Assignment 2

Innopolis University

Machine Learning Fall 2021 - Bachelors

General Instructions

In this assignment, you will solve several tasks on classification and clustering. There will be two theoretical questions on K-means clustering and SVM. Additionally, there will be two practical questions on Ensemble Learning and CNN. We have added a bonus task on GANs, this task will be added to the total grades of the course not only the assignment. Partial solution for the bonus task won't be considered, only full solution will be graded and you will be graded based on the quality of your generated images.

You are required to submit your solutions via Moodle as a single zip file. The zip archive should contain a single ipynb, and a single PDF for the theoretical sections 1 and 2. Please, put your name and email at Innopolis.university as the first line in the notebook.

There's one more ipynb file, you can submit if you have solved the bonus part. We won't give partial grades for the partial solution of the bonus part. There will be grade only for the bonus part if it is complete and it will be graded based on the model performance and the quality of the generated faces. Source code should be clean, easy to read, and well documented. Bonus points may be awarded for elegant solutions. However, these bonus points will only be able to cancel the effect of penalties.

Do not just copy and paste solutions from the Internet. You are allowed to collaborate on general ideas with other students as well as consult books and Internet resources. However, be sure to credit the sources you use and type all the code, documentation by yourself.

1 Theoretical question on K-means Clustering (10%)

Consider a set of $n = 2m + 1$ one-dimensional samples, m of which coincide at $x = -2$, m at $x = 0$, and one sits at $x = a > 0$, i.e.,

$$D = \left\{ \underbrace{-2, -2, ..., -2}_{\text{m times}}, \underbrace{0, 0, ..., 0}_{\text{m times}}, a \right\}.$$

In fact D is a multi-set since it allows for multiple instances of the same element. (NOTE: The variable m has nothing to do with the symbol K often used for the number of clusters.) Let $D_1, D_2 \subset D$ be a two-cluster partitioning of the dataset D so $D_2 = D - D_1$. It can be shown that the two-cluster partitioning that minimizes the sum-of-squared errors (used in the K-means algorithm, though this part is about the globally optimal solution)

$$J(D_1, \mu_1, \mu_2) = \sum_{i: D_i \neq \phi} \sum_{x \in D_i} (x - \mu_i)^2$$

,

groups the m samples at $x = 0$ with the one at $x = a$ (i.e., $D_2 = 0, ..., 0, a$) if

$$a^2 < f(m)$$

where $f(m)$ is a function of m. Derive this function accordingly.

2 Theoretical question on SVM (20%)

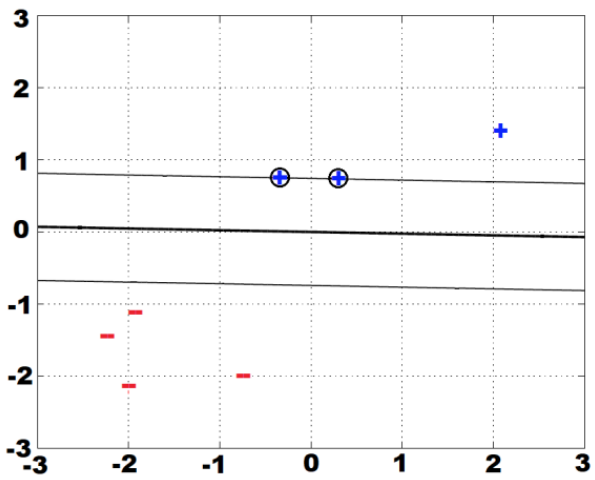
Consider the following SVM formulations:

(I) minimize$_\theta$ $\frac{1}{2}||\theta||^2$ subject to $y_t\theta^T x_t \geq 1$ for all $t \in \{1, ..., n\}$.

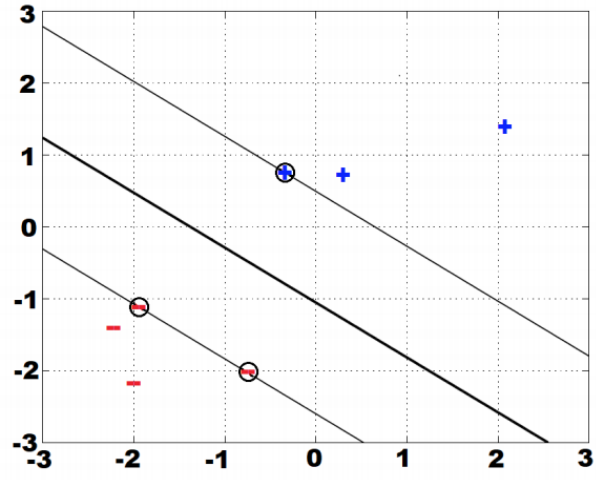
(II) minimize$_{\theta,\theta_0}$ $\frac{1}{2}||\theta||^2$ subject to $y_t(\theta^T x_t + \theta_0) \geq 1$ for all $t \in \{1, ..., n\}$.

