Artificial Intelligence and Machine Learning - Unit II
13 June 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 column is a sample. What is the dimensionality of the samples in $\mathbf{X}$ and how many samples do we have? Complete the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to compute the empirical average and the covariance matrix associated with $\mathbf{X}$. [½]

$$\mathbf{X} = \begin{bmatrix} -4 & 9 & 4 & 0 & -3 \\ 10 & -4 & 8 & 0 & 0 \end{bmatrix} \quad \boldsymbol{\mu} = [\qquad\qquad] \quad \boldsymbol{\Sigma} = \begin{bmatrix} \qquad\qquad \end{bmatrix} \tag{1}$$

Solution: The number of samples is 5 and the dimensionality is 2. $\boldsymbol{\Sigma}$ is $\frac{1}{N}(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^\top$

$$\boldsymbol{\mu} = [1.2 \;\; 2.8] \quad \boldsymbol{\Sigma} = \begin{bmatrix} 22.96 & -12.16 \\ -12.16 & 28.16 \end{bmatrix} \tag{2}$$

   (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 after a series of transformation: the process transforms the point cloud so to swap the principal components (PC). [2]

   - Describe precisely the mathematical process using linear algebra that applies these transformations to change the point cloud from Fig. 1(a) to Fig. 1(b). *Solution with explicit cosine and sine rotation matrices will NOT be considered.*
   - The covariance of $\mathbf{X}$ is of type $\begin{pmatrix} \sigma^2_{x_1x_1} & -\sigma^2_{x_1x_2} \\ -\sigma^2_{x_2x_1} & \sigma^2_{x_2x_2} \end{pmatrix}$, can you rewrite the covariance of the transformed point cloud using the previous covariance values?
   - What can you say about the absolute value of the determinant of the transformations that you described above?
   - Now edit the transformation so that it also scales each PC to increase it by 5%.

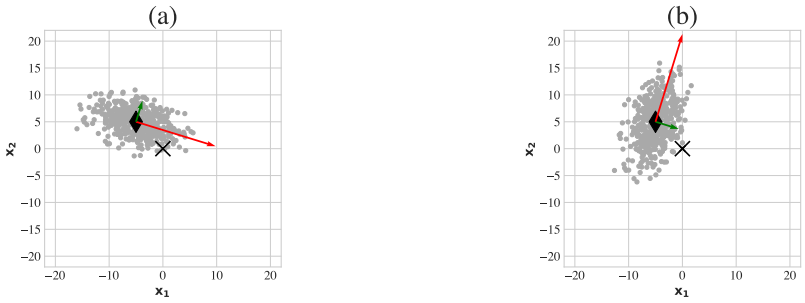


Figure 1: Point cloud transformation

Solution: We center the point cloud by removing the empirical mean $\bar{\mathbf{X}} \doteq \mathbf{X} - \boldsymbol{\mu}$. We compute the PCs by eigendecomposition of the covariance matrix $\frac{1}{N}\bar{\mathbf{X}}^\top\bar{\mathbf{X}}$. This gives a set of column-vectors of PCs in $\mathbf{U}$ where each i-th column is associated with eigenvalue λ_i. Assuming the orientation of PCs is as in Fig. 1–left To swap the two PCs (and the associated points) as Fig. 1–right, we swap the two columns of $\mathbf{U}$ by creating $\mathbf{U}_s$ (`Us=U[...,::-1]`). Then we project $\bar{\mathbf{X}}$ with $\mathbf{U}_s$ and unprojected it with $\mathbf{U}$ (vice versa is also OK). We finally restore the mean. Full process: $\mathbf{U}\mathbf{U}_s^\top\bar{\mathbf{X}}^\top + \boldsymbol{\mu}$. This can also be done as:

$$\mathbf{U}\Gamma\mathbf{U}^\top\bar{\mathbf{X}}^\top + \boldsymbol{\mu} \qquad \Gamma = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The covariance becomes $\begin{pmatrix} \sigma^2_{x_2x_2} & \sigma^2_{x_1x_2} \\ \sigma^2_{x_2x_1} & \sigma^2_{x_1x_1} \end{pmatrix}$, variance is swapped the cloud from anticorrelated becomes correlated; off-diagonal flips from negative to positive. The transformation does not increase the volume of the pointcloud thus the abs. value of determinant is one. Full process with scaling: $\mathbf{U}\Delta\mathbf{U}_s^\top\bar{\mathbf{X}}^\top + \boldsymbol{\mu}$ where $\Delta = \begin{bmatrix} 1.05 & 0 \\ 0 & 1.05 \end{bmatrix}$.

