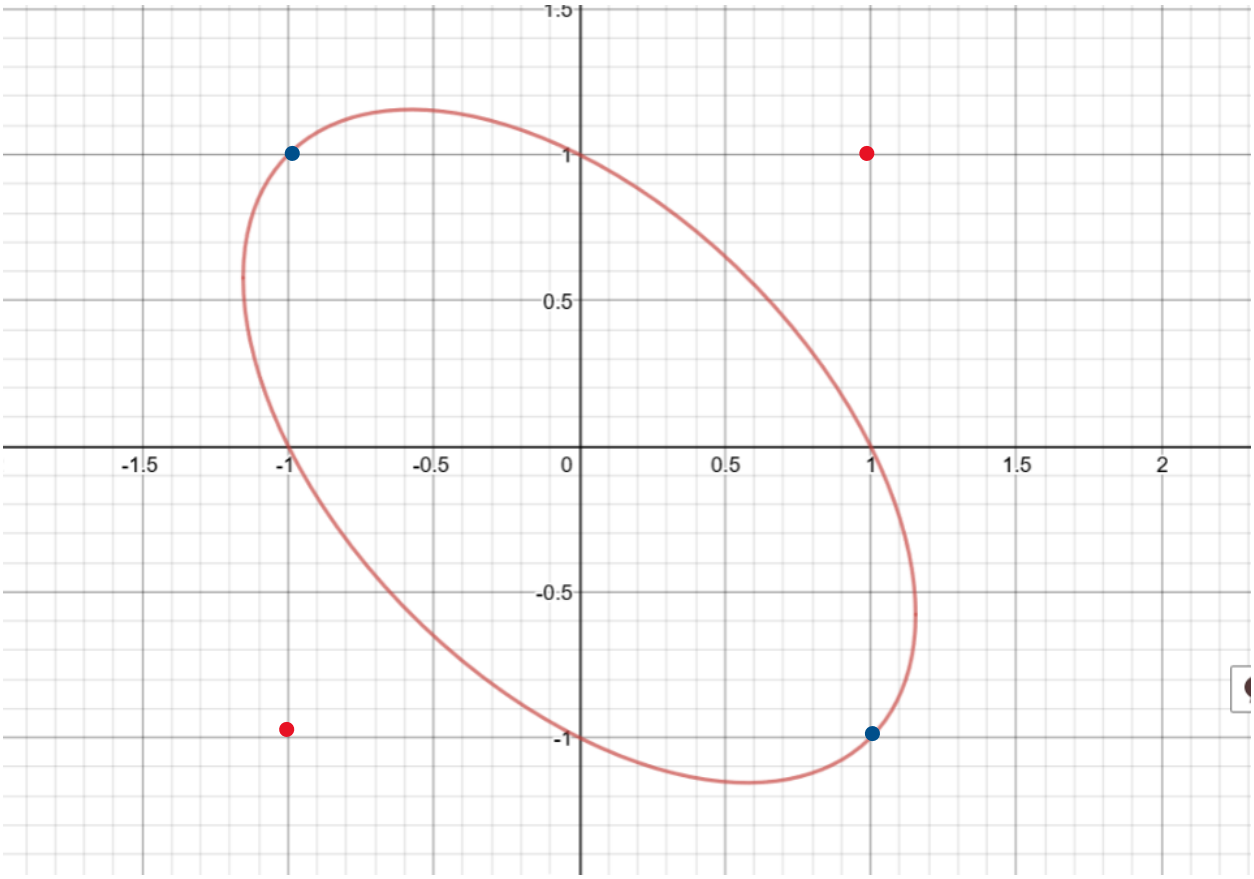
#1-a



No it is impossible to separate them linearly. We cannot draw a straight line that separate them. But, we can be convert them to be linearly separable in higher dimension.

