1. 1) support vectors are:

Point #2 = $\langle 0.91, 0.32 \rangle$

Point #18 = $\langle 2.05, 1.54 \rangle$

Point #19 = $\langle 2.34, 0.72 \rangle$

2) $\vec{w} = 0.9492 \begin{bmatrix} .91 \\ .32 \end{bmatrix} - 0.3030 \begin{bmatrix} 2.05 \\ 1.54 \end{bmatrix} - 0.9053 \begin{bmatrix} 2.34 \\ 0.72 \end{bmatrix}$

$$\vec{w} = \begin{bmatrix} -1.87578 \\ -0.814692 \end{bmatrix}$$

3) $b = 1 - [-1.87578, -0.814692] \begin{bmatrix} .91 \\ .32 \end{bmatrix}$

$$+ (-1 - [-1.87578, -0.814692] \begin{bmatrix} 2.05 \\ 1.54 \end{bmatrix}$$

$$+ (-1 - [-1.87578, -0.814692] \begin{bmatrix} 2.34 \\ 0.72 \end{bmatrix}$$

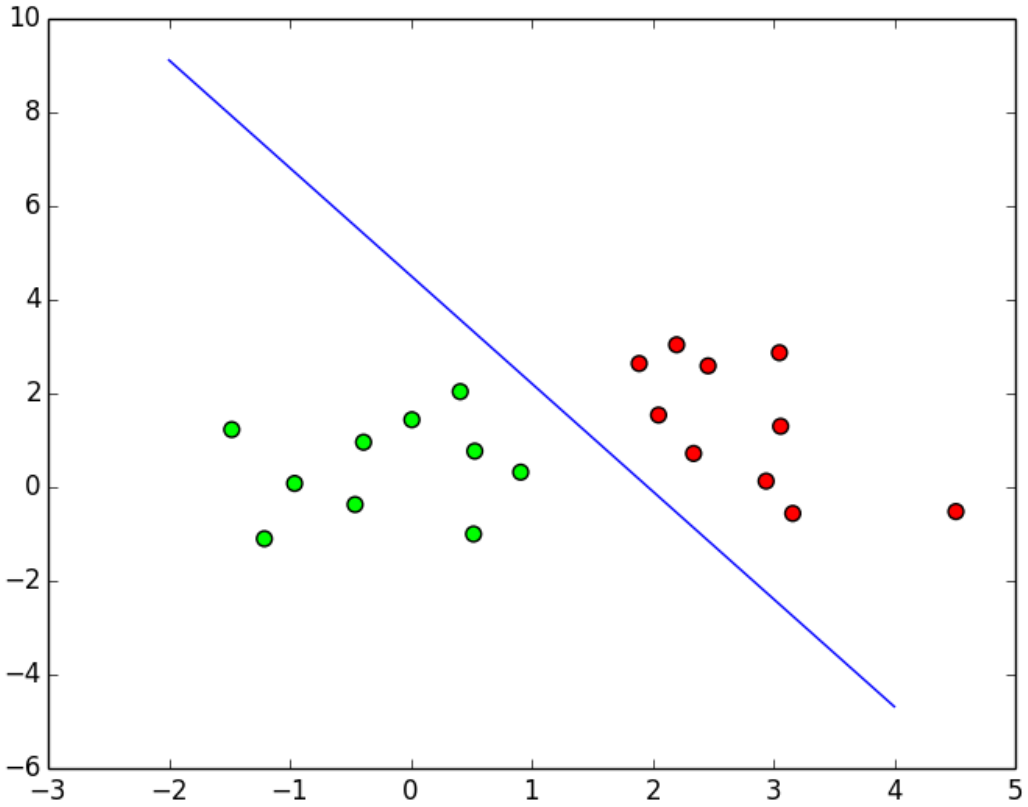
$$= 3.68$$

4) $f(x) = [-1.87578, -0.814692] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 3.68$

5) $f(-1, 2) = [-1.87578, -0.814692] \begin{bmatrix} -1 \\ 2 \end{bmatrix} + 3.62$

$3.866 > 1$

Class label 1



1.6) Plot of data points and decision boundary. (X1 feature on x-axis, X2 feature on y-axis)

2.1) $4p+24+5k$ (assumed bias node for every layer except output)

2.2) The major steps involved in backpropagation are as follows:

For each training example

propagate the inputs by applying activation function

For a hidden or output layer unit j

- calculate net input: $I_j = \sum_i w_{ij} O_i + \Theta_j$
- Calculate output of unit j: $O_j = \sigma(I_j) = \frac{1}{1+e^{-I_j}}$

Backpropagate the error by updating weights and biases

- For unit j in the output layer
  $Err_j = O_j(1-O_j)(T_j - O_j)$
- For unit j in a hidden layer
  $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$
- Update weights
  $w_{ij} = w_{ij} + \eta Err_j O_i$
- Update bias:
  $\Theta_j = \Theta_j + \eta Err_j$

Terminate when we obtain sufficiently small error

2.3.

