**Exercise 1.2**

Suppose that we use a perceptron to detect spam messages. Let's say that each email message is represented by the frequency of occurrence of keywords, and the output is $+1$ if the message is considered spam.

(a) Can you think of some keywords that will end up with a large positive weight in the perceptron?

(b) How about keywords that will get a negative weight?

(c) What parameter in the perceptron directly affects how many borderline messages end up being classified as spam?

a) si palabras como promoción, Sólo por hoy, amenaza, Exclusivo, Últimas horas, etc.

b) información, caso, adjunto, Confirmación, check in, registro, exitoso.

c) principalmente son los usuarios quienes envian la información, si los correos muchas personas agregan usuarios sospechosos esos van a ser clasificados.

**Exercise 1.3**

The weight update rule in (1.3) has the nice interpretation that it moves in the direction of classifying $\mathbf{x}(t)$ correctly.

(a) Show that $y(t)\mathbf{w}^{\mathrm{T}}(t)\mathbf{x}(t) < 0$. *[Hint: $\mathbf{x}(t)$ is misclassified by $\mathbf{w}(t)$.]*

(b) Show that $y(t)\mathbf{w}^{\mathrm{T}}(t+1)\mathbf{x}(t) > y(t)\mathbf{w}^{\mathrm{T}}(t)\mathbf{x}(t)$. *[Hint: Use (1.3).]*

(c) As far as classifying $\mathbf{x}(t)$ is concerned, argue that the move from $\mathbf{w}(t)$ to $\mathbf{w}(t+1)$ is a move 'in the right direction'.

por definicion de la regla de actualización de pesos:

$w(t+1) = w(t) + y(t) * x(t)$

Sea $x(t)$ tal que este clasificado incorrectamente $y(t) * w(t) * x(t) \leq 0$ lo cual implica que el modelo predijo mal para este valor.

sea $w(t+1)$ y $x(t)$ su producto acalar:

- $w(t+1) * x(t)$
- $(w(t) + y(t) * x(t)) * x(t)$
- $w(t) * x(t) + y(t) * x(t) * x(t)$
- $w(t+1) * x(t) = w(t) * x(t) + y(t) * ||x(t)||^2$ esto por $(t) * x(t) \geq 0$

$y(t) * ||x(t)||^2$ es un factor de escala positivo de $x(t)$ esto tiene una afectación positiva en los pesos

En el caso $x(t)$ está clasificado incorrectamente por $w(t)$ entonces $y(t) * w(t) * x(t) \leq 0$, para este caso $y(t) * ||x(t)||^2$ es positivo lo cual implica que el vector de pesos se dirige hacia $x(t)$ logrando corregir la clasificación correcta.

b)

$*= y(t)w^T(t+1)x(t)$

$*= y(t)(w(t) + y(t)x(t))^T x(t)$

$*= y(t)\left(w^T(t) + y(t)x^T(t)\right)x(t)$

$*= y(t)w^T(t)x(t) + y(t)y(t)x^T(t)x(t) > y(t)w^T(t)x(t)$

podemos observar que $y(t)w^T(t)x(t)$ es creciente en cada actualización.

Si $y(t)$ es positivo pero $w^T(t)x(t)$ es negativo, podemos movernos desde $w^T(t)x(t)$ hacia un valor positivo.

Sin embargo, si $y(t)$ es negativo pero $w^T(t)x(t)$ es positivo, $y(t)w^T(t)x(t)$ sigue siendo creciente mientras que $w^T(t)x(t)$ es decreciente.

Por lo tanto, al movernos desde $w(t)$ hacia $w(t+1)$, estamos avanzando en la dirección correcta siempre y cuando nos centremos en la clasificación de $x(t)$.

**Exercise 1.10**

Here is an experiment that illustrates the difference between a single bin and multiple bins. Run a computer simulation for flipping $1,000$ fair coins. Flip each coin independently $10$ times. Let's focus on 3 coins as follows: $c_1$ is the first coin flipped; $c_{\text{rand}}$ is a coin you choose at random; $c_{\min}$ is the coin that had the minimum frequency of heads (pick the earlier one in case of a tie). Let $\nu_1$, $\nu_{\text{rand}}$ and $\nu_{\min}$ be the fraction of heads you obtain for the respective three coins.

