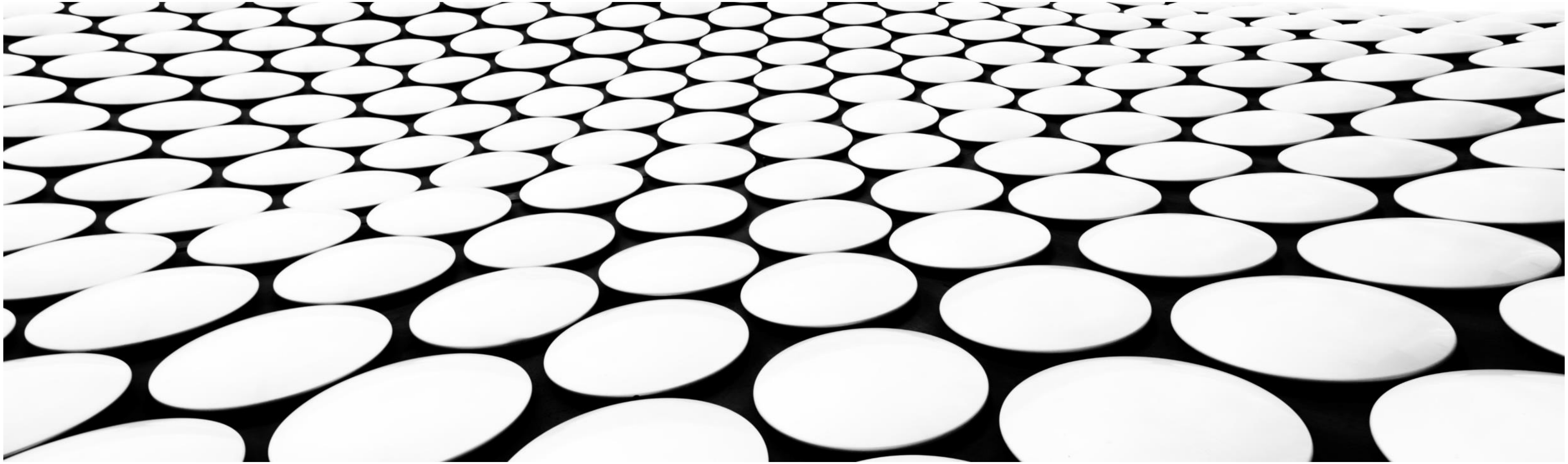
IMAGE CLASSIFICATION USING LOGISTIC REGRESSION

COMPUTER VISION – FINAL PROJECT



OUTLINE

- Dataset
- Preprocessing
- Logistic Regression
- Results
- Conclusion

DATASET

- Project 1: Detect images of smiles using factor analysis
- Smiles dataset from project 1
- Number of smiles: ~ 3500 images
- Number of frowns: ~ 9500 images
- Size of the images: 64x64 pixels
- Already greyscaled

Smile Image

Frown Image

OUTLINE

- Dataset
- Preprocessing
- Logistic Regression
- Results
- Conclusion

PREPROCESSING

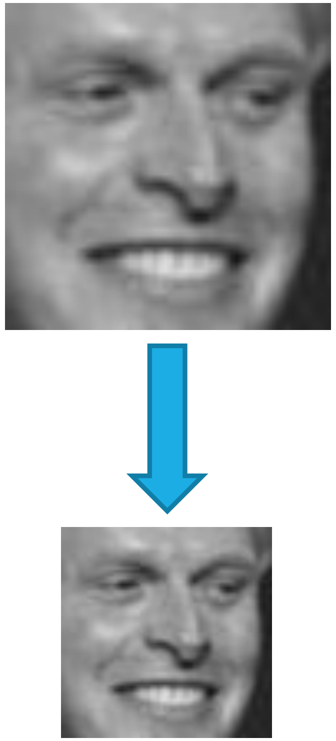
RESIZING THE IMAGES

- Logistic regression model trains faster with smaller images
- Resized images should not be too small

