COMP4432 Machine Learning

Assignment #1 (Reference Answers)

1. Given the following training data for the regression task of f(x).

x	-2	0	2	4	6	8
$f(x)$	-10	-8	10	10	4	4

Build a decision tree for regression (i.e., a regression tree; cf. slide 32 of the decision tree lecture notes) to perfectly fit all six points. You may use the mean absolute error (MAE) to measure the impurity. Thus, your predicted values $\hat{f}(x)$, being the same as the $f(x)$, will have

$$MAE = \frac{1}{6}\sum_{x=-2,0,2,4,6,8}\left|\hat{f}(x) - f(x)\right| = 0.$$

Show your steps and draw the regression tree. You may assume that the possible splits are x= -1, 1, 3, 5, 7.

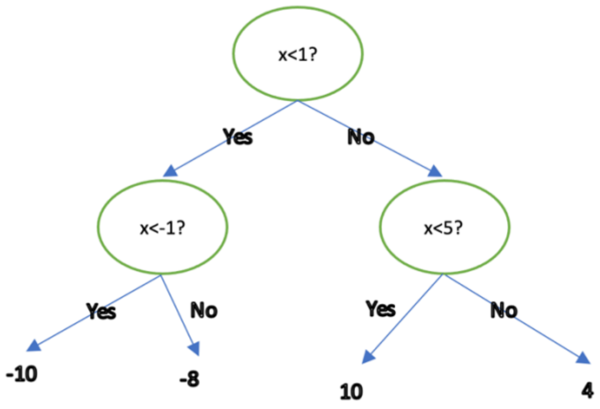
Answer: [25 marks]
For split points, we consider -1, 1, 3, 5, & 7.

To determine the root node's split, we have

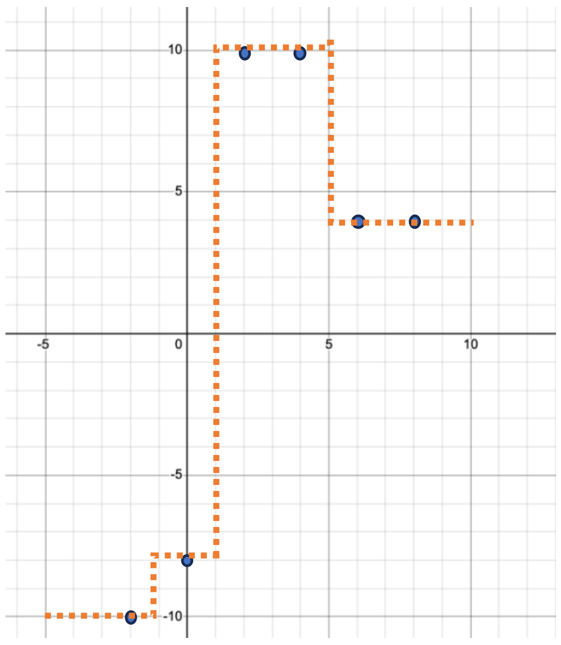
Split point (sp)	Average of points for x<sp	Absolute error of x<sp	Average of points for x≥sp	Absolute error of x≥sp	MAE after split
-1	-10/1	0	20/5	24	(0+24)/6
1	-18/2	2	28/4	12	(2+12)/6
3	-8/3	25.33	18/3	8	(25.33+8)/6
5	2/4	38	8/2	0	(38+0)/6
7	6/5	26	4/1	0	(26+0)/6

So, split point=1 can minimize the MAE (see the green one) and therefore is chosen as the root node. It is logical because -10 and -8 are close to each other while 10, 10, 4, 4 are close to each other and the two parts are NOT do close to each other.

The steps above can then be repeated until MAE=0, basically, the regression tree passes through every point of x=-2, 0, 2, 4, 6, 8. Taking into consideration of f(x)'s distribution, we can easily draw the following regression tree and regression result.

