CSCI 4587/5587, Fall 2023
Machine Learning I
Study Guide for Test#4

(Please don't distribute this study guide.
The guide is for your study purpose only)

Chapter 04

01. (a) What is unsupervised learning? (b) Given the square matrix A= $\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}$, show your detailed works to compute the corresponding Eigen-values and their corresponding Eigen-vectors.

Ans: 01 (a) [Chapter #4, Page 1].

Ans: 01 (b)

We know Av = λv, where v is the non-zero column vector, and λ is the corresponding Eigen-value.

Therefore, using the given Matrix of A, from Av = λv, we can write:

⇨ (A-λI) ***v*** = 0 … (G)
[I = identity matrix, we need to introduce I since λ is scalar]

⇨ (A-λI) = 0 [∵ ***v*** is a non-zero matrix]

⇨ $\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = 0$

⇨ $\begin{pmatrix} 3-\lambda & 2 \\ 2 & 6-\lambda \end{pmatrix} = 0$

⇨ (3-λ) (6-λ)-4=0

⇨ λ^2- 9λ+14=0

⇨ λ =7, 2; being larger λ =7 is the 1st principal eigenvalue.

Now, the question is what would be the corresponding ***v***s. of λ = 7, 2.

Using λ = 7 and Equation (*i*):

$$\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 7 \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

=>

⇨ we get: $3v_1 + 2v_2 = 7v_1$ and $2v_1 + 6v_2 = 7v_2$

⇨ From both equations, we get: $2\boldsymbol{v}_1 = \boldsymbol{v}_2$ => $\frac{v_1}{v_2} = \frac{1}{2}$ => $\boldsymbol{v} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

Further, to make it a unit vector, we can divide its components by $\sqrt{5}$. Thus, the principal (unit) eigenvector can be $\begin{pmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix}$.

Similarly, using $\lambda = 2$:

We get, (unit) eigenvector $\begin{pmatrix} \frac{2}{\sqrt{5}} \\ -\frac{1}{\sqrt{5}} \end{pmatrix}$

02. Write down the PCA algorithm.
Ans:

Algorithm: PCA

Inputs: Parameter q (desired lower dimension) where, $q < p$,
dataset D $\{(x_1), \ldots, (x_N)\}$.

1. For j=1: p // Compute the means or centroids of all the p features

   mean, $\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$

   End
2. For all, update $x = (x - \mu)$ // this is to have 0 means.
3. Scale, $x = \frac{x}{\sigma}$ // where, σ is the standard deviation. This step is needed when the features are in different scales.
4. Compute, Sigma (S) = $\frac{1}{N}\sum_{i=1}^{N} x_i x_i^T$
5. [U,M,V] =svd (Sigma); // Octave/MATLAB function
6. U_q = U(:, 1:q);
7. $\tilde{x} = U_q^T * x$; // Projected approximate values ($\tilde{x}$) are computed.

---

03. What is Lagrange-multiplier or, Lagrange-optimization? Describe using an example.

Ans: *Lagrange Multipliers*, also known as *undetermined multipliers*, are used to find the stationary points of a function of several variables subject to one or more constraints.

Consider the problem of finding the maximum of a function $f(x_1, x_2)$ subject to a constraint relating x_1 and x_2, which we write in the form.

$$g(x_1, x_2) = 0 \qquad \text{(A.1)}$$

To solve, a more elegant and often simpler approach is based on the introduction of a parameter λ called a *Lagrange multiplier*. Then, the problem can be solved by optimizing the *Lagrangian function*

$$L(\mathrm{x}, \lambda) \equiv f(\mathrm{x}) + \lambda\, g(\mathrm{x}) \qquad \text{(A.2)}$$

We then differentiate A.2 with respect to the variables (i.e., x, λ) and equate the equation(s) to zero, and then we get others equations, using which we can get suitable solutions.