(a) What is $\mu$ for the three coins selected?

(b) Repeat this entire experiment a large number of times (e.g., $100,000$ runs of the entire experiment) to get several instances of $\nu_1$, $\nu_{\text{rand}}$ and $\nu_{\min}$ and plot the histograms of the distributions of $\nu_1$, $\nu_{\text{rand}}$ and $\nu_{\min}$. Notice that which coins end up being $c_{\text{rand}}$ and $c_{\min}$ may differ from one run to another.

(c) Using (b), plot estimates for $\mathbb{P}[|\nu-\mu|>\epsilon]$ as a function of $\epsilon$, together with the Hoeffding bound $2e^{-2\epsilon^2 N}$ (on the same graph).

(d) Which coins obey the Hoeffding bound, and which ones do not? Explain why.

(e) Relate part (d) to the multiple bins in Figure 1.10.

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import math
from sklearn.preprocessing import normalize
from functools import partial
```

```
#Se Lanzan las 1000 monedas 10 veces
flips = np.random.binomial(1, 0.5, (1000, 10))

# Obtener los resultados de cada moneda
coin1 = flips[0]
coin2 = flips[np.random.choice(1000)]
coin3 = flips[np.argmin(np.sum(flips, axis=1))]

# Calcular las fracciones de caras para cada moneda
f1 = np.mean(coin1)
f2 = np.mean(coin2)
f3 = np.mean(coin3)
```

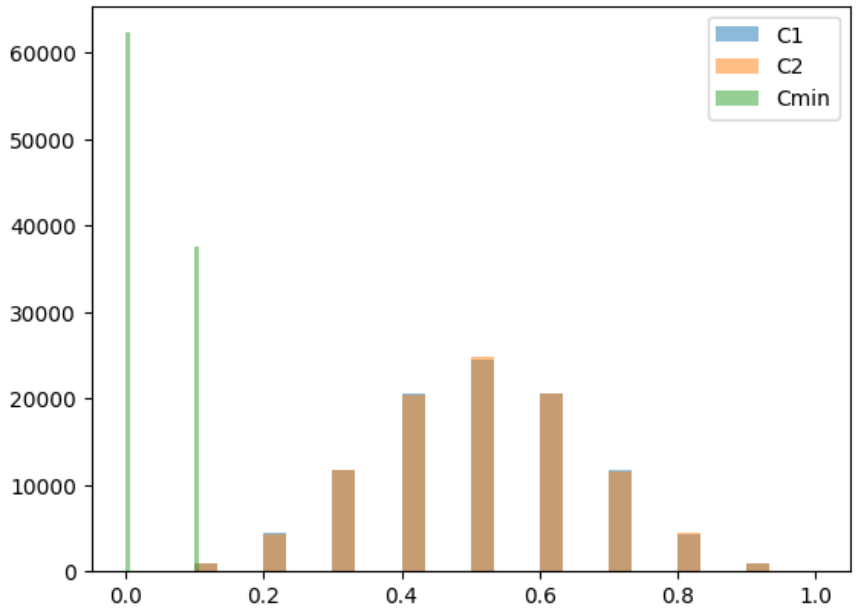
```
f1, f2, f3
```

