Universidade do Minho

Mestrado em Matemática e Computação

Métricas em Machine Learning

Eigenfaces: reconhecimento de faces

Hugo Filipe de Sá Rocha (PG52250)

Simão Pedro Batista Caridade Quintela (PG52257)

Eduardo Teixeira Dias (PG52249)

Tiago Augusto Lopes Monteiro (PG52258)

2 de janeiro de 2024

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular de Métricas em Machine Learning foi-nos proposto, enquanto grupo, um trabalho que incide sobre o uso do algoritmo PCA para reconhecimento de faces. O algoritmo procura comprimir a informação das imagens reduzindo o número de dimensões do problema de forma a preservar uma boa parte da informação presente em cada imagem. Desta forma, a informação de pouca relevância é descartada e é possível fazer um reconhecimento fiável de novas faces (não presentes na base de dados) através de projeções sobre o espaço gerado pelas dimensões de maior importância.

1.1 Base de Dados

A nossa base de dados é composta por 9 pessoas onde atribuímos, a cada uma delas, uma pasta com 11 imagens onde 10 delas são usadas para treino e constituem a base de dados em si e uma delas é usada para teste/reconhecimento.

Capítulo 2

Concepção das soluções

2.1 Criação da base de dados no Python

Para criar base de dados no Python, fizemos uso da biblioteca PIL onde iterámos sobre as diferentes pastas e armazenámos a informação das imagens (convertidas a preto e branco) numa matriz **X**. Cada linha desta matriz, corresponde às 10800 componentes de informação de cada imagem, ou seja, visto termos 90 imagens para treino, a matriz **X** apresenta uma dimensão de **90x10800**.

```
1     base = [Image.open(f'database/p{i}_resized/p{i}_{j}.jpeg').convert('L') for i in
          range(1, num_cobaias+1) for j in range(1, num_pics+1)]
2
3     size = len(base)
4
5     X = np.array([base[i].getdata() for i in range(size)])
```

Após isto, centrámos todos os dados da matriz na média 0.

```
1     media = np.mean(X, 0)
2     phi = X-media
```

2.2 Cara média

Para efeitos de teste foi calculada a cara média da nossa base de dados e mostrada através do seguinte comando:

```
1     display(plt.matshow(np.reshape(media, (120, 90)), cmap='gray'))
```

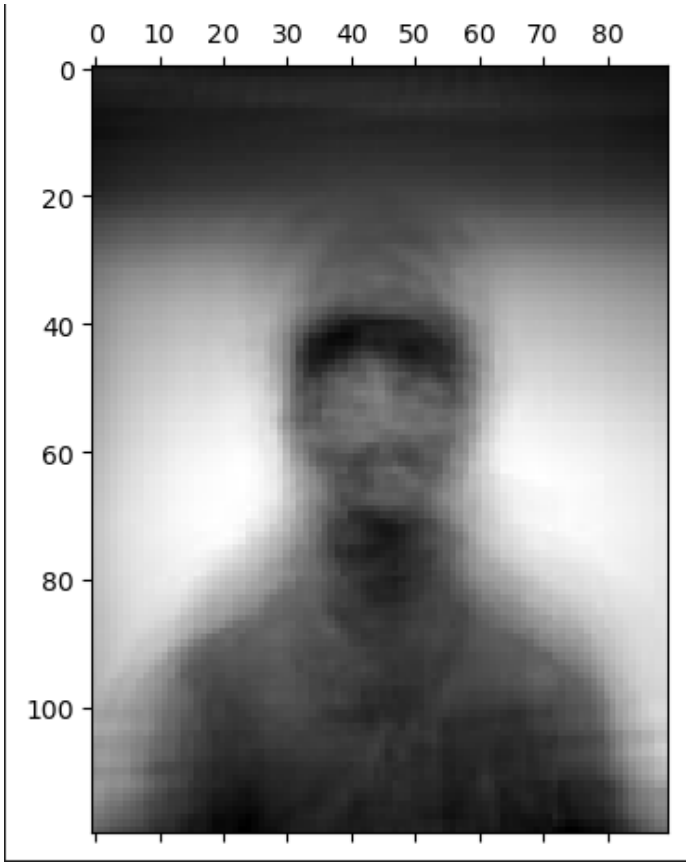

Figura 2.1: `Imagem da cara média.`

2.3 Aplicação do SVD

Para obtenção dos valores e dos vetores próprios da matriz com os dados, utilizou-se a decomposição **SVD** calculando-se logo de seguida o traço da matriz como sendo a soma dos valores próprios.

```
1    e_faces, sigma, v = np.linalg.svd(phi.transpose(), full_matrices=False)
2    val_prop = sigma*sigma
3    traco = sum(val_prop)
```

