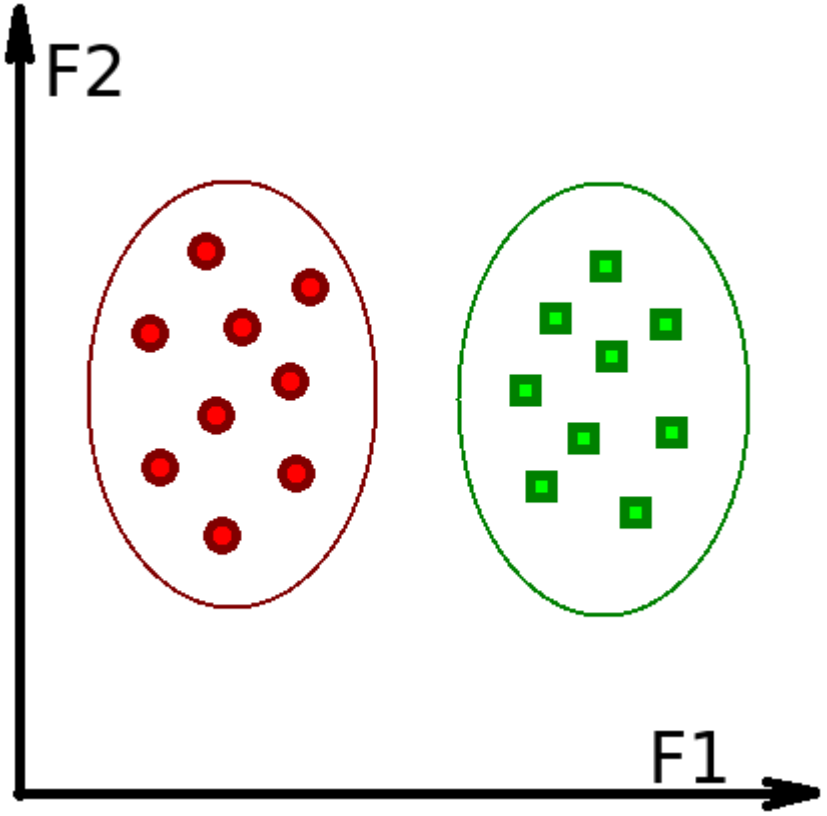
Liniowa analiza dyskryminacyjna

Statystyka biomedyczna

Piotr M. Szczypiński

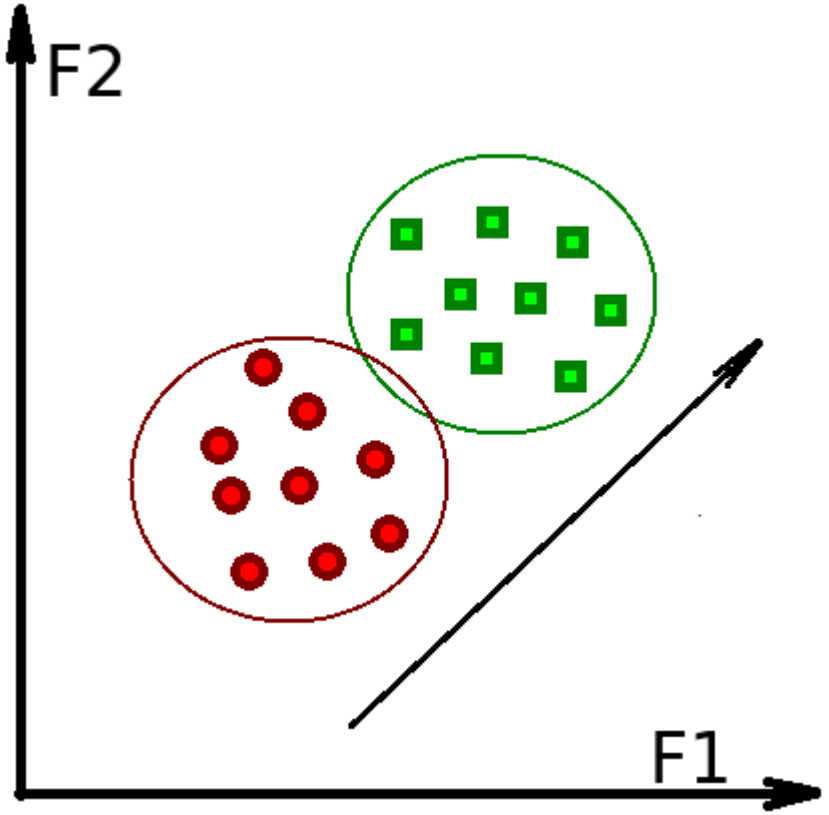
Dyskryminacja

Analiza dyskryminacyjna (ang. discriminant analysis) – zespół metod wielowymiarowej analizy danych, których zadaniem jest rozstrzyganie, które zmienne w najlepszy sposób dzielą dany zbiór przypadków na występujące w nim grupy (klasy).

