IMPERIAL COLLEGE LONDON

MEng and MSc EXAMINATIONS 2023

Part IV and Advanced Mechanical Engineering

for Internal Students of the Imperial College of Science, Technology and Medicine
This paper is also taken for the relevant examination for the Associateship or Diploma

MACHINE LEARNING

Wednesday, 18th January 2023, 14:00 to 16:00

This paper contains SEVEN questions. Attempt all questions. The numbers shown by each question are for your guidance; they indicate approximately how the examiners intend to distribute the marks for this paper.

A Data and Formulæ Book is provided.

This is a CLOSED BOOK Examination.

Turn over

Instructions

A Jupyter notebook has been provided which contains some initial code, including providing access to the necessary datasets. You should use this as a starting point for your work. Upload it to Colab via 'File -> Upload Notebook'. Through this exam you should only make use of the libraries numpy, matplotlib and pandas, i.e. your code must not use other libraries such as keras and scikit-learn. At the end of this exam paper there are some annotated code examples which may be useful for completing the exam. These examples are also included in the notebook. It is not compulsory to use these.

You should write out any necessary working, or any written answers, in the Jupyter notebook (either as code or text) and should add comments as necessary as part of this working. Note that this exam is not to assess programming but rather the understanding of the material on the course, so comments will be assessed in line with working given in a standard exam, rather than for programming correctness. Also note that the quality of the code (including how optimised it is) will not be assessed, but the focus will be on performing the calculations necessary to answer the questions correctly.

You should ensure that all answers are provided as visible output within your notebook. The markers should not need to execute any code from the notebook to confirm that you have got the correct answer. However, you should ensure that your submitted notebook runs without error in case the markers do need to execute particular sections as part of the marking process.

At the end of the exam, submit your Jupyter notebook (.ipynb file), which you can download from Colab via 'File -> Download -> Download .ipynb'. Do not submit any other files.

1. I have three factories producing a specific bike part. 50% of these come from Factory A, 20% from Factory B and the rest from Factory C. The provided file [d1.csv] contains a set of data, from each factory, indicating the distance in km that the component lasted for.

   (a) Use a top hat Parzen window of full width 800km to estimate the probability distribution functions for distance from each of the three factories and plot on the same axes, with A in blue, B in red and C in green. Make your x limits 0 to 20,000km. [12%]

   (b) I have a part provided to me at random which fails at 15,000 km. Using the result of (a), estimate the respective probabilities that it came from each factory. [10%]

2. Consider the words “their”, “there” and “they’re”. I define three metrics: one based on the sound, one on spelling and one on meaning.

   (a) Which of these three metrics will be zero when comparing any two of the given words? [3%]

   (b) What is the technical name for the property of a metric that will cause this to be the case? [2%]

3. You are given a linear hard margin SVM which can be described by a linear discriminant function with $\hat{\mathbf{w}}=(2.4, -1.8)^t$ and $w_0=5$. For given dataset [d3.csv], what is the margin? [15%]

4. Your colleague has been using a SVM to fit a given dataset. They have found that they can get much better fitting on this dataset by using a radial basis kernel over simple linear kernels. What can you conclude about how well their model will perform in general, and why? [8%]

5. I now have a linear discriminant machine consisting of the weighting terms for each class given in Table Q5. Using the dataset [d5.csv], there is one point which is misclassified by this linear discriminant machine. Give the coordinates of this point. [20%]

Table Q5

Class	**w**	w_0
0	$\begin{pmatrix}0.1\\-1\end{pmatrix}$	0.1
1	$\begin{pmatrix}-4\\0.5\end{pmatrix}$	0.5
2	$\begin{pmatrix}0.3\\0.1\end{pmatrix}$	0

6. You are given a dataset [d6.csv]. Given a function $y = Ax^2 + B\cos C\pi x$, estimate what constants A, B and C are and output these values. If using gradient descent, you can use a starting point of A = -2, B = 0 and C = 2.5 and 40 steps with a constant step size of 0.0005. Produce a plot showing the data points provided in black and the final fitted curve in red, in the range $-1.5 < x < 1.5$. [15%]

7. You are given the neural network presented in Figure Q7. Sigmoid, i.e. $y = 1/(1 + e^{-x})$, transfer functions are used throughout. Plot the resulting output as the input x varies between -2 and +2. [15%]