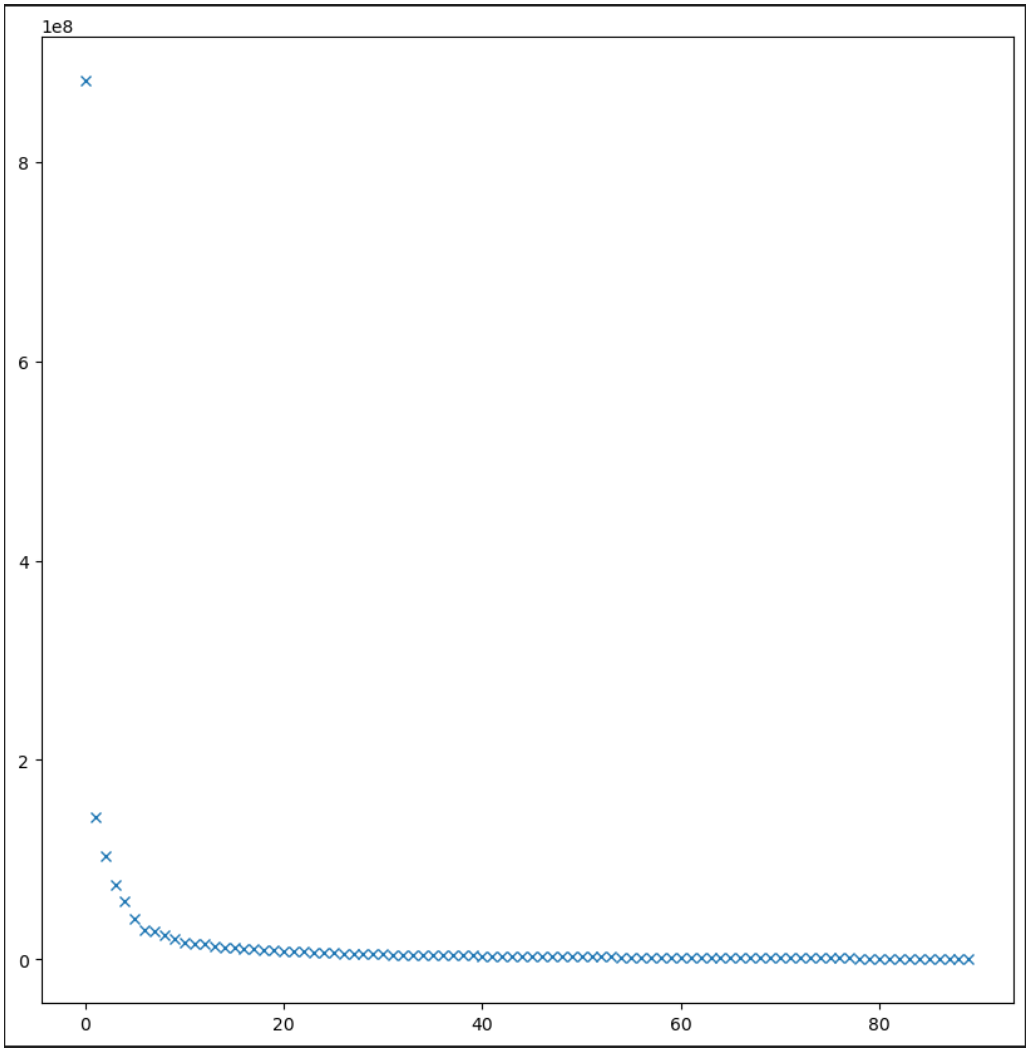

Figura 2.2: *Plot* dos valores próprios.

Por observação da imagem acima, decidimos selecionar através da aplicação da "regra do cotovelo", 7 valores próprios que preservam cerca de 77% da informação, ou seja, no contexto do problema, a dimensão será reduzida de 90 para 7.