As an example, suppose we wish to find the stationary points of the function $f(x_1, x_2) = 1 - x_1^2 - x_2^2$ subject to the constraint $g(x_1, x_2) \Rightarrow x_1 + x_2 - 1 = 0$. The corresponding *Lagrangian function* is given by:

$$L(\mathrm{x}, \lambda) = 1 - x_1^2 - x_2^2 + \lambda\,(x_1 + x_2 - 1) \qquad \text{(A.3)}$$

The conditions for this *Lagrangian* to be stationary with respect to x_1 , x_2 and λ give the following coupled equations:

$$-2x_1 + \lambda = 0 \qquad \text{(A.4)}$$
$$-2x_2 + \lambda = 0 \qquad \text{(A.5)}$$
$$x_1 + x_2 - 1 = 0 \qquad \text{(A.6)}$$

The solution of these equations then gives the stationary point as $(x_1^*, x_2^*) = \left(\frac{1}{2}, \frac{1}{2}\right)$ and the corresponding value for the Lagrange multiplier is λ = 1.

04. (a) What are manifold and manifold learning? (b) describe manifold learning in terms of machine learning (c) How does Locally Linear Embedding (LLE) work?

Ans: (a) A d-dimensional manifold is a d-dimensional shape that can be bent and twisted in a higher-dimensional space. More generally, a d-dimensional manifold is a part of an *n*-dimensional space (where $d < n$) that locally resembles a *d*-dimensional hyperplane. Many dimensionality reduction algorithms work by modeling the manifold on which the training instances lie; this is called manifold learning. It relies on the manifold assumption, also

called the manifold hypothesis, which holds that most real-world high-dimensional datasets lie close to a much lower-dimensional manifold.

(b) In terms of machine learning, although the data points may consist of thousands of features, they may be described as a function of only a few underlying parameters. That is, the data points are actually samples from a low-dimensional manifold that is embedded in a high-dimensional space. Manifold learning algorithms attempt to uncover these parameters in order to find a low-dimensional representation of the data. There are some approaches to solving this problem like Isomap, Locally Linear Embedding, Laplacian Eigenmaps, Semidefinite Embedding, etc. These algorithms work towards extracting the low-dimensional manifold that can be used to describe the high-dimensional data.

(c) Locally Linear Embedding (LLE) is a Manifold Learning technique that does not rely on projections. LLE works by first measuring how each training instance linearly relates to its closest neighbors (c.n.) and then looking for a low-dimensional representation of the training set where these local relationships are best preserved. This approach makes it particularly good at unrolling twisted manifolds, especially when there is not too much noise.

Chapter 06

05. (a) How does face detection differ from face recognition? (b) what is the vision.CascadeObjectDetector refer to in MATLAB?

Ans: (a) Face detection refers to the process of identifying human faces in digital images, whereas face recognition is a process of identifying an individual by comparing a given image with the previously recorded image of that person. However, face-detection can be an important preprocessing step of the face-recognition approach from given random images.

(b) The cascade object detector is a series of models in the AdaBoost algorithm, which uses the Viola-Jones algorithm to detect people's faces, noses, eyes, mouth, or upper body.

06. (a) What was the motivation behind the boosting algorithm? (b) Write down the steps of the AdaBoost.M1 Algorithm (chapter 6, see the slides)

Ans:

(a) The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee."

(b) The AdaBoost.M1 algorithm is given as follows:

Algorithm: AdaBoost.M1.

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. For m = 1 to M

(a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

(b) Compute

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

(c) Compute $\alpha_m = \log((1 - err_m)/err_m)$.

(d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1,2,...,N$.

3. Output $G(x)$ = sign $[\sum_{m=1}^{M} \alpha_m G_m(x)]$.

---

07. (a) What are the major stages of the Viola-Jones Algorithm for face detection? (b) Define integral image. (c) What is/are the formula/s to compute the integral image? (d) Using at least one go-through step, compute the integral image of the following image-matrix:

Image (i)

0	8	6	1
1	5	9	0
0	7	5	0
2	8	9	2

Ans:

(a) The major steps are:

- Features: Feature generation/extraction using Haar-like feature-set.
- Integral Image: A novel image presentation scheme helps compute features faster.
- Feature Classifier: Boost the learning using AdaBoost.
- Cascading: Reject negative class fast, but thoroughly compute potential positive class - thus enhance the robustness.

