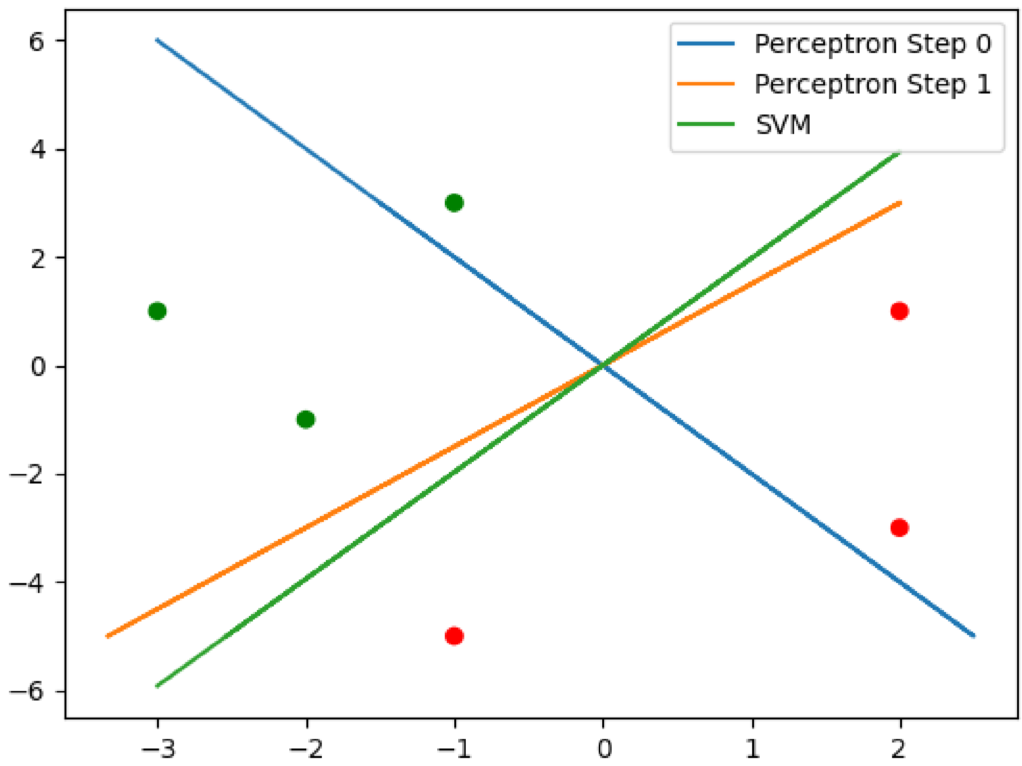
Exercise 1

See Referencesappendix for code.



(a) Perceptron Learning:

Updating vector $w = (0,0)$ using $(x,y) = ((2,1),-1)$
$w = (-2,-1) \rightarrow w = (0,0) + y = -1 * x = (2,1)$

Updating vector $w = (-2,-1)$ using $(x,y) = ((-1,3),1)$
$w = (-3,2) \rightarrow w = (-2,-1) + y = 1 * x = (-1,3)$

Margin $(\min_{(x,y)\in S} \frac{|\langle w,x\rangle|}{||w||})$: 1.1094003924504583

(b) SVM Learning:

$w^* : (-0.6074814098863559, 0.30790724037405226)$
Margin $(\min_{(x,y)\in S} \frac{|\langle w,x\rangle|}{||w||})$: 1.331824259573374

Exercise 2

(a) $\hat{w} = \mathbf{1} = (1, \dots, 1) \in \{1\}^n$ is a suitable weight vector, since $\langle \hat{w}, x \rangle$ is only positive, iff x contains more 1's than -1's.

(b) $\lambda = n$, since $\|x\|$ is maximum when x consists of either only 1's or only -1's.
$\gamma = \frac{1}{n}$ since the margin is minimal for a x which consists of an by one number off amount of 1's and $-1'$. Thus, $\frac{|\langle w,x \rangle|}{\|w\|} = \frac{1}{n}$
Using Theorem 1.13 we can derive that the perceptron algorithm finds a linear separator after at most $\left(\frac{\lambda}{\gamma}\right)^2 = \left(\frac{n}{\frac{1}{n}}\right)^2 = n^4$ updates.

