19	20	21	Σ
5/5	2,5/5	10/10	16,5/10

5 4,5 10 => 19,5/20

Übungsblatt 7

Julia Sobolewski, Lars Kolk und Jannine Salewski

Abgabe: 06.12.2018

TU Dortmund - Fakultät Physik

SMD - Blatt 7

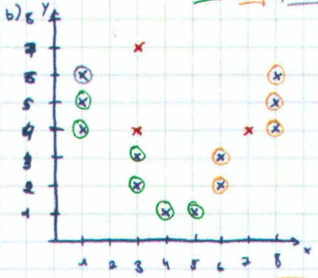
Aufgabe 19

Population: $\begin{pmatrix}1\\4\end{pmatrix}, \begin{pmatrix}1\\5\end{pmatrix}, \begin{pmatrix}1\\6\end{pmatrix}, \begin{pmatrix}3\\3\end{pmatrix}, \begin{pmatrix}3\\2\end{pmatrix}, \begin{pmatrix}4\\1\end{pmatrix}, \begin{pmatrix}5\\1\end{pmatrix}, \begin{pmatrix}6\\2\end{pmatrix}, \begin{pmatrix}6\\3\end{pmatrix}, \begin{pmatrix}8\\4\end{pmatrix}, \begin{pmatrix}8\\5\end{pmatrix}, \begin{pmatrix}8\\6\end{pmatrix}$

a)/ Clusterzentren: $\boxed{\begin{pmatrix}3\\4\end{pmatrix}}, \boxed{\begin{pmatrix}7\\4\end{pmatrix}}, \boxed{\begin{pmatrix}3\\7\end{pmatrix}}$

b)

0. Iteration

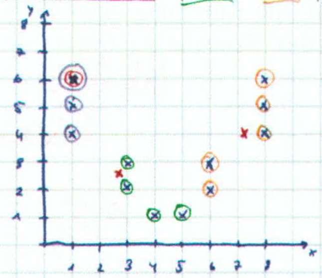


Abstände:	$\begin{pmatrix}5\\1\end{pmatrix}$
$\begin{pmatrix}3\\4\end{pmatrix}$	3,61
$\begin{pmatrix}7\\4\end{pmatrix}$	3,61

← $\begin{pmatrix}5\\1\end{pmatrix}$ wird $\begin{pmatrix}3\\4\end{pmatrix}$ zugeordnet

Neue Clusterzentren: $\boxed{\begin{pmatrix}2,83\\2,67\end{pmatrix}}, \boxed{\begin{pmatrix}7,2\\4\end{pmatrix}}, \boxed{\begin{pmatrix}1\\6\end{pmatrix}}$

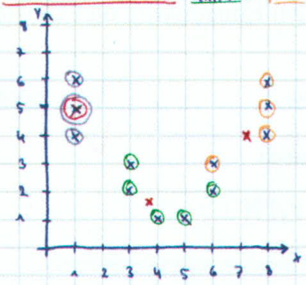
1. Iteration



Abstände	$\begin{pmatrix}1\\4\end{pmatrix}$	$\begin{pmatrix}5\\1\end{pmatrix}$
$\begin{pmatrix}1\\5\end{pmatrix}$	2	/
$\begin{pmatrix}2,83\\2,67\end{pmatrix}$	2,27	2,73
$\begin{pmatrix}7,2\\4\end{pmatrix}$	/	3,72

Neue Clusterzentren: $\boxed{\begin{pmatrix}3,75\\1,75\end{pmatrix}}, \boxed{\begin{pmatrix}7,2\\4\end{pmatrix}}, \boxed{\begin{pmatrix}1\\5\end{pmatrix}}$

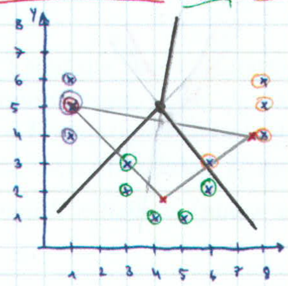
2. Iteration



Abstände	$\begin{pmatrix}6\\2\end{pmatrix}$
$\begin{pmatrix}3,75\\1,75\end{pmatrix}$	2,26
$\begin{pmatrix}7,2\\4\end{pmatrix}$	2,33

