant, while temperature's coefficient of 10 suggests it drives the prediction (e.g., each degree increases consumption by 10 units). This aligns with Lasso's feature selection capability, simplifying the model by focusing on temperature.

3. **Gradient Descent Step Size**
Scenario: A student trains a linear regression model using gradient descent to predict house prices based on size. She experiments with learning rates to minimize the Mean Squared Error (MSE) cost function. With a learning rate of 0.001, convergence is slow, taking 10,000 iterations. With a learning rate of 1.0, the model diverges, with MSE increasing wildly. Which learning rate is likely to balance speed and stability for her dataset?

A) 0.0001

B) 0.01

C) 1.5

D) 10.0

Answer: **B) 0.01**
Explanation: Gradient descent updates parameters by stepping in the direction opposite the gradient, scaled by the learning rate ($\theta := \theta - \eta \nabla J(\theta)$). A learning rate of 0.001 is too small, causing slow convergence (like cautious steps), while 1.0 is too large, leading to divergence (overshooting the minimum). A moderate rate like 0.01 strikes a balance, allowing steady progress without instability, as it's larger than 0.001 but smaller than 1.0. Option 0.0001 is even slower, and 1.5 or 10.0 would likely exacerbate divergence. The optimal rate depends on the dataset, but 0.01 is a reasonable compromise based on typical practice.

4. **Ridge vs. Linear Regression (Difficult)**
Scenario: A researcher predicts student test scores using hours studied and sleep duration. She compares Linear and Ridge regression models:

```python
import numpy as np
from sklearn.linear_model import LinearRegression,
    Ridge

X = np.array([[1, 8], [2, 7], [3, 6], [4, 5], [5,
    4]])
y = np.array([60, 65, 70, 75, 80])

lr = LinearRegression().fit(X, y)
ridge = Ridge(alpha=3.0).fit(X, y)

new_data = np.array([[6, 3]])
lr_pred = lr.predict(new_data)[0]
ridge_pred = ridge.predict(new_data)[0]

print(f"Linear Prediction: {lr_pred:.2f}")
print(f"Ridge Prediction: {ridge_pred:.2f}")
```

Output:

```
Linear Prediction: 85.00
Ridge Prediction: 83.04
```

Question: Why does the Ridge model predict a lower score than Linear regression for a student studying 6 hours with 3 hours of sleep?

A) Ridge underfits the data
B) Ridge penalizes large coefficients
C) Linear regression ignores sleep duration
D) Ridge eliminates the sleep feature

Answer: **B) Ridge penalizes large coefficients**
Explanation: Ridge regression adds an L2 penalty ($\alpha \sum \beta_i^2$) to the cost function, shrinking coefficients to prevent overfitting. The lower prediction (83.04 vs. 85.00) reflects smaller coefficients, making the model less sensitive to input changes (e.g., high study hours). Linear regression, without regularization, may assign larger coefficients, risking overfitting. Ridge doesn't eliminate features (unlike Lasso), and underfitting

would reduce fit quality, not just prediction magnitude. Sleep duration isn't ignored, as both models use both features.

5. **Lasso Coefficient Shrinkage (Difficult)**
Scenario: A city planner models traffic congestion based on vehicle count and road width. She uses Lasso regression to identify key factors:(round off to nearest 1 decimal)

```python
import numpy as np
from sklearn.linear_model import Lasso

X = np.array([[100, 10], [200, 12], [300, 14],
    [400, 16], [500, 18]])
y = np.array([20, 40, 60, 80, 100])

lasso = Lasso(alpha=2.0)
lasso.fit(X, y)

print(f"Coefficients: {lasso.coef_}")
print(f"Intercept: {lasso.intercept_:.2f}")
```

Output:
```
Coefficients: [0.2 0.  ]
Intercept: 0.00
```
Question: What does the Lasso model indicate about the factors affecting congestion?

A) Vehicle count has no impact
B) Road width is the only factor
C) Both factors are equally weighted
D) Vehicle count is the primary factor

Answer: **D) Vehicle count is the primary factor**
Explanation: The coefficients [0.2, 0.0] correspond to vehicle count and road width. Lasso's L1 penalty ($\alpha \sum |\beta_i|$) zeros out road width's coefficient, indicating it has no impact in this model. Vehicle count's coefficient of 0.2 suggests congestion increases by 0.2 units per vehicle, making it the primary factor. This demonstrates Lasso's feature selection, simplifying the model by focusing on vehicle count. The options

implying the dominance of the road width or the equal weighting are incorrect due to the zero coefficient.