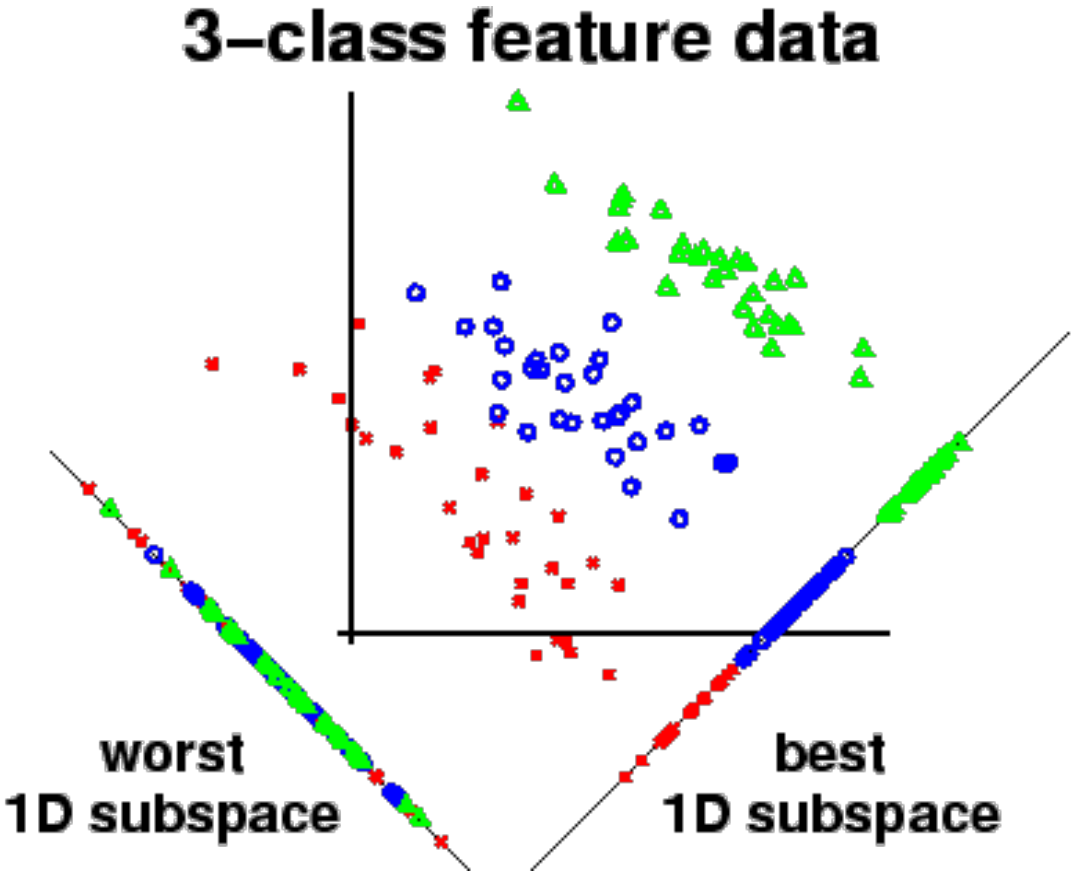


Dyskryminacja liniowa

Liniowa analiza dyskryminacyjna (ang. linear discriminant analysis, LDA) jest używana w uczeniu maszynowym do znalezienia liniowej kombinacji cech, które najlepiej rozróżniają dwie lub więcej klas obiektów lub zdarzeń.