(c) The smallest possible number of updates is 1. Let $(x_1 \coloneqq (1, \dots, 1), y_1 \coloneqq 1)$ be the first element in S.
Initially the weight vector is $w = ((0, \dots, 0))$. Thus, $sgn(\langle w, x_1 \rangle) \neq y_1$ holds and we update $w \leftarrow w + y_1 x_1 = (1, \dots, 1)$.
As discussed previously, this is already a suitable weight vector, therefore we do not have to update w ever again.

(d) No, we cannot. Counterexample:
Let $n = 3$, $x = (1, 1, x_3) \in \mathbb{R}^3$. We know, that x should be classified as 1. However, for every weight vector $w = (w_1, w_2, w_3) \in \mathbb{R}^3$ we can set x_3 to the value of $x_3 \coloneqq -\frac{w_1+w_2}{w_3}$ ($x_3 \coloneqq -(w_1 + w_2)$ iff $w_3 = 0$), that would lead to a false classification of x:

$$sgn(\langle w, x \rangle) = sgn(w_1 + w_2 + x_3 \cdot w_3) = sgn(0) = 0 \neq 1$$

So for every w we can compute a vector x so that w is not consistent with $\{(x, maj(x))\}$. Thus, no such linear separator exists.

Exercise 3

(a)

$$\tau : \mathbb{R}^2 \to \mathbb{R}^4, (x, y) \mapsto (x, y, (\sin(xy))^2, (\cos(xy))^2)$$

(b)

$$\tau' : \mathbb{R}^2 \to \mathbb{R}^3, (x, y) \mapsto (x, y, (\sin(xy))^2)$$

We make use of $1 = (\sin(\phi))^2 + (\cos(\phi))^2$. Hence:

$$\begin{aligned}
h(x, y) &= sgn(w_1 x + w_2 y + w_3(\sin(xy))^2 + w_4(\cos(xy))^2 - b) \\
&= sgn(w_1 x + w_2 y + w_3(\sin(xy))^2 + w_4(1 - (\sin(xy)))^2 - b) \\
&= sgn(w_1 x + w_2 y + w_3(\sin(xy))^2 + w_4 - w_4(\sin(xy))^2 - b) \\
&= sgn(w_1 x + w_2 y + (w_3 + w_4) \cdot (\sin(xy))^2 - (b - w_4)) \\
&= sgn(w_1 x + w_2 y + w_3' \cdot (\sin(xy))^2 - b') \\
&=: h'(x, y)
\end{aligned}$$

where $w_3' \coloneqq w_3 + w_4$ and $b' \coloneqq b - w_4$.

Exercise 4

See Referencesappendix for code.

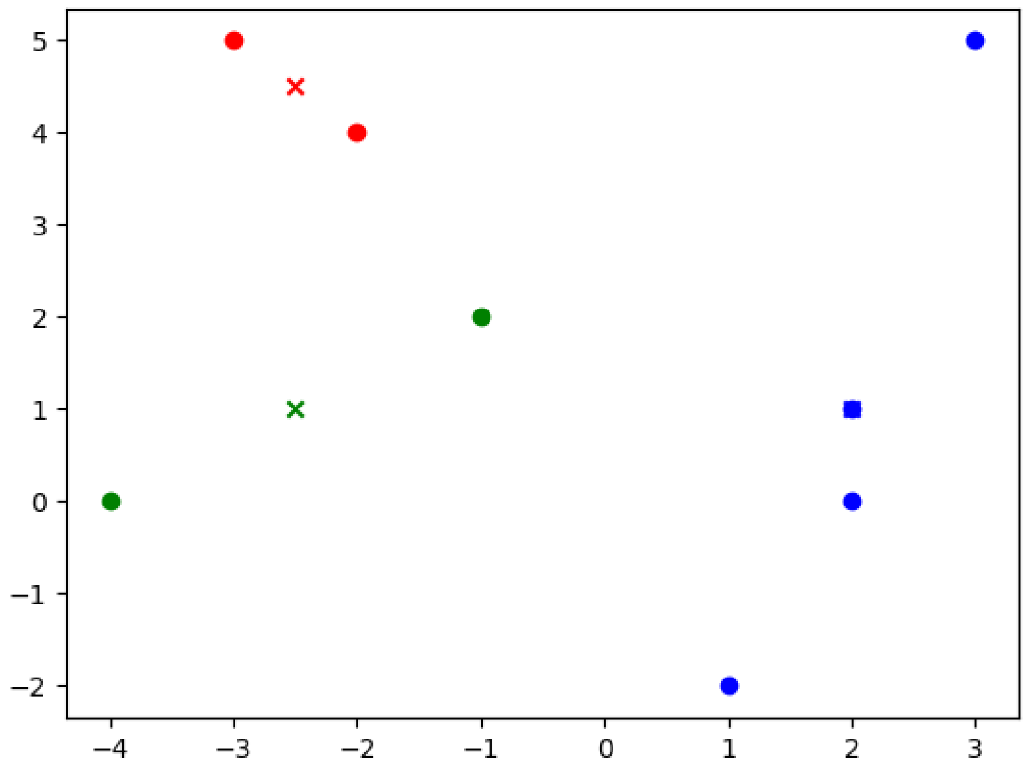
(a) Clusters: [[], [], []]
Centres: [(-3, 5), (-2, 4), (-1, 2)]

