Introduction to Machine Learning
Problem Set 3: SVM and PCA

Problem 1

Consider the following training data,

class	x_1	x_2
+	2	2
+	3	3
+	3	0
−	0	0
−	2	0
−	0	2

(a) Plot these six training points. Are the classes $\{+, -\}$ linearly separable?

(b) Construct the weight vector of the maximum margin hyperplane by inspection and identify the support vectors.

(c) If you remove one of the support vectors, does the size of the optimal margin decrease, stay the same, or increase?

Problem 2

Beginning with the optimization problem on slide 11 on the slides "SVM: Part 1," prove that the solution to the hard-margin optimization problem on page 13 provides a separating hyper-plane with maximum margin

(a) Suppose that w, b is optimal for the hard margin optimization problem on page 13. We must show that w, b gives a hyperplane that maximizes the margin. First show that the margin for w, b (distance from hyperplane to nearest training example) is $1/\|w\|$. To do this, you'll want to use the explicit expression derived in class for the distance between a training example $x^{(i)}$, $y^{(i)}$ and the the hyperplane defined by w and b. You'll also need to make use of the fact w, b satisfy the constraints in the hard margin optimization problem (since it is feasible) and that meets at least one of the constraints with equality (since it is optimal).

(b) Now let z, d be any other separable hyperplane, and let M denote its margin for the data set. Define $z' = z/\|z\|M$ and $d' = d/\|z\|M$. Show that z', d' is a feasible solution for the hard-margin optimization problem, and therefore $\|w'\|^2 \leq \|z'\|^2$ and hence $\|w\| \leq \|z'\|$.

(c) Use (a) and (b) to show that margin for $w, b(= 1/\|w\|)$ is greater than or equal to the margin for $z, d(= M)$.

Hint: Feel free to take a look at Andrew Ng's lecture notes. If you borrow material from those notes, be sure to use the notation used in class.

Problem 3

Consider a supervised machine learning problem with two features (x_1, x_2) and 4 training points $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$:

data	class	x_1	x_2
$x^{(1)}$	$+$	0	1
$x^{(2)}$	$+$	2	3
$x^{(3)}$	$-$	0	3
$x^{(4)}$	$-$	2	1

Denote w_0 & w_1 for the weights and b for the bias.

1. Argue that there is no solution that satisfies the constraints for the hard-margin SVM problem.

2. Show that there is a solution for the soft margin SVM problem. Explicitly provide such a solution $(w_0, w_1, b, \xi^{(1)}, \xi^{(2)}, \xi^{(3)}, \xi^{(4)})$.

Problem 4

We use the Olivetti face dataset[1]. The data contains 400 face images of size 64×64. In faces.csv, each row represents a face image. The first 64 values represent the first column of the image, and the next 64 values represent the second column and so on.

We provide some sample code. The sample code shows how to load the data and display the first image. There are also hints in the comments which will aid you in writing your own code.

In the sample code, you will find the lines "Your Code Here", below which you may start writing your code.

Your tasks are as follows,

1. Display a face image chosen randomly from the 400 images.
2. Compute and display the mean of the faces. Subtract the mean from each faces to get the "centered faces".
3. For each centered face, compute its covariance matrix. Then, compute the average covariance matrix V by averaging the 400 covariance matrices. Note that V is a 64×64 matrix.
4. Calculate the eigenvalues and eigenvectors of the covariance matrix V using the method stated in the eigenface tutorial.
5. Display the first 16 principal components.
6. Reconstruct the first face using the first two principal components.
7. Randomly choose a face, reconstruct it using 5, 10, 25, 50, 100, 200, 300, 399 principle components, and show the reconstructed images.
8. Recall that in principal component analysis, the total variance of the data is given by the sum of all eigenvalues, i.e., $\sum_{j=1} \lambda_j$. The proportion of variance explained by the i-th principal component is given by $\frac{\lambda_i}{\sum_j \lambda_i}$. Plot the proportion of variance explained by all the principal components.

Eigenfaces Tutorial

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The main idea is to represent a face using a linear composition of base features or images (called eigenface). For more information on eigenfaces, see [1] or the eigenfaces wikipedia page.