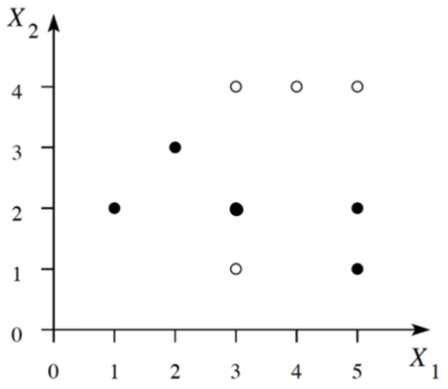


Regression tree constructed



Regression result

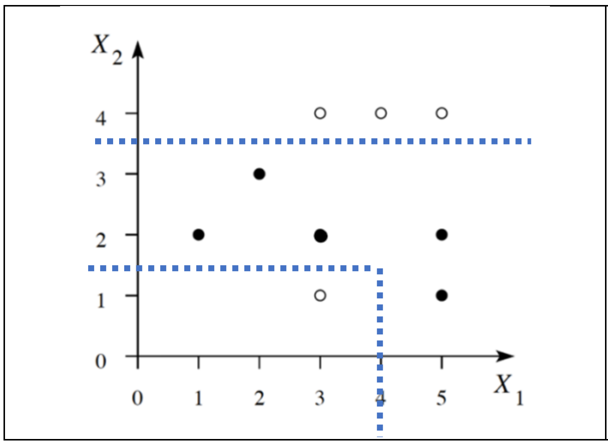
2. For the following dataset with 4 white dots and 5 black dots (data points), with integer aligned coordinate values, construct a decision tree to attain 100% accuracy on the 9 points.
   a) Draw the tree and label all the nodes appropriately. Note that you are NOT required to show the steps.
   b) Draw the corresponding decision boundary of the tree you provided in part (a).

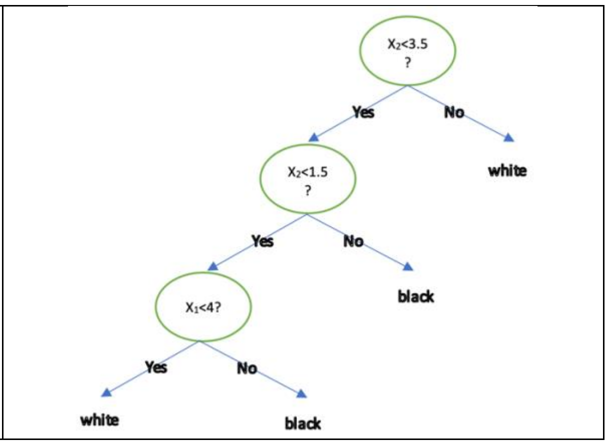


Answer: [25 marks]

Any result satisfying the given criterion (100% accuracy) will be acceptable. For example:

A solution with 3 levels





For the root node:

It is obvious that $X_2=3.5$ should be adopted because there exists only 1 error point at (3,1) with this split.

Under the root node to check $X_2<3.5$, the "no" branch will be terminated and the upmost 3 white points are conquered while the "yes" branch to handle the remaining 6 points may proceed as follows.

For the next level:

There exist two potential splits, i.e., X_2=1.5 and X_1=4. Let's compute the impurity for these two splits.

For X_2<1.5, the impurity with this split:

X_2<1.5?	p_i (white)	p_i (black)	$I(p_i, n_i)$
Yes	1	1	1
No	0	4	0

$$Entropy_{X2<1.5?} = \frac{2}{6}(1) + \frac{4}{6}(0) = \frac{2}{6}$$

For X_1<4, the impurity with this split:

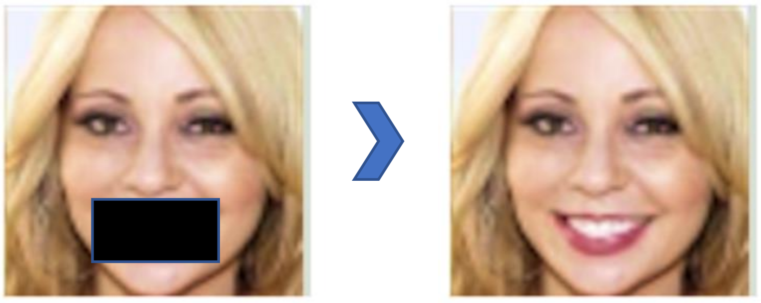
X_1<4?	p_i (white)	p_i (black)	$I(p_i, n_i)$
Yes	1	3	~0.81
No	0	2	0

$$Entropy_{X2<1.5} = \frac{4}{6}(0.81) + \frac{2}{6}(0) = \sim 0.54$$

So, X_2<1.5 can have higher information gain.

