Artificial Intelligence and Machine Learning - Unit II
06 July 2023

1. Answer the following questions by reporting the mathematical procedure. If you have to compute the actual value, please write the procedure that leads you to the numerical values. **Read well the text before proceeding!**

   (a) In Eq. (1) left, $\mathbf{X}$ is a design matrix where each row is a sample. $\boxed{1}$

   - What how many samples do we have?
   - What is the dimensionality of the covariance matrix?
   - Complete the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to compute the empirical average and the covariance matrix associated with $\mathbf{X}$.

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 2 \end{bmatrix} \quad \boldsymbol{\mu} = [\qquad\qquad\qquad] \quad \boldsymbol{\Sigma} = \begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix} \tag{1}$$

Solution: The number of samples is 2 and the dimensionality is 3. $\boldsymbol{\Sigma}$ is $\frac{1}{N}(\mathbf{X}-\boldsymbol{\mu})^\top(\mathbf{X}-\boldsymbol{\mu})$

$$\boldsymbol{\mu} = [0.5 \;\; 1. \;\; 1.] \quad \boldsymbol{\Sigma} = \begin{bmatrix} 0.25 & 0. & 0.5 \\ 0 & 0 & \\ 0.5 & 0. & 1. \end{bmatrix} \tag{2}$$

   (b) A point cloud with $N = 500$ poins $\mathbf{X} \doteq \{\mathbf{x}_i\}_{i=1}^N$ is sampled from a multi-variate Gaussian distribution in 2D and shown in Fig. 1 (a). The black diamond indicate the empirical mean of the distribution, while the black cross indicates the origin of the space. Fig. 1 (b) shows the same point cloud but the point cloud is "classified" in three quadrants following the principal components. There are three cases for the "classification": case A (rounded points), base B (cross) and case C (diamond). See the end of the exam for a zoomed version of Fig. 1 (b). $\boxed{1}$

   - Describe with linear algebra how you can project all the points to the principal component that has the highest variance.
   - Considering the classification shown in Fig. 1 (b), describe an algorithm that uses linear algebra to classify the point cloud in the three cases: case A, case B, case C.

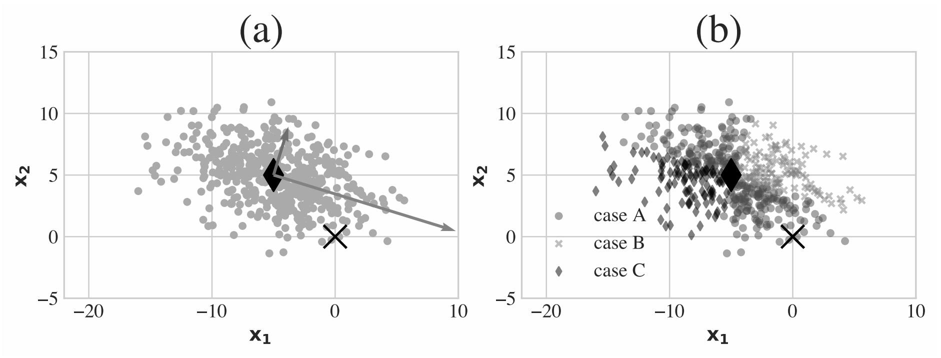

Figure 1: Point cloud classification

Solution: From now on I will have less and less time to release solution so I will describe them very briefly. ◇ The first question is simply a definition. You have to say that to project all the points on the PC with highest variance you have to. 1) center the pointcloud 2) compute the cov. matrix 3) obtain the eigvalues and eigvectors $\mathbf{U}$ from eigendecomposition of cov. matrix. Pick the index i of highest eigvalues and select the associated column of matrix $\mathbf{u} = \mathbf{U}_i$ with that index to select the strongest PC. At this point to project simply to the dot product between $\mathbf{u}$ and each point or you cal also write it in matrix notation. ◇ There might be different ways to solve it. The easiest given what we saw in class is to 1) center the point cloud 2) apply decomposition mentioned before in the definition. We know that the PCs will be oriented in that way, so we can rotate the point cloud to make it axis-aligned (the PCs will be standard bases) $\mathbf{p} = \mathbf{U}^\top(\mathbf{X}-\mu)$. At this point you simply have to check the sign inside $\mathbf{p} = [p_1, p_2] = [\mathbf{U}_1^\top\mathbf{x}, \mathbf{U}_2^\top\mathbf{x},]$. You can do smt like

```
    if p_1>0 AND p_2 >0 case B;    IF XOR(p_1,p_2) case A;    else  case C
```

There might be another solution to consider the angles between PCs but it could be tricky because you need still need to center the cloud and the make the classification relative to the angles that the PCs have in this reference system.

