bytewise.

Bytewise Fellowship Program

DATA SCIENCE

Task 14

BWT- Data Science (Group1)

Submitted to: Mahrukh Khan

Submitted by: Usama Malik

Task: Linear Regression

Linear Regression (Chapter 4)

1. Linear Regression

Definition: Linear regression models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to observed data.

Example:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([1.2, 1.8, 3.6, 3.8, 5.1])
# Train linear regression model
model = LinearRegression()
model.fit(X, y)
# Predict and plot
y_pred = model.predict(X)
plt.scatter(X, y)
plt.plot(X, y_pred, color='red')
plt.show()
```

2. The Normal Equation

Definition: A method to compute the parameters of the linear regression model analytically by minimizing the cost function.

Example:

```
# Using the normal equation to compute theta
X_b = np.c_[np.ones((5, 1)), X]  # Add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

```
print(f'Theta: {theta_best}')
```

3. Computational Complexity

Definition: The computational cost of training a linear regression model using the normal equation is O(n3)O(n^3)O(n3) due to the matrix inversion.

4. Gradient Descent

Definition: An iterative optimization algorithm to minimize the cost function by updating the model parameters in the direction of the steepest descent.

Example:

```
eta = 0.1  # Learning rate
n_iterations = 1000
m = 5

theta = np.random.randn(2,1)  # Random initialization
for iteration in range(n_iterations):
    gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
    theta = theta - eta * gradients

print(f'Gradient Descent Theta: {theta}')
```

5. Polynomial Regression

Definition: Extends linear regression by adding polynomial features to the model, allowing it to fit a wider range of data patterns.

Example:

```
from sklearn.preprocessing import PolynomialFeatures
# Generate polynomial features
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
# Train polynomial regression model
poly_model = LinearRegression()
```

```
poly_model.fit(X_poly, y)
```

6. Learning Curves

Definition: Graphs that plot the model performance on the training set and validation set over time to diagnose bias and variance.

Example:

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
# Split data
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2)
train_errors, val_errors = [], []
for m in range(1, len(X_train)):
    model.fit(X_train[:m], y_train[:m])
    y_train_predict = model.predict(X_train[:m])
    y_val_predict = model.predict(X_val)
    train_errors.append(mean_squared_error(y_train[:m], y_train_predict))
    val_errors.append(mean_squared_error(y_val, y_val_predict))
plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="train")
plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="val")
plt.legend(loc="upper right")
plt.xlabel("Training set size")
plt.ylabel("RMSE")
plt.show()
```

7. Regularized Linear Models

Definition: Techniques to constrain or regularize the model to reduce overfitting, including Ridge, Lasso, and Elastic Net regression.

8. Ridge Regression

Definition: A regularized version of linear regression that adds a penalty equal to the square of the magnitude of the coefficients.

Example:

```
from sklearn.linear_model import Ridge

ridge_reg = Ridge(alpha=1, solver="cholesky")

ridge_reg.fit(X, y)
```

9. Logistic Regression

Definition: A regression model used for binary classification that estimates probabilities using a logistic function.

Example:

```
from sklearn.linear_model import LogisticRegression

# Sample classification data

X_class = np.array([[0.1], [0.3], [0.5], [0.7], [0.9]])

y_class = np.array([0, 0, 1, 1, 1])

log_reg = LogisticRegression()

log_reg.fit(X_class, y_class)
```

10. Estimating Probabilities

Definition: Logistic regression provides estimated probabilities that an instance belongs to a particular class.

Example:

```
# Estimating probabilities

probs = log_reg.predict_proba([[0.6]])

print(f'Probabilities: {probs}')
```

11. Training and Cost Function

Definition: The cost function for logistic regression is the log loss (binary cross-entropy), and training involves finding the parameters that minimize this cost.

12. Decision Boundaries

Definition: Logistic regression models have decision boundaries that separate the classes by predicting probabilities above or below a certain threshold (usually 0.5).

Example:

```
# Plotting decision boundary

X_new = np.linspace(0, 1, 1000).reshape(-1, 1)

y_proba = log_reg.predict_proba(X_new)

plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")

plt.xlabel("Feature")

plt.ylabel("Probability")

plt.legend()

plt.show()
```