1)

$$\cos\theta = \frac{A.B}{\|A\| \|B\|}$$

Geometrically speaking,

+1, when both vectors have the same direction

0, when the vectors are perpendicular to each other

-1, when the vectors are in exactly opposite directions

case of +1, $A = [1, 1, 1, 1]$ $B = [2, 2, 2, 2]$

$$\cos\theta = \frac{A.B}{\|A\| \|B\|} = \frac{2+2+2+2}{\sqrt{4\times1^2}\ \sqrt{4\times2^2}} = 1$$

Hence, angle between vectors $(\theta) = 0^\circ$

case of 0, $A = [1, 1, 1, 1]$ $B = [1, 1, -1, -1]$

$$\cos\theta = \frac{1+1-1-1}{\sqrt{4\times1^2}\ \sqrt{4\times1^2}} = 0$$

Hence, perpendicular vectors.

case of -1, $A = [1, 1, 1, 1]$ $B = [-2, -2, -2, -2]$

$$\cos\theta = \frac{-2-2-2-2}{\sqrt{4\times1^2}\ \sqrt{4\times(-2)^2}} = -1$$

Hence, vectors are at 180°, i.e., in opposite directions.

2)

2) $D = \{x_i, i \in 0, 1, 2, ---, n\}$

$W^T x_i$

$$\cos\theta = \frac{x_i \, \alpha(x_i)}{\|x_i\| \, \|\alpha x_i\|}$$

$$= \frac{\alpha}{|\alpha|} \frac{x_i \cdot x_i}{\|x_i\| \|x_i\|}$$

→ Positive samples

$W^T x_i > 0$

Multiplying by α,

$W^T(\alpha x_i) > 0$

$\cos\theta$ is either 1 or -1.

Since, $\alpha \in R^+$, $\cos\theta = 1$.

As we can see, the classification is not influenced

Similarly, negative samples will also remain unchanged only for positive scalars. ~~All negative scalars give 0 accuracy~~

Misinterpreted (Redone at End of Answer 2,)

→ $W^T \cdot \begin{bmatrix} \alpha_1 x_{i1} \\ \alpha_2 x_{i2} \\ \vdots \end{bmatrix}$

Independent random scalars will change the accuracy of the classifier.

This happens because these scalars influence not only the magnitude of the vector but also its direction.

→ Orthogonal matrix.

The rows form an orthonormal basis.

When we use this matrix A for transformation, there is a change in the original basis.

The transformed vectors x_i will have the same magnitude but have different direction,

i.e., the angle 'θ' between x_i & Ax_i will not be $0°$.

Hence, the accuracy will change for the given classifier.

The accuracy will obviously remain unchanged if the same tranformation is applied to the ~~vector~~ vector W^T.

→ Rank deficient matrix 'A'.

This implies that row(s)/column(s) are linearly dependent.

When such a matrix is ~~no~~ used to transform a vector, it will lead to dimensionality reduction, which means loss of information.

The transformed vector will have ~~&~~ value(s) which are scalar multiples of each other.

This will enable us to represent the ~~vector~~ vector components as a linear combination of each other.

For example

$$\begin{bmatrix} a & b & c \\ 2a & 2b & 2c \\ x & y & z \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \Rightarrow \begin{bmatrix} ax_1 + bx_2 + cx_3 \\ 2(ax_1 + bx_2 + cx_3) \\ \vdots \\ \vdots \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} k \\ 2k \\ \vdots \\ \vdots \end{bmatrix}$$

$$\begin{bmatrix} W_{11} & W_{12} & \cdots \\ W_{21} & W_{22} & \\ \vdots & & \\ \vdots & & \end{bmatrix} \begin{bmatrix} k \\ 2k \\ \vdots \\ \vdots \end{bmatrix} \Rightarrow \begin{bmatrix} (W_{11} + 2W_{12})k + \cdots \\ (W_{21} + 2W_{22})k + \cdots \\ \\ \end{bmatrix}$$

The weights can be merged into a single vector.

The accuracy will change as not only the direction of vectors change, but their information is also lost.

The loss of information means that it would not be possible to linearly transform the classifier to gain similar accuracy.

→ The accuracy will remain unchanged when,

1) The directions of vectors (x_i) do not change with respect to the classifier which is linear.

In other words, it remains unchanged if the cosine similarity between x_i & Ax_i (transformed) is ~~0~~ 1, i.e., $\theta = 0°$. OR

If the classifier & x_i both change by the same degree θ.

→ Redoing 2nd part -

$w^T x_i > 0$ $\quad \alpha \in \mathbb{R}$

Similar to the first part of the question, we would arrive at $\cos\theta = \frac{\alpha_i}{|\alpha_i|}$ for each vector.