→ Could reduce the performance

→ There is no perfect size for images

Before: 64x64 Image



After: 16x16 Image

PREPROCESSING

NORMALIZATION

- Reduction of unwanted variations of:
  - Intensity
  - Contrast

→ All images have a similar data distribution

→ All images have a mean of 0 and a standard deviation of 1

Before

After

OUTLINE

- Dataset
- Preprocessing
- Logistic Regression
- Results
- Conclusion

LOGISTIC REGRESSION
IN GENERAL

- Classification algorithm to predict a binary outcome with the help of data

→ Data: Images of smiles and frowns

→ Binary outcome: Either smile or frown

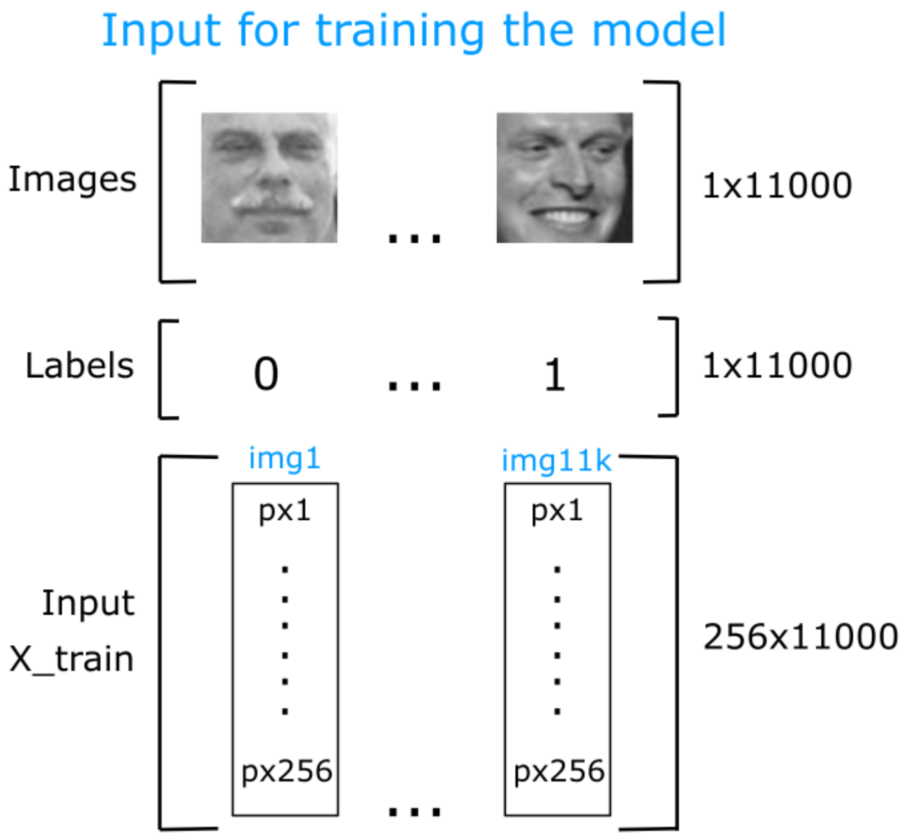
- No closed form solution using ML method

→ Iterative non-linear optimization is needed

→ Simple single layer network

LOGISTIC REGRESSION

INPUT



LOGISTIC REGRESSION

MODEL

- Parameters: Weights and bias
- Output z:

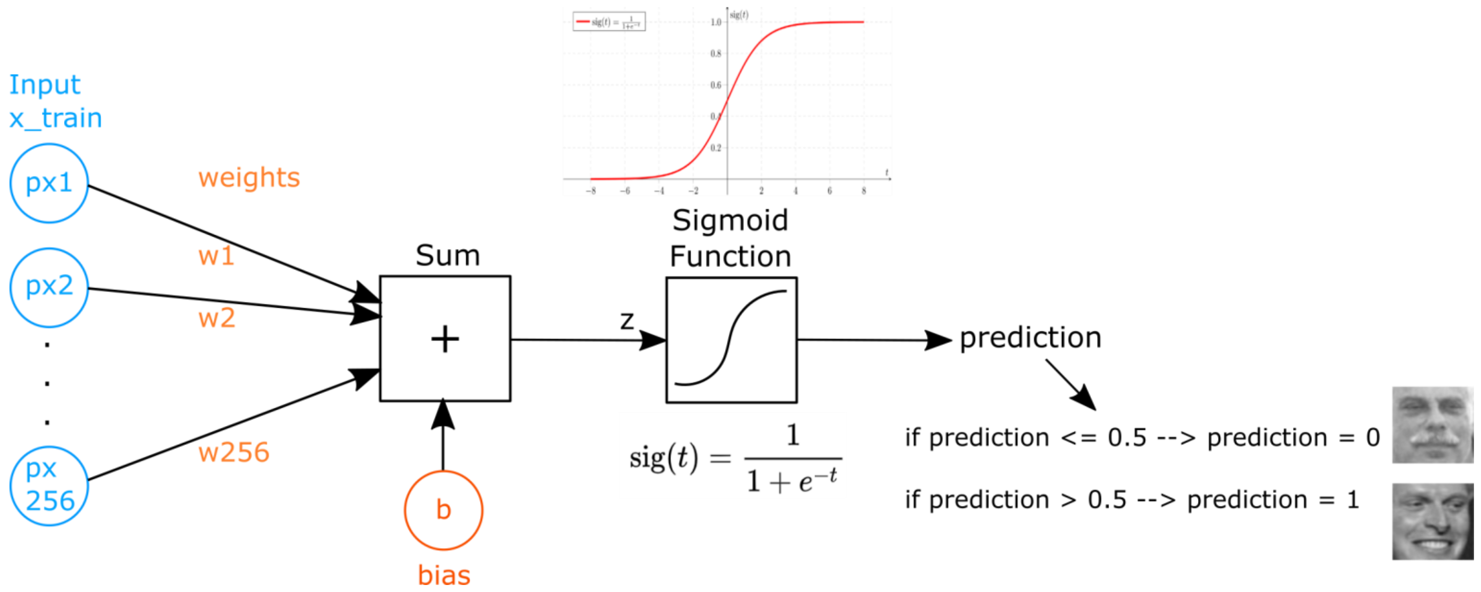
$$z = w^T x + b$$

- Prediction:

$$prediction = sigmoid(z)$$

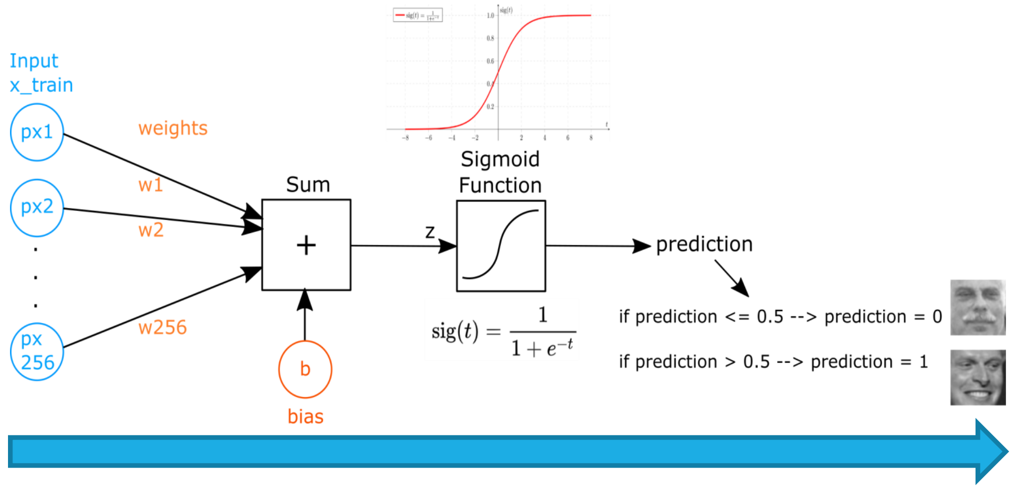
→ Prediction is between 0 and 1 → Probability