$$LD1 = a1\ F1 + a2\ F2$$

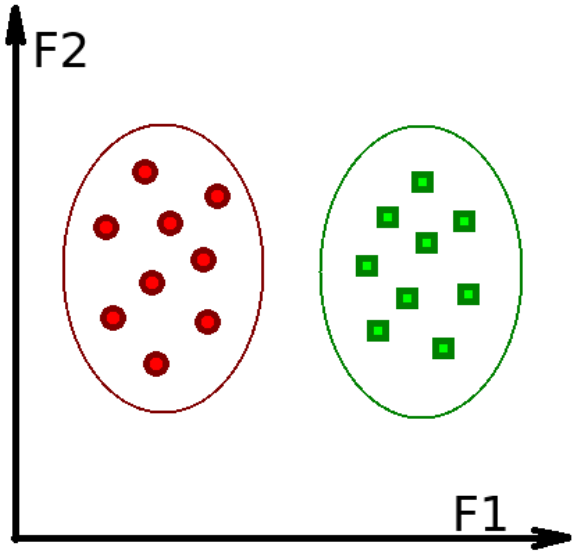

3–class feature data
worst
1D subspace
best
1D subspace

Jak znaleźć kombinację?

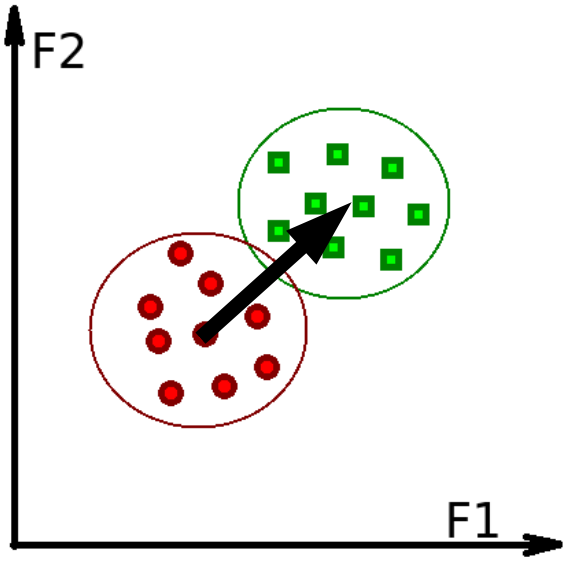
$$LD1 = a1\ F1 + a2\ F2 + a3\ F3 \ldots$$

$$LD2 = b1\ F1 + b2\ F2 + b3\ F3 \ldots$$

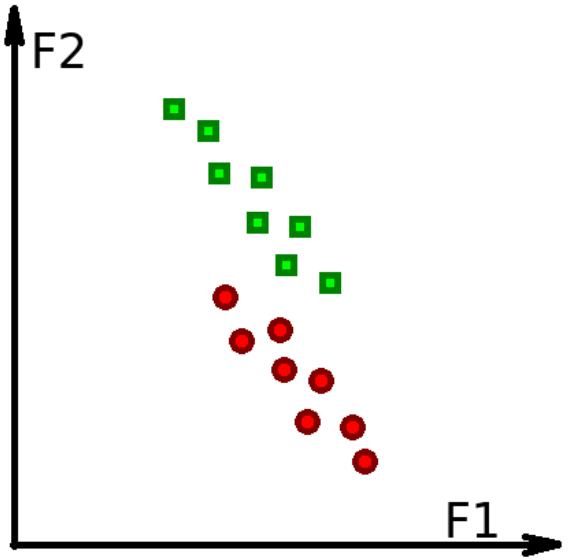
...



Test studenta?