```
(0.4, 0.3, 0.0)
```

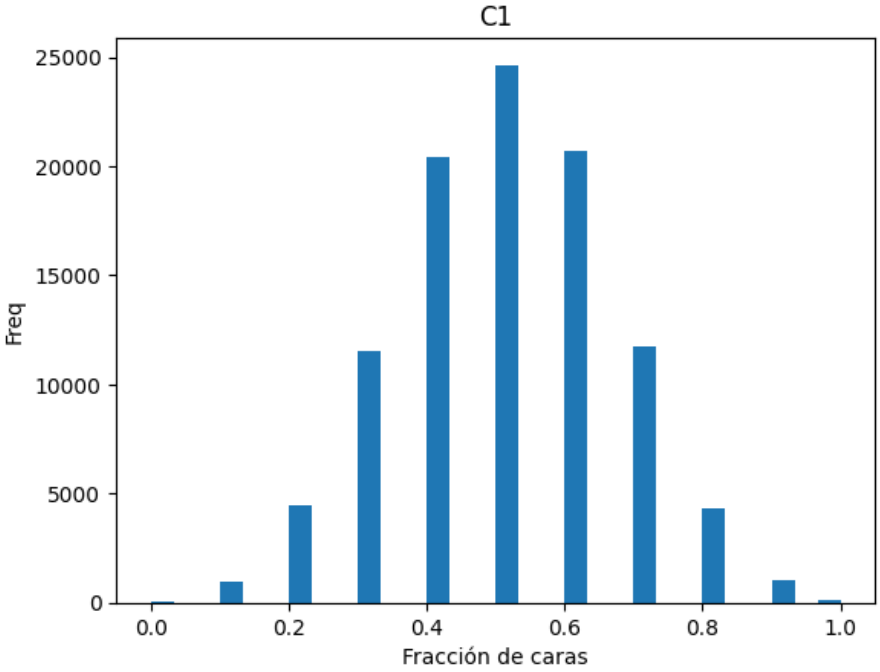
b) repetir el modelo 1000 veces

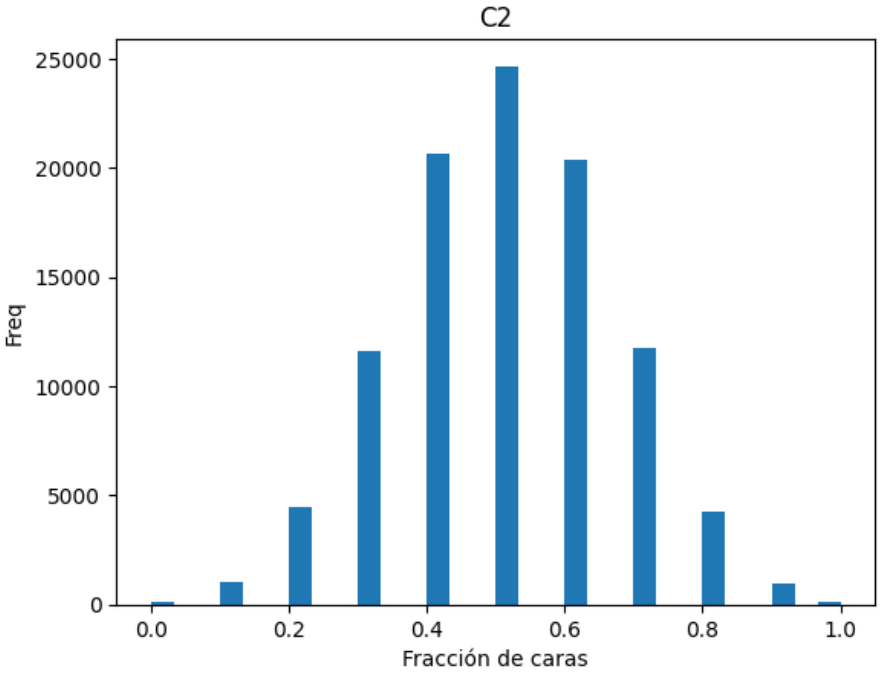
```
# Repetir el experimento 100000 veces
n_trials = 100000
results = np.zeros((n_trials, 3)) #Se crea un array de numpy llamado results que tendrá n_trials filas y 3 columnas. Esta m
for i in range(n_trials):#e inicia un bucle for que se ejecutará n_trials veces.

