|  |  |
| --- | --- |
| [Machine Learning]  [2024-1] |  |
| Homework 1 |  |
| [Due Date] 2024.04.04  Student ID : 2021111000  Name : Minjun Cho  Professor : Juntae Kim | logo-placeholder |

1. Write python codes to solve each of the following problem, and attach the result and description. (20 pts)

1-1. Numpy:

, ,

Compute where mean of each column of X

Compute where

|  |
| --- |
| Code |
| *import* numpy *as* np  X = np.array([[1, 2, 3, 4],  [5, 6, 7, 8],  [9, 10, 11, 12]])  a = np.array([0.4, 0.2, 0.1])  m = X.mean(*axis*=0)  y = np.dot(a, X)  (m, y) |
| Result(Captured images) |
| A screenshot of a computer program  Description automatically generated |

1-2. Pandas: Read data from From boston.csv (Boston Housing Price dataset), make a dataframe by selecting data with CRIM values < 1.0. Then from this data, compute “MEDV” column’s mean, and show the distribution of “MEDV” using a Histogram.

|  |
| --- |
| Code |
| *import* pandas *as* pd  *import* matplotlib.pyplot *as* plt  boston\_df = pd.read\_csv('boston.csv')  filtered\_df = boston\_df[boston\_df['CRIM'] < 1.0]  medv\_mean = filtered\_df['MEDV'].mean()  plt.hist(filtered\_df['MEDV'], *bins*=30, *edgecolor*='black')  plt.title('Distribution of MEDV Values')  plt.xlabel('MEDV')  plt.ylabel('Frequency')  plt.show()  medv\_mean |
| Result(Captured images) |
| A graph of a distribution of medv values  Description automatically generated |

1-3. Matplotlib : For plot the graph of in red color. The noise is normal distribution random value with mean 0, standard deviation 0.1

|  |
| --- |
| Code |
| *import* numpy *as* np  *import* matplotlib.pyplot *as* plt  x = np.arange(0, 101)  y = np.log2(x[1:])  noise = np.random.normal(0, 0.1, *size*=y.shape)  y\_noisy = y + noise  plt.figure(*figsize*=(10,5))  plt.plot(x[1:], y\_noisy, 'r', *label*='y = log2(x) + noise')  plt.title('Graph of y = log2(x) + noise')  plt.xlabel('x')  plt.ylabel('y')  plt.legend()  plt.grid(True)  plt.show() |
| Result(Captured images) |
| A graph with a red line  Description automatically generated |

1-4. Scikit-learn: Following is a dataset for student score vs. study hours and sleep ours. Use scikit-learn LinearRegression to learn a prediction model, and predict the score of a student with study hours = 2 and sleep ours = 2.

|  |  |  |
| --- | --- | --- |
| Study hours | Sleep hours | Score |
| 7.0 | 9.0 | 91 |
| 3.5 | 4.0 | 54 |
| 8.5 | 2.5 | 77 |
| 1.0 | 9.5 | 26 |
| 5.0 | 5.0 | 65 |

|  |
| --- |
| Code |
| *from* sklearn.linear\_model *import* LinearRegression  data = {  'Study hours': [7.0, 3.5, 8.5, 1.0, 5.0],  'Sleep hours': [9.0, 4.0, 2.5, 9.5, 5.0],  'Score': [91, 54, 77, 26, 65]  }  df = pd.DataFrame(data)  X = df[['Study hours', 'Sleep hours']]  y = df['Score']  model = LinearRegression()  model.fit(X, y)  predicted\_score = model.predict([[2, 2]])  predicted\_score[0] |
| Result(Captured images) |
| A screen shot of a computer program  Description automatically generated |

