$$\nabla f_o(x) + \lambda_1 \nabla f_1(x) + \lambda_2 \nabla f_2(x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0 \cdot \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 - \lambda_2 \end{bmatrix} = 0$$

No existe λ_2 y por tanto λ que satisfaga la ecuación anterior. Por lo tanto sí existe Δx de descenso y x no es mínimo.

Comentario

En resúmen de los 3 ejemplos anteriores: si x^* es una solución local del problema estándar de optimización entonces existen vectores multiplicadores de Lagrange v^*, λ^* para las restricciones de igualdad y desigualdad respectivamente tales que:

$$\nabla_x \mathcal{L}(x^*, v^*, \lambda^*) = 0$$

$$h_i(x^*) = 0 \quad \forall i = 1, \ldots, p$$

$$f_i(x^*) \leq 0 \quad \forall i = 1, \ldots, m$$

$$\lambda_i^* \geq 0 \quad \forall i = 1, \ldots, m$$

$$\lambda_i^* f_i(x^*) = 0 \quad \forall i = 1, \ldots, m$$

faltan considerar suposiciones importantes para tener completo el resultado pero los ejemplos anteriores abren camino hacia las **condiciones de Karush-Kuhn-Tucker, *aka KKT*, de optimalidad**.

Un problema estándar de optimización es:

$$\min f_o(x)$$

sujeto a:

$$f_i(x) \leq 0, \quad \forall i = 1, \ldots, m$$

$$h_i(x) = 0, \quad \forall i = 1, \ldots, p$$

con $x = (x_1, x_2, \ldots, x_n)^T$ es la **variable de optimización del problema**, $f_o : \mathrm{R}^n \to \mathrm{R}$ es la **función objetivo**, $f_i : \mathrm{R}^n \to \mathrm{R}$ $\forall i = 1, \ldots, m$ son las **restricciones de desigualdad**, $h_i : \mathrm{R}^n \to \mathrm{R}$, $\forall i = 1, \ldots, p$ son las **restricciones de igualdad**.

Observación

Obsérvese que las condiciones KKT de optimalidad son condiciones **necesarias** e involucran información de primer orden.

Máquina de Soporte Vectorial (SVM) para datos linealmente separables, ejemplo de *Constrained Inequality Convex Optimization* (CICO):

Clasificador lineal

Considérese dos conjuntos de puntos en R^n cuyos atributos o *features* están dados por $\{x_1, x_2, \ldots, x_N\}$, $\{y_1, y_2, \ldots, y_M\}$ y tales puntos tienen etiquetas $\{-1, 1\}$ respectivamente.

El objetivo es encontrar una función $f : \mathrm{R}^n \to \mathrm{R}$ que sea negativa en el primer conjunto de puntos y positiva en el segundo conjunto de puntos:

$$f(x_i) < 0 \quad \forall i = 1, \ldots, N$$

$$f(y_i) > 0 \quad \forall i = 1, \ldots, M$$

Si existe tal función entonces f o el conjunto $\{x : f(x) = 0\}$ separa o clasifica los 2 conjuntos de puntos.

Observación

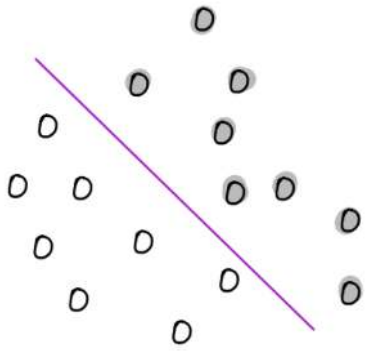
La clasificación puede ser débil que significa: $f(x_i) \leq 0$, $f(y_i) \geq 0$.

Para el caso en el que los datos son linealmente separables se busca una función afín de la forma $f(x) = a^T x - b$ que clasifique los puntos, esto es:

$$a^T x_i - b < 0, \quad \forall i = 1, \ldots, N$$

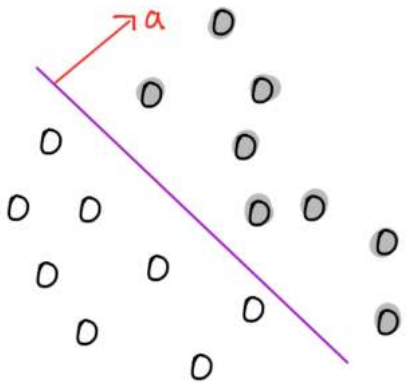
$$a^T y_i - b > 0, \quad \forall i = 1, \ldots, M$$

Geométricamente se busca un hiperplano de dimensión $n-1$ que separe ambos conjuntos:

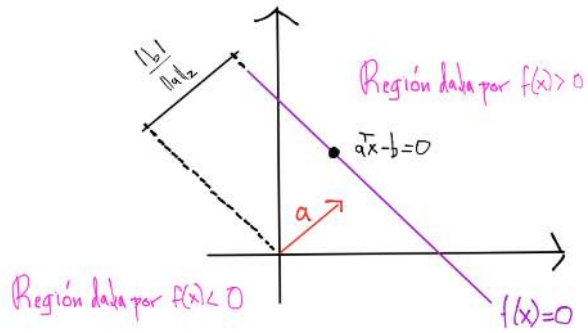


En el dibujo tómese los puntos sin rellenar como los pertenecientes a una clase de $\{-1, 1\}$.

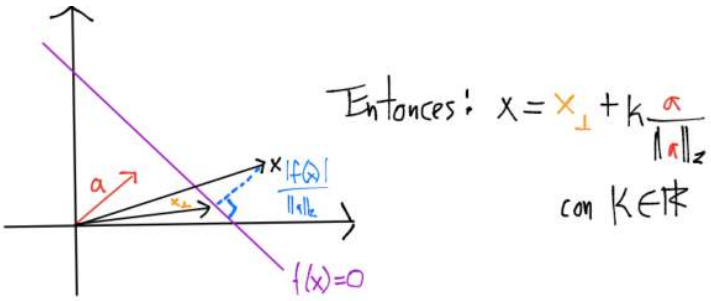
Si x, y son puntos en el hiperplano entonces $f(x) = f(y) = 0$ y además $a^T(x-y) = b - b = 0$, así a es ortogonal a todo punto en el hiperplano y determina la orientación de este:



Si x es un punto en el hiperplano tenemos $f(x) = 0$ por lo tanto la distancia perpendicular del origen al hiperplano está dada por: $\frac{|a^T x|}{||a||_2} = \frac{|b|}{||a||_2}$ y el parámetro b determina la localización del hiperplano:



Sean $x \in \mathrm{R}^n$ y $x_\perp \in \mathrm{R}^n$ la proyección ortogonal en el hiperplano:



tenemos: $a^T x = a^T x_\perp + k||a||_2 = b + k||a||_2$ por lo que $a^T x - b = k||a||_2$ y:

$$k = \frac{a^T x - b}{||a||_2} = \frac{f(x)}{||a||_2},$$

esto es, $|f(x)|$ da una medida de distancia perpendicular de x al hiperplano.

Modelo de SVM

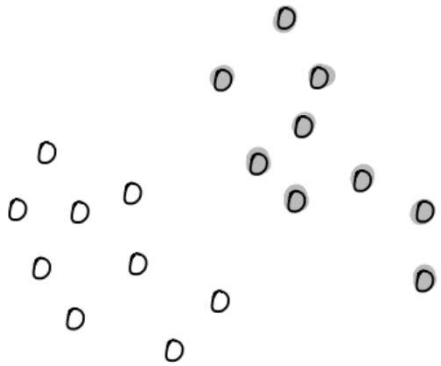
En este modelo se desea obtener:

$$a^T x_i - b \leq -1, \quad \forall i = 1, \ldots, N$$

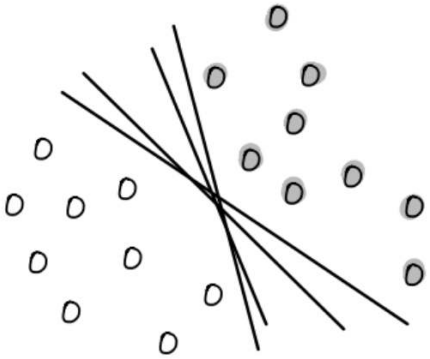
$$a^T y_i - b \geq 1, \quad \forall i = 1, \ldots, M$$

> **Observación**
>
> El uso de $\{-1, 1\}$ ayuda a la escritura y formulación matemática del modelo.

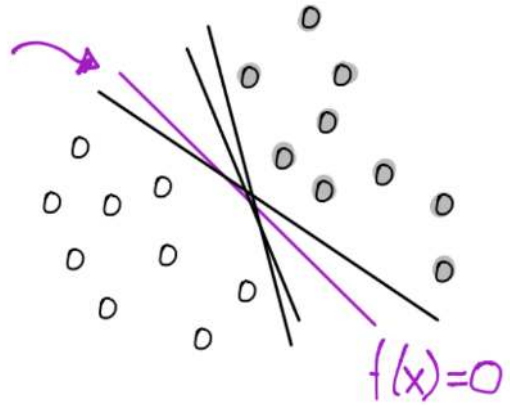
para clasificar a datos linealmente separables:



Obsérvese que existen una infinidad de hiperplanos que separan a los datos anteriores:



En la SVM buscamos un hiperplano que tenga la máxima separación de cada una de las distancias de los datos al mismo:



Una distancia al hiperplano para los datos $x_i, \forall i = 1, \ldots, N$ y otra distancia para los datos $y_i \forall i = 1, \ldots, M$.

Por la sección anterior, la distancia al hiperplano para cada conjunto de datos está dada por:

$$\min_{i=1,\dots,N} \frac{|f(x_i)|}{||a||_2}$$

$$\min_{i=1,\dots,M} \frac{|f(y_i)|}{||a||_2}$$

Entonces encontrar un hiperplano que tenga la máxima separación de cada una de las distancias de los datos al hiperplano se puede escribir como el problema:

$$\max_{a,b} \left\{ \min_{i=1,\dots,N} \frac{|f(x_i)|}{||a||_2} + \min_{i=1,\dots,M} \frac{|f(y_i)|}{||a||_2} \right\}$$

Obsérvese que el cociente $\frac{|f(x_i)|}{||a||_2}$ es invariante ante reescalamientos por ejemplo:

$$\frac{|ka^T x - kb|}{||ka||_2} = \frac{a^T x - b}{||a||_2} \forall k \neq 0$$

Por tanto, si los índices i_1, i_2 cumplen:

$$i_1 = \text{argmin}_{i=1,\dots,N} \left\{ \frac{|f(x_i)|}{||a||_2} \right\}$$

$$i_2 = \text{argmin}_{i=1,\dots,M} \left\{ \frac{|f(y_i)|}{||a||_2} \right\}$$

se puede hacer un reescalamiento para obtener:

$$|f(x_{i_1})| = 1, \quad |f(x_i)| > 1 \quad \forall i = 1, \dots, N, i \neq i_1$$

$$|f(y_{i_2})| = 1, \quad |f(y_i)| > 1 \quad \forall i = 1, \dots, M, i \neq i_2$$

Esto es, $|f(x_i)| \geq 1, \forall i = 1, \dots, N$, $|f(y_i)| \geq 1, \forall i = 1, \dots, M$.

Problema de optimización en SVM

El problema canónico o estándar de la máquina de soporte vectorial es:

$$\max_{a,b} \frac{2}{||a||_2}$$

sujeto a:

$$f(x_i) \leq -1, \quad \forall i = 1, \dots, N$$

$$f(y_i) \geq 1, \quad \forall i = 1, \dots, M$$

El cual es equivalente a:

$$\min_{a,b} \frac{||a||_2^2}{2}$$

sujeto a:

$$a^T x_i - b \leq -1, \quad \forall i = 1, \dots, N$$

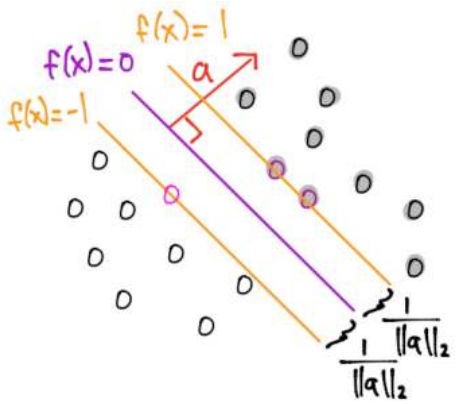
$$a^T y_i - b \geq 1, \quad \forall i = 1, \dots, M$$

Comentarios

- Se ha quitado el valor absoluto $|f(x_i)|$, $|f(y_i)|$ pues x_i, y_i se asumen cumplen $a^T x_i - b \geq 1$, $a^T y_i - b \leq -1$, esto es, x_i, y_i son clasificados correctamente.
- Este problema es de optimización convexa con restricciones de desigualdad.

Vectores de soporte

Al encontrar este hiperplano tenemos otros dos hiperplanos paralelos ortogonales al vector a sin ningún dato entre ellos a una distancia de $\frac{1}{||a||_2}$:



Definición

Aquellas restricciones activas son originadas por puntos con el nombre de vectores de soporte.

Comentarios

- Al resolver el problema de optimización siempre existirán al menos 2 restricciones activas pues siempre hay una distancia mínima para cada conjunto de puntos.
- Las otras dos rectas forman lo que se conoce como margen.

Ejemplo Iris *dataset*

Utilizamos el conocido *dataset* de *Iris* en el que se muestran **tres especies del género *Iris***. Las especies son: *I. setosa*, *I. virginica* y *I. versicolor*:

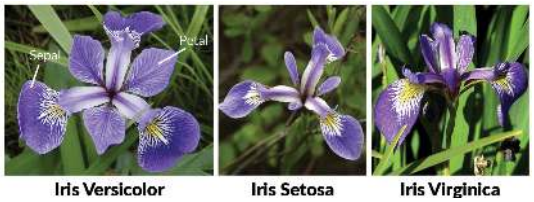


Imagen obtenida de Iris Dataset.

La clase $\mathcal{C}_{-1}$ es `setosa` y $\mathcal{C}_1$ es `versicolor` codificadas como $-1, 1$ respectivamente.

```
data_iris_setosa_versicolor = data_iris[0:100].copy()
classes = iris["target"][0:100].copy()
```

```
print(classes)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
classes[0:50] = classes[0:50].copy()-1
```

```
print(classes)
```

```
[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1]
```

Estandarizamos la matriz de datos:

```
data_iris_setosa_versicolor = (data_iris_setosa_versicolor-

data_iris_setosa_versicolor.mean(axis=0))/data_iris_setosa_versicolor.std(axis=0)
```

Añadimos la columna que indica uso de intercepto:

```
m,n = data_iris_setosa_versicolor.shape
```

```
data_iris_setosa_versicolor = np.column_stack((-1*np.ones((m,1)),
data_iris_setosa_versicolor))
```

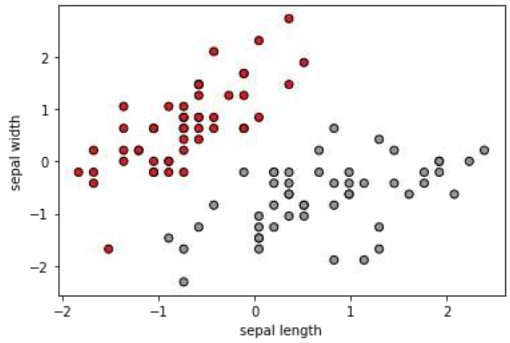
```
print(data_iris_setosa_versicolor[0:10, 0:10])
```

```
[[-1.   -0.58  0.84 -1.01 -1.04]
 [-1.   -0.89 -0.21 -1.01 -1.04]
 [-1.   -1.21  0.21 -1.08 -1.04]
 [-1.   -1.36  0.   -0.94 -1.04]
 [-1.   -0.74  1.05 -1.01 -1.04]
 [-1.   -0.11  1.68 -0.8  -0.69]
 [-1.   -1.36  0.63 -1.01 -0.86]
 [-1.   -0.74  0.63 -0.94 -1.04]
 [-1.   -1.68 -0.42 -1.01 -1.04]
 [-1.   -0.89  0.   -0.94 -1.22]]
```

Visualizamos para algunos atributos:

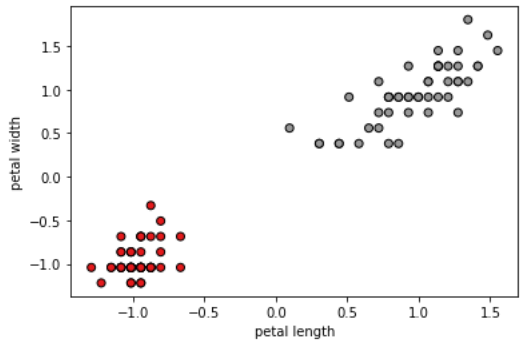
```
plt.scatter(data_iris_setosa_versicolor[:, 1],
            data_iris_setosa_versicolor[:, 2], c=classes, cmap=plt.cm.Set1,
            edgecolor='k')
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.show()
```

Obsérvese que considerando los atributos `Sepal.Length` y `Sepal.Width` se pueden separar la clase *I. setosa* codificada como -1 y la clase *I. versicolor* codificada como 1 de forma lineal.




```
plt.scatter(data_iris_setosa_versicolor[:, 3],
            data_iris_setosa_versicolor[:, 4], c=classes, cmap=plt.cm.Set1,
            edgecolor='k')
plt.xlabel("petal length")
plt.ylabel("petal width")
plt.show()
```

Obsérvese que considerando los atributos `Petal.Length` y `Petal.Width` se pueden separar la clase *I. setosa* codificada como -1 y la clase *I. versicolor* codificada como 1 de forma lineal.



Función objetivo:

$$\frac{||a||_2^2}{2}$$

y restricciones:

$$a^T x_i - b \leq -1, \quad \forall i = 1, \ldots, N$$

$$a^T y_i - b \geq 1, \quad \forall i = 1, \ldots, M$$

```
n = 5 #number of variables
vec = cp.Variable(n) #optimization variable
```

```
fo_cvxpy = 1/2*cp.norm(vec[1:n],2)**2 #fo just includes a not intercept
```

```
constraints = [data_iris_setosa_versicolor[0:50,:]@vec <=-1,
               data_iris_setosa_versicolor[50:100,:]@vec >= 1]
```

```
obj = cp.Minimize(fo_cvxpy)
```

```
prob = cp.Problem(obj, constraints)
print(prob.solve())
```

```
0.6190278040023801
```

```
print("\nThe optimal value is", prob.value)
print("The optimal vector with intercept is")
print(vec.value)
```

```
The optimal value is 0.6190278040023801
The optimal vector with intercept is
[-0.24  0.27 -0.34  0.7   0.75]
```

Ver cvxpy: svm

El vector `vec` contiene al intercepto b y al vector a.

`vec[0]` es b, `vec[1:n]` es a.

Los primeros 50 renglones pertenecen a la clase $\mathcal{C}_{-1}$: *I. setosa* y los restantes 50 renglones pertenecen a la clase $\mathcal{C}_1$: *I. versicolor*. Se clasifica de acuerdo al signo de: $a^T x - b$:

```
print(np.sign(data_iris_setosa_versicolor[0:50,:]@vec.value))
```

```
[-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
```

```
print(np.sign(data_iris_setosa_versicolor[50:100,:]@vec.value))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1.]
```

Visualización utilizando únicamente `Sepal.Length` y `Sepal.Width`:

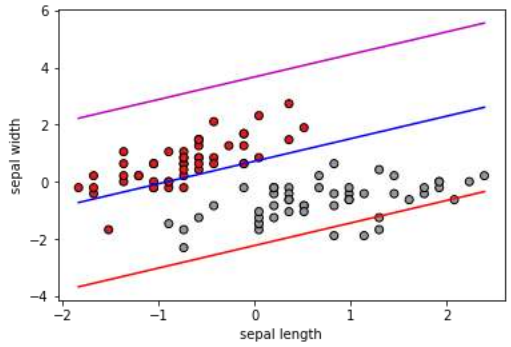
```
x_min = np.min(data_iris_setosa_versicolor[:,1])
x_max = np.max(data_iris_setosa_versicolor[:,1])
y_min = np.min(data_iris_setosa_versicolor[:,2])
y_max = np.max(data_iris_setosa_versicolor[:,2])
x_plot = np.linspace(x_min, x_max, 100)
```

```
b, a = vec.value[0], vec.value[1:n]
```

```
y_plot = 1/a[1]*(-a[0]*x_plot + b)
y_plot_minus_1 = 1/a[1]*(-a[0]*x_plot + b -1)

y_plot_plus_1 = 1/a[1]*(-a[0]*x_plot + b + 1)
```

```
plt.scatter(data_iris_setosa_versicolor[:, 1],
            data_iris_setosa_versicolor[:, 2], c=classes, cmap=plt.cm.Set1,
            edgecolor='k')
plt.plot(x_plot,y_plot,'b',
         x_plot,y_plot_minus_1, 'm',
         x_plot, y_plot_plus_1,'r')
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.show()
```



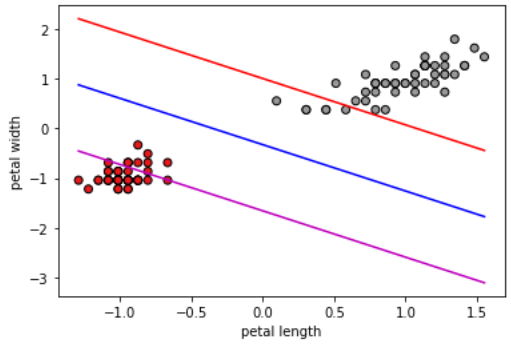
Visualización utilizando únicamente `Petal.Length` y `Petal.Width`:

```
x_min = np.min(data_iris_setosa_versicolor[:,3])
x_max = np.max(data_iris_setosa_versicolor[:,3])
y_min = np.min(data_iris_setosa_versicolor[:,4])
y_max = np.max(data_iris_setosa_versicolor[:,4])
x_plot = np.linspace(x_min, x_max, 100)
```

```
y_plot = 1/a[3]*(-a[2]*x_plot + b)
y_plot_minus_1 = 1/a[3]*(-a[2]*x_plot + b -1)

y_plot_plus_1 = 1/a[3]*(-a[2]*x_plot + b + 1)
```

```
plt.scatter(data_iris_setosa_versicolor[:, 3],
            data_iris_setosa_versicolor[:, 4], c=classes, cmap=plt.cm.Set1,
            edgecolor='k')
plt.plot(x_plot,y_plot,'b',
         x_plot,y_plot_minus_1, 'm',
         x_plot, y_plot_plus_1,'r')
plt.xlabel("petal length")
plt.ylabel("petal width")
plt.show()
```



Ejercicio

Realiza la clasificación anterior para las clases `virginica` y `versicolor`.

Ajustando el modelo de la SVM sólo para dos atributos: `Sepal.Length` y `Sepal.Width`:

```
data_iris_setosa_versicolor = data_iris[0:100, 0:2].copy()
```

Aquí no estandarizamos.

```
data_iris_setosa_versicolor = np.column_stack((-1*np.ones((m,1)),
data_iris_setosa_versicolor))
```

```
n = 3 #number of variables
vec = cp.Variable(n) #optimization variable
```

```
fo_cvxpy = 1/2*cp.norm(vec[1:n],2)**2 #fo just includes a not intercept
```

```
constraints = [data_iris_setosa_versicolor[0:50,:]@vec <=-1,
               data_iris_setosa_versicolor[50:100,:]@vec >= 1]
```

```
obj = cp.Minimize(fo_cvxpy)
```

```
prob = cp.Problem(obj, constraints)
print(prob.solve())
```

```
33.79501292632983
```

```
print("\nThe optimal value is", prob.value)
print("The optimal vector with intercept is")
print(vec.value)
```

```
The optimal value is 33.79501292632983
The optimal vector with intercept is
[17.32  6.32 -5.26]
```

```
print(np.sign(data_iris_setosa_versicolor[0:50,:]@vec.value))
```

```
[-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
 -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
```

```
print(np.sign(data_iris_setosa_versicolor[50:100,:]@vec.value))
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1.]
```

```
b, a = vec.value[0], vec.value[1:n]
```

```
print(b)
```

```
17.315789240731355
```

```
print(a)
```

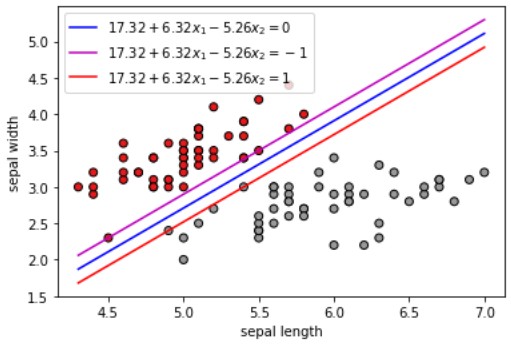
```
[ 6.32 -5.26]
```

```
x_min = np.min(data_iris_setosa_versicolor[:,1])
x_max = np.max(data_iris_setosa_versicolor[:,1])
y_min = np.min(data_iris_setosa_versicolor[:,2])
y_max = np.max(data_iris_setosa_versicolor[:,2])
x_plot = np.linspace(x_min, x_max, 100)
```

```
y_plot = 1/a[1]*(-a[0]*x_plot + b)
y_plot_minus_1 = 1/a[1]*(-a[0]*x_plot + b -1)

y_plot_plus_1 = 1/a[1]*(-a[0]*x_plot + b + 1)
```

```
plt.scatter(data_iris_setosa_versicolor[:, 1],
            data_iris_setosa_versicolor[:, 2], c=classes, cmap=plt.cm.Set1,
            edgecolor='k')
plt.plot(x_plot,y_plot,'b',
         x_plot,y_plot_minus_1, 'm',
         x_plot, y_plot_plus_1,'r')
plt.legend(["$17.32 + 6.32x_1 -5.26x_2=0$",
            "$17.32 + 6.32x_1 -5.26x_2=-1$",
            "$17.32 + 6.32x_1 -5.26x_2=1$"])
plt.xlabel("sepal length")
plt.ylabel("sepal width")
plt.show()
```



Ejercicio

1. Determina para este ajuste los vectores de soporte y márcalos en la gráfica.
2. Realiza el ajuste para los atributos `Petal.Length`, `Petal.Width` estandarizando la matriz de datos para las mismas clases y la visualización de los datos con la recta que separa y las rectas que forman al margen.
3. ¿Es necesario estandarizar si se usan dos atributos para este *dataset*? realiza el ajuste del ejercicio 2 anterior sin estandarizar la matriz de datos para las mismas clases y la visualización de los datos con la recta que separa y las rectas que forman al margen.

Problemas de programación lineal, ejemplo de CIEO:

Una gran cantidad de aplicaciones utilizan formulaciones de problemas de programación lineal que involucran igualdades y desigualdades cuya forma **estándar** está dada por:

$$\min c^T x$$

sujeto a:

$$Ax = b$$

$$x \geq 0$$

donde: $A \in \mathrm{R}^{m \times n}$ y se **asume** $m \leq n$ y tiene *rank* completo por renglones y la última desigualdad se refiere a que todas las componentes del vector x son mayores o iguales a cero.

Observación

Obsérvese que la función objetivo $f_o(x) = c^T x$ y tanto las restricciones de igualdad como desigualdad son funciones lineales.