Here ~~also~~ the magnitude of the scalar does not play a role, only the sign does.

Hence, accuracy is same for positive scalars & inverted accuracy for negative scalars (90% → 10%)

3) Code ::

```
import sys
import numpy as np
import csv
import matplotlib.pyplot as plt
import random

def parse_w(given_str):
parsed_arr = [float(x) for x in given_str[1:-1].split(',')]
n = len(parsed_arr)
if n > 3 or n < 3:
return None
return parsed_arr

def plot_single(class_A, class_B, np_w, check_np_w):
np_class_A = np.asarray(class_A)
np_class_B = np.asarray(class_B)
sizes=[5,4,3,2,1]
colors=["red","blue","green","black","brown"]
A_x = np_class_A[:,0]
A_y = np_class_A[:,1]
B_x = np_class_B[:,0]
B_y = np_class_B[:,1]
plt.plot(A_x, A_y, "ob", markersize=1, color="blue")
plt.plot(B_x, B_y, "ob", markersize=1, color="pink")

color_index=0
x=[]
y=[]
for _ in range(0,10):
x.append(random.randint(-2,2))
for iter_x in x:
y.append(((-check_np_w[0]*iter_x) - check_np_w[2])/check_np_w[1])
plt.plot(x,y,markersize=sizes[color_index], color=colors[color_index])
color_index+=1
plt.plot([0,-2,2],[0,1,-1],color="gray")
plt.show()
def plot_all(class_A, class_B, np_w, check_np_w):
np_class_A = np.asarray(class_A)
np_class_B = np.asarray(class_B)
sizes=[5,4,3,2,1]
colors=["red","blue","green","black","brown"]
A_x = np_class_A[:,0]
A_y = np_class_A[:,1]
```

```
B_x = np_class_B[:,0]
B_y = np_class_B[:,1]
plt.plot(A_x, A_y, "ob", markersize=2, color="blue")
plt.plot(B_x, B_y, "ob", markersize=2, color="pink")

color_index=0
for plane in check_np_w:
x=[]
y=[]
print(plane, " color: ", colors[color_index])
for _ in range(0,10):
x.append(random.randint(-2,2))
for iter_x in x:
y.append(((-plane[0]*iter_x) - plane[2])/plane[1])
plt.plot(x,y,markersize=sizes[color_index], color=colors[color_index])
color_index+=1
plt.plot([0,-2,2],[0,1,-1],color="gray")
plt.show()

def generate_classes(w):
class_A = []
class_B = []

while len(class_A)<50:
rand_x = np.random.rand(2)
x = np.append(rand_x, 1)
if np.dot(w,x)>0:
class_A.append(x)
while len(class_B)<50:
rand_x = np.random.rand(2)
x = np.append(-rand_x, 1)
if np.dot(w,x)<0:
class_B.append(x)

return class_A, class_B

def get_accuracy(class_A, class_B, w):
correct_count = 0
for x in class_A:
if np.dot(w,x)>0:
correct_count+=1
for x in class_B:
if np.dot(w,x)<0:
correct_count+=1
return correct_count/(len(class_A)+len(class_B))
```

```
if __name__=="__main__":
w = [0.5,1,0]
np_w = np.asarray(w)
class_A, class_B = generate_classes(np_w)

check_w = []
check_np_w = []
check_w.append([1,1,0])
# check_w.append([-1,-1,0])
# check_w.append([0,0.5,0])
check_w.append([1,-1,5])
check_w.append([1,1,0.3])
check_np_w.append(np.asarray(check_w[0]))
check_np_w.append(np.asarray(check_w[1]))
check_np_w.append(np.asarray(check_w[2]))
# check_np_w.append(np.asarray(check_w[3]))
# check_np_w.append(np.asarray(check_w[4]))

if w is None:
print("Error: Please give an array of length 3")
exit(0)
print("w is: ", np_w)
for weights in check_np_w:
print("For ",weights, " accuracy is: ", get_accuracy(class_A, class_B, weights))
x = 0
plot_all(class_A, class_B, np_w, check_np_w)
# plot_single(class_A, class_B, np_w, check_np_w[x])
```

3)

iii) Vectors a:[1, 1, 0] & b:[-1, -1, 0] have accuracies which sum to 1.

This is expected as they are exactly opposite vectors

vector c: [0, 0.5, 0]

This causes dimensions to be ignored.

Since the generated class_A points in ~~the~~ my experiment had positive 'y' values & negative 'y' values for class_B, it was expected.

ii) The graphs were plotted for the 2-D plane of points, in the plane z=1.

The classifier for each 'w' is the intersection of the plane represented by 'w' with plane z=1.