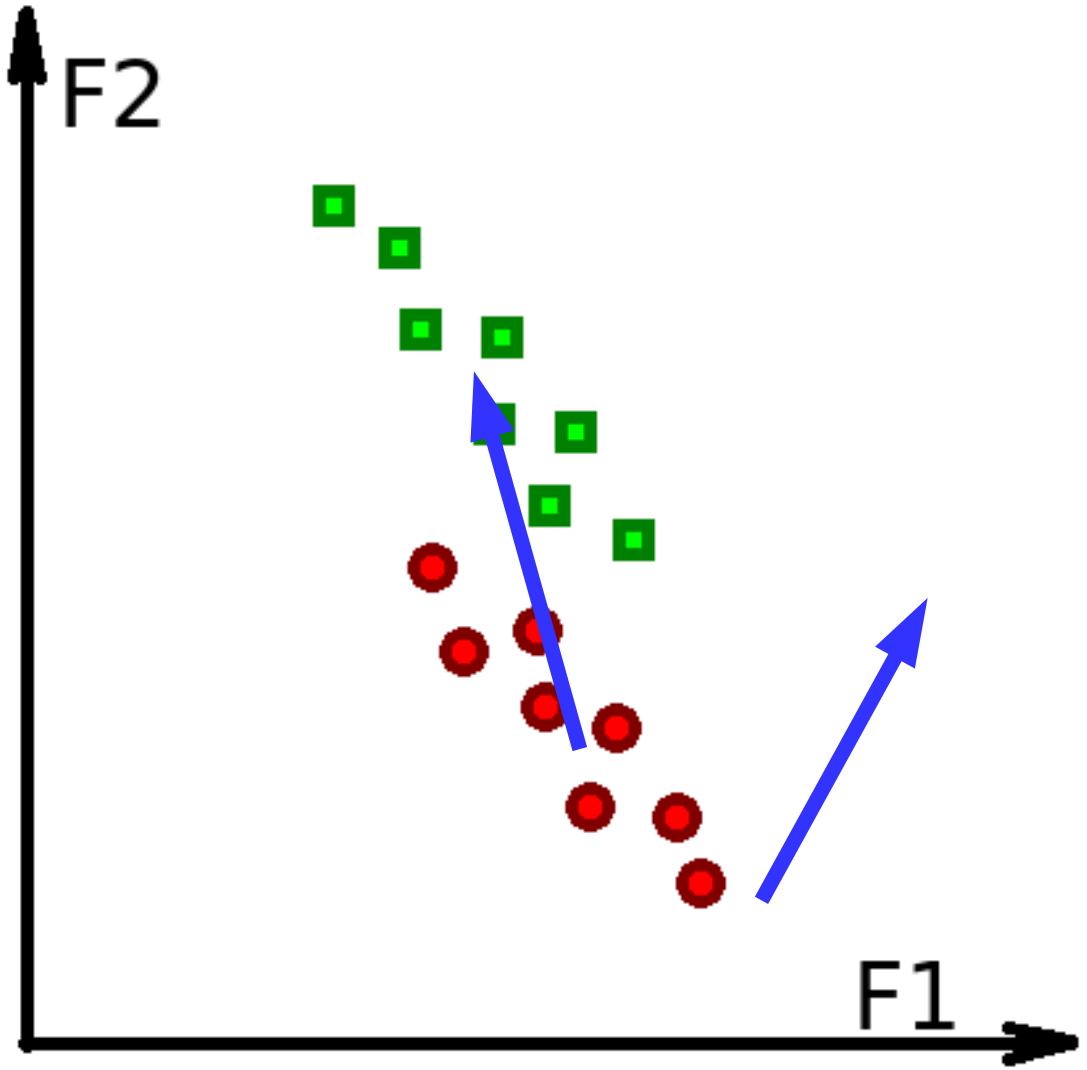


Połączyć środki skupień?



?

Jak znaleźć kombinację?