(b) We define the integral image (*ii*) at location *x*, *y* contains the sum of the pixels above and to the left of *x*, *y*, inclusive as indicated in equation (C) and figure #A:

$$ii(x,\ y) = \sum_{x' \le x,\ y' \le y} i(x',\ y') \qquad \text{(C)}$$

where, *ii* (*x*, *y*) is the integral image, and *i* (*x'*, *y'*) is the original image.

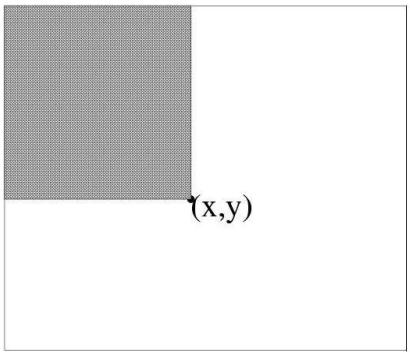


Figure A: The value of the integral image at point (*x*, *y*) is the sum of all the pixels above and to the left.

(c) Integral image, *ii* (*x*, *y*) can be computed using the following pair of recurrences:

$$s(x, y) = s\ (x, y - 1) + i\ (x, y) \qquad \text{(D)}$$

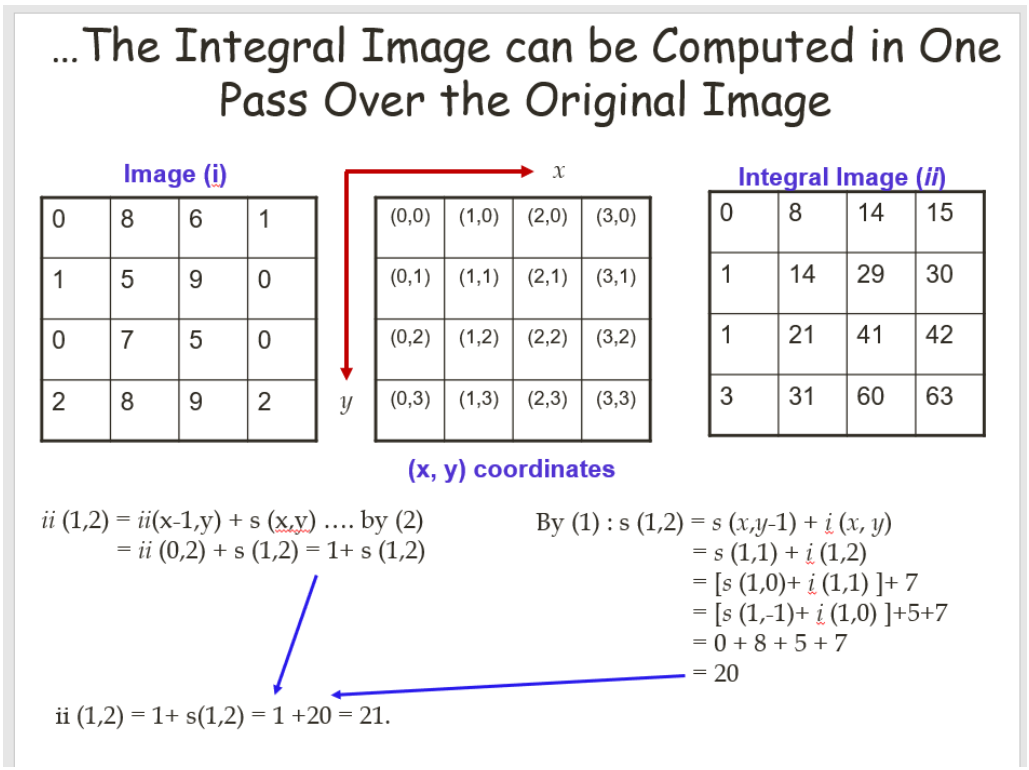
$$ii\ (x, y) = ii\ (x - 1, y) + s(x, y) \qquad \text{(E)}$$

where, *s* (*x*, *y*) is the cumulative row sum,
s (*x*,−1) = 0, and
ii (−1, y) = 0