    flips = np.random.binomial(1, 0.5, (1000, 10))#se generan aleatoriamente 1000 series de 10 lanzamientos de monedas
    coin1 = flips[0] #e define la variable coin1 como la primera serie de lanzamientos.
    coin2 = flips[np.random.choice(1000)] #se elige aleatoriamente una serie de lanzamientos de moneda y se define como coi
    coin3 = flips[np.argmin(np.sum(flips, axis=1))] #se busca la serie de lanzamientos de moneda con la menor cantidad de c
    results[i] = [np.mean(coin1), np.mean(coin2), np.mean(coin3)] #se calcula la fracción de caras para cada una de las tre
```

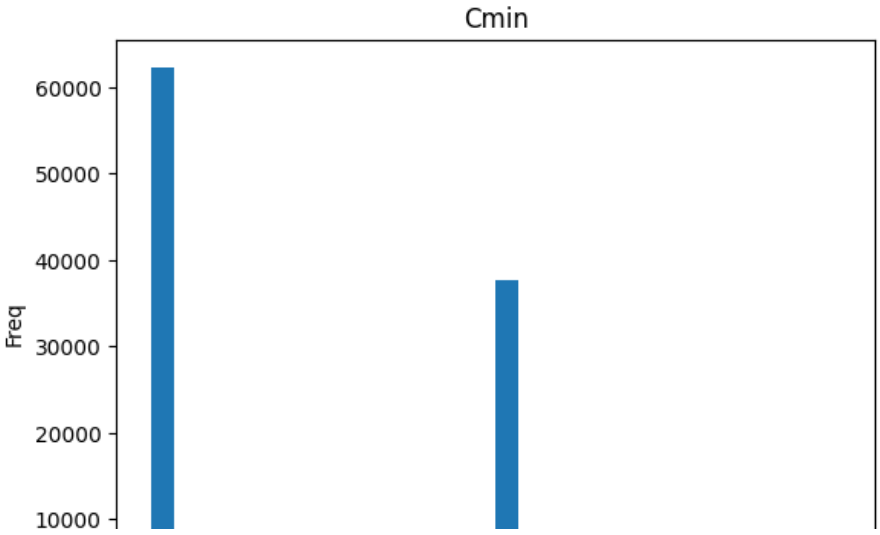
```
# Graficar los histogramas de las distribuciones
plt.hist(results[:, 0], bins=30, alpha=0.5, label='C1')
plt.hist(results[:, 1], bins=30, alpha=0.5, label='C2')
plt.hist(results[:, 2], bins=30, alpha=0.5, label='Cmin')
plt.legend(loc='upper right')
plt.show()
```




```
n_trials = 100000
results = np.zeros((n_trials, 3))
for i in range(n_trials):
    flips = np.random.binomial(1, 0.5, (1000, 10))
    coin1 = flips[0]
    coin2 = flips[np.random.choice(1000)]
    coin3 = flips[np.argmin(np.sum(flips, axis=1))]
    results[i] = [np.mean(coin1), np.mean(coin2), np.mean(coin3)]

# Graficar el histograma de El
plt.hist(results[:, 0], bins=30)
plt.title('C1')
plt.xlabel('Fracción de caras')
plt.ylabel('Freq')
plt.show()

# Graficar el histograma de Mrand
plt.hist(results[:, 1], bins=30)
plt.title('C2')
plt.xlabel('Fracción de caras')
plt.ylabel('Freq')
plt.show()