2 Numeric Type Questions

1. **Ridge Model MSE**
   Scenario: A chef predicts cooking time based on ingredient count and oven temperature. To avoid overfitting, she uses Ridge regression:

```python
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

X = np.array([[2, 200], [3, 220], [4, 240], [5,
    260], [6, 280]])
y = np.array([30, 45, 60.95, 74.05, 90])

ridge = Ridge(alpha=1.5)
ridge.fit(X, y)

y_pred = ridge.predict(X)
mse = mean_squared_error(y, y_pred)
print(f"MSE: {mse:.2f}")
```

   Question: What is the Mean Squared Error (MSE) of the Ridge model on the training data?
   Answer: **0.34**
   Explanation: The MSE, output as 0.34, measures the average squared difference between predicted and actual cooking times (MSE $= \frac{1}{n}\sum(y_i - \hat{y}_i)^2$). A low MSE indicates good training fit, but Ridge's L2 regularization ensures coefficients are moderated to reduce overfitting risk. The chef should verify performance on a test set, as a low training MSE alone doesn't guarantee generalization, but the question focuses on the output value. The small MSE reflects the model's ability to closely match the training data while being regularized.

2. **Gradient Descent Iterations**
   Scenario: A programmer optimizes a linear regression model for ad click prediction using gradient descent. With a learning rate of 0.05,

the cost function (MSE) drops below 0.01 after 500 iterations. She tests a smaller learning rate of 0.005 and finds the MSE is still 0.1 after 500 iterations. How many iterations would she need at 0.005 to reach an MSE below 0.01, assuming linear convergence?
Answer: **5000**
Explanation: Gradient descent convergence speed is roughly proportional to the learning rate (η). With $\eta = 0.05$, the MSE reaches below 0.01 in 500 iterations. Reducing η to 0.005 (1/10th) slows convergence by a factor of 10, assuming linear convergence (a simplification for small steps). Thus, it would take $500 \times 10 = 5000$ iterations to achieve the same MSE reduction. This reflects the trade-off of smaller steps requiring more iterations to reach the cost function's minimum, as smaller η ensures stability but slows progress.

3. **Lasso Prediction**
Scenario: A retailer predicts sales based on advertising budget and store size using Lasso regression:

```python
import numpy as np
from sklearn.linear_model import Lasso

X = np.array([[10, 100], [20, 200], [30, 300], [40,
    400]])
y = np.array([50, 100, 150, 200])

lasso = Lasso(alpha=1.0)
lasso.fit(X, y)

new_data = np.array([[50, 500]])
pred = lasso.predict(new_data)[0]
print(f"Predicted sales: {pred:.0f}")
```

Output: `Predicted sales: 250.00`
Question: What are the predicted sales for an advertising budget of 50 and store size of 500?
Answer: **250.00**
Explanation: The output shows a predicted sales value of 250.00 for the input [50, 500]. Lasso regression minimizes the cost function with an L1 penalty, potentially zeroing out coefficients (e.g., store size may be irrelevant if its coefficient is 0). The prediction reflects the model's

learned relationship, likely driven by advertising budget, as Lasso simplifies by selecting key features. The exact coefficients aren't shown, but the output directly provides the answer, consistent with the model's fit to the training data ($y \approx 5x_1$).

4. **Ridge Coefficient Magnitude**

   Scenario: A biologist models fish growth based on water temperature and food supply. She uses Ridge regression to constrain coefficients:

```python
import numpy as np
from sklearn.linear_model import Ridge

X = np.array([[10, 1], [15, 2], [20, 3], [25, 4]])
y = np.array([5, 10, 15, 20])

ridge = Ridge(alpha=4.0)
ridge.fit(X, y)

coef_sum = np.sum(np.abs(ridge.coef_))
print(f"Sum of absolute coefficients:
   {coef_sum:.2f}")
```

   Output: `Sum of absolute coefficients: 1.12`
   Question: What is the sum of the absolute values of the Ridge model's coefficients?
   Answer: **1.12**
   Explanation: The output gives 1.12 as the sum of absolute coefficients ($\sum |\beta_i|$). Ridge regression minimizes MSE $+ \alpha \sum \beta_i^2$, shrinking coefficients to reduce overfitting. A smaller sum compared to linear regression (without regularization) indicates moderation, balancing fit and simplicity. The value 1.12 reflects the combined influence of temperature and food supply, constrained by $\alpha = 4.0$. This metric helps assess regularization strength, as higher α further reduces coefficient magnitudes.

