Homework 5

Question 1

a) $x_1 \Rightarrow \sqrt{(3-0)^2+(1-0)^2},\ \sqrt{(3-4)^2+(1-3)^2},\ \sqrt{(3-2)^2+(1+3)^2} = \boxed{\sqrt{5}}$

b) $x_2 \Rightarrow \sqrt{(1-0)^2+(-2+0)^2},\ \sqrt{(1-4)^2+(-2-3)^2},\ \sqrt{(1-2)^2+(-2+3)^2} = \boxed{\sqrt{2}}$

c) $x_3 \Rightarrow \sqrt{(6-0)^2+(10-0)^2},\ \sqrt{(6-4)^2+(10-3)^2},\ \sqrt{(6-2)^2+(10+3)^2} = \boxed{\sqrt{53}}$

d) $x_4 \Rightarrow \sqrt{(-3-0)^2+(6-0)^2},\ \sqrt{(-3-4)^2+(6-3)^2},\ \sqrt{(-3-2)^2+(6+3)^2} = \boxed{\sqrt{45}}$

e) $x_5 \Rightarrow \sqrt{(1-0)^2+(-1-0)^2},\ \sqrt{(1-4)^2+(-1-3)^2},\ \sqrt{(1-2)^2+(-1+3)^2} = \boxed{\sqrt{2}}$

site 1 site 2 site 3

f) $$f_\mu \Sigma(x) = \frac{1}{2\pi^{d/2}} \frac{1}{\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

Σ_1

$$f_\mu \Sigma_1(x_1) = \frac{1}{2\pi}\frac{1}{4.0}\exp\left(-\frac{1}{2}\begin{bmatrix}3\\1\end{bmatrix}\frac{1}{4.0}\begin{bmatrix}2.0 & 0\\0 & 2.0\end{bmatrix}[1,3]\right)$$

$$f_\mu \Sigma_1(x_2) = \frac{1}{2\pi}\frac{1}{4.0}\exp\left(-\frac{1}{2}\begin{bmatrix}1\\-2\end{bmatrix}\frac{1}{4.0}\begin{bmatrix}2.0 & 0\\0 & 2.0\end{bmatrix}[-2,1]\right)$$

$$f_\mu \Sigma_1(x_3) = \frac{1}{2\pi}\frac{1}{4.0}\exp\left(-\frac{1}{2}\begin{bmatrix}6\\10\end{bmatrix}\frac{1}{4.0}\begin{bmatrix}2.0 & 0\\0 & 2.0\end{bmatrix}[10,6]\right)$$

$$f_\mu \Sigma_1(x_4) = \frac{1}{2\pi}\frac{1}{4.0}\exp\left(-\frac{1}{2}\begin{bmatrix}-3\\6\end{bmatrix}\frac{1}{4.0}\begin{bmatrix}2.0 & 0\\0 & 2.0\end{bmatrix}[6,-3]\right)$$

$$f_\mu \Sigma_1(x_5) = \frac{1}{2\pi}\frac{1}{4.0}\exp\left(-\frac{1}{2}\begin{bmatrix}1\\-1\end{bmatrix}\frac{1}{4.0}\begin{bmatrix}2.0 & 0\\0 & 2.0\end{bmatrix}[-1,1]\right)$$

Σ_2

$$f_\mu \Sigma_2(x_1) = \frac{1}{2\pi}\frac{1}{16.0}\exp\left(-\frac{1}{2}\begin{bmatrix}-1\\-2\end{bmatrix}\frac{1}{16.0}\begin{bmatrix}4.0 & 0\\0 & 4.0\end{bmatrix}[-2,-1]\right)$$

$$f_\mu \Sigma_2(x_2) = \frac{1}{2\pi}\frac{1}{16.0}\exp\left(-\frac{1}{2}\begin{bmatrix}-3\\-5\end{bmatrix}\frac{1}{16.0}\begin{bmatrix}4.0 & 0\\0 & 4.0\end{bmatrix}[-5,-3]\right)$$