Clusters: [[(-3, 5)], [(-2, 4)], [(-1, 2), (-4, 0), (1, -2), (2, 0), (2, 1), (3, 5)]]
Centres: [(-3.0, 5.0), (-2.0, 4.0), (0.5, 1.0)]

Clusters: [[(-3, 5)], [(-2, 4), (-4, 0)], [(-1, 2), (1, -2), (2, 0), (2, 1), (3, 5)]]
Centres: [(-3.0, 5.0), (-3.0, 2.0), (1.4, 1.2)]

Clusters: [[(-3, 5), (-2, 4)], [(-1, 2), (-4, 0)], [(1, -2), (2, 0), (2, 1), (3, 5)]]
Centres: [(-2.5, 4.5), (-2.5, 1.0), (2.0, 1.0)]

Final Clusters: [[(-3, 5), (-2, 4)], [(-1, 2), (-4, 0)], [(1, -2), (2, 0), (2, 1), (3, 5)]]
Final Centers: [(-2.5, 4.5), (-2.5, 1.0), (2.0, 1.0)]



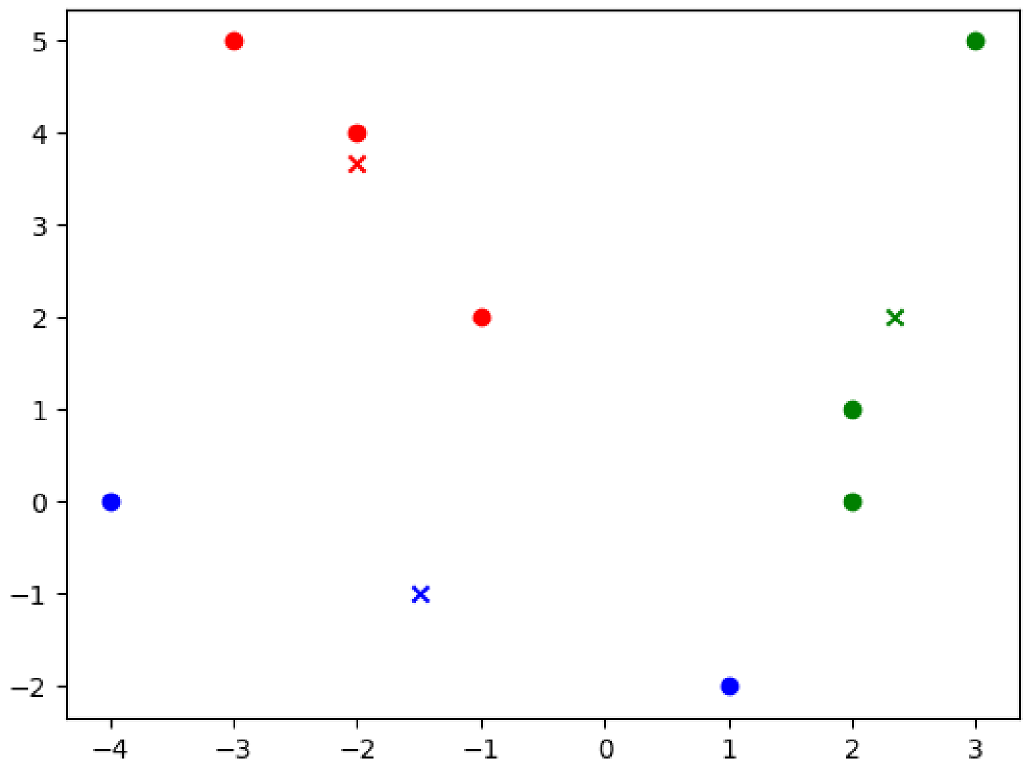
(b)

