To,

IITD-AIA Foundation on Smart Manufacturing

Subject: Weekly Progress Report for Week-9

Respected sir,

Following is the required progress report to the best of my knowledge considering relevant topics to be covered.

@1 Naive Bayes

@2 Principal Component Analysis .

@ 3 Applications of PCA

@ 4 Reading images and displaying images.

@ 5 Data preprocessing steps .

@ 6 Understanding haar classifiers

Day 1:

Naive Bayes

Naive Bayes is a straightforward classification algorithm that assumes features are unrelated.

It calculates probabilities to predict class labels based on observed features.

Naïve Bayes algorithms are often used in sentiment analysis, spam filtering, recommendation systems, etc. They are quick and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent.

Why should we use Naive Bayes ?

- It is extremely fast for both training and prediction.
- It provide straightforward probabilistic prediction.
- It is often very easily interpretable..
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Day 2

Principal Component Analysis .

(PCA) in machine learning is a technique for reducing the dimensionality of data while retaining important information.

Steps :

1. Standardization: Standardize data to have zero mean and unit variance.
2. Covariance Matrix: Calculate the covariance matrix of standardized data.
3. Eigenvalue Decomposition: Find eigenvalues and eigenvectors of the covariance matrix.
4. Eigenvalue Sorting: Sort eigenvalues in descending order to identify principal components.
5. Dimensionality Reduction: Choose top eigenvectors to retain desired variance.

6. Projection: Transform data by projecting onto selected principal components.
7. Interpretation: Principal components represent meaningful data directions.

Algorithm points:

1. Standardize: Make sure data has zero mean and unit variance.
2. Covariance: Calculate the covariance matrix of standardized data.
3. Eigenvectors: Find eigenvectors and eigenvalues of the covariance matrix.
4. Transform: Create a new data matrix by projecting onto selected eigenvectors.
5. Interpret: Understand data patterns using the transformed components.

Reference:

https://www.javatpoint.com/principal-component-analysis

Day 3

Applications of PCA :

- PCA is used to visualize multidimensional data.
- It is used to reduce the number of dimensions in healthcare data.
- PCA can help resize an image.
- It can be used in finance to analyze stock data and forecast returns.
- PCA helps to find patterns in the high-dimensional datasets.

Steps :

1. Normalize the Data
2. Build the Covariance Matrix
3. Find the Eigenvectors and Eigenvalues
4. Sort the Eigenvectors
5. Select Principal Components
6. Class Separation Using 2 Features
7. Plot the Correlation Matrix
8. Normalize the Data for PCA
9. Describe the Scaled Data
10. Import the PCA Module and Plot the Variance Ratio
11. Transform the Data
12. Visualize with Principal Components

Reference:

https://www.enjoyalgorithms.com/blog/principal-component-analysis-in-ml/

Day4:

Reading images and displaying images.

Reading images :

In OpenCV, the cv2.imread() function is used to read images from files. It takes the image file path as an argument and returns a NumPy array representing the image. By default, it reads images in the Blue-Green-Red color format.

Syntax for this:

import cv2

image = cv2.imread('image_path.jpg')

Displaying Images:

To display images, you can use the cv2.imshow() function. It takes two arguments: the window name and the image to be displayed. However, for the image to be displayed properly, you need to add a few more lines of code to handle user interaction and window destruction.

Syntax for this:

```
import cv2

image = cv2.imread('image_path.jpg')

cv2.imshow('Image', image)

cv2.waitKey(0)  # Waits for a key press

cv2.destroyAllWindows()  # Closes all windows
```

Reference :

https://www.cis.rit.edu/class/simg211/pgm_ex8.html

Day 5:

Data preprocessing steps:

```
Importing the libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('user_data.csv')
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

References:

https://www.youtube.com/watch?v=3I8oX3OUL6I&pp=ygUWIE5haXZlIEJheWVzIGluIFB5dGhvbg%3D%3D

https://www.kaggle.com/discussions/general/230154

Day 6:

Understanding haar classifiers:

Think of them as pattern-searching pros that help computers recognize objects in pictures.

By learning from examples, they become experts at finding things like faces or eyes

Points related to this are:

- Pattern Recognition: Haar classifiers spot specific shapes in images.
- Quick Math: They use shortcuts to find shapes fast.
- Learning from Examples: Trained on pictures with and without the shape.
- Tell the Difference: Learn what the shape looks like vs. what it doesn't.
- Step-by-Step: Organized stages help them decide.
- Fast Filtering: Ignore areas without the shape.
- Real-Time Ready: Speedy for live video tasks.
- Viola and Jones Method: Well-known for spotting faces.
- Adaptable: Not just faces, can find other things too.

Haar classifiers are quick tools that learn to find shapes in pictures and are often used for faces and other things in real-time task

Detecting faces in a group photo:

Look for simple patterns like edges or lines in the image.

Train with pictures of faces and non-faces.

Calculate pixel intensity differences for the defined patterns.

Create detectors for each pattern (e.g., a line across the eyes).

Combine detectors into a strong face recognizer.

Organize recognizers in a sequence, discarding non-faces early.

Slide a window over the photo, checking if it matches.

Repeat at different sizes to spot faces of all sizes.

Remove extra detections of the same face.

Reference :

https://www.analyticsvidhya.com/blog**/2022/04/object-detection-using-haar-cascade-opencv/**