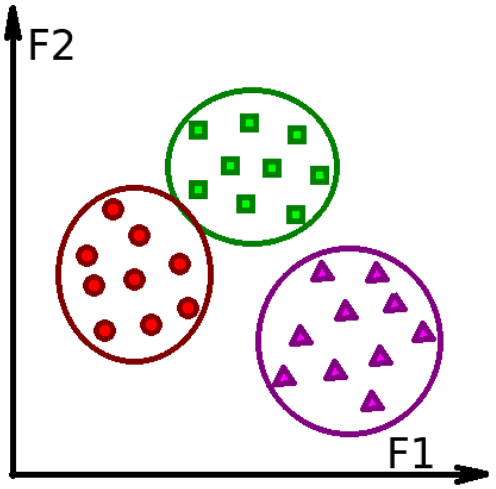
Twórca LDA

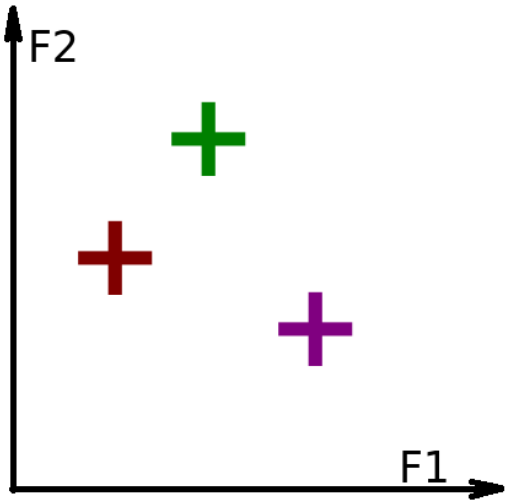
Ronald Aylmer Fisher (ur. 17 lutego 1890 w Londynie, zm. 29 lipca 1962 w Adelaide) – genetyk i statystyk brytyjski, profesor eugeniki London School of Economics w latach (1933-1943) i profesor genetyki Uniwersytetu w Cambridge (1943-1957), członek Royal Society w Londynie (Towarzystwa Królewskiego).

Ronald Fisher stworzył m.in. statystyczną metodę największej wiarygodności (ang. maximum likelihood), analizę wariancji (ANOVA) oraz **liniową analizę dyskryminacyjną**.

https://pl.wikipedia.org/wiki/Ronald_Fisher

Procedura LDA





1) Obliczamy macierze kowariancji w poszczególnych klasach:
   $\Sigma_1, \Sigma_2, \ldots \Sigma_n$
2) Obliczamy średnią macierzy kowariancji (macierz kowariancji wewnątrzklasowych): **$\Sigma_w = (\Sigma_1+\Sigma_2\ldots+\Sigma_n)/n$**
3) Obliczamy macierz kowariancji pomiędzy klasami (środkami rozkładów) **Σ_b**
4) Obliczamy wektory I wartości własne macierzy symetrycznej: **$\Sigma_w^{-1} \Sigma_b$**
5) Szeregujemy wartości własne nierosnąco oraz wektorów własne zgodnie z kolejnością wartosci własnych
6) Transformujemy liniowo przestrzeń za pomocą macierzy wektorów własnych.

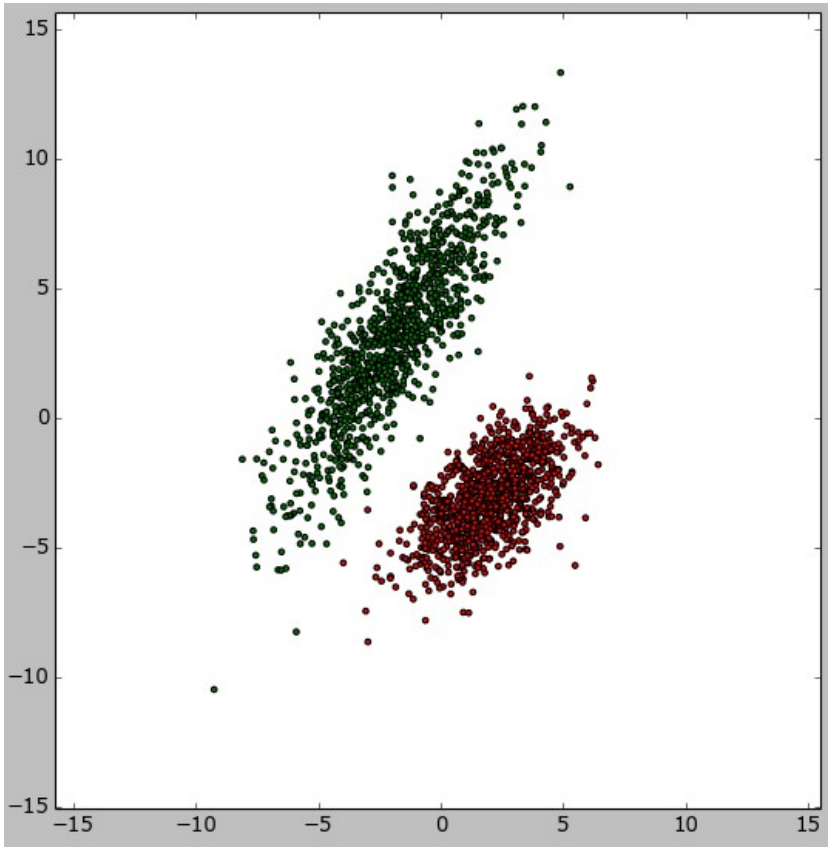
LDA vs. ANOVA vs. PCA

1) Dla przypadku jednowymiarowego LDA to ANOVA a test F jest tożsamy z wartością własną LDA
2) Celem LDA jest zmniejszenie wymiarowości przestrzeni cech poprzez wybranie nowej przestrzeni zmiennych. Do rzutowania wykorzystuje się wektory o największych wartościach własnych.
3) Nie należy mylić PCA i LDA. W PCA nowe zmienne mają maksymalizować wariancję, w LDA mają maksymalizować zdolność separacji (dyskryminacji) klas.

