Computer Vision

CVI620

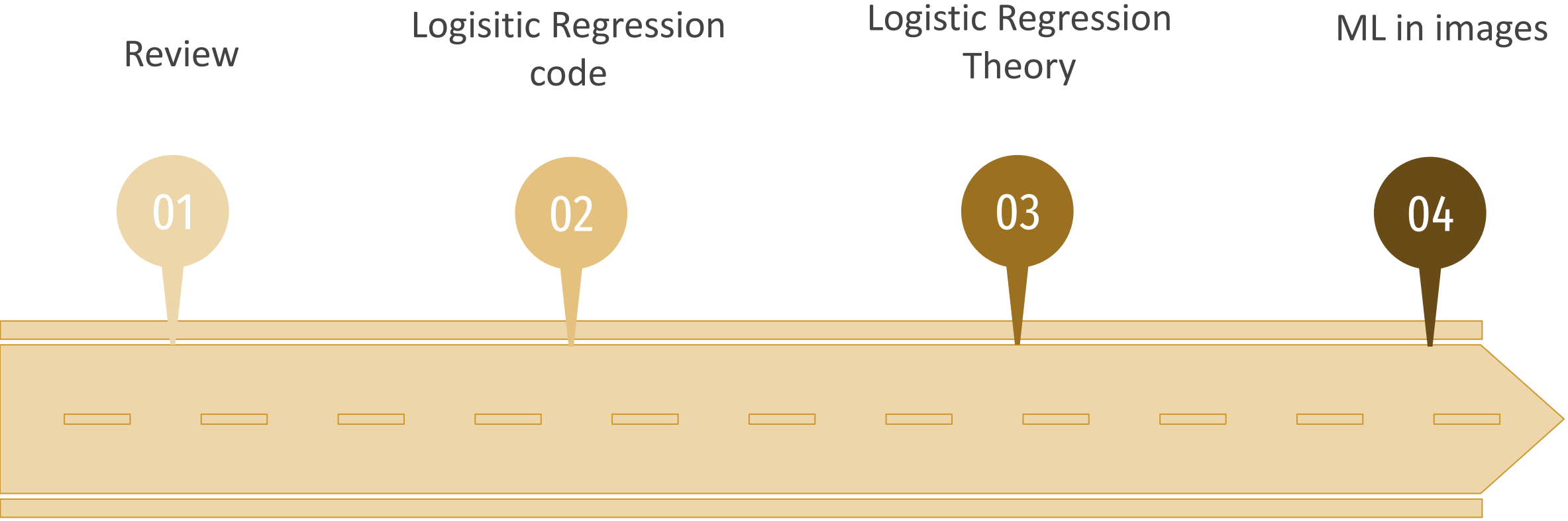
Session 18
03/2025

What is Left?

10 sessions

1. Optimization and Loss Function
2. Code + Logistic Regression
3. ML and Images
4. Perceptron and Neural Networks
5. Deep Neural Networks
6. Convolution Neural Networks (CNN)
7. Advanced CNNs
8. Project
9. Segmentation
10. Introduction to object detection and image generation methods with AI
11. Project

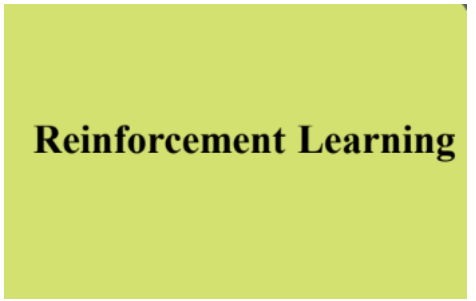
Agenda


Review
Logisitic Regression code
Logistic Regression Theory
ML in images
01
02
03
04