$$f_\mu \Sigma_2(x_3) = \frac{1}{2\pi}\frac{1}{16.0}\exp\left(-\frac{1}{2}\begin{bmatrix}2\\7\end{bmatrix}\frac{1}{16.0}\begin{bmatrix}4.0 & 0\\0 & 4.0\end{bmatrix}[7,2]\right)$$

$$f_\mu \Sigma_2(x_4) = \frac{1}{2\pi}\frac{1}{16.0}\exp\left(-\frac{1}{2}\begin{bmatrix}-7\\3\end{bmatrix}\frac{1}{16.0}\begin{bmatrix}4.0 & 0\\0 & 4.0\end{bmatrix}[3,-7]\right)$$

$$f_\mu \Sigma_2(x_5) = \frac{1}{2\pi}\frac{1}{16.0}\exp\left(-\frac{1}{2}\begin{bmatrix}-3\\-4\end{bmatrix}\frac{1}{16.0}\begin{bmatrix}4.0 & 0\\0 & 4.0\end{bmatrix}[-4,-3]\right)$$

Σ_3

$$f_u \Sigma_3(x_1) = \frac{1}{2\pi}\,\frac{1}{25.0}\,\exp\left(-\frac{1}{2}\begin{bmatrix}1\\4\end{bmatrix}\frac{1}{25.0}\begin{bmatrix}5&0\\0&5\end{bmatrix}[4,1]\right)$$

$$f_u \Sigma_3(x_2) = \frac{1}{2\pi}\,\frac{1}{25.0}\,\exp\left(-\frac{1}{2}\begin{bmatrix}-1\\1\end{bmatrix}\frac{1}{25.0}\begin{bmatrix}5&0\\0&5\end{bmatrix}[1,-1]\right)$$

$$f_u \Sigma_3(x_3) = \frac{1}{2\pi}\,\frac{1}{25.0}\,\exp\left(-\frac{1}{2}\begin{bmatrix}4\\13\end{bmatrix}\frac{1}{25.0}\begin{bmatrix}5&0\\0&5\end{bmatrix}[13,4]\right)$$

$$f_u \Sigma_3(x_4) = \frac{1}{2\pi}\,\frac{1}{25.0}\,\exp\left(-\frac{1}{2}\begin{bmatrix}-5\\9\end{bmatrix}\frac{1}{25.0}\begin{bmatrix}5&0\\0&5\end{bmatrix}[9,-5]\right)$$

$$f_u \Sigma_4(x_5) = \frac{1}{2\pi}\,\frac{1}{25.0}\,\exp\left(-\frac{1}{2}\begin{bmatrix}-1\\2\end{bmatrix}\frac{1}{25.0}\begin{bmatrix}5&0\\0&5\end{bmatrix}[2,-1]\right)$$

Question 3

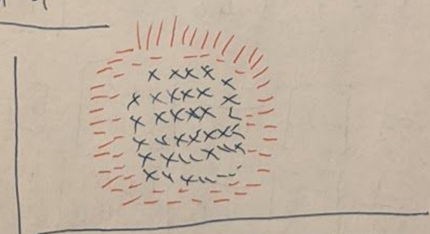
a) $$\Delta(z_i) = \begin{cases} 1 & \text{if } z(i) < -2 \\ 1/2 & \text{if } z(i) \le 2 \text{ and } -2 \le z(i) \\ 0 & \text{if } z(i) > 2 \end{cases}$$

b) $$l_\Delta(z) = \ln(1 + \exp(-x)) + 2$$

(See the picture in write up for the graph)

Question 4

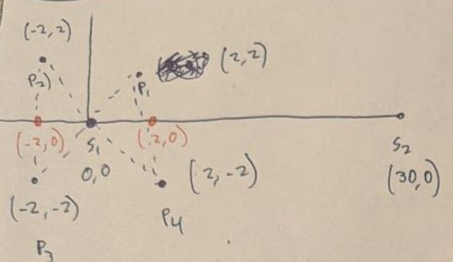
a)



b) The perceptron algorithm will run forever since there will always be a misclassified point in the example given

c) Feature Expansion can be used to map the data to a higher dimension which will make it possible to use a linear classifier.