Then, for the third level, we only need to divide (3,1) and (5,1). Hence, the split X_1<4 can be applied. The final decision tree can be seen above.

3. Image Inpainting refers to the task of rebuilding missing or damaged patches of an image. Given color images of fixed size 300(h)x200(w) pixels with a rectangular patch of pixels of size 60(h)x100(w) pixels missing as exemplified on the left photo below, we want to rebuild the missing pixels like the right photo on the right. Recall that color images consist of 3 channels with pixel value from 0 to 255 (represented by 1 byte).



   a) Suppose you are asked to use a multilayer perceptron (MLP) neural network as shown in Fig.1 to carry out the inpainting task, i.e., predicting the missing image patch.

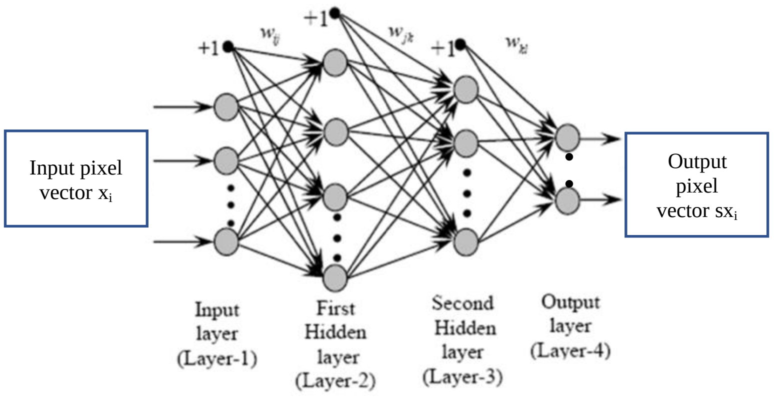


Fig.1

      (i) Describe how you formulate the problem by describing what should be the input (pixel vector x_i) and what should be the output (pixel vector sx_i).
      (ii) How training data should be **prepared** to train the MLP based image inpainting model?

Answers of part (a): [10 marks]

(i) [5 marks] There can be other formulations. One is that the problem can be viewed as predicting the missing color pixels by the existing ones, i.e., the non-masked pixels (300x200-60x100=54000 pixels). So, the input pixel vector consists of 54000x3-pixel values (wrt 3 color channels) while the output color pixel vector consists of 6000x3-pixel values. Note that the output layer should just employ a linear activation function to output. On the other hand, using the whole image, including the masked pixels, can also be used as input, resulting to have a 60000x3-pixel vector. This formulation will be preferred if the 2D spatial information needs to be exploited.

(ii) [5 marks] There may involve different aspects. A key one is how to prepare the input and output pixel vectors. A simple solution is just to collect sufficient number of color facial images and separate the mouth pixel patch from the remaining one. It will also involve unifying the image size to 300x200 and the collected images should be upscaled or downscaled accordingly. Other issues like aligning and/or normalizing the mouth to the mask position should be considered. Also, separation of male and female facial images can be added to discuss.

b) Suppose now the MLP in part (a) is enhanced with convolutional layers and pooling layers so that a CNN is resulted to carry out color image inpainting. For the following table of architecture, how many learnable parameters are there in each of the specified layers? Show the formula or calculations in your answers. Also, there exist TWO unknown parameters in the specification column, namely, U_1 and U_2, write down their correct values.