Neue Clusterzentren: $\boxed{\begin{pmatrix}4,2\\1,8\end{pmatrix}}, \boxed{\begin{pmatrix}7,5\\4,5\end{pmatrix}}, \boxed{\begin{pmatrix}1\\5\end{pmatrix}}$

3. Iteration



Abstände:	$\begin{pmatrix}6\\3\end{pmatrix}$
$\begin{pmatrix}4,2\\1,8\end{pmatrix}$	2,16
$\begin{pmatrix}7,5\\4,5\end{pmatrix}$	2,12

✓

c) Nach der 3. Iteration werden sich die Clusterzentren nicht weiter ändern → Algorithmus ist konvergiert
Das Ergebnis entspricht nicht ganz den Erwartungen, da man intuitiv Punkt $\begin{pmatrix}6\\3\end{pmatrix}$ dem grünen Cluster zuordnen würde.

✓

5/5

Aufgabe 20

a) Die Lossfunktion ist ein Maßfür die Ungenauigkeit zwischen einem vorhergesagten Wert $\hat{y}$ und einem richtigen Wert y. Eine gebräuchliche Kostenfunktion ist in etwa die Kreuzentropie

$$H(p,q) = -\sum_k p(k) \log q(k) \tag{1}$$

($p(x) \widehat{=}$ wahre Wahrscheinlichkeitsdichte, $q(x) \widehat{=}$ geschätzte Wahrscheinlichkeitsdichte)

Die Gleichung (1) liefert kleinere Werte je näher p und q sind.

Alternativ: mean squared error $\sum_i (E_{est,i} - E_{true,i})^2$

1/1

b) Die Lossfunktion kann minimiert werden, indem man der Richtung des Gradienten in jedem Schritt folgt.

1/1

c) Aktivierungsfunktionen haben die Funktion, die Anregung einer Zelle im Zellkern zu simulieren und somit die Ausgabefunktion des Neurons zu bestimmen. 3 gängige Aktivierungsfunktionen sind:

1. sigmoid-Funktion:$\sigma(x) = \frac{1}{1+\exp -x}$
2. Tangens hyperbolicus :$\tanh(x)$
3. Rectified Linear Unit: $max(0,x)$

Welches Problem lösen sie?

Problem: viele Matrizen -> lineare Abbildung -> hätte auch eine Matrix gereicht

0,5/1

d) Ein Neuron bildet die Basis eines neuronalen Netzwerkes und ist einem Neuron in der Biologie nachempfunden. Mit den zugeweisenden Eingaben x_i und Gewichte W_i wird die Ausgabefunktion

$$\text{net}_j = \sum_{i=0}^{n} x_i w_{ij} \tag{2}$$

definiert. Die Aktivierung dagegen ist definiert durch

$$o_j = \phi(\text{net}_j - \theta_j) \tag{3}$$

($\phi \widehat{=}$ Aktivierungsfunktion, $\theta \widehat{=}$ 'Schellenwert' zur Verschiebung der Gewichtung)

1/1

e) Anwendungsbeispiele sind zum Beispiel:

1. Gesichts/Bilderkennung
2. Texterkennung
3. Spracherkennung

Diese Beispiele eignen sich besonders, da kein (oder wenig) explizites Wissen vorhanden sein muss, um diese zu identifizieren.

1/1

4,5/5

1 Aufgabe 21

1.1 a)-c)

siehe Anhang.

1.2 d)

Siehe Python-Datei.

$$W = \begin{pmatrix} -0,58 & 1,42 \\ 1,83 & -0,56 \end{pmatrix}$$

$$b = \begin{pmatrix} -0,48 \\ 0,61 \end{pmatrix}$$

1.3 e)

Mit

$$\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \tag{4}$$

und $f_1 = f_2$ folgt

$$W_{11}x + W_{12}y + b_1 = W_{21}x + W_{22}y + b_2$$
$$=> y(x) = \frac{b_2 - b_1 + x \cdot (W_{21} - W_{11})}{W_{12} - W_{22}}$$

In Abbildung 1 sind die zwei Populationen dargestellt und die oben berechnete Trennung der beiden Populationen. $f_1 = f_2$ kann angenommen werden, da die Punkte auf der Geraden keiner Population zugeordnet werden können, da sie weder unter noch über der Geraden liegen, deshalb müssen f_1 und f_2 gleich sein.