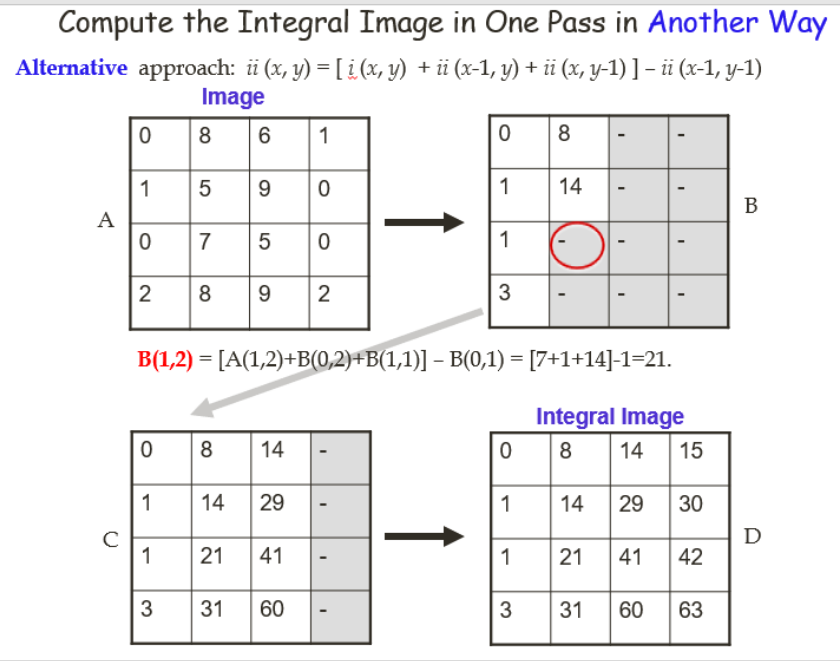
Or, the Integral image *ii* (*x*, *y*) can be computed using the following equation (F):

$$ii\ (x, y) = [\ i\ (x, y)\ + ii\ (x\text{-}1, y) + ii\ (x, y\text{-}1)\ ] - ii\ (x\text{-}1, y\text{-}1) \qquad \text{(F)}$$

(d) [Hints: use slide #(19) or slide #(20) of chapter 06]


…The Integral Image can be Computed in One Pass Over the Original Image
Image (i)
0 8 6 1
1 5 9 0
0 7 5 0
2 8 9 2
x
y
(0,0) (1,0) (2,0) (3,0)
(0,1) (1,1) (2,1) (3,1)
(0,2) (1,2) (2,2) (3,2)
(0,3) (1,3) (2,3) (3,3)
(x, y) coordinates
Integral Image (ii)
0 8 14 15
1 14 29 30
1 21 41 42
3 31 60 63
ii (1,2) = ii(x-1,y) + s (x,y) …. by (2)
= ii (0,2) + s (1,2) = 1+ s (1,2)
By (1) : s (1,2) = s (x,y-1) + i (x, y)
= s (1,1) + i (1,2)
= [s (1,0)+ i (1,1)]+ 7
= [s (1,-1)+ i (1,0)]+5+7
= 0 + 8 + 5 + 7
= 20
ii (1,2) = 1+ s(1,2) = 1 +20 = 21.

Or,



Alternative approach: ii (x, y) = [i (x, y) + ii (x-1, y) + ii (x, y-1)] – ii (x-1, y-1)

Question: How to compute B(1,2) (applying alternative way)?