Layer in CNN	Specification	Number of learnable parameters (**formula answer is acceptable**)
Input Layer	Your solution in part (a)	0
1st Convolutional Layer	64 3x3xU_1 filters; stride=2; no zero padding	
1st Max Pooling Layer	2x2 window; stride=2	
2nd Convolutional Layer	256 3x3xU_2 filters; stride=1; no zero padding	
2nd Max Pooling Layer	2x2 window; stride=2	
Input layer of fully connected (fc) feedforward network	Just the flattened output from previous layer	0
1st hidden layer of fc feedforward network	N_{L2} hidden neurons	
2nd hidden layer of fc feedforward network	N_{L3} hidden neurons	
Output layer	Your solution in part (a)	

Answers of part (b): [15 marks]

Layer in CNN	Specification	Number of learnable parameters (**formula answer is acceptable**)
Input Layer	Your solution in part (a) CNN typically assumes a 3D data input and a simple formulation should be a 300x200x3, i.e., including the masked pixels.	0
1st Convolutional Layer	64 3x3xU_1 filters; stride=2; no zero padding U_1=3	64x3x3x3
1st Max Pooling Layer	2x2 window; stride=2	0
2nd Convolutional Layer	256 3x3xU_2 filters; stride=1; no zero padding U_2=64	256x3x3x64
2nd Max Pooling Layer	2x2 window; stride=2	0
Input layer of fully connected (fc) feedforward network	Just the flattened output from previous layer	0

1st hidden layer of fc feedforward network	N_{L2} hidden neurons	(36x21x256+1)xN_{L2}
2nd hidden layer of fc feedforward network	N_{L3} hidden neurons	(N_{L2}+1)x N_{L3}
Output layer	Your solution in part (a)	(N_{L3}+1)x60x100x3

The answers above are reference ones wrt the adopted formulation. For other input pixel matrix and output pixel matrix, the answers should be adjusted accordingly. Some remarks for the answers above are:

- U_1=3 because there are 3 color channels
- U_2=64 because 64 filters are used in the first convolutional layer.
- The mapping between (convolutional and pooling) layers are: 300x200x3 => 149x99x64 (for stride=2)=> 74x49x64 => 72x47x256 => 36x23x256.
- Please refer to the CNN lecture notes slide 44 to check with the answer above, e.g., for the 1st convolutional layer, it produce a volume of [(300-3+2*0)/2+1=149]x[(200-3+2*0)/2+1=99]x64.

4. In our lecture notes, we have already seen the derivative of a sigmoid function w.r.t. its weight parameters as follows.

$$f_{sigmoid}(x) = \frac{1}{1+e^{-(wx+b)}}$$

we have

$$\frac{\partial}{\partial w}\left(\frac{1}{1+e^{-(wx+b)}}\right)$$

$$= \frac{-1}{(1+e^{-(wx+b)})^2}\frac{\partial}{\partial w}(e^{-(wx+b)}))$$

$$= \frac{-1}{(1+e^{-(wx+b)})^2} * (e^{-(wx+b)})\frac{\partial}{\partial w}(-(wx+b)))$$

$$= \frac{-1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (-x)$$

$$= \frac{1}{(1+e^{-(wx+b)})} * \frac{e^{-(wx+b)}}{(1+e^{-(wx+b)})} * (x)$$

$$= f(x) * (1 - f(x)) * x$$

Let us now consider the weighted softmax function $f_{smax}(z_j)$, i.e.,

$$f_{smax}(z_j) = \frac{m_j e^{z_j}}{\sum_k m_k e^{z_k}}$$

where m_k is a user specified weight for the corresponding z_k, i.e., $m_k's$ are fixed (not a variable).

i) Derive the derivative of weighted softmax function w.r.t. z_j. You MUST use the symbols above, i.e., z_j, $f_{smax}(z_j)$, etc., to present your answer.