etwas schwammig formuliert
-> Softmax normiert
-> deshalb auf bei $f_1 = f_2 = 0,5$

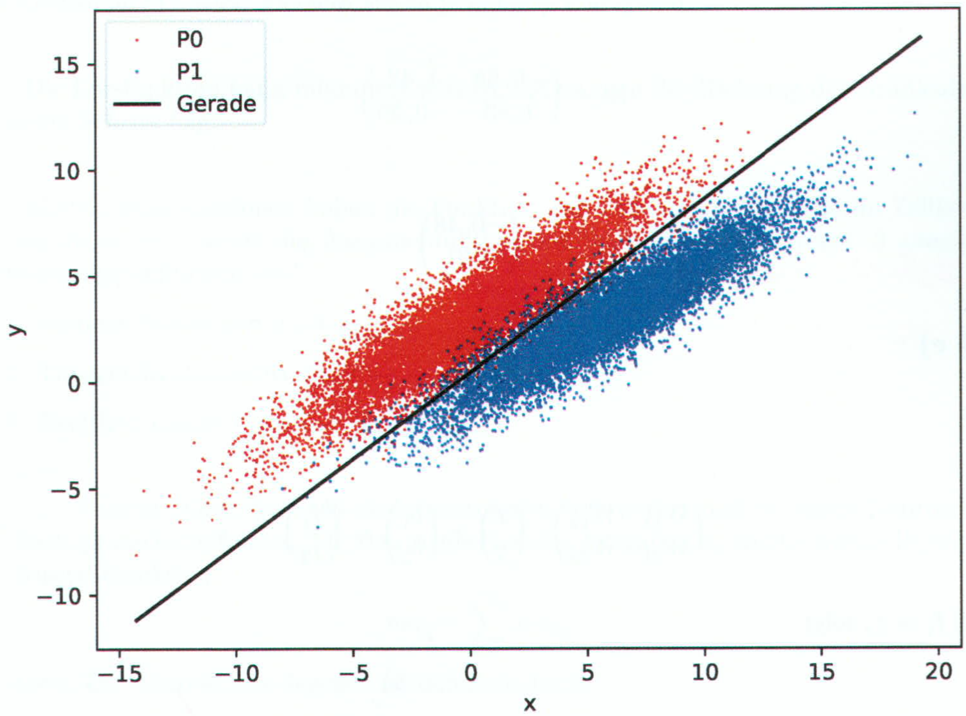


Abbildung 1: Populationen mit der berechneten Trennung.

SMD-Blatt 7 - Aufgabe 21

Lars Kolk
Julia Sobolewski
Jannine Salewski

a) - K Klassen
- m Beispiele x_i mit jeweils M Komponenten

$$\begin{pmatrix} \omega_{1,1} & \cdots & \omega_{1,M} \\ \vdots & \ddots & \vdots \\ \omega_{K,1} & \cdots & \omega_{K,M} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_M \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_K \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_K \end{pmatrix}$$

x_i: Spaltenvektor $x_i \in \mathbb{R}^{M \times 1}$ ✓

C: Kostenfunktion $C \in \mathbb{R}$ ✓

W: Gewichtungsmatrix $W \in \mathbb{R}^{K \times M}$ ✓

b: Bias-Vektor $b \in \mathbb{R}^{K \times 1}$ ✓

$$\nabla_W \hat{C} = \sum_{k=1}^{K} \underbrace{\frac{\partial \hat{C}}{\partial f_{ki}}}_{\in \mathbb{R}} \cdot \underbrace{\frac{\partial f_{ki}}{\partial W}}_{\in \mathbb{R}^{K \times M}} \Rightarrow \nabla_W \hat{C} \in \mathbb{R}^{K \times M}$$ ✓

$$\nabla_{f_i} \hat{C} = \sum_{k=1}^{K} \underbrace{\frac{\partial \hat{C}}{\partial f_{ki}}}_{\mathbb{R}} \underbrace{\frac{\partial f_{ki}}{\partial f_i}}_{\mathbb{R}^{K \times 1}} \Rightarrow \nabla_{f_i} \hat{C} \in \mathbb{R}^{K \times 1}$$ ✓