(c) Circumcenter $c = (-0.25, 2.75)$
Line 1: $y = 2.75$ (separating red and green)
Line 2: $x = -0.25$ (separating green and blue)
Line 3: $y = 2.75 + \frac{x+0.25}{3.5} \cdot 0.5$ (separating red and blue)

(d)

$$z^1 = x_1 = (-3, 5), z^2 = x_8 = (3, 5), z^2 = x_4 = (-4, 0)$$

produces following clustering:



(e) Yes, for example $z^1 = \begin{bmatrix} -4 \\ 3 \end{bmatrix}$, $z^2 = \begin{bmatrix} -1.5 \\ 3 \end{bmatrix}$, $z^3 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$,

Exercise 5

(a)

$$\mathbb{E}[Y_n] = \mathbb{E}[\prod_{i=1}^{n} X_i] = \prod_{i=1}^{n} \mathbb{E}[X_i] = \prod_{i=1}^{n} 3.5 = 3.5^n$$

$$\begin{aligned} \mathrm{Var}(Y_n) &= \mathbb{E}[(Y_n - \mathbb{E}[Y_n])^2] = \mathbb{E}[(Y_n - 3.5^n)^2] = \mathbb{E}[Y_n^2 - 2 \cdot 3.5^n \cdot Y_n + 3.5^{2n}] \\ &= \mathbb{E}[Y_n^2] - 2 \cdot 3.5^n \cdot \mathbb{E}[Y_n] + 3.5^{2n} \\ &= \mathbb{E}[Y_n] \cdot \mathbb{E}[Y_n] - 2 \cdot 3.5^n \cdot 3.5^n + 3.5^{2n} \\ &= 3.5^n \cdot 3.5^n - 2 \cdot 3.5^n \cdot 3.5^n + 3.5^{2n} \\ &= 0 \end{aligned}$$

(b) (i)

$$Pr(Z_{300} \leq 10) = 1 - Pr(Z_{300} \geq 11) \geq 1 - \frac{\frac{50}{11}}{300} = 1 - \frac{65}{66} = \frac{1}{66} = 0.015$$

(ii)

$$P(|Z_{300} - E(X)| \leq 10) = P(|Z_{300} - E(X)| < 11) \geq 1 - \frac{V(X)}{11^2} = 1 - \frac{\frac{125}{3}}{11^2} = \frac{238}{363} = 0.656$$

(iii) Chernoff:

$$P\left[\sum X_i \leq (1-\delta)\cdot pn\right] \leq \exp\left(-\frac{\delta^2}{2}pn\right)$$

$$\Rightarrow (1-\delta)\cdot 50 = 10 \Leftrightarrow 1-\delta = \frac{1}{5} \Leftrightarrow -\delta = -\frac{4}{5} \Leftrightarrow \delta = \frac{4}{5}$$

$$\Rightarrow P(X \leq 10) \leq \exp(-\frac{\frac{4}{5}^2}{2}\cdot 50) = 1.125\cdot 10^{-7}$$

(iv)

$$\Pr\left[\sum(X_i - \mathrm{E}[X_i]) \geq c\right] \leq \exp\left(\frac{-2c^2}{\sum(b_i-a_i)^2}\right)$$

with $a = 0$ and $b = 1$. Centering around the mean of 50 leads to $a = -50$ and $b = -49$ and $c = 10 - 50 = -40$:

$$P(X \leq 10) = 1 - P(X10) > 1 - \exp\left(\frac{-2(-40)^2}{300\cdot(-49+50)^2}\right) = 0.999$$

The best bound is given by the Chernoff inequality as the actual probability is binomially distributed with

$$P(X \leq 10) = 3.03\cdot 10^{-13}$$

with $n = 300$ and $p = \frac{1}{6}$.

Exercise 6

(a)

$$H(X|Y) + H(X) \Leftrightarrow (\sum_{y \in rg(Y)} Pr(Y=y) \left(\sum_{x \in rg(X)} Pr(X=x|Y=y) \cdot \log(\frac{1}{Pr(X=x|Y=y)} \right))$$

$$+ (\sum_{y \in rg(Y)} Pr(Y=y) \cdot \log(\frac{1}{Pr(Y=y)}))$$

$$\Leftrightarrow (\sum_{y \in rg(Y)} \sum_{x \in rg(X)} \left(Pr(X=x, Y=y) \cdot \log(\frac{Pr(Y=y)}{Pr(X=x, Y=y)} \right))$$