→ Sigmoid function is derivative → Gradient Descent



LOGISTIC REGRESSION

TRAINING



- **Forward Propagation:**

1. Compute output z
2. Put z into the sigmoid function to get the prediction
3. Compute loss function for each image
4. Compute cost function which is the summation of all loss functions

$$J = -\frac{1}{m}[\sum_{i=1}^{m} trueLabel_i * log(prediction_i) + (1 - trueLabel_i) * log(1 - prediction_i)]$$

→ Cost is the error

LOGISTIC REGRESSION
TRAINING

- **Backward Propagation:**
- Decrease the cost (error)

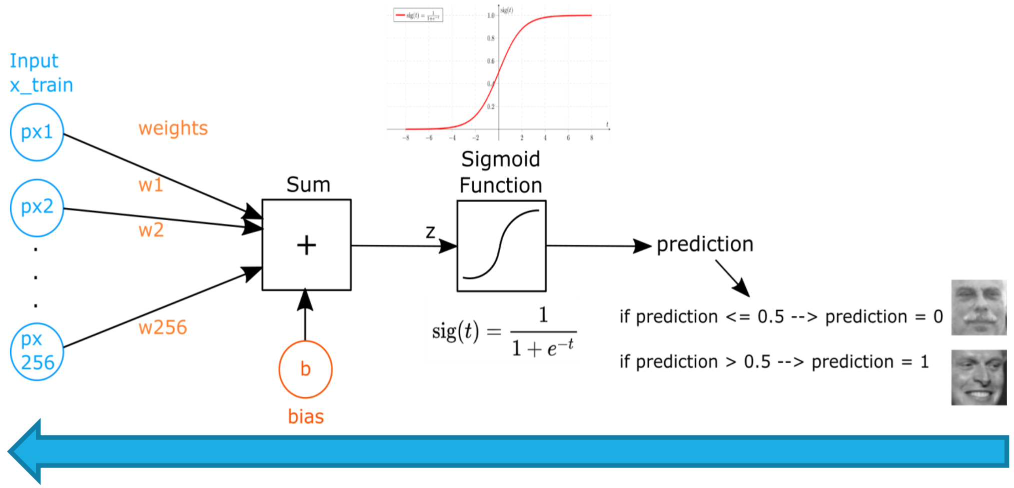
→ The weights and bias have to be updated

→ Gradient Descent Algorithm

- Steps:

1. Take derivative of the cost function for weights and bias
2. Update the weights and bias

$$w = w - \alpha \frac{\partial J(w,b)}{\partial (w,b)} \qquad b = b - \alpha \frac{\partial J(w,b)}{\partial (w,b)}$$



$$\frac{\partial J}{\partial w} = \frac{1}{m} x(prediction - trueLabel)^T$$
$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (prediction_i - trueLabel_i)$$