$\frac{\partial f_{ki}}{\partial W} \in \mathbb{R}^{K \times M}$ ✓

$\frac{\partial f_{ki}}{\partial b} \in \mathbb{R}^{K \times 1}$ ✓

2/2P.

b) $\nabla_{f_a} C(f) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{f_a} \hat{C}(f_i)$

$$\nabla_{f_a} \hat{C}(f_i) = - \nabla_{f_a} \sum_{k=1}^{K} \mathbb{1}(y_i = k) \log\left(\frac{e^{f_{ki}}}{\sum_j e^{f_{ji}}}\right)$$

$$= - \nabla_{f_a} \log\left(\frac{e^{f_{y_i,i}}}{\sum_j e^{f_{ji}}}\right)$$

$$= \underbrace{\frac{-\sum_j e^{f_{ji}}}{e^{f_{y_i i}}}}_{\text{äußere Ableitung}} \cdot \underbrace{\left[\frac{(\nabla_{f_a} e^{f_{y_i,i}}) \cdot \sum_j e^{f_{ji}} - e^{f_{y_i,i}} \cdot (\nabla_{f_a} \sum_j e^{f_{j,i}})}{(\sum_j e^{f_{j,i}})^2}\right]}_{\text{innere Ableitung}}$$ ✓

$$= - \frac{1}{e^{f_{y_i,i}}} \cdot \left[\frac{e^{f_{ai}} \mathbb{1}(y_i = a) \cdot \sum_j e^{f_{j,i}} - e^{f_{y_i i}} e^{f_{ai}}}{\sum_j e^{f_{ji}}}\right]$$

$$= - \frac{e^{f_{ai}} \mathbb{1}(y_i = a)}{e^{f_{y_i i}}} + \frac{e^{f_{y_i,i}} e^{f_{ai}}}{e^{f_{y_i i}} \sum_j e^{f_{ji}}} = \frac{e^{f_{ai}}}{\sum_j e^{f_{ji}}} - \mathbb{1}(y_i = a)$$ ✓

$$\Rightarrow \nabla_{f_a} C(f) = \frac{1}{m} \sum_{i=1}^{m} \left[\frac{e^{f_{ai}}}{\sum_j e^{f_{ji}}} - \mathbb{1}(y_i = a)\right] \quad \text{qed.}$$ ✓

c) $\frac{\partial f_{ki}}{\partial W} = \frac{\partial(W_k x_i + b_k)}{\partial W} = \underbrace{\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & & \\ x_{i,1} & \cdots & x_{i,M} \\ 0 & \cdots & 0 \end{pmatrix}}_{M} \leftarrow$ k-te Zeile

$\frac{\partial f_{ki}}{\partial b} = \frac{\partial(W_k x_i + b_k)}{\partial b} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \rightarrow$ k-te Zeile ✓

- ~~Plot fehlt (hab ihn erstellt, sieht gut aus !)~~
- ~~Warum gilt die Bedingung $f_1 = f_2$?~~

sorry, habe erst nicht gesehen

~~→ 8/10 P.~~

10/10

→ Aufgabe sehr gut bearbeitet :)

Code fuer Blatt07

Kolk, Sobolewski, Salewski

10. Dezember 2018

../B/7/Blatt07_Kolk_Sobolewski_Salewski/Aufgabe21.py

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


np.random.seed(100)

def softmax(f):
    fexp = np.exp(f)
    return fexp/np.sum(fexp, axis=1, keepdims=True)


# Einlesen
P0 = pd.read_hdf('populationen.hdf5', key='P_0')
P1 = pd.read_hdf('populationen.hdf5', key='P_1')

# Zusammenfassen und Labels
Px = np.append(P0.x, P1.x)
Py = np.append(P0.y, P1.y)
Label = np.append(np.zeros(len(P0)), np.ones(len(P1)))
Population = np.vstack([Px, Py, Label])
input = Population.T[:, :2]
```

einmal mischen/shuffle wäre noch sinnvoll → np.random.shuffle()

```python
##Training
rate = 0.5
Epochen = 100

W = np.random.rand(2, 2)
b = np.random.rand(2)
```

Transformation hier nicht nötig dann input = Population[:2,:]

```python
N = len(Px)

for i in range(Epochen):
    f_i = input @ W + b
```

→ np.matmul(W,input)+b.reshape(2,1)