$$+ (\sum_{y \in rg(Y)} Pr(Y=y) \cdot \log(\frac{1}{Pr(Y=y)}))$$

$$\Leftrightarrow (\sum_{y \in rg(Y)} \sum_{x \in rg(X)} (Pr(X=x, Y=y) \cdot (\log(Pr(Y=y) - \log(Pr(X=x, Y=y))))$$

$$- (\sum_{y \in rg(Y)} Pr(Y=y) \cdot \log(Pr(Y=y)))$$

$$\Leftrightarrow (\sum_{y \in rg(Y)} \sum_{x \in rg(X)} (Pr(X=x, Y=y) \cdot \log(Pr(Y=y))))$$

$$+ (\sum_{y \in rg(Y)} \sum_{x \in rg(X)} (Pr(X=x, Y=y) \cdot -\log(Pr(X=x, Y=y))))$$

$$- (\sum_{\substack{y \in rg(Y) \\ x \in rg(X)}} Pr(Y=y, X=x) \cdot \log(Pr(Y=y)))$$

$$\Leftrightarrow (\sum_{y \in rg(Y)} \sum_{x \in rg(X)} (Pr(X=x, Y=y) \cdot \log(Pr(Y=y))))$$

$$+ H(X,Y)$$

$$- (\sum_{\substack{y \in rg(Y) \\ x \in rg(X)}} Pr(Y=y, X=x) \cdot \log(Pr(Y=y)))$$

$$\Leftrightarrow H(X,Y)$$

(b)

$$H(X|Y) \Leftrightarrow \sum_{y \in rg(Y)} Pr(Y=y) \left(\sum_{x \in rg(X)} Pr(X=x|Y=y) \cdot \log(\frac{1}{Pr(X=x|Y=y)}) \right)$$

$$\Leftrightarrow \sum_{y \in rg(Y)} Pr(Y=y) \left(\sum_{x \in rg(X)} Pr(X=x) \cdot \log(\frac{1}{Pr(X=x)}) \right)$$

$$\Leftrightarrow \sum_{y \in rg(Y)} Pr(Y=y) \, (H(X))$$

$$\Leftrightarrow H(X) \cdot \sum_{y \in rg(Y)} Pr(Y=y)$$

$$\Leftrightarrow H(X)$$

Appendix

Code for Exercise 1

```
from matplotlib import pyplot
from sklearn.svm import LinearSVC
from math import sqrt

S = [
    ((2, 1), -1),
    ((-1, 3), 1),
    ((-3, 1), 1),
    ((-2, -1), 1),
    ((-1, -5), -1),
    ((2, -3), -1),
]


def plot(S, w_percs, w_svm):
    # scatter points
    x_values = [s[0][0] for s in S]
    y_values = [s[0][1] for s in S]
    colors = ['green' if s[1] == 1 else 'red' for s in S]

    pyplot.scatter(x_values, y_values, c=colors)

    # plot linear seperators
    x_min = min(x_values)
    x_max = max(x_values)
    y_min = min(y_values)
    y_max = max(y_values)

    w_list = list(w_percs + [w_svm])

    for i in range(len(w_list)):
        w = w_list[i]
        ortho_w = (-w[1], w[0])

        p_1 = (x_min, ortho_w[1] * (x_min / ortho_w[0]))
        p_2 = (x_max, ortho_w[1] * (x_max / ortho_w[0]))
        p_3 = (ortho_w[0] * (y_min / ortho_w[1]), y_min)
        p_4 = (ortho_w[0] * (y_max / ortho_w[1]), y_max)

        p_x_values = (p_1[0], p_2[0], p_3[0], p_4[0])
        p_y_values = (p_1[1], p_2[1], p_3[1], p_4[1])

        pyplot.plot(p_x_values, p_y_values, label=('SVM' if w==w_svm else f'Perceptron Step {i}'))#

    pyplot.legend()

    # save to file
    pyplot.savefig(f'exercise_01.png')


def sgn(value) -> int:
    if value > 0:
        return 1
    elif value == 0:
        return 0
```

```python
    else:
        return -1


def dot_product(a, b) -> int:
    return a[0] * b[0] + a[1] * b[1]


def check_consistency(S, w) -> bool:
    for s in S:
        if sgn(dot_product(s[0], w)) != s[1]:
            return False
    return True


def perceptron(S) -> list:
    w_list = list()
    w = (0, 0)
    while not check_consistency(S, w):
        for s in S:
            if sgn(dot_product(s[0], w)) != s[1]:
                w_old = w
                # w <- w + yx
                w_x = w[0] + s[1] * s[0][0]
                w_y = w[1] + s[1] * s[0][1]
                w = (w_x, w_y)
                w_list.append(w)
                # printing formatted for latex. Just copy and paste
                print(f'Updating vector $w={w_old}$ using $(x,y)={s}$ \\\\')
                print(
                    f'$w={w} \\rightarrow w={w_old} + y={s[1]} * x={s[0]}$
    \\\\ \n\\bigskip \n')
    return w_list


def margin(S, w) -> float:
    distances = [abs(dot_product(w, s[0]))/sqrt(dot_product(w, w)) for s in
    S]
    distances = sorted(distances)
    return distances[0]


def svm(S) -> tuple:
    classifier = LinearSVC(fit_intercept=False) # force heterogenous (
    fit_intercept=False)
    classifier.fit([[s[0][0], s[0][1]] for s in S], [s[1] for s in S])
    return (classifier.coef_[0][0], classifier.coef_[0][1])


if __name__ == '__main__':
    print(f'Perceptron Learning: \\\\ \n\\bigskip \n')
    w_percs = perceptron(S)
    print(f'Margin: ${margin(S,w_percs[-1])}$')

    print(f'SVM Learning: \\\\ \n\\bigskip \n')
    w_svm = svm(S)
    print(f'$w^*: {w_svm}$ \\\\')
    print(f'Margin: ${margin(S, w_svm)}$')

    plot(S, w_percs, w_svm)
```