(c) You have to help a colleague of yours that implemented a pipeline for supervised classification. The pipeline classifies the data linearly and it does not work well. When you debug the pipeline you realize that as a first step, the approach compressed the data to remove some attributes using Principal Component Analysis (PCA). In particular you find out that the two classes distribute in 2D as Fig. 2, where each color indicates a different class and PCA projects the data on the dimension with higher variance. 1

- Explain why the supervised classification with linear boundaries may not work when the data distributes as Fig. 2 using the pipeline described above.
- What can you change in the pipeline to let the linear classifier separate the data?

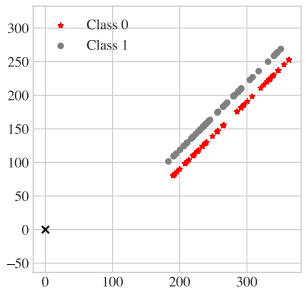


Figure 2: Supervised Classification with PCA

Solution:

- A supervised linear learning algorithm is able to separate that data in Fig. 2 yet the PCA compressing the 2D points into a single dimension with the higher variance is the issue. This causes the two classes to be non-linearly separable anymore. Points from the two classes will overlap once projected onto the major variance axis.
- A quick solution is to project it to axis with less variance where visually we can see that are separable easily with a line (actually all the points of a class collapse to a single point).

Total for Question 1: 3½

2. We are in the i-th step of the Expectation-Maximization (EM) for learning the parameters of a GMM. Let us assume the **maximization** part just finished. The parameters of the GMM and the training points x are given in Tab. 1. Assume the estimate for GMM is maximum likelihood.

x	1	0	10	-5
	z_1	z_2	z_3	z_4
μ	1.80	1.13	1.05	1.12
σ^2	46.81	6.91	3.31	7.62
π	0.57	0.11	0.22	0.10

Table 1: Training set of a GMM with parameters.

(a) Write the density of the GMM with equations using the parameters in Tab. 1, for a point x. [½]

Solution: $p(x) = 0.57 \cdot \mathcal{N}(x; 1.80, 46.81) + 0.11 \cdot \mathcal{N}(x; 1.13, 6.91) + 0.22 \cdot \mathcal{N}(x; 1.05, 3.31) + 0.10 \cdot \mathcal{N}(x; 1.12, 7.62)$.

(b) In the definition of a single multi-variate Gaussian pdf, give an intuition why the determinant of the covariance matrix $\boldsymbol{\Sigma}$ is at the denominator. Moreover, although $\boldsymbol{\mu}$ is a parameter, why is not in the denominator? [1]

Solution: The intuition is that the absolute value of the determinant of a matrix encodes the change in volume that transformation induces. With a gaussian pdf the only way to change the volume is to edit the covariance matrix $\boldsymbol{\Sigma}$. This latter acts as the shape of the distribution. Thus for a shape that gets bigger, the pdf has to be normalized more so that the volume of the pdf is kept one. The location parameter $\boldsymbol{\mu}$ is just translating the distribution in space somewhere thus does not not affect the volume.

(c) Write down the equation to compute the responsibilities $\boldsymbol{\gamma}$. Compute $\boldsymbol{\gamma}$ for the training point $x = 0$, assuming $p(x == 0|z_k) = 95\% \quad \forall k \in 1, \dots, 4$. You may need only *some* parameters in Tab. 1. [1]

Solution: To compute the vector $\boldsymbol{\gamma}$, we need to compute the probability that a point x may have been generated by the k-th Gaussian. We can use Bayes theorem as:

$$\boldsymbol{\gamma}_k = p(z == k|x == 0) = \frac{p(x == 0|z_k)p(z_k)}{\sum_{j\in\{1,2,3,4\}} p(x == 0|z_j)p(z_j)} \quad \forall k \in 1, \dots, 4.$$

In case of GMM, $p(z_j)$ are the mixing coefficients and $p(x == 0|z_k) = 0.95$ for all the modes, thus $\boldsymbol{\gamma} = \boldsymbol{\pi} = [0.57, 0.11, 0.22, 0.10]$.