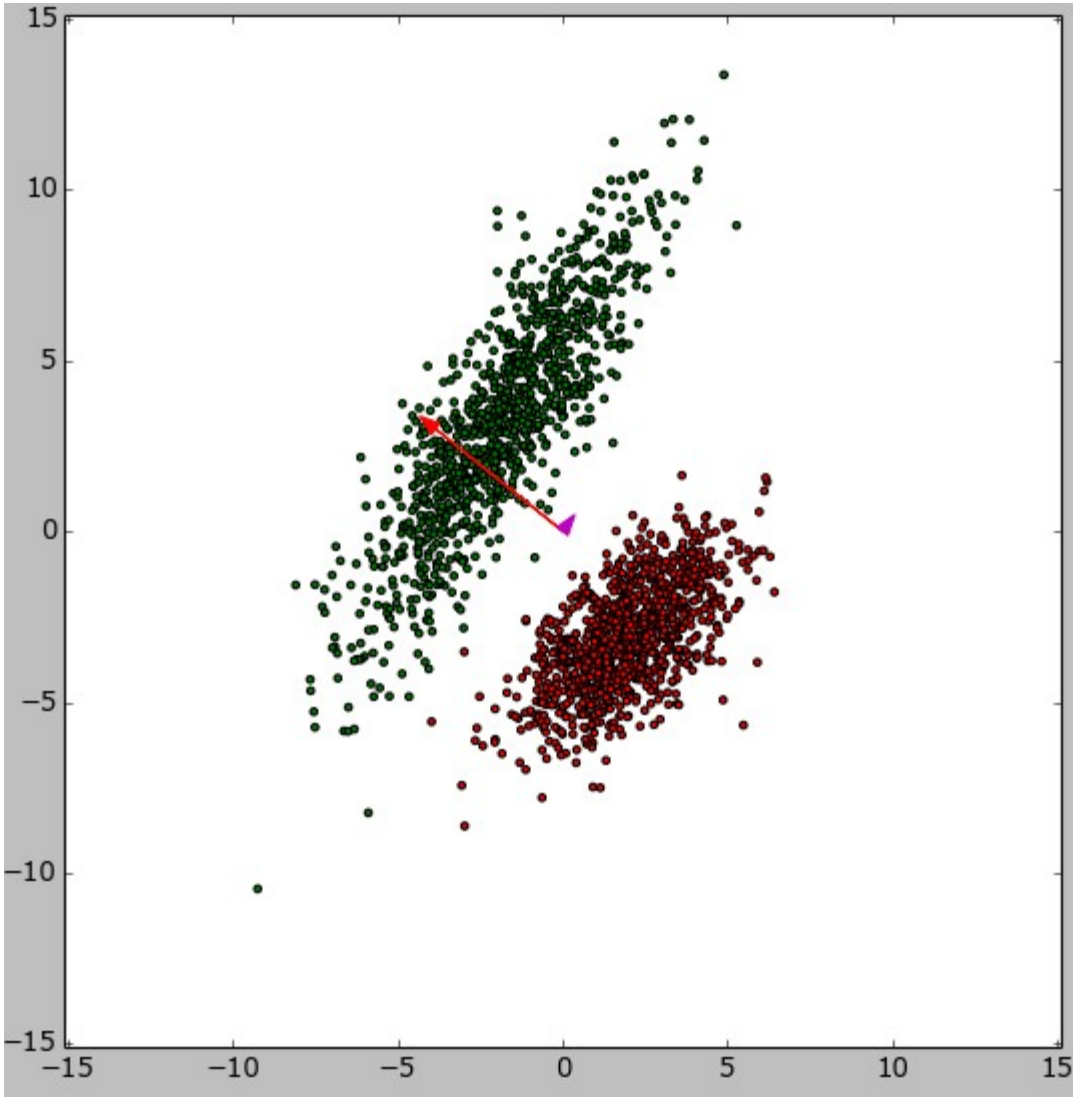
Przykład LDA w Pythonie

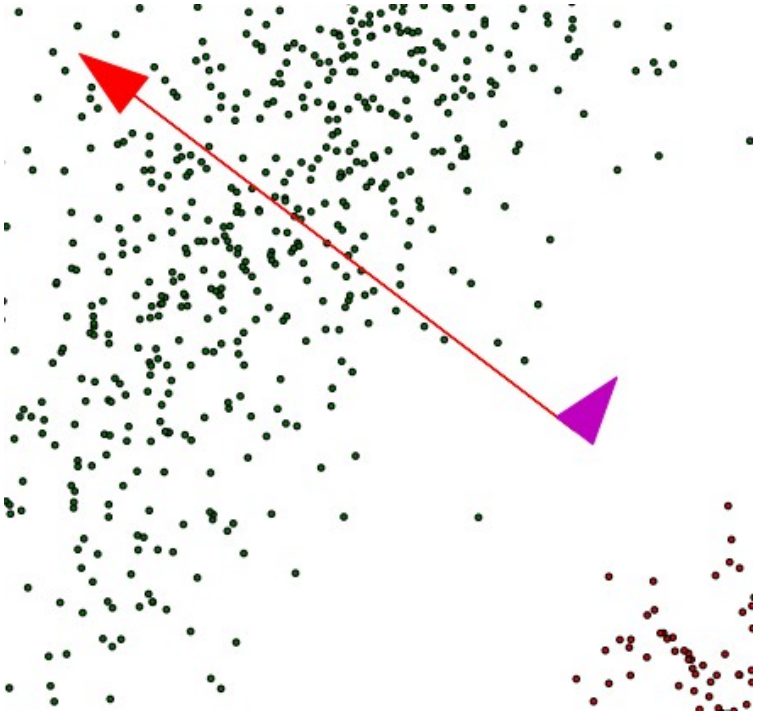
```
from numpy import *
from matplotlib.pyplot import *
from scipy.stats import *

standardowy = norm(loc=0.0, scale=1.0)

#Liczba próbek
N=1000
#Zadane odchylenia standardowe
StdB1 = 1
StdC1 = 2
StdB2 = 1
StdC2 = 4
#Zadane kąty obrotu
A1=0.80
A2=0.56
#Zadane przesunięcia skupień
ShiftX1 = 2
ShiftY1 = -3
ShiftX2 = -2
ShiftY2 = 3
```

```
#Przygotowanie danych
B1 = StdB1 * standardowy.rvs(N)
C1 = StdC1 * standardowy.rvs(N)
B2 = StdB2 * standardowy.rvs(N)
C2 = StdC2 * standardowy.rvs(N)
CosA = cos(A1)
SinA = sin(A1)
X1 = CosA*B1+SinA*C1+ShiftX1
Y1 = -SinA*B1+CosA*C1+ShiftY1
CosA = cos(A2)
SinA = sin(A2)
X2 = CosA*B2+SinA*C2+ShiftX2
Y2 = -SinA*B2+CosA*C2+ShiftY2
```