(c) You have a zero-mean point cloud sampled from a multi-variate Gaussian distribution in a d dimensional space and then you have a transformation $\mathbf{T}$ that consists of two other transformations: a rotation matrix $\mathbf{R}$ and a diagonal scaling matrix $\boldsymbol{\Delta}$ where each value in the diagonal is $1/2$. The formal definition of $\mathbf{T}$ is $\mathbf{T} = \mathbf{R}\boldsymbol{\Delta}\mathbf{R}^\top$. Now you want to apply the transformation $\mathbf{T}$ to the point cloud n times, where $n = 2$. $\boxed{1}$

- Can you describe what happens to the generic point cloud without running a single code in `numpy`? If so, describe the process from a mathematical perspective.
- Now assume you run it for a very large number of steps, n is very large ($n = 100$): you apply it once; then you applied once again to the resulting point cloud from previous step; and so on and so forth n times. What happens to the point cloud when n is very large?

Solution:

- Algebrically what happens is $\mathbf{X}' = \mathbf{T}^2\mathbf{X} = \mathbf{T}\mathbf{T}\mathbf{X} = \mathbf{R}\boldsymbol{\Delta}\mathbf{R}^\top\mathbf{R}\boldsymbol{\Delta}\mathbf{R}^\top = \mathbf{R}\boldsymbol{\Delta}^2\mathbf{R}^\top\mathbf{X}$ sine $\mathbf{R}$ is a rotation matrix is also orthonormal $\mathbf{R}^T\mathbf{R} = \mathbf{Id}$. So the point cloud is already centered, get only shrinked by 4 times.
- Repeating the above n times with n towards infinite, $\mathbf{R}\boldsymbol{\Delta}^n\mathbf{R}^\top\mathbf{X}$ you get a point in zero assuming the machine has finite precision.

Total for Question 1: 3

2. Answer the following questions about Gaussian Mixture Models (GMM) and K-means.

(a) Write, using equations, the probability density function (pdf) used when learning a GMM for a point x, assuming you have k modes. $\boxed{1/2}$

Solution: see previous exam. the pdf is MoG Mixture of Gaussian; linear combination of Gaussian; the weights to combine sums up to 1 and are non-negative.

(b) In the definition of the pdf used in GMM, give an intuition why the mixing coefficients have to: $\boxed{1 1/2}$

- sum up to one
- be all non-negative *(Think by "contradiction": assume they can be negative and wonder what happens to the pdf.)*

Solution: To sum up to one, we can say that it needs to be like that, given that the total volume under the MoG pdf still needs to integrate to one; other way to look at this is from a generative point of view: if the mix. coefficients are a categorical distribution that is used to sample which Gaussian, they have to be as such by definition. Similar intuition holds for the non-negativity (if the pdf goes negative then integrating the pdf into the negative interval may give you negative probability).

(c) $\boxed{1}$

- Write down the equation to compute the responsibilities. $\boldsymbol{\gamma}$. If you have k modes, how many dimensions has the vector $\boldsymbol{\gamma}$?
- Now assume that the mixing coefficients are $\pi = [0.57, 0.11, 0.22, 0.10]$, and you have a point x. Assume you know that:

$$p(x|z_1) = 1\% \quad p(x|z_2) = 25\% \quad p(x|z_3) = 5\% \quad p(x|z_4) = 50\%.$$

Can you compute the $\boldsymbol{\gamma}$ for the point x? If yes, please do it; if no then state why you cannot.

Solution: Definition of responsibilities see previous exam. The dimensionality is k. For the second part, given the eq. of responsibilities just compute the denominator once first. you can do that by putting the two $p(x|z)$ and π into two vectors. They have the same dimensionality. Compute the dot product between them. This is the denominator you need. The result in this case is 0.0942. Then, for each component of γ_k just consider $\frac{p(x|z_k)\pi_k}{0.0942}$. The result is

$$\gamma = [0.061 \quad 0.292 \quad 0.117 \quad 0.531]$$

. The $\sum_{i=1}^{k} p(x|z_k)$ does NOT have to sum up to one, $\int_{-\infty}^{\infty} p(x|z_k)\, dx = 1$ and $\sum_{i=1}^{k} \gamma_k = 1$.