5. **Gradient Descent Convergence**

   Scenario: An engineer optimizes a model predicting battery life using gradient descent with MSE as the cost function. With a learning rate of 0.02, the gradient magnitude drops below 0.001 after 2000 iterations. If she increases the learning rate to 0.04, how many iterations are needed

to achieve the same gradient magnitude, assuming proportional convergence?
Answer: 1000
Explanation: Gradient descent updates parameters as $\theta := \theta - \eta \nabla J(\theta)$. Doubling the learning rate from 0.02 to 0.04 (2x) roughly halves the number of iterations needed for the same gradient reduction, assuming linear convergence near the minimum. Thus, $2000/2 = 1000$ iterations suffice. This simplification holds for small gradients, but in practice, larger learning rates risk instability, though the question assumes convergence. The result reflects faster progress per step with a larger η.
Hint: Consider how doubling the learning rate affects iteration count.

3 Multiple Select Questions (MSQs)

1. **Comparing Regression Models**
Scenario: A meteorologist predicts rainfall based on humidity and wind speed, comparing Linear, Ridge, and Lasso regression:

```python
import numpy as np
from sklearn.linear_model import LinearRegression,
    Ridge, Lasso

X = np.array([[50, 10], [60, 12], [70, 14], [80,
    16]])
y = np.array([20, 25, 30, 35])

lr = LinearRegression().fit(X, y)
ridge = Ridge(alpha=2.0).fit(X, y)
lasso = Lasso(alpha=1.0).fit(X, y)

print(f"Linear Coef: {lr.coef_}")
print(f"Ridge Coef: {ridge.coef_}")
print(f"Lasso Coef: {lasso.coef_}")
```

Output:
```
Linear Coef: [0.48 0.09]
Ridge Coef: [0.47 0.09]
Lasso Coef: [0.49 0.0]
```
Question: Which statements are true about the models?

A) Ridge coefficients are smaller than Linear regression's
B) Lasso eliminates wind speed as a feature
C) Linear regression risks overfitting more than Ridge
D) Lasso assigns equal weight to both features

Answers: **A, B, C**
Explanation:
- A) True: Ridge adds L2 regularization, which shrinks coefficients, so they are slightly smaller than those from linear regression.
- B) True: the second feature (0.0 in Lasso coef) is eliminated by Lasso (set to 0).
- C) True in general. Ridge adds regularization, which helps control overfitting by shrinking coefficients.
- D)False: Lasso assigned 0.49 to the first feature and 0.0 to the second — not equal.

2. **Gradient Descent Behavior**
Scenario: A data scientist trains a model to predict customer purchases using gradient descent. She observes different learning rates: 0.001 (slow), 0.1 (oscillating), and 0.01 (steady). Which statements describe gradient descent outcomes?

A) A learning rate of 0.001 ensures stable but slow convergence
B) A learning rate of 0.1 may cause divergence
C) A learning rate of 0.01 risks overshooting the minimum
D) The cost function always decreases monotonically

Answers: **A, B**
Explanation:
- A) A small learning rate (0.001) takes tiny steps, ensuring stability but slowing convergence, true.
- B) A large learning rate (0.1) can overshoot, causing oscillation or divergence, true.
- C) A learning rate of 0.01 is moderate, typically stable, not prone to overshooting, false.
- D) With large learning rates, the cost may increase (e.g., oscillation), so it's not always monotonic, false.
Hint: Think about how learning rate size affects gradient descent's path.