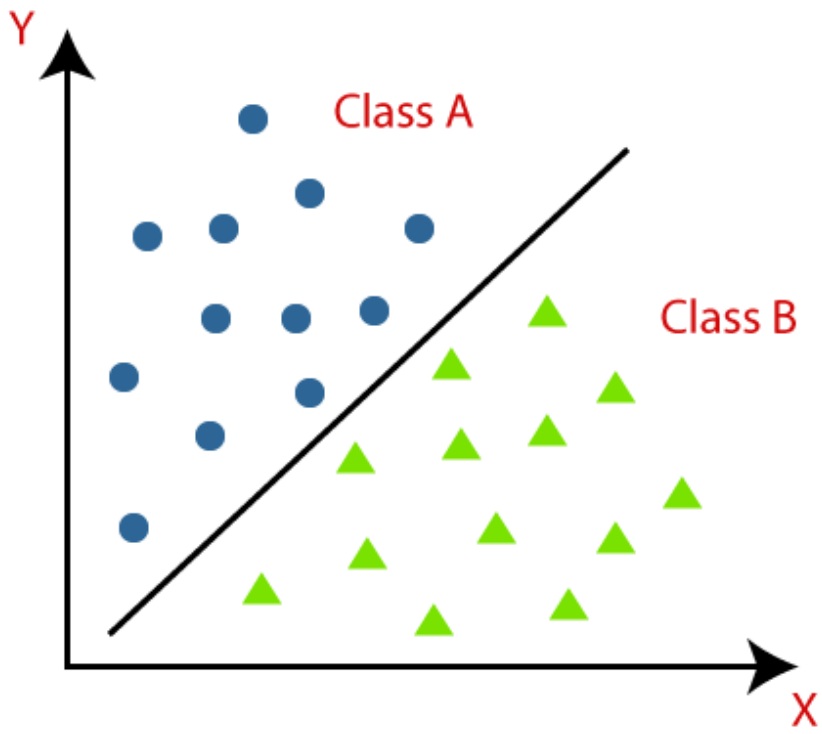
Terms overview

- Train-Test sets
- Normalization
- KNN for classification
- Linear regression for prediction
- Gradient Descent for prediction
- Epoch
- Batch size
- Accuracy
- Loss function

ML Algorithm Categorizations

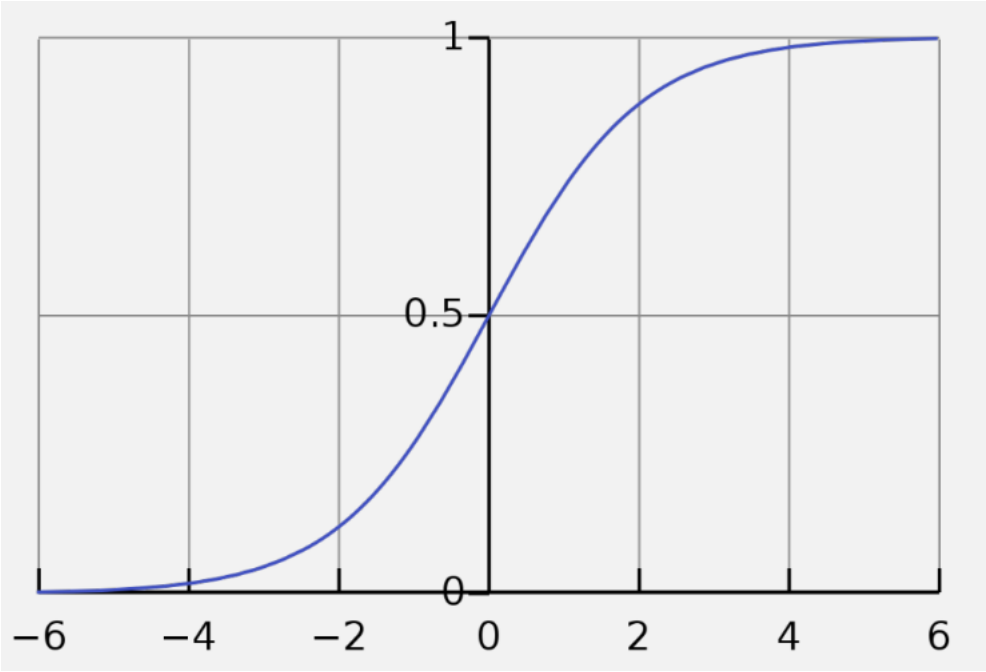


Logistic Regression



```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('S17/diabetes.csv')

zero_not_accepted = ['Glucose', 'BloodPressure',
                     'SkinThickness', 'Insulin', 'BMI']

for columns in zero_not_accepted:
    df[columns] = df[columns].replace(0, np.nan)
    mean = int(df[columns].mean(skipna=True))
    df[columns] = df[columns].replace(np.nan, mean)


X = df.drop(columns=['Outcome'])
y = df['Outcome']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

model = LogisticRegression()
model.fit(x_train, y_train)

preds = model.predict(x_test)
print(accuracy_score(y_test, preds))
```