B(1,2) = [A (1,2)+ B (1-1,2) +B (1,2-1)] – B(1-1,2-1)
= [A(1,2)+B(0,2)+B(1,1)] – B(0,1)
= [7+ 1+14]-1
= 22-1
= 21

Chapter 07

08. What are the main tasks that autoencoders are used for?

Ans: Here are some of the main tasks that autoencoders are used for:

- Feature extraction
- Unsupervised pretraining
- Dimensionality reduction
- Generative models
- Anomaly detection (an autoencoder is generally bad at reconstructing outliers).

09. Suppose you want to train a classifier, and you have plenty of unlabeled training data but only a few thousand labeled instances. How can autoencoders help? How would you proceed?

Ans: If you want to train a classifier and you have plenty of unlabeled training data but only a few thousand labeled instances, then you could first train a deep autoencoder on the full dataset (labeled + unlabeled), then reuse its lower half for the classifier (i.e., reuse the

layers up to the codings layer, included) and train the classifier using the labeled data. If you have little labeled data, you probably want to freeze the reused layers when training the classifier.

10. What are undercomplete and overcomplete autoencoders? What is the main risk of an excessively undercomplete autoencoder? What about the main risk of an overcomplete autoencoder?

Ans: An undercomplete autoencoder is one whose codings layer is smaller than the input and output layers. If it is larger, then it is an overcomplete autoencoder. The main risk of an excessively undercomplete autoencoder is that it may fail to reconstruct the inputs. An overcomplete autoencoder's main risk is that it may just copy the inputs to the outputs without learning any useful features.

11. What are the two major parts of the Generative Adversarial Network (GAN). Draw a schematic diagram of GAN.

Ans:

(a) GANs are composed of two neural networks:

(*i*) a **generator** that tries to generate data that looks similar to the training data, and

(*ii*) a **discriminator** that tries to tell real data from fake data.

(b)

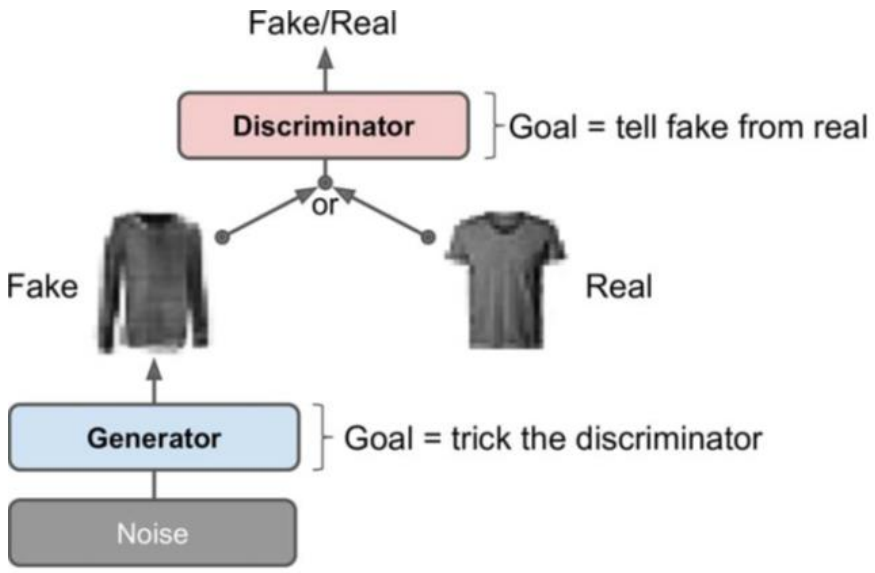


Figure: A schematic diagram of the generative adversarial network (GAN).

12. What are the two classes of classical approaches for numerically representing words?

Ans:

They are:

(*i*) approaches that use **external resources** for representing words, such as WordNet, and (*ii*) approaches that **do not use external resources**, such as one-hot encoding and Term Frequency-Inverse Document Frequency (TF-IDF).

--- x ---