(III) For fixed $C \in (0, \infty)$, minimize$_{\theta,\zeta}$ $\frac{1}{2}||\theta||^2 + C\sum_{t=1}^{n} \zeta_t$ subject to $y_t\theta^T x_t \geq 1 - \zeta_t$ and $\zeta_t \geq 0$ for all $t \in \{1, ..., n\}$.

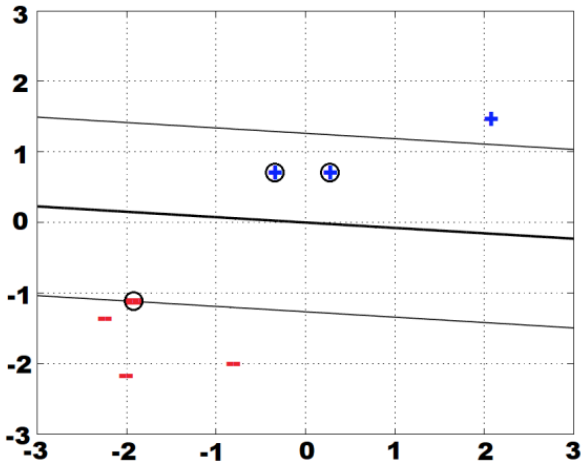
Note that in (I) and (III) we have $\theta_0 = 0$. Consider the following illustrations, with positive points marked '+', negative points marked '−', three straight lines indicating the decision boundary and margins, and circles placed around the support vectors:



(a)



(b)



(c)

Figure 1: Different SVM models on the same set of points.

For each of the above SVM formulations (I), (II), and (III), indicate whether it is **possible** or **impossible** for it to have produced each classifier in figures (a), (b), and (c). If it is possible, simply write "**Yes**" with no further explanation. If it is impossible, write "**No**" followed by a brief explanation. (Note: These explanations only need to be one short sentence long, as opposed to detailed justifications.)

3 Practical Tasks on Ensemble Learning (35%)

Malware detection refers to the process of detecting the presence of malware on a host system or of distinguishing whether a specific program is 'malicious' or 'benign'. In this task, you will use some network layer features such as Duration, Number of packets, etc. to build a machine learning classification model that will detect Android malware applications, using app features. You can download the dataset from here.

Note: The output column is the **type** column.

1. Read data in Python. Split your data into train and test sets (80% and 20% respectively).
2. Create the following three models:
   - `RandomForestClassifier(max_depth=15)`
   - `BaggingClassifier(base_estimator=DecisionTreeClassifier(max_depth=15))`
   - `AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=15))`
3. Tune the following hyper-parameters of the estimators in all ensemble models using grid search:
   - `n_estimators`
   - `max_features` $\longrightarrow$ for the base estimators
   - `min_impurity_decrease` $\longrightarrow$ for the base estimators

   You should tune these hyper-parameters within the ensemble not separately. Create the final models using the best values of the hyper-parameters and evaluate your models on the test set. Which model performed the best on the test set? Why do you think that is the case?
4. Answer this question before doing the next part. If you fine-tuned the hyper-parameter `max_depth` as well, which of the three ensemble models would you expect to have deeper (larger `max_depth` value) base learners and which would have shallower base learners? Why do you think that would be the case?
5. Initialize the models with the best parameters you got from the third step. Fine tune `max_depth` from 5 to 25. Draw 3 plots on the same graph. Put the `max_depth` parameter on the horizontal axis and the cross validation accuracy of your ensemble models on the vertical axis. Do the results agree with your answer in the previous part?

4 Practical Tasks on CNN (35%)

ML applications are frequently used to replace a person while doing some complex tasks, like driving (self-driving cars). However ML is also useful to assist person, to make sure that the person is OK. One of such ML approaches is Eye tracking. This technology allows to look after driver (if he is not looking on the road) or it could be used to measure level of stress or absent-mindedness of a person while doing some responsible task. It is also a solution for disabled people to manipulate computer with just eye movements. In this task you are going to implement CNN for calculating human iris center. This CNN architecture is proposed in https://ieeexplore.ieee.org/abstract/document/8803121 as a fully convolutional network which consists of a base network and auxiliary network. It has two losses: reconstruction loss for ensuring the model saves important positional features and second loss which measures the distance between predicted eye center and ground truth one. The architecture also has skip connection which brings positional information to deep layer. Architecture is shown on Figure2. Dataset with face images and labels (iris center coordinates + eye corners) is here https://www.unavarra.es/gi4e/databases/gi4e/.

You can use cv2 as tool in that task to read, convert color of images, to draw something on image and to do all the other needed operations with images. However you are not restricted to use it and you can use any libraries.