2. Explain what Supervised Learning, Unsupervised Learning, and Reinforcement Learning are, and describe the differences. (10 pts)

|  |
| --- |
| Your Answer |
| **Supervised Learning** is a type of machine learning where the model is trained on a labeled dataset, which means that each training example is paired with an output label. The model learns to make predictions based on this data. It is commonly used for tasks such as regression (predicting a continuous output) and classification (predicting a discrete output).  **Unsupervised Learning**, on the other hand, involves training a model on data that does not have labeled responses. The model tries to learn the underlying patterns and structure from the data without any guidance on what the output should be. Common unsupervised learning tasks include clustering (grouping similar instances together) and dimensionality reduction (simplifying the data without losing important information).  **Reinforcement Learning** is a type of learning where an agent learns to make decisions by performing certain actions and observing the rewards or penalties from those actions. It is different from supervised and unsupervised learning in that the learning process is based on interaction with the environment and is aimed at long-term goals. Reinforcement learning is commonly used in areas such as robotics, games, and navigation where an agent must make a sequence of decisions that lead to a reward.  The key differences among these learning paradigms are:   * **Supervised Learning** uses labeled data to train models. * **Unsupervised Learning** works with unlabeled data and aims to find structure within the data. * **Reinforcement Learning** interacts with an environment to learn policies for achieving long-term goals. |

3. Describe the concept of “overfitting”, and explain how you can prevent overfitting in supervised learning. (10 pts)

|  |
| --- |
| Your Answer |
| Overfitting in supervised learning occurs when a model learns not only the underlying patterns in the training data but also the noise and random fluctuations to an extent that it negatively impacts the performance of the model on new, unseen data. Essentially, an overfitted model is too complex and too closely fit to the training dataset, resulting in poor generalization to other datasets.  To prevent overfitting in supervised learning, you can use several techniques:   1. **Simplify the Model**: Use a simpler model with fewer parameters or less complexity. This can be achieved by reducing the number of features, using linear models instead of high-degree polynomial models, or applying techniques like dimensionality reduction. 2. **Cross-Validation**: Instead of using the entire dataset for training, use a portion for training and another portion for validation. Cross-validation, such as k-fold cross-validation, allows you to test the model's performance on unseen data and ensures that the model generalizes well. 3. **Regularization**: Techniques such as L1 (Lasso) and L2 (Ridge) regularization add a penalty for larger coefficients to the loss function. This discourages the model from fitting too closely to the training data and promotes simpler models that generalize better. 4. **Early Stopping**: During the training process, you can monitor the performance of the model on a validation set and stop the training once the performance begins to degrade, which is a sign that the model might be starting to overfit. 5. **Pruning**: In certain models, like decision trees, you can remove branches that have little importance and are likely to be noise. This process is known as pruning and can lead to simpler and more general models. 6. **Ensemble Methods**: Combining the predictions of several models can reduce the risk of overfitting. Techniques like bagging and boosting help in forming a more robust model by aggregating the results of multiple models. 7. **Data Augmentation**: Increasing the size of the training set using data augmentation can help in preventing overfitting, especially in domains like computer vision where it is possible to generate new training samples by altering the existing ones in realistic ways. 8. **Including More Data**: If possible, increasing the amount of training data can help the model to generalize better and reduce overfitting. |

4. is m x n matrix, is m x 1 vector, is n x 1 vector. .

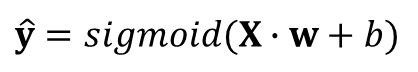
Let

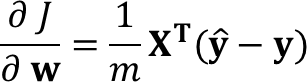
Show that (10 pts)

|  |
| --- |
| Your Answer |
| A math equations on a graph paper  Description automatically generated |

5. For

Compute followings by hand: (20 pts)





|  |
| --- |
| Your Answer |
| A math equations on a graph paper  Description automatically generated |

6. The heart\_disease.csv dataset represents 13 attributes of a patient and the presence of heart disease. Meaning of attributes are as below. The ‘num’ is the target value, 0 means no disease, 1~4 means different types of disease.