Logistic Function



$$y = \frac{1}{1 + e^{-x}}$$

$$e \approx 2.71828$$

Where do you think it is useful?

Classification

$$out = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_{n-1} x_{n-1}$$

$$y = \frac{1}{1 + e^{-out}}$$

Logistic Regression Loss Function

If y=1 and y'=0 or the opposite -> loss should be high

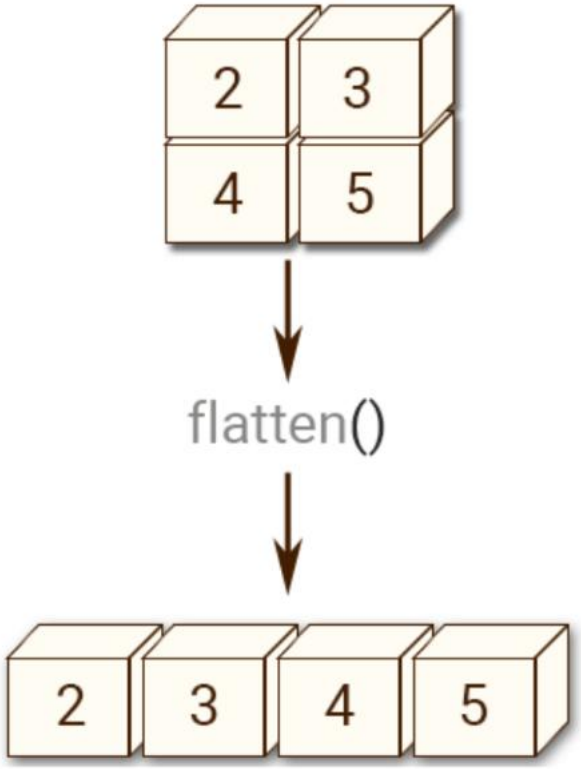
Else -> loss low

$$Loss = -ylog(y') - (1-y)\log(1-y')$$

Loss can be anything
in optimization!

ML in Images

ML in Images



Fire Detection

```python
import glob
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
from joblib import dump

def load_data():
    data_list = []
    labels = []

    for i, address in enumerate(glob.glob('S18/fire_dataset\\*\\*')):
        img = cv2.imread(address)
        img = cv2.resize(img, (32,32))
        img = img/255
        img = img.flatten()

        data_list.append(img)
        label = address.split("\\")[-1].split(".")[0]
        labels.append(label)

        if i % 100 == 0:
            print(f"[INFO]: {i}/1000 processed")

    data_list = np.array(data_list)

    X_train, X_test, y_train, y_test = train_test_split(data_list, labels, test_size=0.2)

    return X_train, X_test, y_train, y_test

X_train, X_test, y_train, y_test = load_data()
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)
print(accuracy_score(y_test, predictions))
dump(clf, 'fire_detection.z')
```

```python
import cv2
import numpy as np
import glob
from joblib import load

clf = load('S18/fire_detector.z')

for item in glob.glob("test_images\\*"):
    img = cv2.imread(item)
    r_img = cv2.resize(img, (32,32))
    r_img = img/255
    r_img = img.flatten()
    r_img = np.array([r_img])

    label = clf.predict(r_img)[0]

    cv2.putText(img, label, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 1.9, (0,255,0), 2)
    cv2.imshow()
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

Types of Preprocessing in Images

- Resize
- Normalization
- Flat


Gender Classification

MTCNN

- pip install mtcnn

Or use haarcascades