- Preprocess and visualize the dataset: [50%]

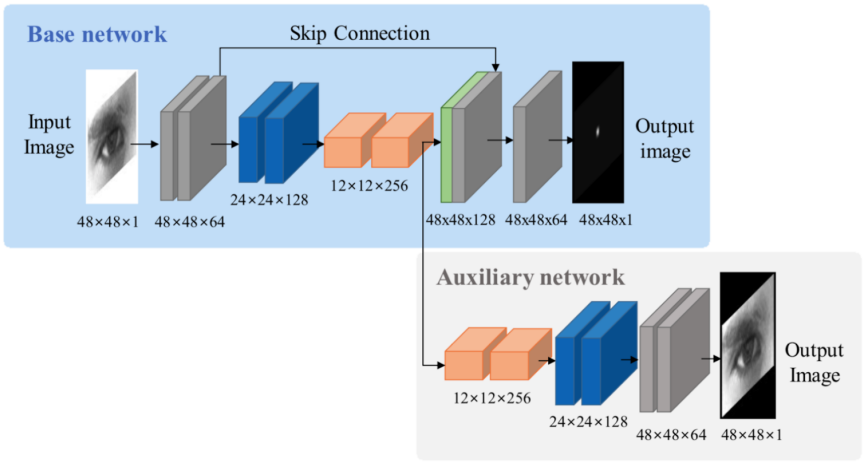


Figure 2: Eye center estimation CNN architecture.

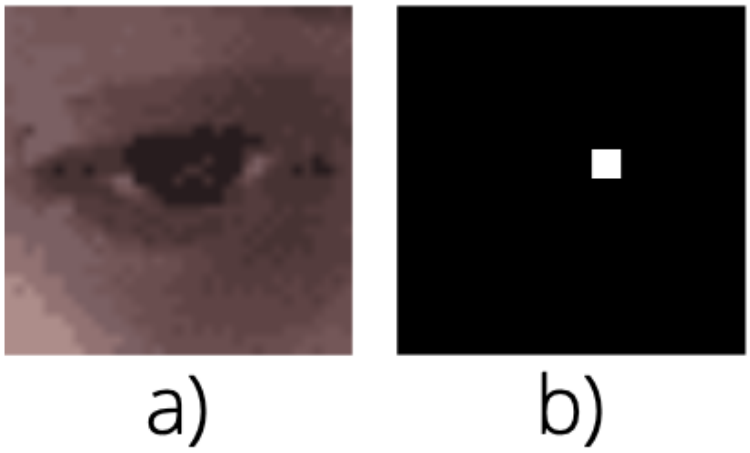


Figure 3

- Download dataset. Description of folders and naming are inside dataset folder in README.txt
- Read all images and convert them to gray with (cv2.ctvColor())
- Read annotation for images. It contains eye corners and eye centers of 2 eyes for each image.
- Visualize one image, draw eye corners and iris centers on it
- Normalize images (divide by 255)
- Crop eye regions (and resize if needed) to be (48x48) image with the help of eye corners. Do that for all images. It should look like Figure3(a)
- Now data is ready to create a final dataset, which you will use for CNN training. You should create two np arrays X and Y:

  - X contains (48x48) images of eye regions which you crop on previous step

  - Your labels (Y) are coordinates of eye center for each image in X (don't forget to convert iris center from whole image coordinate system to coordinate system of eye region). You should make one more step to cook Y set. For each eye center in Y you should create a (48x48) image (with zero values) and assign value=1 to pixel which coordinate is an iris center. Do it for all images. It should look like Figure3(b)

  Finally, your X and Y sets are lists of 48x48 images. X contains images of eye and Y images with white pixel on the place of iris center.
- Split dataset

- Build a CNN model using Keras. Layers that might be helpful: Conv2, Conv2DTranspose, concatenate. [20%]
- Compile and train CNN with different optimizers [sgd, adam, adamax, rmsprop], loss functions [mse, mae] and activations [tanh, relu, sigmoid]. Report best combination. [20%]
- Make a prediction for 10 test images. Draw predicted centers on them and visualize it. (You can draw iris center with cv2.circle()) [10%]

  HINTS: Use colab with GPU to speed up training and hyperparameter tuning.

5 Bonus on GANs

Fake Face Generation using GANs. For this question, please check and fill the attached notebook.

Notes

- Cheating is a serious academic offense and will be strictly treated for all parties involved. So delivering nothing is always better than delivering a copy.
- Late assignments will not be accepted and will receive **ZERO** mark.
- Code cleanness and style are assessed. So maybe you want to take a look at our references: Link 1 and Link 2.
- Organize your notebook appropriately. Divide it into sections and cells with clear titles for each task and subtask.