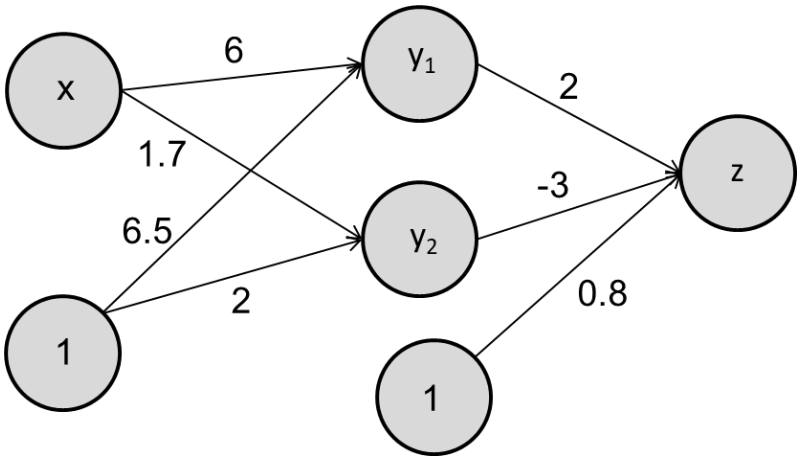

Figure Q7

```
#You may find lines or sections of the following code useful:
def useful_code():
  x_low = 0
  x_high = 2*np.pi
  n_x_values = 30
  #define a set of n_x_values points between x_low and x_high (0 to
  #2 pi in this case):
  x = np.linspace(x_low,x_high,n_x_values)

  #Normalise a vector (i.e. turn it into a unit vector):
  v = np.array([4, 8, 1])
  v_hat = v/np.linalg.norm(v)
  print("Normalised vector: ", v_hat)

  #find the minimum value within vector v:
  print("Minimum of v (should be 1): ", np.min(v))
  #find where the minimum occured - NB zero indexed
  #(NB can also do argmax() for maximum point)
  print("Minimum occurs at element: ", np.argmin(v,axis=0))

  #calculate y values for the given x values, using y = cos(x)
  y = np.cos(x)
  integral = np.trapz(y, x=x) #integrate y using the trapezium rule
  print("Integral (should be approx 0):",integral)

  loc = np.pi #set loc to be pi (3.14...)
  #interpolate function y(x) to find value at loc - i.e. y(loc):
  val = np.interp(loc, x, y)
  print("Value at pi (should be approx -1):",val)

  #summing and squaring:
  print("Sum of y squared: ", np.sum(y**2))

  #do a for loop
  for cnt in range(4):
    print(cnt)

  #Reshape matrix a such that it has the same shape as b:
  A = np.array([[1,2],[3,4]])
  print("A before reshaping:\n",A)
  B = np.array([6,7,8,9])
  print("B:\n",B)
  A = np.reshape(A,B.shape)
  print("A after reshaping to match B:\n",A)

  #Multiply two matrices together:
  A = np.array([[1,2],[3,4]])
  B = np.array([[5,6],[7,8]])
  C = np.matmul(A, B)
  print("Multiplication of \n", A, "\nand\n", B, "\ngives:\n", C)
```

```
  #Plotting functions/points:
  fig, ax = plt.subplots()  #define a figure
  #plot the points given by vectors x and y with a blue
  #solid line ('b-'). k is black, r red and g green.
  plt.plot(x, y, 'b-')
  plt.plot(x, y, 'k.')      #as above, but plot as black dots


  #2D plotting and routines:

  #generate two random sets of values x1 and x2
  n_points = 20
  x1 = np.random.normal(loc=0,scale=1,size=n_points)
  x2 = np.random.normal(loc=0,scale=1,size=n_points)
  #Combine two vectors x1 and x2, of length n_points, into
  # a matrix of size n_points x 2
  X = np.concatenate((np.reshape(x1,[n_points,1]),
                      np.reshape(x2,[n_points,1])),axis=1)

  y = np.zeros([n_points]) #define y and set all values to zero
  #now put all the class values to 1 where the x1 value is
  # greater than 0.2
  y[x1 > 0.2] = 1

  #points in x and y:
  npx = 200
  npy = 200
  #generate the grid to sample 2D space:
  Xgrid,x1line,x2line = gen_sample_grid(npx,npy,3)
  #generate an arbitrary 2D function - here do x1^2 + x2:
  z = Xgrid[:,0]**2+Xgrid[:,1]
  #and reshape it back to the grid
  z = z.reshape([npx,npy])
  fig, ax = plt.subplots()
  #plot the values in z sampled at values given by the
  #vectors x1line, x2line:
  plt.contourf(x1line, x2line, z)
  #plot scattered values in the n_points by 2 matrix X where
  #corresponding values in the y vector equal 0:
  ax.scatter(X[y == 0, 0], X[y == 0, 1])
  #then plot where y == 1:
  ax.scatter(X[y == 1, 0], X[y == 1, 1])

#uncomment to run the code if you wish:
#useful_code()
```