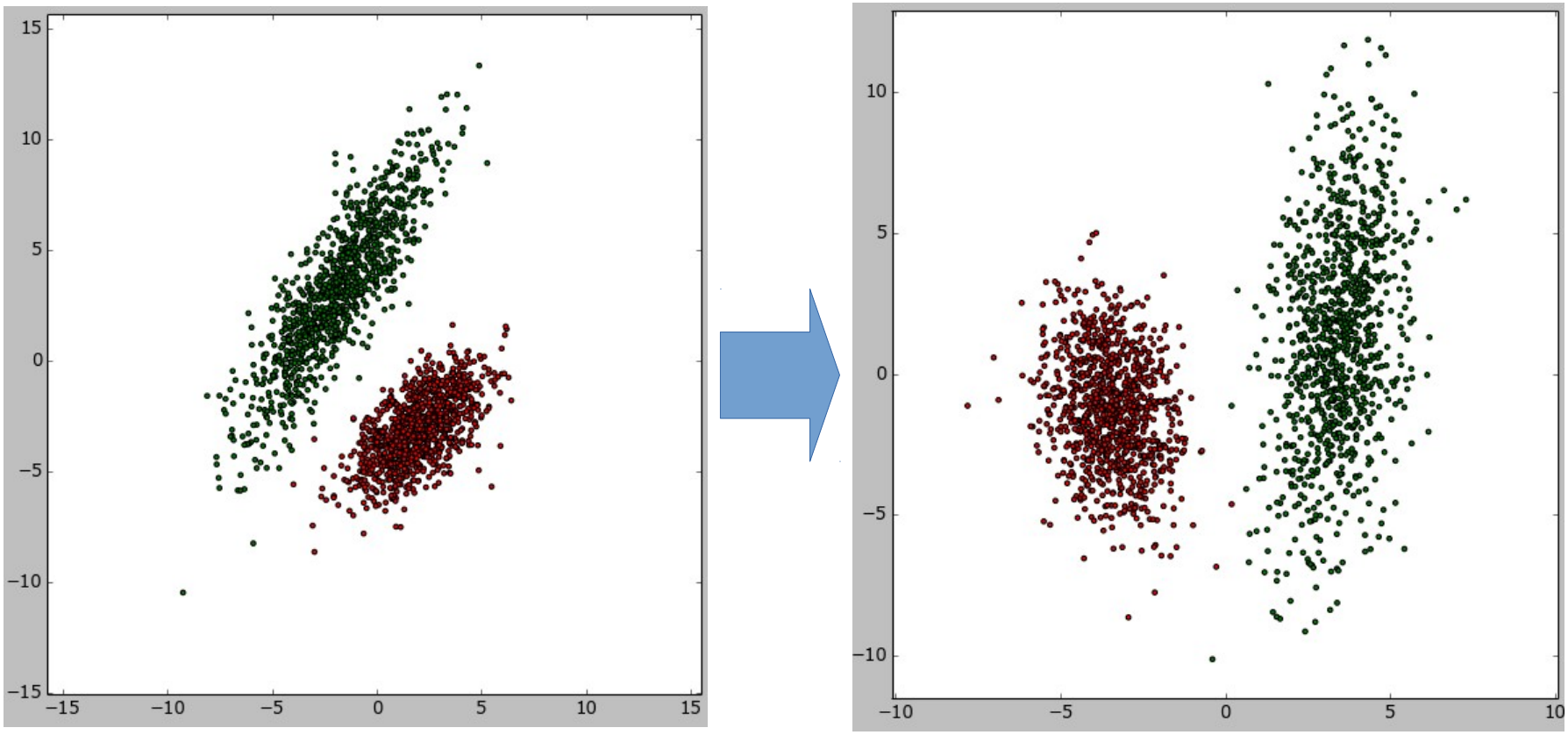
```
#Obliczanie kowariancji wewnątrzklasowej
Ew = (cov(X1,Y1) + cov(X2,Y2)) / 2
#Obliczanie kowariancji międzyklasowej
Xmu = array([mean(X1), mean(X2)])
Ymu = array([mean(Y1), mean(Y2)])
Eb = cov(Xmu, Ymu)
# E = Ew^-1 * Eb
E = linalg.inv(Ew).dot(Eb)
#Obliczenie wektorów i wartości własnych macierzy E
eigenvectors, eigenvalues, V =
linalg.svd(E, full_matrices=False)
#Wypisanie wektorów i wartości własnych macierzy E
print("Macierz wektorów własnych:")
print eigenvectors
print("Wartości własne:")
print eigenvalues
```





Macierz wektorów własnych:
[[-0.79037185 0.61262741]
[0.61262741 0.79037185]]
Wartości własne:
[2.37640962e+01   9.35185815e-16]

```
#Rzutowanie na kierunki głównych składowych LDA
XY1 = hstack((X1,Y1))
XY2 = hstack((X2,Y2))
LDX_1 = eigenvectors[0][0]*X1+eigenvectors[0][1]*Y1
LDX_2 = eigenvectors[0][0]*X2+eigenvectors[0][1]*Y2
LDY_1 = eigenvectors[1][0]*X1+eigenvectors[1][1]*Y1
LDY_2 = eigenvectors[1][0]*X2+eigenvectors[1][1]*Y2
```



Iris data set:
https://archive.ics.uci.edu/ml/datasets/Iris
https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Więcej o LDA w Pythonie:
http://sebastianraschka.com/Articles/2014_python_lda.html
http://scikit-learn.sourceforge.net/0.6/auto_examples/plot_lda_qda.html