(d) You work in a company yet unfortunately a cyberattack deleted all the training set images that your partner used to train a GMM. The training set consisted of 2D grayscale images of digits of size 28×28 and the grayscale is quantized with 8 bits. The number of images were $N = 10K$. Thankfully, a parametric model of the GMM was NOT destroyed. The parametric model was computed with the following steps: [1½]

- First of all, PCA was applied to the images, flattened into a vector, to learn a low-rank mapping from $28 \times 28 \mapsto 50$. The PCA mapping stores the $\boldsymbol{\mu}_{\text{pca}} \in \mathbb{R}^{784}$ and the principal components $\mathbf{U} \in \mathbb{R}^{784\times 50}$. Each training point is centered and compressed with PCA to form a 50-D vector.
- Then GMM is ran on the 50-D space to estimate 10 clusters with no singularities and no overfitting. Assume you have the parameters of the GMM.

Can you help your friend to generate images of the training set? (Cross the multiple true answers)

- ◯ No, it is not possible to generate any samples.
- ◯ Yes, but not the exact same samples of the training set.
- ◯ Yes, we can but we will generate an approximation of the training samples.
- ◯ We can definitely restore the original training samples.

If you answered that you can generate, describe precisely with math, how you can do it. If you answered no, state why.

Solution: The right answers are the 2. and 3. We can generate similar samples but not the same ones. How to do it: We built the cumulative mass function C from the π of GMM. We repeat 10K times. 1. We random sample $u \sim \mathcal{U}(0, 1)$ 2. We use inverse transform sampling applied to u and C to select the k mode of the GMM. 3. We random sample from $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ we then multiply by $\boldsymbol{\Sigma}_k$ to $\mathbf{z}$ and add $\boldsymbol{\mu}_k$ 4. We unproject as $\mathbf{x} = \mathbf{U}\mathbf{z} + \boldsymbol{\mu}_{\text{pca}}$. The generated $\mathbf{x}$ should be similar samples to the ones used for training. Thanks to student 1982785 we also need to convert

$\mathbf{x}$ from float to int (if you assume you quantize colors to 8 bits). You can do that by assigning to the nearest integer and eventually clipping the values in $[0, 255]$.

Total for Question 2: 4

3. We are given an already learned decision tree for multi-class classification shown in Fig. 3 below. Each shape represents a training sample where the shape identifies the class. The feature of each point are two-dimensional $\mathbf{x} = [x_1, x_2]$.

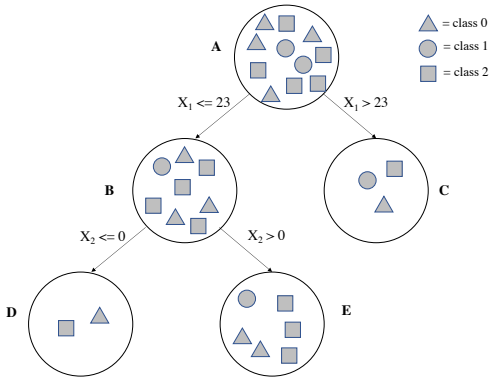


Figure 3: Decision Tree for multi-class classification.

(a) Compute the Impurity using the entropy **for the entire TREE.** [1]

Solution: The entropy of the tree is the sum of the entropies of leaves weighted by the number of samples of each leaf. See previous exams.

(b) You have a test sample $\mathbf{x} = [x_1, x_2] = [-1, -1000]$, which class you assign to $\mathbf{x}$—triangle, square or circle or maybe it is better to return that the algorithm is unsure? [½]

Solution: The path on the tree is $A \mapsto B \mapsto D$. We have one vote for square and one vote for triangle. Very high impurity better to return algorithm is unsure.

(c) Let us assume that you have **enough computational power that you can afford to learn and do inference on multiple** decision trees. At the same time you are troubled because the single decision tree that you learned is over-fitting badly given that you have a deep tree. Describe how you can mitigate overfitting without reducing the depth of three. [1]

Solution: We can apply the concept of Bagging and Ensemble to reduce the variance of the algorithm without sacrificing the depth. Keeping the depth the same, we can random sample **with replacement** multiple M datatsets from the main one and train different trees over each dataset. Given a new samples we can perform inference on each of the tree and pool the final results using average or majority vote depending on the problem (regression or classification).