Code for Exercise 4

```
from math import sqrt
from matplotlib import pyplot

X = [
    (-3, 5),
    (-2, 4),
    (-1, 2),
    (-4, 0),
    (1, -2),
    (2, 0),
    (2, 1),
    (3, 5),
]

Z = [X[0], X[1], X[2]] # a+b
#Z = [X[0], X[7], X[3]] # d

def plot(C, Z, k=3):
    colors = ['red', 'green', 'blue']
    for j in range(k):
        x_values = [cj[0] for cj in C[j]]
        y_values = [cj[1] for cj in C[j]]
        pyplot.scatter(x_values, y_values, c=colors[j])
        pyplot.scatter([Z[j][0]], [Z[j][1]], marker='x', c=colors[j])

    # save to file
    pyplot.savefig(f'exercise_04.png')


def equals_list_of_lists(C_1, C_2) -> bool:
    ll_1 = list([set(l_1) for l_1 in C_1])
    ll_2 = list([set(l_2) for l_2 in C_2])
    for l_1 in list(ll_1):
        for l_2 in list(ll_2):
            if l_1 == l_2:
                ll_1.remove(l_1)
                ll_2.remove(l_2)
                break

    return len(ll_1) == 0 and len(ll_2) == 0


def dot_product(a, b) -> float:
    return a[0] * b[0] + a[1] * b[1]


def k_means(X, Z, k=3) -> tuple:
    C = [[] for j in range(k)]
    C_ = list(C) # copy of C_
    first_iteration = True
    while not equals_list_of_lists(C, C_) or first_iteration:
        first_iteration = False
        C = list(C_)
        C_ = [[] for j in range(k)]

        print(f'Clusters: {C} \\\\')
        print(f'Centres: {Z} \\\\\n')

```

```
        for x in X:
            distances = list()
            for j in range(k):
                # x_i - z_j
                tmp = (x[0] - Z[j][0], x[1] - Z[j][1])
                distances.append((j, sqrt(dot_product(tmp, tmp))))
            # get min j
            min_distance = float('inf')
            min_j = float('inf')
            for d in distances:
                j = d[0]
                distance = d[1]
                if distance < min_distance:
                    min_distance = distance
                    min_j = j
                elif distance == min_distance and j < min_j:
                    min_j = j
            # add x_i to C_j
            C_ = [C_[j] if j != min_j else C_[j] + [x] for j in range(k)]
        # update z_j
        Z = [(sum([x[0] for x in C_[j]])/len(C_[j]), sum([x[1] for x in C_[j
    ]])/len(C_[j])) if len(C_[j]) != 0 else Z[j] for j in range(k)]
    return (C_, Z)


if __name__ == '__main__':
    (C, Z) = k_means(X, Z, k=3)
    print(f'Final Clusters: {C} \\\\')
    print(f'Final Centers: {Z} \\\\\n')
    plot(C, Z, k=3)
```