# Graficar el histograma de Cmin
plt.hist(results[:, 2], bins=30)
plt.title('Cmin')
plt.xlabel('Fracción de caras')
plt.ylabel('Freq')
plt.show()
```


C1
25000
20000
15000
10000
5000
0
Freq
0.0
0.2
0.4
0.6
0.8
1.0
Fracción de caras


C2
25000
20000
15000
10000
5000
0
Freq
0.0
0.2
0.4
0.6
0.8
1.0
Fracción de caras


Cmin
60000
50000
40000
30000
20000
10000
Freq

**Exercise 1.11**

We are given a data set $\mathcal{D}$ of 25 training examples from an unknown target function $f: \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \{-1, +1\}$. To learn $f$, we use a simple hypothesis set $\mathcal{H} = \{h_1, h_2\}$ where $h_1$ is the constant $+1$ function and $h_2$ is the constant $-1$.

We consider two learning algorithms, S (smart) and C (crazy). S chooses the hypothesis that agrees the most with $\mathcal{D}$ and C chooses the other hypothesis deliberately. Let us see how these algorithms perform out of sample from the deterministic and probabilistic points of view. Assume in the probabilistic view that there is a probability distribution on $\mathcal{X}$, and let $\mathbb{P}[f(\mathbf{x}) = +1] = p$.

(a) Can S produce a hypothesis that is *guaranteed* to perform better than random on any point outside $\mathcal{D}$?

(b) Assume for the rest of the exercise that all the examples in $\mathcal{D}$ have $y_n = +1$. Is it *possible* that the hypothesis that C produces turns out to be better than the hypothesis that S produces?

(c) If $p = 0.9$, what is the probability that S will produce a better hypothesis than C?

(d) Is there any value of $p$ for which it is more likely than not that C will produce a better hypothesis than S?

a) H consta de dos hipótesis: $h_1$, que siempre predice +1, y $h_2$, que siempre predice -1. El algoritmo de aprendizaje S elige la hipótesis que coincide más con los datos de entrenamiento D.

Dado que S selecciona la hipótesis que coincide con la mayoría de los ejemplos en D, siempre elegirá $h_1$ si p ≥ 0,5 (la mayoría de los ejemplos tienen la etiqueta +1) y $h_2$ si p < 0,5 (la mayoría de los ejemplos tienen la etiqueta -1). Por lo tanto, S nunca superará al azar fuera de D, ya que simplemente selecciona la hipótesis que coincide con la mayoría de los ejemplos de entrenamiento.

b) S como C eligen la misma hipótesis $h_1$ en este escenario, no hay diferencia entre las hipótesis producidas por S y C. Por lo tanto, no es posible que la hipótesis producida por C sea mejor que la hipótesis producida por S, ya que son idénticas.

c) Si p = 0.9, significa que la función objetivo f asigna la etiqueta +1 al 90%. En este caso, S elegirá la hipótesis $h_1$ (que siempre predice +1). C elige deliberadamente la hipótesis $h_2$ (que siempre predice -1). Dado que $h_2$ predice lo opuesto a las etiquetas verdaderas en D, tendrá una tasa de error del 0.9 o 90% en los ejemplos de entrenamiento. En este caso, la probabilidad de que f asigne la etiqueta -1 a un punto es (1 - p) = (1 - 0.9) = 0.1. Por lo tanto, la probabilidad de que S produzca una hipótesis mejor que C es del 0.1 o 10%

d) Cuándo es más probable que C produzca una hipótesis mejor que S, necesitamos encontrar el valor de p para el cual la tasa de error de C es menor que la de S. En otras palabras, queremos:

0.5 - p < p

Simplificando la desigualdad, obtenemos:

0.5 < 2p

Dividiendo ambos lados por 2, obtenemos:

0,25 < p

---