OUTLINE

- Dataset
- Preprocessing
- Logistic Regression
- Results
- Conclusion

RESULTS

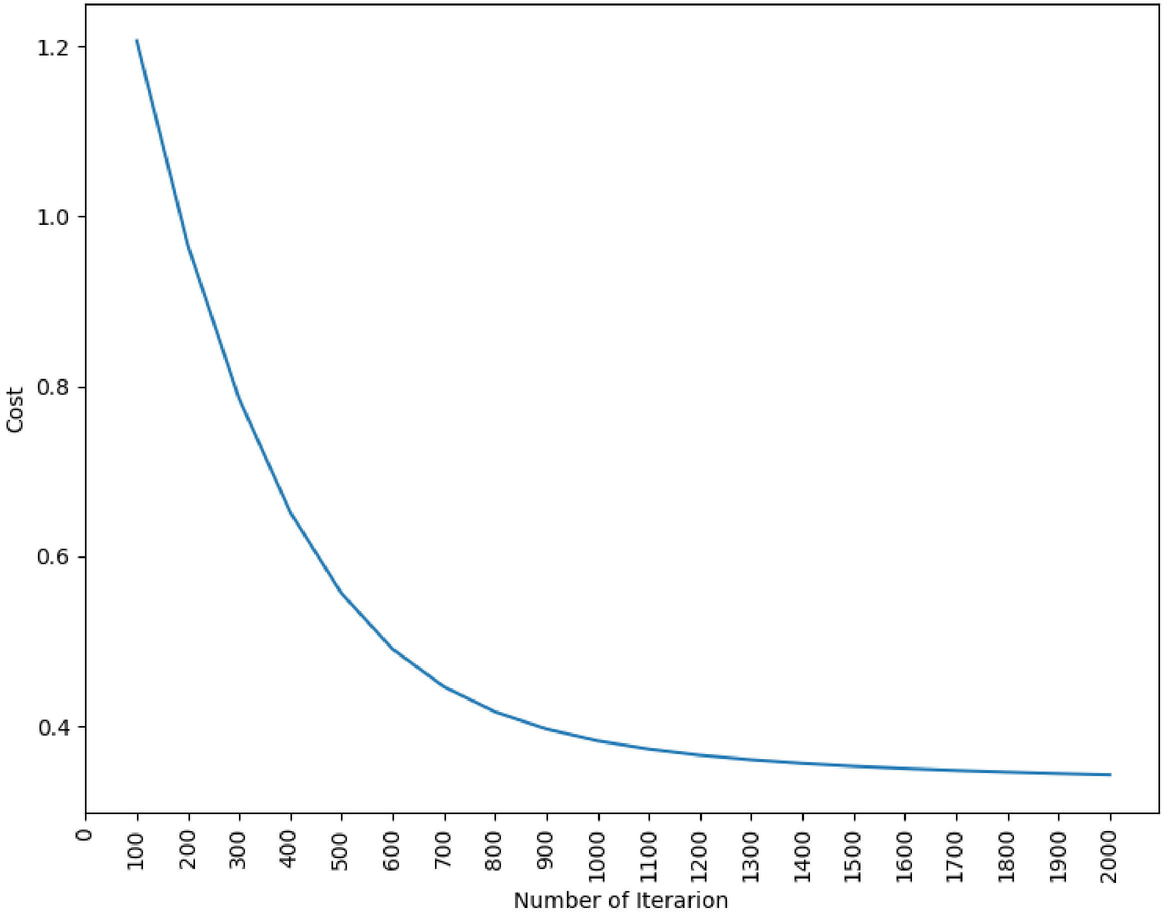
- Split data into train (~ 11000 images) and test set (~ 2000 images)
- After trying different combinations of the learning rate and the sizes of the images

→ Learning rate = 10^(-5)

→ Resized image = 16x16 pixels

RESULTS

COST FUNCTION



RESULTS

COMPARISON WITH PROJECT 1

```
Test Accuracy: 84.88 %

Train Accuracy: 85.1 %
```

Accuracy of the logistic regression

```
smile_accuracy =

    0.6806

>> frown_accuracy

frown_accuracy =

    0.6684
```

Accuracy of the factor analysis

RESULTS
SUMMARY

- After 2000 iterations the cost function converges
- Accuracy of the logistic regression is better than the factor analysis

OUTLINE

- Dataset
- Preprocessing
- Logistic Regression
- Results
- Conclusion

CONCLUSION

- Logistic regression was implemented

→ Dataset from project 1 for a comparison

- Better performance than the factor analysis
- More potential

→ Change the initialization of the weights and bias

QUESTIONS?