2.4 Cálculo das projeções

Após a seleção do número de dimensões a utilizar, projetámos todas as imagens no espaço gerado pelos 7 vetores próprios associados aos 7 maiores valores próprios, através do seguinte comando:

```
1     coef_proj = [np.dot(phi[i], e_faces[:, 0:k]) for i in range(size)]
```

2.5 Reconhecimento de novas imagens

Para reconhecer um novo *input*, é necessário centrá-lo na média 0 e projetá-lo sobre o espaço gerado pelos vetores próprios selecionados. Depois disto, é necessário calcular a distância da projeção do novo *input* a todas as projeções da base de dados e selecionar aquela que minimiza a distância. Para isso, como pedido no enunciado, utilizámos duas métricas distintas: **métrica euclidiana** e **métrica de Mahalanobis**.

2.5.1 Métrica euclidiana

$$d(x, y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}$$

Cálculo das distâncias usando a métrica euclidiana e seleção da imagem mais próxima:

```
    test_coef_proj = np.dot(test_phi, e_faces[:, 0:k])
    dist = [np.linalg.norm(coef_proj[i] - test_coef_proj) for i in range(size)]
    d_min = np.min(dist)


    if d_min < 7000:
        pasta_rec = np.floor(np.argmin(dist)/10)+1
        print(f'Pasta p{pasta_rec}\ndist={d_min}')
    else:
        print('Who the hell are you?')
```

2.5.2 Métrica de Mahalanobis

$$d(x, y) = \sum \frac{1}{\lambda_i}(x_i - y_i)^2$$

Definição da função para o cálculo da distância de Mahalanobis entre duas faces:

```
    def mahalanobis(face1, face2):
        l = []

        for i,elem in enumerate(face1-face2):
            l.append((1/val_prop[i])*elem*elem)

        return sum(l)
```

Cálculo das distâncias usando a métrica de Mahalanobis e seleção da imagem mais próxima:

```
    test_coef_proj = np.dot(test_phi, e_faces[:,0:k])
    dist = [mahalanobis(coef_proj[i], test_coef_proj) for i in range(size)]
    d_min = np.min(dist)

    pasta_rec = np.floor(np.argmin(dist)/10)+1
    print(f'Pasta p{pasta_rec}\ndist={d_min}')
```

2.6 Validação/teste do modelo

Para validar o modelo feito, implementámos um ciclo que seleciona todas as imagens destinadas ao reconhecimento e projeta-as sobre o espaço dos vetores próprios reconhecendo-as usando as duas métricas no espaço das projeções: **euclidiana** e **Mahalanobis**.

```
    dic = {}

    for i in range(1,num_cobaias+1):
        pasta_input = i
        input_img = Image.open(f"database/p{pasta_input}_resized/p{pasta_input}
            _reconhecimento.jpeg").convert('L')

        gamma = np.array(input_img.getdata())
        test_phi = gamma - media

        test_coef_proj = np.dot(test_phi, e_faces[:, 0:k])

        dist = [np.linalg.norm(coef_proj[i] - test_coef_proj) for i in range(size)]
        d_min = np.min(dist)
        pasta_rec_eucl = np.floor(np.argmin(dist)/10)+1

        dist = [mahalanobis(coef_proj[i], test_coef_proj) for i in range(size)]
        d_min = np.min(dist)
        pasta_rec_maha = np.floor(np.argmin(dist)/10)+1

        dic[i] = (pasta_rec_eucl, pasta_rec_maha)


    print("PASTA INPUT|RECONHECIMENTO C/ DIST. EUCL.|RECONHECIMENTO C/ DIST. MAHA.")
    for k in dic:
        print(f"  {int(k)}   |    {int(dic[k][0])}    |      {int(dic[k][1])}")
```

PASTA INPUT	RECONHECIMENTO C/ DIST. EUCL.	RECONHECIMENTO C/ DIST. MAHA.
1	1	1
2	7	7
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9

Figura 2.3: `Tabela de reconhecimentos.`

Como podemos ver na tabela de resultados, o modelo consegue reconhecer corretamente, usando ambas as métricas, 8 dos 9 sujeitos. No entanto, o sujeito correspondente à pasta 2 é reconhecido incorretamente para ambas as métricas. Esta falha no reconhecimento dever-se-á ao facto de preservarmos 77% da informação ao invés de uma percentagem mais elevada.