**Exercise 1.12**

A friend comes to you with a learning problem. She says the target function $f$ is *completely* unknown, but she has $4,000$ data points. She is willing to pay you to solve her problem and produce for her a $g$ which approximates $f$. What is the best that you can promise her among the following:

(a) After learning you will provide her with a $g$ that you will guarantee approximates $f$ well out of sample.

(b) After learning you will provide her with a $g$, and with high probability the $g$ which you produce will approximate $f$ well out of sample.

(c) One of two things will happen.

(i) You will produce a hypothesis $g$;

(ii) You will declare that you failed.

If you do return a hypothesis $g$, then with high probability the $g$ which you produce will approximate $f$ well out of sample.

lo mejor que le puedo ofrecer es la opción c, en la cual si la función construida bajo la hipotesis aplica la herramineta podra generalizar y podra ser la f, pero si no no era en cierta medida generalizar.

**Problem 1.2** Consider the perceptron in two dimensions: $h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ where $\mathbf{w} = [w_0, w_1, w_2]^{\mathrm{T}}$ and $\mathbf{x} = [1, x_1, x_2]^{\mathrm{T}}$. Technically, $\mathbf{x}$ has three coordinates, but we call this perceptron two-dimensional because the first coordinate is fixed at $1$.

(a) Show that the regions on the plane where $h(\mathbf{x}) = +1$ and $h(\mathbf{x}) = -1$ are separated by a line. If we express this line by the equation $x_2 = ax_1 + b$, what are the slope $a$ and intercept $b$ in terms of $w_0, w_1, w_2$?

sea $h(x) = sing(\bar{w}^T \bar{x})$, donde $\bar{w} \epsilon R^3$

tambien sea $x_2 = ax_2 + b$

a es la pendiente: para $a = 0$, la linea es horizontal para $a > 0$ esta creciente, b es punto de corte.

Sea el eje $x_2$ en $x_2 = b$, para la linea que pasa por el origen si y solo si b=0.

entonces:

1. $x_2 = ax_2 + b$
2. $x_2 - ax_2 - b = 0$
3. $\begin{vmatrix} 1 & -a & b \end{vmatrix} \begin{vmatrix} x_2 \\ x_1 \\ 1 \end{vmatrix} = 0$

para el caso de $R^3$ $\bar{w}^T \bar{x}$ es el hiperplano

sea para el caso $h(x) = 0$ generalizado para $\begin{vmatrix} w_0 & w_1 & w_2 \end{vmatrix}$

comunmente $w_0$ hace referencia al punto de corte del hiperplano:

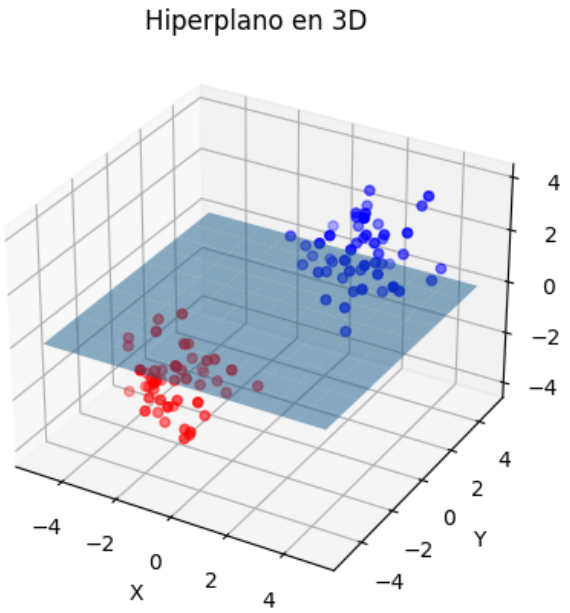
1. $\begin{vmatrix} w_0 & w_1 & w_2 \end{vmatrix} \begin{vmatrix} 1 \\ x_1 \\ x_2 \end{vmatrix} = 0$
2. $w_0 + w_1 x_1 + w_1 x_1 = 0$

tambien este hiperplano separa dos grupos $h(x) = -1$ & $h(x) = 1$

respectivamente, sea $h(x) = sign(y(x)) = 1$ para $x \epsilon H_{+1}$, de la misma forma $h(x) = sign(y(x)) = -1$ para $x \epsilon H_{-1}$

a continuación se presenta un plano que muestra la idea inicial en python

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Generar puntos aleatorios
np.random.seed(0)
positive_points = np.random.randn(50, 3) + 2
negative_points = np.random.randn(50, 3) - 2

# Crear la figura 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Graficar los puntos positivos
ax.scatter(positive_points[:, 0], positive_points[:, 1], positive_points[:, 2], c='b', marker='o')

# Graficar los puntos negativos
ax.scatter(negative_points[:, 0], negative_points[:, 1], negative_points[:, 2], c='r', marker='o')

# Crear el hiperplano
xx, yy = np.meshgrid(range(-5, 6), range(-5, 6))
z = (0) * (1 * xx + -1 * yy) / 1

# Graficar el hiperplano
ax.plot_surface(xx, yy, z, alpha=0.5)

# Configurar el aspecto del gráfico
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Hiperplano en 3D')

# Mostrar el gráfico
plt.show()
```

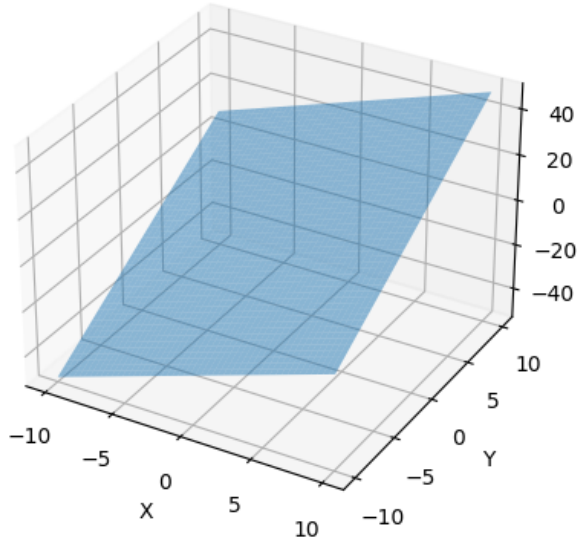



(b) Draw a picture for the cases $\mathbf{w} = [1, 2, 3]^{\mathrm{T}}$ and $\mathbf{w} = -[1, 2, 3]^{\mathrm{T}}$.

In more than two dimensions, the $+1$ and $-1$ regions are separated by a *hyperplane*, the generalization of a line.

para ambos casos el hiperplano es $2x_1 + 3x_2 + 1 = 0$

```
# Crear los datos para el plano
x = np.linspace(-10, 10, 50)
y = np.linspace(-10, 10, 50)
X, Y = np.meshgrid(x, y)
Z = (2*X + 3*Y - 1)  # Aquí dividimos por 0 para obtener la ecuación del hiperplano

# Crear la figura y el eje tridimensional
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Graficar el hiperplano
ax.plot_surface(X, Y, Z, alpha=0.5)

# Configurar los ejes
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

# Mostrar el gráfico
plt.show()
```




**Problem 1.3** Prove that the PLA eventually converges to a linear separator for separable data. The following steps will guide you through the proof. Let $\mathbf{w}^*$ be an optimal set of weights (one which separates the data). The essential idea in this proof is to show that the PLA weights $\mathbf{w}(t)$ get "more aligned" with $\mathbf{w}^*$ with every iteration. For simplicity, assume that $\mathbf{w}(0) = \mathbf{0}$.

(a) Let $\rho = \min_{1 \leq n \leq N} y_n(\mathbf{w}^{*\top}\mathbf{x}_n)$. Show that $\rho > 0$.

a) Sea $P = \min_{1 \le n < N} y_n (w^{*T} x_n)$

muestre que $P > 0$

Por definición $w^*$ es un conjunto optima de medidas que separan los datos

Sea $X_a$ una dato con una serie de caracteristicas $\{x_{1a}, x_{2a}, x_{3a} \ldots\}$ Al testear en los datos $y_N (w^{*T} x_a)$ es mayor que 0 por ser una distancia.

(b) Show that $\mathbf{w}^{\mathrm{T}}(t)\mathbf{w}^* \geq \mathbf{w}^{\mathrm{T}}(t-1)\mathbf{w}^* + \rho$, and conclude that $\mathbf{w}^{\mathrm{T}}(t)\mathbf{w}^* \geq t\rho$.
*[Hint: Use induction.]*

b) Nosotros tenemos.

$$W^t(t-1) = (w(t) - y(t)\, x(t))^t$$

$$= w^T - y(t)\, x^T(t)$$

entonces tenemos:

$$W^T(t-1)\, W^* + P = W^T(t)\, w^t - y(t)\, X^T(t)$$

Nosotros Adicionalmente tenemos

$$P \le y(t)\, X^T(t)\, w^* \quad \text{Por definición:}$$

$$P - y(t)\, X^T(t)\, w^* \le 0$$

Así: $W^T(t)\,w^* - y(t)\,x^T(t)\,w^* + P \leq W^T(t)\,w^*$

para el caso

$t = 0 \Rightarrow w^T(0)\,w^* \geq 0$

Usando inducción

$$W^T(t+1)\,w^* \geq (t+1)\,P$$

nosotros observamos que:

$(t+1)\,P = tP + P \leq w^T(t)\,w^* + P \leq w^T(t+1)\,w^*$

(c) Show that $\|\mathbf{w}(t)\|^2 \leq \|\mathbf{w}(t-1)\|^2 + \|\mathbf{x}(t-1)\|^2$.

*[Hint: $y(t-1) \cdot (\mathbf{w}^T(t-1)\mathbf{x}(t-1)) \leq 0$ because $\mathbf{x}(t-1)$ was misclassified by $\mathbf{w}(t-1)$.]*

c) Para $y(t-1) = +1$

$\|w(t)\|^2 = \|w(t-1)\|^2 + \|x(t-1)\|^2 - 2\|w(t-1)\|\|x(t-1)\|\cos\theta$

es el angulo entre $-w(t-1)$ y $x(t-1)$ y $\theta \leq \frac{\pi}{2}$ Donde:

$y(t-1)\,w^T(t-1)\,x(t-1) < 0 \Rightarrow w^T(t-1)\,x(t-1) < 0 \Leftrightarrow$

$\cos(\pi - \theta) < 0 \Rightarrow \pi - \theta > \frac{\pi}{2} \Leftrightarrow 0 < \frac{\pi}{2}$

Para el caso $y(t-1) = -1$

Entonces:

$\cos\theta \geq 0 \Leftrightarrow -2\|w(t-1)\|\|x(t-1)\|\cos\theta \leq 0$

(d) Show by induction that $\|\mathbf{w}(t)\|^2 \leq tR^2$, where $R = \max_{1 \leq n \leq N} \|\mathbf{x}_n\|$.

(d) Caso basico $t=0$, $\|w(0)\|^2 \leq 0\,R^2 \Leftrightarrow 0 \leq 0$

Asumomos $\|w(t+1)\|^2 \leq (t+1)R^2$

Nosotros observamos

$$(t+1)R^2 = tR^2 + R^2 \geq \|w(t)\|^2 + R^2 \geq \|w(t)\|^2 + \|x(t-1)\|^2 \geq \|w(t+1)\|^2$$

(e) Using (b) and (d), show that

$$\frac{\mathbf{w}^{\mathrm{T}}(t)}{\|\mathbf{w}(t)\|}\mathbf{w}^* \geq \sqrt{t} \cdot \frac{\rho}{R},$$

and hence prove that

$$t \leq \frac{R^2\|\mathbf{w}^*\|^2}{\rho^2}.$$

[*Hint:* $\frac{\mathbf{w}^{\mathrm{T}}(t)\mathbf{w}^*}{\|\mathbf{w}(t)\|\,\|\mathbf{w}^*\|} \leq 1$. *Why?*]

In practice, PLA converges more quickly than the bound $\frac{R^2\|\mathbf{w}^*\|^2}{\rho^2}$ suggests. Nevertheless, because we do not know $\rho$ in advance, we can't determine the number of iterations to convergence, which does pose a problem if the data is non-separable.

e) Haciendo Referencia a b tenemos

$$\frac{w^T(t)}{\|w(t)\|} w^* \geq \frac{w^T(t-1) w^* + P}{\|w(t)\|} \geq \frac{tP}{\|w(t)\|}$$

Por el caso de d, nosotros tenemos

$$\frac{tP}{\|w(t)\|} \geq \frac{tP}{\sqrt{t}R} = \frac{\sqrt{t}P}{R}$$