3. **Regularization Effects**

Scenario: A gardener predicts plant height using water amount and sunlight hours, applying Ridge regression to control overfitting:

```python
import numpy as np
from sklearn.linear_model import Ridge

X = np.array([[1, 8], [2, 7], [3, 6], [4, 5]])
y = np.array([10, 15, 20, 25])

ridge = Ridge(alpha=5.0)
ridge.fit(X, y)

print(f"Coefficients: {ridge.coef_}")
print(f"Intercept: {ridge.intercept_:.2f}")
```

Output:

```
Coefficients: [1.66 -1.66]
Intercept: 24.17
```

Question: Which statements are true about the Ridge model?

A) Water amount has a stronger effect than sunlight

B) The model includes a non-zero intercept

C) Coefficients are shrunk to prevent overfitting

D) Some coefficients are exactly zero

Answers: **B, C**

Explanation:

- A) False: The coefficients for both features (water amount and sunlight) are equal in magnitude (1.66 and -1.66), so neither has a stronger effect. The sign indicates direction (positive or negative impact), not strength.

- B) True: The intercept is 24.17, which is clearly non-zero. This means the model does not pass through the origin, but rather has a baseline value of 24.17 when both features are zero.

- C) True: Ridge regression applies L2 regularization, which shrinks the coefficients to prevent overfitting. The coefficients are not excessively large, which indicates regularization is at work

- D) False: The coefficients are 1.66 and -1.66, which are non-zero. Ridge does not shrink coefficients to exactly zero; that's the domain of

Lasso (L1 regularization).

4. **Cost Function Analysis**
   Scenario: A physicist models particle motion using a regression model with MSE as the cost function. She visualizes the cost landscape for two parameters and applies gradient descent:

   A) The cost function forms a bowl-shaped landscape
   B) Gradient descent moves opposite the gradient's direction
   C) A flat gradient indicates convergence
   D) The cost always reaches zero

   Answers: **A, B, C**
   Explanation:
   - A) MSE for two parameters typically forms a convex, bowl-shaped surface, true.
   - B) Gradient descent updates parameters as $\theta := \theta - \eta \nabla J$, moving downhill, true.
   - C) A small gradient ($\nabla J \approx 0$) signals the minimum, indicating convergence, true.
   - D) The cost may not reach zero due to noise or model limitations, false.
   Hint: Visualize the cost function and gradient descent's mechanics.

5. **Model Evaluation Metrics**
   Scenario: A marketer evaluates three regression models (Linear, Ridge, Lasso) for predicting campaign revenue:

```
import numpy as np
from sklearn.linear_model import LinearRegression,
   Ridge, Lasso
from sklearn.metrics import r2_score

X = np.array([[1, 10], [2, 20], [3, 30], [4, 40]])
y = np.array([100, 200, 300, 400])

lr = LinearRegression().fit(X, y)
ridge = Ridge(alpha=2.0).fit(X, y)
lasso = Lasso(alpha=2.0).fit(X, y)
```

```

r2_lr = r2_score(y, lr.predict(X))
r2_ridge = r2_score(y, ridge.predict(X))
r2_lasso = r2_score(y, lasso.predict(X))

print(f"R^2 Linear: {r2_lr:.4f}")
print(f"R^2 Ridge: {r2_ridge:.4f}")
print(f"R^2 Lasso: {r2_lasso:.4f}")
```

Output:

```
R^2 Linear:  1.0000
R^2 Ridge:  1.0000
R^2 Lasso:  1.0000
```

Question: Which statements are true about the models' training performance?

A) Linear regression perfectly fits the training data

B) Ridge has a slightly worse fit than Linear

C) Lasso's fit is the weakest among the three

D) All models have poor generalization

Answers: **A**

Explanation:

- A)True: An R^2 value of 1.0000 means the model explains 100- B)False: Ridge also has an R^2 value of 1.0000, which means it also fits the training data perfectly, just like Linear regression. There is no sign of a worse fit

- C) False: Lasso also has an R^2 value of 1.0000, which indicates a perfect fit to the training data. There is no evidence that its fit is weaker than the others

- D)False: The R^2 value of 1.0000 indicates perfect training fit, but it does not necessarily reflect poor generalization. Overfitting is a concern if the models have high training performance but low test performance. This statement cannot be verified solely based on the training R^2 value. However, the statement about "poor generalization" is not accurate without test data performance.