Question 2



Points : $(2,2)$ $(-2,2)$ $(-2,-2)$ $(2,-2)$

Let S represent sites

Let S' represent prime sites

If all the points are grouped around 1 site, then Lloyd's will terminate since the set S is unchanged. S' is better than S because using the pythagorean theorem we can see that some points are close to S' and we get two clusters

---

Extra Credit

Question 6

a) i) $\|A_2\|_2^2 = \boxed{\sigma_1^2}$

ii) $\|A_3\|_F^2 = \boxed{\sum_{i=1}^{3} \sigma_i^2}$

b) i) $A_3 = \boxed{\sum_{j=1}^{3} \sigma_j u_j v_j^T}$

ii) $A_3 - A_2 = \boxed{\sum_{j=1}^{3} \sigma_j U_j V_j^T - \sum_{i=1}^{2} \sigma_i U_i V_i^T}$

c) $\boxed{\pi_{A_3}(x) = \sum_{i=1}^{3} v_i \langle x, v_i \rangle}$

Question 5

a) $\boxed{\langle \alpha, (1, (a_i, b_i), (a_i, b_i)^2) \rangle}$

b) $\boxed{2 \sum_{i=1}^{n} (M_\alpha(x_i) - y_i)(1, x_i, x_i^2)}$

Results for 1 g-k

w1(x1) = 0.22270013882530787
w1(x2) = 0.7772998611746875
w1(x3) = 4.6499074378633745e-15
w1(x4) = 3.528935910172004e-05
w1(x5) = 1.6455438190198433
w2(x1) = 0.9716965685781364
w2(x2) = 0.025894836437441376
w2(x3) = 0.0024085949844222863
w2(x4) = 0.001289227992085548
w2(x5) = 0.0797617114945835
w3(x1) = 19776402.658497844
w3(x2) = 88631687.64519446
w3(x3) = 1.0
w3(x4) = 2697.28232826852
w3(x5) = 65659969.137330756