(d) Let's assume that you are in a leaf of the tree, e.g. the node E, and you want to make the decision tree a randomized algorithm for prediction. Describe how you can make the decision tree prediction random with probabilities that are given by the labels you find in the leaf. [½]

Solution: We apply inverse transform sampling on the probabilties you have on the leaves. See answer previous exams.

Total for Question 3: 3

4. We want to perform some evaluation of a binary classifier using logistic regression that reports probability for an input $\mathbf{x}$ being of class $+1$ as $p(y = +1|\mathbf{x})$. The ground-truth labels y_{gt} are show in Tab. 2 as positive labels $+1$ and negative labels -1.

y_{gt}	+1	-1	-1	-1	-1	+1	-1	+1
$p(y = +1\|\mathbf{x})$	0.25	0.01	0.3	0.01	0.165	0.15	0.02	0.5

Table 2: Labels and probabilities for a binary classifier.

(a) Give a definition of True Positive Rate (TPR) and False Positive Rate (FPR). Given a binary classifier with probability of $\mathbf{x}$ being positive $p(y = +1|\mathbf{x})$—compute the ROC curve for the values in Tab. 2 by showing the TPR and FPR in a table. [1]

Solution: The True Positive Rate is the ratio between the positive samples that are correctly classified (TP) divided by the number of ground-truth positive we have in the population (P_{gt}). The False Positive Rate is the ratio between the negative samples that are incorrectly classified as positive FP divided by the total number of ground-truth negative in the population N_{gt}.

$$TPR = \frac{TP}{P_{gt}} \quad FPR = \frac{FP}{N_{gt}}.$$

See previous exams for computation, briefly:

```
thrs -> [1.5   0.5   0.3   0.25  0.165 0.15  0.02  0.01 ]
fpr -> [0. 0. 1. 1. 2. 2. 3. 5.]
neg -> [5, 5, 5, 5, 5, 5, 5, 5]

tpr -> [0. 1. 1. 2. 2. 3. 3. 3.]
pos -> [3, 3, 3, 3, 3, 3, 3, 3]
```

(b) Compute the Area Under the Curve (AUC) of the above ROC. [1]

Solution: See previous exams for solution. `auc--> 0.8`.

(c) Two interns are working on a machine learning approach to spam detection. Each student has their own set of 100 labeled emails, 90% of which are used for training and 10% for validating the model. Student A runs a k-NN classification algorithm and reports 80% accuracy on her validation set. Student B experiments with over 100 different learning algorithms and variations of them, training each one on his training set, and recording the accuracy on the validation set. His best formulation achieves 95% accuracy. Whose algorithm would you pick for protecting a corporate network from spam? Motivate your answer. [½]

Solution: First of all, the case described above is not best practice because of the low number of both training and validation samples. Although both of them are validating the approach in the validation set, the second intern chooses the best algorithm conditioned on the fact that before tried 100 different algorithms and variations! Only after seeing in the performance on the validation set chose the method, he is missing a final test to see what could be a fair accuracy or even better cross-validation. Thus student B selection may suffer from high-variance problem and may over fit more than the previous solution and it should be considered less. Also: Thanks to a student that made me notice this: how can be possible that with 10 samples student B report 95% accuracy?

Total for Question 4: 2½

5. We have to analyze a neural network in the form of a multi-layer perceptron (MLP). The neural network details follow in the sub questions below.

   (a) Deduce and write how many trainable parameters you have with a MLP with input feature vectors with dimension equal to 2048, a first layer $\mathbf{W}_1$ with 1024 units/neurons. This layer $\mathbf{W}_1$ is not trained, acts as a random projection and in pytorch is set `required_grad=False`. A second layer $\mathbf{W}_2$ with 512 units/neurons, and a final multi-class classification layer $\mathbf{W}_3$ with 10 units/neurons. Assume all layers do NOT have the bias term. The network uses ReLU activation function after each layer except the classification that uses softmax. Write down the equation for the computation, not just the final value. Moreover, what is the dimensionality of the logit vector that the network produces? [1]

   Solution: The network params are:

   $$\overbrace{\underbrace{2048 \times 1024}_{\text{1st layer NOT trainable; so DO NOT COUNT}}}^{\text{matrix}} \qquad \underbrace{1024 \times 512}_{\text{2nd layer}} + \underbrace{512 \times 10}_{\text{3rd layer}}.$$

   The total number of trainable parameters is $1024 \times 512 + 512 \times 10 = 529.408$. Given that we have to 10 classification units the logit vector will have 10 dimensions.