```python
    #Gradienten berechnen
    df_i = softmax(f_i)
    df_i[range(N), [int(d) for d in Population.T[:, 2]]] -=1
    df_i /= N

    dW = input.T @ df_i
    db = np.array([np.sum(df_i[:, 0]), np.sum(df_i[:, 1])])

    W -= rate * dW
    b -= rate * db

print(W)
print(b)

def Gerade(X, W, b):
    return (b[1]-b[0]+X*(W[0][1]-W[1][1]))/(W[1][0]-W[0][0])
```

1 0 0 0 0 1 1 1 stimmt nicht mit eurer Rechnung überein

```python
xPlot = np.linspace(np.min(Px), np.max(Px), 10000)
plt.plot(P0.x, P0.y, 'r.', alpha=0.7, markersize=0.7, label='P0')
plt.plot(P1.x, P1.y, 'b.', alpha=0.7, markersize=0.7, label='P1')
plt.plot(xPlot, Gerade(xPlot, W, b), 'black', label='Gerade')
plt.legend(loc='best')
plt.xlabel('x')
```

```
plt.ylabel('y')
plt.tight_layout()
plt.savefig('Plot.pdf')
```

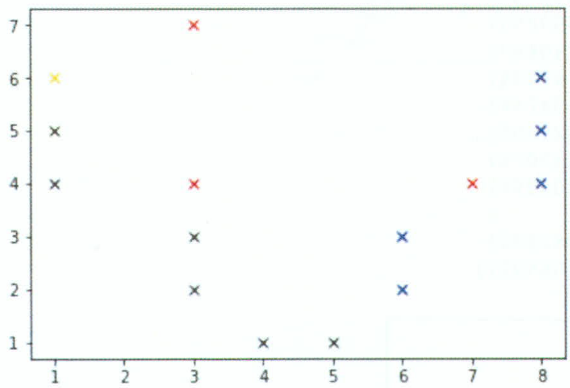
LLD in die pdf ???

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
```

In [17]:
```
def pltk(p, center):
    distance = cdist(p, center)
    print(distance)
    labels = []

    for i in range(len(p)):
        if np.argsort(distance)[i,0] == 0:
            labels.append('black')
        if np.argsort(distance)[i,0] == 1:
            labels.append('blue')
        if np.argsort(distance)[i,0] == 2:
            labels.append('gold')
        plt.plot(p[i,0], p[i,1], color=labels[i], marker='x', linestyle='None')

    plt.plot(center[:,0], center[:,1], 'rx')
```

In [18]:
```
p = np.array([(1,4),(1,5),(1,6),(3,3),(3,2),(4,1),(5,1),(6,2),(6,3),(8,4),(8,5),
(8,6)])
center = np.array([(3,4),(7,4),(3,7)])

pltk(p, center)
```

```
[[2.         6.         3.60555128]
 [2.23606798 6.08276253 2.82842712]
 [2.82842712 6.32455532 2.23606798]
 [1.         4.12310563 4.        ]
 [2.         4.47213595 5.        ]
 [3.16227766 4.24264069 6.08276253]
 [3.60555128 3.60555128 6.32455532]
 [3.60555128 2.23606798 5.83095189]
 [3.16227766 1.41421356 5.        ]
 [5.         1.         5.83095189]
 [5.09901951 1.41421356 5.38516481]
 [5.38516481 2.23606798 5.09901951]]
```

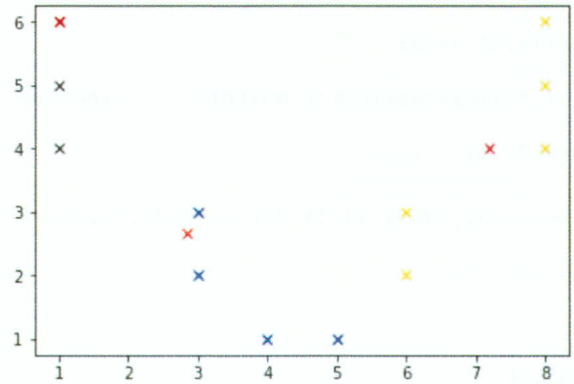


In [19]:
```
center2 = np.array([(1,6),((sum(p[0:7])-p[2])/6), (sum(p[7:])/5)])
center2
```