Code for 1 g-k

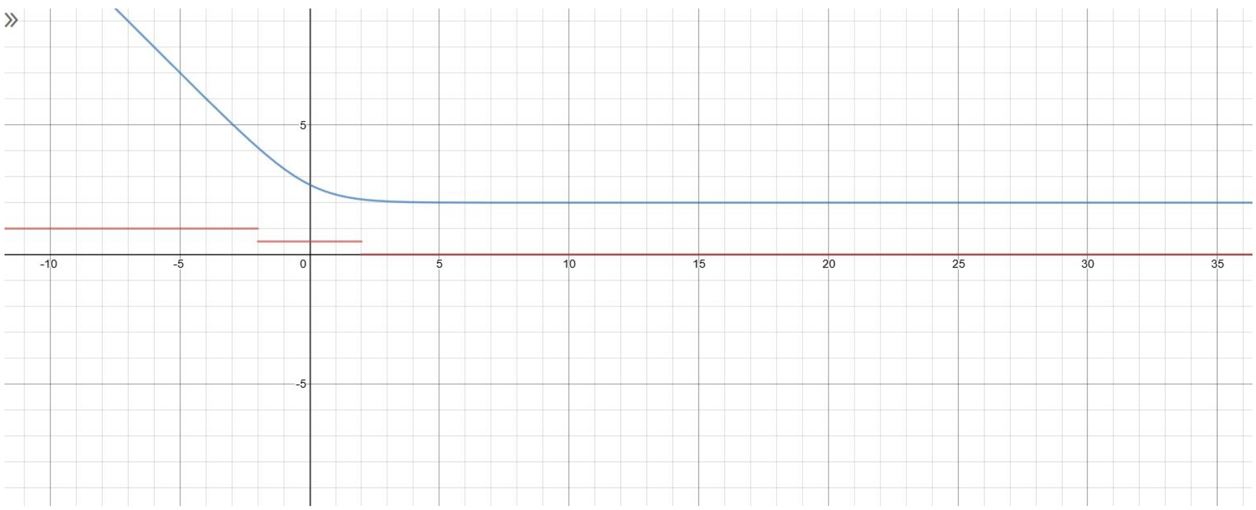
```
import numpy as np
import math
import array
# the sites
s1x = 0
s1y = 0
s2x = 3
s2y = 4
s2x = -3
s2y = 2

# sigma values for the matrix
sig1 = 2.0
sig2 = 4.0
sig3 = 5

# points
x1x = 1
x1y = 3
x2x = -2
x2y = 1
x3x = 10
x3y = 6

# foo = np.array([[1, 2]])
# print(foo)
# foo_transpose = np.transpose(foo)
# print(foo_transpose)
# def calculate_weight(pointx, pointy, sitex, sitey, sigma):
#     point = np.array([pointx, pointy])
#     site = np.array([sitex, sitey])
#     difference = point - site
#     convariance_matrix = np.identity(2) * sigma
#     inverse_covariance = np.linalg.inv(convariance_matrix)
#     determinant = np.linalg.det(convariance_matrix)
#     determinant_constant = 1/math.sqrt(determinant)
#     exponent = math.exp((-1/2 * difference.T).dot(inverse_covariance).dot(point))
#     result = (1/2 * math.pi) * determinant_constant * exponent
#     return result

def calculate_weight_array(point, site, sigma):
  pointx = point[0]
  pointy = point[1]
  sitex = site[0]
  sitey = site[1]
  point = np.array([pointx, pointy])
```

```python
    site = np.array([sitex, sitey])
    difference = point - site
    convariance_matrix = np.identity(2) * sigma
    inverse_covariance = np.linalg.inv(convariance_matrix)
    determinant = np.linalg.det(convariance_matrix)
    determinant_constant = 1/math.sqrt(determinant)
    exponent = math.exp((-1/2 * difference.T).dot(inverse_covariance).dot(difference))
    result = (1/2 * math.pi) * determinant_constant * exponent
    return result

point_array = [np.array([1, 3]), np.array([-2, 1]), np.array([10, 6]), np.array([6, -3]), np.array([-1, 1])]
site_array = [np.array([0, 0]), np.array([3, 4]), np.array([-3, 2])]

# Weight1
w1_x1_numerator = calculate_weight_array(point_array[0], site_array[0], sig1)
w1_x1_denominator = 0
for g in range(0, 3):
    w1_x1_denominator += calculate_weight_array(point_array[g], site_array[0], sig1)
w1_x1 = w1_x1_numerator / w1_x1_denominator
print(f"w1(x1) = {w1_x1}")
w1_x2_numerator = calculate_weight_array(point_array[1], site_array[0], sig1)
w1_x2_denominator = 0
for g in range(0, 3):
    w1_x2_denominator += calculate_weight_array(point_array[g], site_array[0], sig1)
w1_x2 = w1_x2_numerator / w1_x2_denominator
print(f"w1(x2) = {w1_x2}")
w1_x3_numerator = calculate_weight_array(point_array[2], site_array[0], sig1)
w1_x3_denominator = 0
for g in range(0, 3):
    w1_x3_denominator += calculate_weight_array(point_array[g], site_array[0], sig1)
w1_x3 = w1_x3_numerator / w1_x3_denominator
print(f"w1(x3) = {w1_x3}")
w1_x4_numerator = calculate_weight_array(point_array[3], site_array[0], sig1)
w1_x4_denominator = 0
for g in range(0, 3):
    w1_x4_denominator += calculate_weight_array(point_array[g], site_array[0], sig1)
w1_x4 = w1_x4_numerator / w1_x4_denominator
print(f"w1(x4) = {w1_x4}")
w1_x5_numerator = calculate_weight_array(point_array[4], site_array[0], sig1)
w1_x5_denominator = 0
for g in range(0, 3):
    w1_x5_denominator += calculate_weight_array(point_array[g], site_array[0], sig1)