Unit j	Net Input I_j	Output O_j
3	0.6	$\frac{1}{1+e^{-0.6}} = 0.645$
4	$2(0)+(-.1)(1)+(-.4)$ $= -0.5$	$\frac{1}{1+e^{0.5}} = 0.378$
5	$(-0.2)(0.645)+(-.3)$ $\times(.378)+0.1$ $= -0.142$	$\frac{1}{1+e^{0.142}} = 0.465$

Error Unit j	Err_j
5	$0.465(1-0.465)(1-0.465) = 0.133$
4	$0.378(1-0.378)(0.133)(0.3) = 0.009$
3	$0.645(1-0.645)(0.133)(-0.2) = -0.006$

$Err_j = O_j(1-O_j)(T_j - O_j)$

$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$

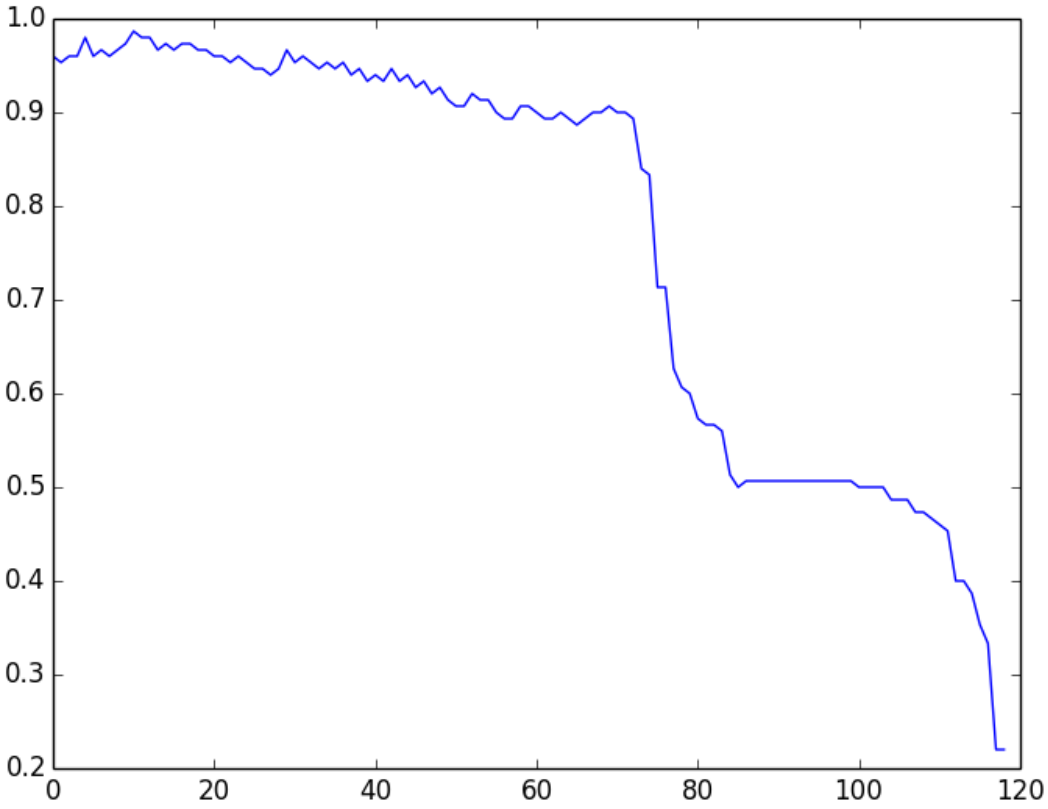
$\theta_j = \theta_j + \eta Err_j$

$w_{ij} = w_{ij} + \eta Err_j O_i$

Weight or Bias	New Val
w_{35}	$-0.2 + 0.8(.133)(.645) = -.131$
w_{45}	$-0.3 + 0.8(.133)(.378) = -.2597808$
w_{13}	$-0.3 + 0.8(-.006)(0) = -0.3$
w_{14}	$0.2 + 0.8(-.009)(0) = 0.2$
w_{23}	$0.4 + 0.8(-.006)(1) = 0.3952$
w_{24}	$-0.1 + 0.8(-.009)(1) = -0.1072$
θ_5	$0.1 + 0.8(.133) = 0.2064$
θ_4	$-0.4 + 0.8(-.009) = -0.4072$
θ_3	$0.2 + 0.8(-.006) = 0.1952$

3.1) The best value of K obtained was 11. In general the the best K-value returned by my program usually ranged from 7-14, but sometimes I would obtain K values as low as 4.

3.2) Plot of K-value vs average accuracy over 5 folds. K-values are on the x-axis while average accuracies are on the y-axis.



3.3) My reported K-value gives a better average accuracy than extreme K-values such as K=1 or K=120 because these extreme values are prone to overfitting and underfitting respectively. This is because when K is extremely small, the classification is oversensitive to the labels of the small number of surrounding points, while if K is huge the opposite is true (if you look at too many neighbors it becomes unclear what you should classify as).