(d) Your mate wants to perform classification with K-means. The dataset is 2D grayscale images of digits of size 28×28, similarly to what we used in the course. To do that she fits the k-means on a training set of n samples. Then, using the assignments, she computes the mode of the labels for a given cluster. So now every cluster as a label. The classification criteria is as follows: the predicted label on a new sample to $\mathbf{x}$ is given by first searching for the nearest centroid and then assigning the centroid label $\mathbf{x}$. $\boxed{1 1/2}$

- Describe the classification results when the number k of clusters that she set is equal to 1.
- Now instead consider, that she leans k-means with k very close to n, i.e. $k \approx n$. The more k approaches n, does her method resemble one of the algorithms that we reviewed in the course? If yes than state which algorithm and why.

Solution: $\diamond$ In the first case $k = 1$ the classification result is invariant to the input. No matter which input, it is always going to return the most frequent labels in the dataset. It acts as base constant classifier. We can also say it underfits but was not asked. $\diamond$ In the second case $k = n$ then the classification algorithm becomes k-nn with $k = 1$.

Total for Question 2: 4½

3. Given the training points below for $y \in \{0, 1\}$ binary classification:

$$(x_0 = 0; y_1 = +1) \quad (x_2 = 1/4; y_2 = 0) \quad (x_3 = 1/2; y_3 = +1) \quad (x_4 = 1; y_4 = 0)$$

(a) Determine the output of a **K Nearest Neighbour (K-NN)** classifier for all points on the interval $0 \leq x \leq 1$ using 1-NN. $\boxed{1}$

Solution: see previous exams

(b) In a (K-NN) classifier for binary classification, explain why is convenient to choose k as a odd number instead of even number. $\boxed{1}$

Solution: see previous exams for longer solution: you avoid the case of a tie.

(c) Assume you want to regress continuous values—thus $y \in \mathbb{R}$. The regressed output is the mean of the **K Nearest Neighbour (K-NN)** of a test point. Determine the output on the interval $0 \leq x \leq 1$ using the same training data above for $K = 2$. $\boxed{1}$

Solution: see previous exams for longer solution. in all interval is the predicted is y=0.5. Watch out now the task is regression not classification.

Total for Question 3: 3

4. We want to perform some evaluation of classifier that returns a raw score for an input $\mathbf{x}$. The higher the score the more is likely to be class +1. The ground-truth labels y_{gt} are show in Tab. 1 as positive labels +1 and negative labels 0.

y_{gt}	0	+1	0	0
score	-1	0.01	0	2

Table 1: Labels and scores for a binary classifier.

(a) Give a definition of True Positive Rate (TPR) and False Positive Rate (FPR). Given the binary classifier mentioned above, compute the ROC curve for the values in Tab. 1 by showing the TPR and FPR in a table. $\boxed{1}$

Solution: See previous exam for the definition. A lot of you in the exam skipped the definition. You still have to define what true positives are. They are only simple "the positive" over all the positives. See previous exams or slide for definition. See previous exams for computation, briefly:

```
thrs -> [ 3.    2.    0.01  0.   -1.  ]
fpr -> [0. 1. 1. 2. 3.]
neg -> [3, 3, 3, 3, 3]

tpr -> [0. 0. 1. 1. 1.]
pos -> [1, 1, 1, 1, 1]
```

(b) Compute the Area Under the Curve (AUC) of the above ROC. $\boxed{1}$

Solution: See previous exams for solution. `auc-->` $\frac{2}{3}$.

(c) You friend is torn because he developed a method for binary classification that has 99% accuracy on the test set yet the method is computational heavy. You ask him to tell you what is the distribution of the labels of the dataset. The dataset has a 1000 samples in total and is split as train 70%, validation 20%, and test 10%. He tells you that the labels distribute as: $\boxed{1/2}$

—	label 0	label 1
train	693	7
valid	198	2
test	99	1

Table 2: Dataset label distribution

- Explain what is stratified sampling when partitioning a dataset.
- State if you friend used or not stratified sampling when partitioning the dataset into the three splits as Tab. 2. Motivate your choice.
- Given the description of the problem above explain if an accuracy of 99% is a "good" accuracy or if there is a very basic method to obtain 99% accuracy on this dataset.

Solution: $\diamond$ Stratified sampling is a sampling technique that preserves some underline factors present in the entire dataset (they can be labels or groups etc). So when performing sampling to split into partitions, stratified sampling takes care of the fact that labels proportion needs to be kept the same. $\diamond$ Yes, she did used Stratified sampling. it can be seen by computing the values for the entire dataset from which she started from, that was obviously not shown in the table. For the total dataset class 0 is 990 while class 1 is 10, for a total samples of 1000. Fixing a class, let's say class 1, the ratio of class 1 is thus $\frac{10}{1000}$. Now if you look at the partition you see that

$$\frac{10}{1000} = \frac{7}{700} = \frac{2}{200} = \frac{1}{100}$$

stratied sampling kept the ratio of class labels the same between splits. If you split simply randomly you may get no label 1 (given that you have only a few of them over 1000), if the split is very small.