[1] https://cs.nyu.edu/ roweis/data.html

Suppose we have a set of m images $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(m)}$. Each image is represented by an $n \times n$ matrix, where

$$\mathbf{x}^{(r)} = \begin{bmatrix} p_{11}^{(r)} & p_{12}^{(r)} & \cdots & p_{1n}^{(r)} \\ p_{21}^{(r)} & p_{22}^{(r)} & \cdots & p_{2n}^{(r)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1}^{(r)} & p_{n2}^{(r)} & \cdots & p_{nn}^{(r)} \end{bmatrix}_{n \times n}, \text{ for all } r = 1, 2, \ldots, m.$$

$0 \leq p_{ij}^{(r)} \leq 255$ represents the pixel intensity, and $n \times n$ represents the numbers of pixels in the image. Now we wish to change the representation of our image into a vector of dimension n^2, we do this by concatenating all the columns of the matrix $\mathbf{x}^{(r)}$ as follows,

$$\mathbf{x}^{(r)} = \begin{bmatrix} p_{11}^{(r)} \\ p_{21}^{(r)} \\ \vdots \\ p_{n1}^{(r)} \\ p_{12}^{(r)} \\ p_{22}^{(r)} \\ \vdots \\ p_{n2}^{(r)} \\ \vdots \\ p_{1n}^{(r)} \\ p_{2n}^{(r)} \\ \vdots \\ p_{nn}^{(r)} \end{bmatrix}_{n^2 \times 1},$$

where $r = 1, \cdots, m$ and $0 \leq p_{ij}^{(r)} \leq 255$. Our goal, therefore, is to extract a lower dimension set of useful features out of these n^2-dimensional vectors.

Since we are much more interested in the characteristic features of those faces, let's subtract everything that is common among them, i.e., the average face. The average face of the image can be defined as $\mathbf{m} = \frac{1}{m} \sum_{r=1}^{m} \mathbf{x}^{(r)}$. We then redefine:

$$\mathbf{x}^{(r)} \leftarrow \mathbf{x}^{(r)} - \mathbf{m}.$$

So now, $\frac{1}{m} \sum_{r=1}^{m} \mathbf{x}^{(r)} = 0$. In order to find the principal components, we will attempt to find the eigenvectors $\mathbf{z}_j$ and the corresponding eigenvalues λ_j of the covariance matrix

$$\mathbf{V} = \frac{1}{\mathbf{m}} \sum_{r=1}^{m} \mathbf{x}^{(r)} \mathbf{x}^{(r)^T} = A^T A$$

where the matrix $A^T = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}]$. However, because the dimension of the matrix V is $n^2 \times n^2$, the task of determining the eigenvalues and eigenvactors is intractable (to put things into perspective, suppose our set of images are 64×64, then the matrix V would have $16,777,216$ elements). Fortunately, we can make this problem more computationally feasible by solving a much smaller $m \times m$ matrix $L = AA^T$.

Denote the eigenvectors of matrix L by $\mathbf{v}_j$, where $j = 1, \ldots, m$. Observe that for $L\mathbf{v}_j = \lambda_j \mathbf{v}_j$, we have

$$\begin{aligned} A^T L \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \\ A^T A A^T \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \\ V A^T \mathbf{v}_j &= \lambda_j A^T \mathbf{v}_j \end{aligned}$$

and hence $\mathbf{z}_j = A^T \mathbf{v}_j$ are the eigenvectors and eigenvalues of V, respectively. Thus, we can find the eigenvectors of V by first finding the eigenvectors of L, then multiplying each eigenvector $\mathbf{v}_j$ by A^T. One final step is that $\mathbf{z}_j$ needs to be normalized, i.e., $||\mathbf{z}_j|| = 1$.

The eigenvectors $\mathbf{z}_j$ are the eigenfaces. You can view these faces by scaling them to 255. We can discard the components with the smallest eigenvalues. Only the $k (k \leq m)$ ones with the largest eigenvalues (i.e., only the ones making the greatest contribution to the variance of the original images set) and chuck them into the matrix

$$U = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k]_{n^2 \times k}.$$

Once we have a new face image $\mathbf{x}$, it can then be transformed into its eigenface components by a simple operation

$$\Omega = U^T(\mathbf{x} - \mathbf{m}) = [\omega_1, \omega_2, \cdots, \omega_k]^T.$$

Notice that we have reduced an image of size $n \times n$ into a vector length k. To approximate the original image $\mathbf{x}'$, all we have to do is to project it into the face space and adjust for the mean by

$$\mathbf{x}' = \mathbf{m} + U\Omega = \mathbf{m} + \sum_{j=1}^{k} \omega_j \mathbf{z}_j.$$

References

[1] M.A. Turk, A.P. Pentland, "Face recognition using eigenfaces", Proceedings CVPR 91'., IEEE Conference on Computer Vision and Pattern Recognition. pp. 586-591.