$z = x^2+y^2+xy$

(red positive and blue is negative)

#1-b

$\phi(x) = [1,x,y,xy]^T$

$\phi(X+) = \{[1,1,1,1]^T,[1,-1,-1,1]^T\}$

$\phi(X-) = \{[1,1,-1,-1]^T,[1,-1,1,-1]^T\}$

#1-c

$y(x) = w^T\phi(x)$

for example if we consider (1,-1) and (-1,-1) then

- ⇨ $1 = x^2+y^2+xy$
- ⇨ $W=[0,0,1,0]$

#2

i and ii

An approach for feature detection is to use dense multi-scale feature sampling by scanning the image at multiple scales at fixed locations on a grid of rectangular patches. While directly using SIFT is based on hand-crafted local features.

The difference is that with dense SIFT you get a SIFT descriptor at every location, while with normal sift you get a SIFT descriptions at the locations determined by Lowe's algorithm.

A disadvantage of directly using SIFT is that all scale information is lost. Though for image classification such an invariance with respect to scale might seem beneficial since instances can appear at different scales, it trades discriminative information for scale invariance.

iii

A simple histogram is obtained by taking a region of an image, assigning a label to each pixel (somehow; via some mapping function), and then computing a histogram of the labels. The histogram counts, for each possible label, how many pixels received that label. You obtain a feature vector: a vector of counts, one count per possible label.

A pyramid allows you to do something at multiple scales. You start with a 32x32 image and compute a feature vector for it. Then, you rescale the image down to a 16x16 image and compute a feature vector for it. Then you rescale down to a 8x8 image and compute a feature vector for that. And so on. Finally, you concatenate all of those feature vectors.

This allows us to capture both coarse features from the highly downscaled images as well as fine features from the large images.

Such a histogram doesn't capture any information about where the labels appear. A spatial histogram addresses that shortcoming by dividing the image up into several smaller patches, computing a histogram for each patch, and concatenating those histograms. For instance, you could take a 32x32 image and break it up into sixteen 8x8 patches, compute a histogram for each patch, and concatenate them to get a feature vector. You still have to choose an appropriate matching.

iv

The idea is mapping the non-linear separable data-set into a higher dimensional space where we can find a hyperplane that can separate the samples.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

we just need to know K and not the mapping function itself. This function is known as Kernel function and it reduces the complexity of finding the mapping function. So, Kernel function defines inner product in the transformed space.

#3

$$softmax(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

X+C means adding a constant c to every dimension of x

$$softmax(x+c)_i = \frac{e^{x_i+c}}{\Sigma_j e^{x_j+c}} = \frac{e^c e^x i}{e^c \Sigma e^{x_j}} = \frac{e^{x_i}}{\Sigma_j e^{x_j}} = softmax(x)_i$$

#4

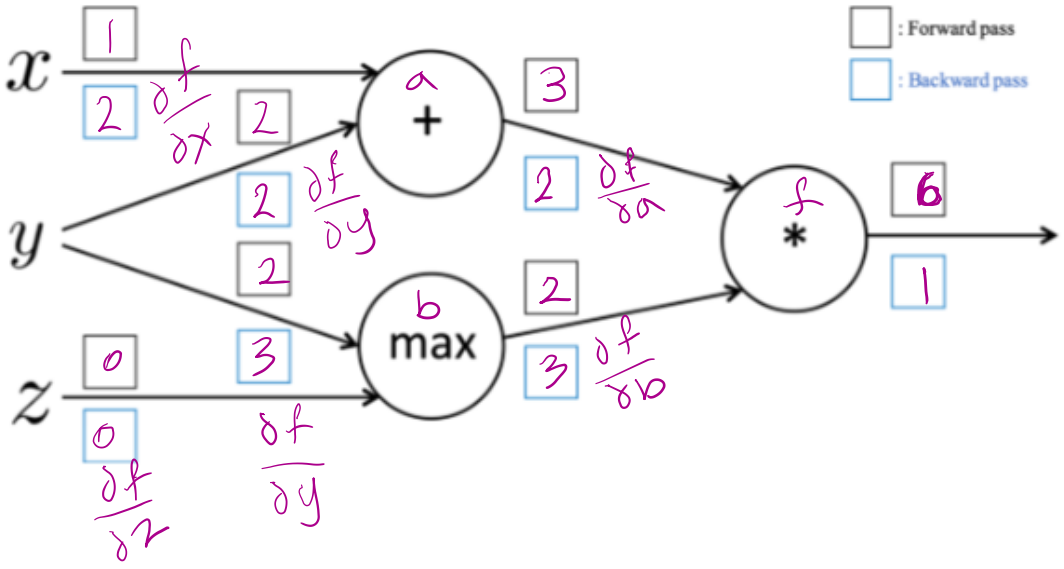
$$\frac{df}{da} = b = \max(2,0) = 2, \quad \frac{df}{db} = a = x + y = 1 + 2 = 3$$