$\diamond$ No, it is not a good accuracy, given that the method is expensive. Constant base classifier that returns always the most frequent class has the same accuracy.

Total for Question 4: $2\frac{1}{2}$

5. You are given to study the following formulation for binary classification $\{-1, +1\}$:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) \doteq \frac{1}{1+\exp^{-\boldsymbol{\theta}^\top \mathbf{x}}} \tag{3}$$

where $p(y=+1|\mathbf{x}) \doteq f_{\boldsymbol{\theta}}(\mathbf{x})$ and $p(y=-1|\mathbf{x}) = 1 - p(y=+1|\mathbf{x})$

(a) Assuming that the input feature $\mathbf{x}$ has 100 dimensions, what is number of parameters that you have to learn in Eq. (3)? $\boxed{1/2}$

Solution: it learns 100 parameters by the definition of dot-product with x.

(b) What is the role of the function $\frac{1}{1+\exp^{-z}}$ and what is usually called? $\boxed{1/2}$

Solution: it is called sigmoid function and is used in logistic regression to squish every input value in the range [0,1] so to bound it to be a probability.

(c) Now considering Eq. (3), the learned parameters $\boldsymbol{\theta} = [0.82, -0.46, 0.1]$ and each sample in Tab. 3. $\boxed{1\,1/2}$

◇ Compute $\boldsymbol{\theta}^\top \mathbf{x}$ for each sample.

◇ Compute the probability for each sample of being positive.

◇ Classify $\hat{y}$ the training samples as either positive (+1) or negative (-1) thresholding the probability at 50%.

$\mathbf{x}$	[-2, -2, 1]	[-1, -1, 1]	[0, 0, 1]	[1, 1, 1]	[2, 2, 1]	[3, 3, 1]
$\boldsymbol{\theta}^\top \mathbf{x}$						
$p(y=+1\|\mathbf{x})$						
$\hat{y}$						

Table 3: Testing point for the probabilistic classifier in Eq. (3).

Solution: Just for the first point. Compute the 3-dimensional dot product between point and the weights. -0.62. Place it inside sigmoid. Be careful that sigmoid flips the sign. prob. is 0.3497. After thresholding at 50%, class is negative -1.

(d) Given a function: $f(\theta; \mathbf{x}) = \left(\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c\right)^2$ where $\theta = \{\mathbf{A}, \mathbf{b}, c\}$ are fixed parameters and a starting point $\mathbf{x} = [1, 1]$, we want to optimize $\mathbf{x}$ to minimize the function using `pytorch`. The code in Listing 1 is supposed to do that but there are 4 issues: 3 bugs and 1 missing piece. Find the 3 bugs and the missing part and explain what is the problem with each of them. *Note: the issues are not syntax errors but fundamental issues.* $\boxed{1\,1/2}$

```
1  import torch
2  import random
3  random.seed(0);torch.manual_seed(0);
4
5  def function(x, A, b, c,):
6      return torch.pow(x.T @ A @ x + b.T@x + c, 2)
7
8  A = torch.tensor([[0., 2.], [2.,0.]], requires_grad=False)
9  b = torch.tensor([[-0.5],[ 0.5]], requires_grad=False)
10 c = torch.tensor([-2.], requires_grad=False)
11 x = torch.tensor([1., 1.], requires_grad=False)
12
13 for epoch in range(200):
14     value = function(x, A, b, c)
15     with torch.no_grad():
16         x += 1e+6*x.grad
```

Listing 1: Example pytorch code for optimization

Solution: `l11`: we are optimizing x so `requires_grad=True` bug

`l14-15`: there is no backpropagation as of now; it misses to compute the gradient of the function over those that requires gradients. `value.backward()` missing part `l16:` minimize x thus the gradient needs to be subtracted `x += -1e+6*x.grad` bug.

`l16:` `1e+6` is exploding learning rate; the notion of the gradient is true in tiny neighborhood `x += -1e-6*x.grad` $1e6 = 10^6$ bug

`l14:` Something I forgot myself:`.backward()` adds the gradients if you call it multiple times, if not zeroed before. So every time in the loop we have to zero the gradients before eval the function `x.grad.zero_()`. bug I did not remove points from this because I never told you this. Those that spotted it, got extra points.

Total for Question 5: 4

You can use this space for writing. The summary of points is at the bottom.

Question:	1	2	3	4	5	Total
Points:	3	4½	3	2½	4	17
Score:						