   (b) Consider the previous network, but now assume that we replace $\mathbf{W}_3$ with another layer, call it $\mathbf{W}$, that maps the input from $512 \mapsto 512$. We also assume the input $\mathbf{h}_3$ to $\mathbf{W}$ to be zero-mean. We "freeze" all layers and we train only $\mathbf{W}$ with SGD over mini-batches. The new configuration of the MLP is thus: $\mathbf{x} \mapsto \mathbf{W}_1 \mapsto \mathbf{h}_2 \mapsto \mathbf{W}_2 \mapsto \mathbf{h}_3 \mapsto \mathbf{W}$. This new network is trained with a strange loss as follows: [2]

   $$\left\|\mathbf{h}_3 - \mathbf{W}\mathbf{W}^\top \mathbf{h}_3\right\|_2^2 + \lambda \left\|\mathbf{W}\mathbf{W}^\top - \mathbf{I}\right\|_2^2 \tag{3}$$

   where $\mathbf{I}$ is the identity matrix of size 512×512, λ is a scalar parameters to trade-off the two-terms, and the second norm over the resulting matrix treats the matrix as a flattened vector.

   - What does this loss compute? State if you have encountered something similar throughout the course.
   - Describe in details what the two terms achieve.
   - Could you replace this entire training process with a simpler one?

   Solution:

   - It computes a linear projection $\mathbf{W}$ that is optimized towards rotating the data by forcing this projection to be orthonormal matrix. It is the same loss we have found as the second way to express PCA as reconstructing the data (not as variance maximizing procedure) though in this case we do not drop any dimension.
   - Note before training $\mathbf{W}$ is a random matrix; the first term of the loss computes a linear layer that reconstructs the input points $\mathbf{h}_3$ by first projecting them with $\mathbf{W}^\top$ and unprojecting them with $\mathbf{W}$. Once the second term approaches zero the first term rotates in 512D and unrotate it back. SGD updates $\mathbf{W}$ so to minimize the "reconstruction" of $\mathbf{h}_3$ under ℓ_2^2 norm. At the same time we have another loss term that forces SGD to go in a configuration of $\mathbf{W}$ that is orthonormal.
   - We can simply extract the hidden vector $\mathbf{h}_3$ for all training samples once. Compute the covariance matrix and recover $\mathbf{W}$ with PCA. No need for iterative SGD and to tune λ. Then we can put $\mathbf{W}$ inside the NN.

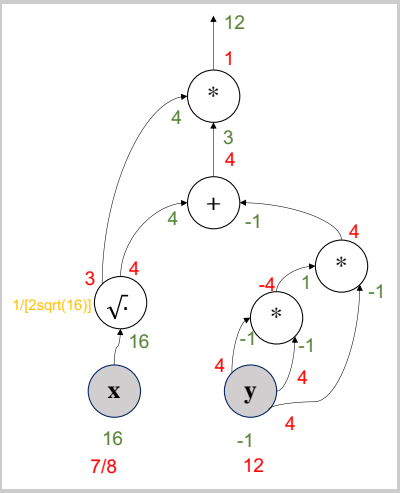
   (c) Assume you have a loss function of the type [1]

   $$\ell(x, y) = \left(\sqrt{x} + y \cdot y \cdot y\right) \cdot \sqrt{x}$$

   .

   - You have 3 types of gates that takes at most two input elements: multiplicative $\otimes$, summation $\oplus$, and square root $\boxed{\surd}$ gates. Draw the computational DAG that represents the loss function reusing the computation as much as possible.
   - Assume $x = 16$ and $y = -1$ forming a vector $\mathbf{v} \doteq [x, y]$. Compute the gradient $\nabla_{\mathbf{v}} \ell(\mathbf{v})$ by showing all the gradient flow at each gate from the loss to x and y.
   - Given the gradient that you calculated, assume you can perturb either x or y a bit: which of the two would you choose to increase the loss more and why?

Solution:



$\nabla_{\mathbf{v}}\ell(\mathbf{v}) = [\frac{\partial \ell}{\partial x}, \frac{\partial \ell}{\partial y}] = [7/8, 12]$.

$\frac{\partial \ell}{\partial y}$ is 12 times stronger than $\frac{\partial \ell}{\partial x}$ so is more convenient to perturb y than x of the same tiny bit.

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:	3½	4	3	2½	4	17
Score:						