$$\frac{da}{dx} = 1, \quad \frac{da}{dy} = 1, \quad \frac{db}{dy} = 1, \quad \frac{db}{dz} = 1$$

$$\frac{\partial f}{\partial x} = \frac{df}{da} * \frac{da}{dx} = 2$$

$$\frac{\partial f}{\partial y} = \frac{df}{db} * \frac{db}{dy} = 3\,, \quad \frac{\partial f}{\partial y} = \frac{df}{da} * \frac{da}{dy} = 2$$

$$\frac{\partial f}{\partial z} = 0$$



#4-b

Above diagram shows the intuition level behind the operations that backpropagation performs in order to calculate the gradients on the inputs.

Add distributes gradients equally to all its inputs.

Max matches the gradient to the higher input.

Multiply takes the input activations, swaps them and multiplies by its gradient.

Programming part

#1

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

It halts creating and optimizing clusters when either:

The centroids have stabilized there is no change in their values because the clustering has been successful.

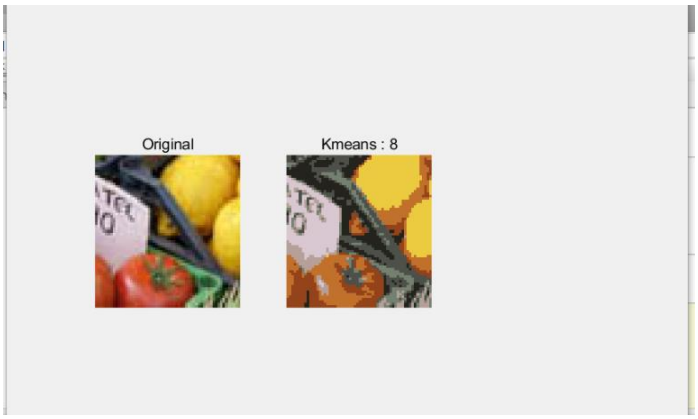
The defined number of iterations has been achieved.

Here are the functions I used:

TF = isnan(A) returns an array the same size as A containing logical 1 (true) where the elements of A are NaNs and logical 0 (false) where they are not.

X = randi(imax) returns a pseudorandom scalar integer between 1 and imax

At the beginning I got the black output image but later I changed number of centroids, number of iterations so the out put came as below



#2

Multi-Layer Perceptron is similar to the theory question

I did implement forward protagation, using currently learned network

Here is the algotithmn

```
n = size(X,2); % num of samples
   % forward propagation [[[here]]]
   for l = 1 : L
      activation{l}=zeros(ei.layer_sizes(l),size(X,2));


   %%%%%%%%%%%%%
    if l==1
       activation{1} = wStack{1}.W * X + repmat(wStack{1}.b, 1, n);
    end
    if l == L
       Y = wStack{l}.W * activation{l-1} + repmat(wStack{l}.b, 1, n);
       Y = exp(Y);
       % C = bsxfun(fun,A,B) applies the element-wise binary operation
       % specified by the function handle fun to arrays A and B.
       % Left array divide >> @rdivide
       activation{l} = bsxfun(@rdivide, Y, sum(Y,1));
       break;
%% here I did not break it at the beginning so the accurency was about 0.1 aand not working later after I did break I got 93 percent
    End
```

명령 창

61	226	3.49485e-02	5.10059e+01	6.28257e-02
62	241	1.62494e-02	5.10054e+01	6.15493e-02
63	256	1.92667e-02	5.10045e+01	5.96600e-02
64	273	2.25667e-02	5.09998e+01	5.80537e-02
65	287	0.00000e+00	5.09998e+01	5.80537e-02

Step Size below progTol

train accuracy: 0.918033

test accuracy: 0.921700

>>