* age: age in years
* sex: sex (1 = male; 0 = female)
* cp: chest pain type
* trestbps: resting blood pressure (in mm Hg on admission to the hospital)
* chol: serum cholestoral in mg/dl
* fbs: fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
* restecg: resting electrocardiographic results

(0: normal, 1: ST-T wave abnormality, 2: left ventricular hypertrophy)

* thalach: maximum heart rate achieved
* exang: exercise induced angina (1 = yes; 0 = no)
* oldpeak = ST depression induced by exercise relative to rest
* slope: the slope of the peak exercise ST segment (1: upsloping, 2: flat, 3: downsloping)
* ca: number of major vessels (0-3) colored by flourosopy
* thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
* num: diagnosis of heart disease

Change the dataset for binary classification (change 1~4 values of ‘num’ to 1), then perform logistic regression and show 1) the cost function graph, 2) learned model, 3) training accuracy of the model, 4) prediction result for the patient with attribute values of [61, 0, 3, 154, 210, 1, 0, 130, 0, 1.5, 2, 2, 3].

Do NOT use scikit learn library. (30 pts)

|  |
| --- |
| Code |
| df = pd.read\_csv('heart\_disease.csv')  df['num'] = df['num'].apply(lambda *x*: 1 *if* *x* >= 1 *else* 0)  def sigmoid(*z*):    *return* 1 / (1 + np.exp(-*z*))  class CustomLogisticRegression:  def \_\_init\_\_(*self*, *learning\_rate*=0.01, *iterations*=1000):  *self*.learning\_rate = *learning\_rate*  *self*.iterations = *iterations*  *self*.weights = None  *self*.bias = None  *self*.cost\_history = []  def fit(*self*, *X*, *y*):  n\_samples, n\_features = *X*.shape  *self*.weights = np.zeros(n\_features)  *self*.bias = 0  *# Gradient descent*  *for* \_ *in* range(*self*.iterations):  model = np.dot(*X*, *self*.weights) + *self*.bias  predictions = sigmoid(model)  *# Compute the cost*  cost = (-1/n\_samples) \* np.sum(*y* \* np.log(predictions) + (1 - *y*) \* np.log(1 - predictions))  *self*.cost\_history.append(cost)  *# Compute gradients*  dw = (1/n\_samples) \* np.dot(*X*.T, (predictions - *y*))  db = (1/n\_samples) \* np.sum(predictions - *y*)  *# Update parameters*  *self*.weights -= *self*.learning\_rate \* dw  *self*.bias -= *self*.learning\_rate \* db  def predict\_prob(*self*, *X*):  model = np.dot(*X*, *self*.weights) + *self*.bias  *return* sigmoid(model)  def predict(*self*, *X*):  probabilities = *self*.predict\_prob(*X*)  *return* [1 *if* i > 0.5 *else* 0 *for* i *in* probabilities]    def accuracy(*self*, *y\_true*, *y\_pred*):  accuracy = np.sum(*y\_true* == *y\_pred*) / len(*y\_true*)  *return* accuracy  X = df.drop('num', *axis*=1).values  y = df['num'].values  X\_mean = np.mean(X, *axis*=0)  X\_std = np.std(X, *axis*=0)  X = (X - X\_mean) / X\_std  model = CustomLogisticRegression(*learning\_rate*=0.01, *iterations*=1000)  model.fit(X, y)  plt.plot(range(model.iterations), model.cost\_history, *label*='Cost Function')  plt.title('Cost Function Over Iterations')  plt.xlabel('Iterations')  plt.ylabel('Cost')  plt.legend()  plt.show()  predictions = model.predict(X)  accuracy = model.accuracy(y, predictions)  print(f'Training Accuracy: {accuracy \* 100:.2f}%')  new\_patient = np.array([[61, 0, 3, 154, 210, 1, 0, 130, 0, 1.5, 2, 2, 3]])  new\_patient = (new\_patient - X\_mean) / X\_std *# Standardize patient data*  prediction = model.predict(new\_patient)  prob = model.predict\_prob(new\_patient)  print(f'Predicted class for the new patient: {prediction[0]}')  print(f'Probability of having heart disease for the new patient: {prob[0]:.4f}') |
| Result(Captured images) |
| A screen shot of a computer screen  Description automatically generated  A screen shot of a computer  Description automatically generated  A screen shot of a computer program  Description automatically generated |

**Note**

1. Submit the file to e-class as pdf

2. Specify your pdf file name as “hw1\_<StudentID>\_<Name>.pdf”

Ex) hw1\_2000123456\_홍길동.pdf