Hint: Using the quotient rule and let $g(z_j) = e^{z_j}$ and $h(z_j) = \sum_k e^{z_k}$, we have

$$\frac{\partial f_{smax}(z_j)}{\partial z_l} = \frac{g\prime(z_j)h(z_j) - g(z_j)h\prime(z_j)}{[h(z_j)]^2}.$$

Answer: [20 marks]

Using the quotient rule and let $g(z_j) = m_j e^{z_j}$ and $h(z_j) = \sum_k m_k e^{z_k}$, we have

$$\frac{\partial f_{smax}(z_j)}{\partial z_l} = \frac{g'(z_j)h(z_j) - g(z_j)h'(z_j)}{[h(z_j)]^2}$$

For $h(z_j) = \sum_k m_k e^{z_k}$, $h'(z_j)$ (i.e., $\frac{\partial h(z_j)}{\partial z_l}$) must be $= m_l e^{z_l}$, no matter which z_l we compute the derivative of $h(z_j)$ for, because it is simply a sum of independent exponent terms. In other words,

$$h'(z_j) = m_l e^{z_l}$$

But for $g(z_j) = m_j e^{z_j}$,

$$g'(z_j) = \frac{\partial g(z_j)}{\partial z_l} = \begin{cases} m_l e^{z_l} & l = j \\ 0 & l \neq j \end{cases} \qquad \text{Eq.(a)}$$

Then, for $l = j$, we have

$$\frac{\partial f_{smax}(z_j)}{\partial z_l} = \frac{m_l e^{z_l}(\sum_k m_k e^{z_k}) - m_j e^{z_j} m_l e^{z_l}}{[h(z_j)]^2}$$

$$= \frac{m_l e^{z_l}(\sum_k m_k e^{z_k})}{(\sum_k m_k e^{z_k}) * (\sum_k m_k e^{z_k})} - \frac{m_j e^{z_j} m_l e^{z_l}}{(\sum_k m_k e^{z_k}) * (\sum_k m_k e^{z_k})}$$

$$= \frac{m_l e^{z_l}}{\sum_k m_k e^{z_k}} \cdot \frac{\sum_k m_k e^{z_k}}{\sum_k m_k e^{z_k}} - \frac{m_j e^{z_j}}{\sum_k m_k e^{z_k}} \cdot \frac{m_l e^{z_l}}{\sum_k m_k e^{z_k}}$$

$= f_{smax}(z_l) \cdot 1 - f_{smax}(z_j) \cdot f_{smax}(z_l)$
$= f_{smax}(z_l)[1 - f_{smax}(z_j)]$

Note and be careful about the subscripts *l* and *j*! So, the final answer for $l = j$ is

$$\frac{\partial f_{smax}(z_j)}{\partial z_j} = f_{smax}(z_j)[1 - f_{smax}(z_j)]$$

The following part is a more general answer:

In fact, *l* may not necessarily equal to *j*.
For $l \neq j$ and recall Eq.(a) above, we have

$$\frac{\partial f_{smax}(z_j)}{\partial z_l} = \frac{0 - m_j e^{z_j} m_l e^{z_l}}{[h(z_j)]^2} = -\frac{m_j e^{z_j} m_l e^{z_l}}{(\sum_k m_k e^{z_k}) * (\sum_k m_k e^{z_k})}$$
$$= -f_{smax}(z_j) f_{smax}(z_l)$$

By using the Kronecker delta function, i.e.,

$$\delta_{lj} = \begin{cases} 1 & l = j \\ 0 & l \neq j \end{cases}$$

we have

$$\frac{\partial f_{smax}(z_j)}{\partial z_l} = f_{smax}(z_l)[\delta_{lj} - f_{smax}(z_j)].$$

This is the answer you may see from the Internet.

ii) With $z_j = \boldsymbol{w}_j^T \boldsymbol{x}$+b, i.e., an ordinary input-to-output in perceptron, derive the derivative of the weighted softmax function w.r.t. a weight parameter w_{ij}.

Answer [5 marks]:
Following $\frac{\partial f_{smax}(z_j)}{\partial z_l} = f_{smax}(z_l)[\delta_{lj} - f_{smax}(z_j)]$.

$$\frac{\partial f_{smax}(z_j)}{\partial w_{ij}} = \frac{\partial f_{smax}(z_j)}{\partial z_l} \cdot \frac{\partial z_l}{\partial w_{ij}}$$
$$= f_{smax}(z_l)[\delta_{lj} - f_{smax}(z_j)] \cdot x_i.$$