w1_x5 = w1_x5_numerator / w1_x5_denominator
print(f"w1(x5) = {w1_x5}")

# Weight 2
w2_x1_numerator = calculate_weight_array(point_array[0], site_array[1], sig2)
w2_x1_denominator = 0
```

```python
for h in range(0, 3):
  w2_x1_denominator += calculate_weight_array(point_array[h], site_array[1], sig2)
w2_x2_numerator = calculate_weight_array(point_array[1], site_array[1], sig2)
w2_x1 = w2_x1_numerator / w2_x1_denominator
print(f"w2(x1) = {w2_x1}")
w2_x2_denominator = 0
for h in range(0, 3):
  w2_x2_denominator += calculate_weight_array(point_array[h], site_array[1], sig2)
w2_x2 = w2_x2_numerator / w2_x2_denominator
print(f"w2(x2) = {w2_x2}")
w2_x3_numerator = calculate_weight_array(point_array[2], site_array[1], sig2)
w2_x3_denominator = 0
for h in range(0, 3):
  w2_x3_denominator += calculate_weight_array(point_array[h], site_array[1], sig2)
w2_x3 = w2_x3_numerator / w2_x3_denominator
print(f"w2(x3) = {w2_x3}")
w2_x4_numerator = calculate_weight_array(point_array[3], site_array[1], sig2)
w2_x4_denominator = 0
for h in range(0, 3):
  w2_x4_denominator += calculate_weight_array(point_array[h], site_array[1], sig2)
w2_x4 = w2_x4_numerator / w2_x4_denominator
print(f"w2(x4) = {w2_x4}")
w2_x5_numerator = calculate_weight_array(point_array[4], site_array[1], sig2)
w2_x5_denominator = 0
for h in range(0, 3):
  w2_x5_denominator += calculate_weight_array(point_array[h], site_array[1], sig2)
w2_x5 = w2_x5_numerator / w2_x5_denominator
print(f"w2(x5) = {w2_x5}")
# Weight 3
w3_x1_numerator = calculate_weight_array(point_array[0], site_array[2], sig3)
w3_x1_denominator = 0
for t in range(0, 3):
  w3_x1_denominator = calculate_weight_array(point_array[t], site_array[2], sig3)
w3_x1 = w3_x1_numerator / w3_x1_denominator
print(f"w3(x1) = {w3_x1}")
w3_x2_numerator = calculate_weight_array(point_array[1], site_array[2], sig3)
w3_x2_denominator = 0
for t in range(0, 3):
  w3_x2_denominator = calculate_weight_array(point_array[t], site_array[2], sig3)
w3_x2 = w3_x2_numerator / w3_x2_denominator
print(f"w3(x2) = {w3_x2}")
w3_x3_numerator = calculate_weight_array(point_array[2], site_array[2], sig3)
w3_x3_denominator = 0
for t in range(0, 3):
  w3_x3_denominator = calculate_weight_array(point_array[t], site_array[2], sig3)
w3_x3 = w3_x3_numerator / w3_x3_denominator
print(f"w3(x3) = {w3_x3}")
w3_x4_numerator = calculate_weight_array(point_array[3], site_array[2], sig3)
```

```python
w3_x4_denominator = 0
for t in range(0, 3):
  w3_x4_denominator = calculate_weight_array(point_array[t], site_array[2], sig3)
w3_x4 = w3_x4_numerator / w3_x4_denominator
print(f"w3(x4) = {w3_x4}")
w3_x5_numerator = calculate_weight_array(point_array[4], site_array[2], sig3)
w3_x5_denominator = 0
for t in range(0, 3):
  w3_x5_denominator = calculate_weight_array(point_array[t], site_array[2], sig3)
w3_x5 = w3_x5_numerator / w3_x5_denominator
print(f"w3(x5) = {w3_x5}")
```

Desmos graph picture for question 3



Citations
(This article was used for understanding question 4)
https://towardsdatascience.com/truly-understanding-the-kernel-trick-1aeb11560769