Out[19]:
```
array([[1.        , 6.        ],
       [2.83333333, 2.66666667],
       [7.2       , 4.        ]])
```

In [20]:
```
pltk(p,center2)
```

```
[[2.         2.26691175 6.2       ]
 [1.         2.96741564 6.28012739]
 [0.         3.8042374  6.51459899]
 [3.60555128 0.372678   4.31740663]
 [4.47213595 0.68718427 4.65188134]
 [5.83095189 2.03442594 4.38634244]
 [6.40312424 2.73353658 3.72021505]
 [6.40312424 3.23608131 2.33238076]
 [5.83095189 3.1841622  1.56204994]
 [7.28010989 5.33593686 0.8       ]
 [7.07106781 5.66911712 1.28062485]
 [7.         6.14862225 2.15406592]]
```

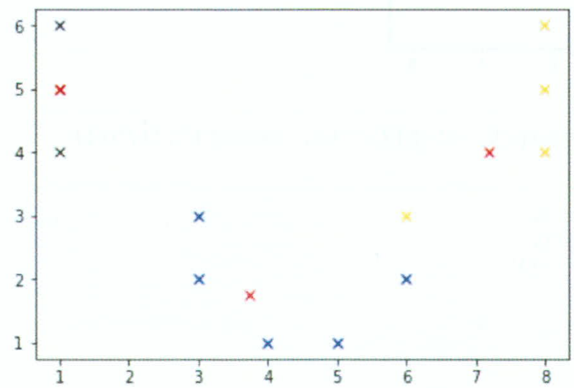


In [21]:
```
center3 = np.array([(sum(p[0:3])/3), (sum(p[3:7])/4), (sum(p[7:])/5)])
center3
```

Out[21]:
```
array([[1.  , 5.  ],
       [3.75, 1.75],
       [7.2 , 4.  ]])
```

In [22]:
```
pltk(p,center3)
```

```
[[1.         3.5531676  6.2       ]
 [0.         4.25734659 6.28012739]
 [1.         5.06211418 6.51459899]
 [2.82842712 1.45773797 4.31740663]
 [3.60555128 0.79056942 4.65188134]
 [5.         0.79056942 4.38634244]
 [5.65685425 1.45773797 3.72021505]
 [5.83095189 2.26384628 2.33238076]
 [5.38516481 2.57390754 1.56204994]
 [7.07106781 4.80884602 0.8       ]
 [7.         5.35023364 1.28062485]
 [7.07106781 6.01040764 2.15406592]]
```

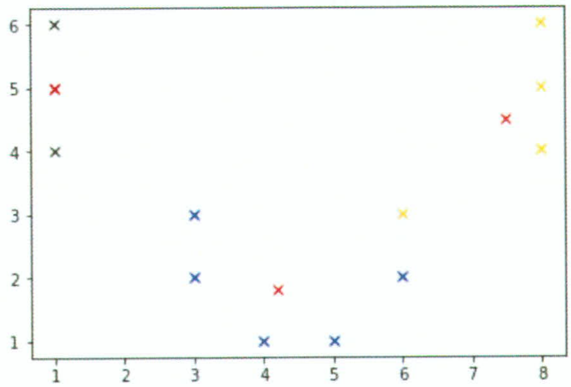


In [26]:
```
center4 = np.array([(sum(p[0:3])/3), (sum(p[3:8])/5), (sum(p[8:])/4)])
center4
```

Out[26]:
```
array([[1. , 5. ],
       [4.2, 1.8],
       [7.5, 4.5]])
```

In [27]:
```
pltk(p,center4)
```

```
[[1.         3.88329757 6.51920241]
 [0.         4.5254834  6.51920241]
 [1.         5.28015151 6.67083203]
 [2.82842712 1.69705627 4.74341649]
 [3.60555128 1.21655251 5.14781507]
 [5.         0.82462113 4.94974747]
 [5.65685425 1.13137085 4.30116263]
 [5.83095189 1.81107703 2.91547595]
 [5.38516481 2.16333077 2.12132034]
 [7.07106781 4.39089968 0.70710678]
 [7.         4.96789694 0.70710678]
 [7.07106781 5.6639209  1.58113883]]
```



In [25]:
```
(1+3+3+5+4)/5
```

Out[25]:
```
3.2
```

In [ ]: