Sessió 2

En aquesta segona sessió aplicarem l'algorisme perceptró a algunes tasques de classificació. El quadern d'iris descriu una implementació senzilla de perceptró i la seua aplicació. L'objectiu principal d'aquesta sessió és estendre l'exemple donat a altres tasques, tractant de minimitzar l'error de test.

Perceptró aplicat a iris

Lectura del corpus: comprovem també que les matrius de dades i etiquetes tenes les files i columnes que toca

In [1]:
```
import numpy as np; from sklearn.datasets import load_iris
iris = load_iris(); X = iris.data.astype(np.float16);
y = iris.target.astype(np.uint).reshape(-1, 1);
print(X.shape, y.shape, "\n", np.hstack([X, y])[:5, :])
```

```
(150, 4) (150, 1)
 [[5.1015625  3.5        1.40039062 0.19995117 0.        ]
 [4.8984375  3.         1.40039062 0.19995117 0.        ]
 [4.69921875 3.19921875 1.29980469 0.19995117 0.        ]
 [4.6015625  3.09960938 1.5        0.19995117 0.        ]
 [5.         3.59960938 1.40039062 0.19995117 0.        ]]
```

Partició del corpus: Creem un split d'iris amb un 20% de dades per a test i la resta per a entrenament (training), barallant prèviament les dades d'acord amb una llavor donada per a la generació de nombres aleatoris. Ací, com en tot codi que incloga aleatorietat (que requerisca generar nombres aleatoris), convé fixar la dita llavor per a poder reproduir experiments amb exactitud.

In [2]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True, random_state=23)
print(X_train.shape, X_test.shape)
```

```
(120, 4) (30, 4)
```

Implementació de Perceptró: torna pesos en notació homogènia, $\mathbf{W} \in \mathbb{R}^{(1+D)\times C}$; també el nombre d'errors i iteracions executades

In [3]:
```python
def perceptro(X, y, b=0.1, a=1.0, K=200):
    N, D = X.shape; Y = np.unique(y); C = Y.size; W = np.zeros((1+D, C))
    for k in range(1, K+1):
        E = 0
        for n in range(N):
            xn = np.array([1, *X[n, :]])
            cn = np.squeeze(np.where(Y==y[n]))
            gn = W[:,cn].T @ xn; err = False
            for c in np.arange(C):
                if c != cn and W[:,c].T @ xn + b >= gn:
                    W[:, c] = W[:, c] - a*xn; err = True
            if err:
                W[:, cn] = W[:, cn] + a*xn; E = E + 1
        if E == 0:
            break;
    return W, E, k
```

Aprenentatge d'un classificador (lineal) amb Perceptró: Perceptró minimitza el nombre d'errors d'entrenament (amb marge)

$$\mathbf{W}^* = \underset{\mathbf{W}=(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_C)}{\operatorname{argmin}} \sum_n \mathbb{I}\left(\max_{c\neq y_n} \boldsymbol{w}_c^t \boldsymbol{x}_n + b \; > \; \boldsymbol{w}_{y_n}^t \boldsymbol{x}_n \right)$$

In [4]:
```python
W, E, k = perceptro(X_train, y_train)
print("Nombre d'iteracions executades: ", k)
print("Nombre d'errors d'entrenament: ", E)
print("Vectors de pesos de les classes (en columnes i amb notació homogènia):\n", W);
```

```
Nombre d'iteracions executades:  200
Nombre d'errors d'entrenament:  2
Vectors de pesos de les classes (en columnes i amb notació homogènia):
 [[  10.           85.         -142.        ]
 [ -49.421875    -68.19140625 -176.47265625]
 [  50.171875     -1.72460938 -181.06445312]
 [-189.91210938  -87.70507812   68.69726562]
 [ -86.40258789 -137.78149414  157.88415527]]
```

Càlcul de la taxa d'error en test:

In [5]:
```
X_testh = np.hstack([np.ones((len(X_test), 1)), X_test])
y_test_pred  = np.argmax(X_testh @ W, axis=1).reshape(-1, 1)
err_test = np.count_nonzero(y_test_pred != y_test) / len(X_test)
print(f"Taxa d'error en test: {err_test:.1%}")
```

```
Taxa d'error en test: 16.7%
```

Ajust del marge: experiment per a aprendre un valor de b

In [6]:
```
for b in (.0, .01, .1, 10, 100):
    W, E, k = perceptro(X_train, y_train, b=b, K=1000)
    print(b, E, k)
```

```
0.0 3 1000
0.01 5 1000
0.1 3 1000
10 6 1000
100 6 1000
```

Interpretació de resultats: les dades d'entrenament no semblen linealment separables; no està clar que un marge major que zero puga millorar resultats, sobretot perquè sols tenim 30 mostres de test; amb marge nul ja hem vist que s'obté un error (en test) del 16.7%