Mašinsko učenje

profesor: Prof. Dr. Milan Milosavljević

asistent: doc. dr. Vladimir Matić
email: vmatic@singidunum.ac.rs

PITANJE 1. UVOD U IPYTHON: NUMPY, SCIPY

Uvod u Ipython:: NumPy, SciPy

Kratak uvod u NumPy - Numerical Python biblioteku

In [1]:
```
# "#" Ovo je znak za komentar.
# Uvezli smo biblioteku numpy kao instancu np koristeci komandu import

import numpy as np
```

In [2]:
```
# np sada mozemo koristiti u velikom broju primera.
# ukoliko otkucamo np. i pritisnemo TAB dobićemo listu mogucih funkcija

# kreiramo niz od 6 elemenata
a= np.array([0, 1, 2, 3, 4, 5])
```

In [3]:
```
# a * 3 u slucaju da je a numpy instanca, mnozi se svaki element sa 3
a = a * 3
```

Uvod u Ipython:: NumPy, SciPy

 # Ovo je vazan koncept u NumPy-ju (slican kao i u Matlabu). # Vectorization:: moguce je procesirati nizove na brz nacin bez koriscenja loop petlji

In [4]:
```
print (a)
```
```
[ 0  3  6  9 12 15]
```

In [5]:
```
# ovim smo generisali numericki niz od 6 elemenata.
# mozemo proveriti dimenzije niza. U ovom slucaju rezultat 1 nam
# govori da se radi o vektoru.
a.ndim
```
Out[5]:
```
1
```

In [6]:
```
# komandom reshape mozemo promeniti oblik niza. Od 1D niza (1x6), mozemo kreirati matricu, 2D niz, dimenzija (2 x 3)
a.reshape(2,3)
```
Out[6]:
```
array([[ 0,  3,  6],
       [ 9, 12, 15]])
```

Uvod u IPython:: NumPy, SciPy

```
In [8]: b = np.zeros((3,4))

In [9]: b

Out[9]: array([[ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.]])

In [10]: x = np.array([[1., 2., 3.], [4., 5., 6.]])

In [11]: print(x)

[[ 1.  2.  3.]
 [ 4.  5.  6.]]

In [12]: x * 3

Out[12]: array([[  3.,   6.,   9.],
                [ 12.,  15.,  18.]])
```

Indeksiranje

Uvod u Ipython:: NumPy, SciPy

Indeksiranje

In [13]:
```
N = 10
x  = np.arange(N)
```

In [14]:
```
# ovom naredbom type mozemo proveriti koji je tip podataka x. --> numpy.ndarray
type(x)
```

Out[14]:
```
numpy.ndarray
```

In [15]:
```
print(x)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

In [16]:
```
# Indeksiranje pocinje od 0, do N-1
x[0]
```

Out[16]:
```
0
```

In [17]:
```
print(x[1], x[5], x[-1])
```

```
(1, 5, 9)
```

In [18]:
```
print(x[2:5])
```

```
[2 3 4]
```

Uvod u Ipython:: NumPy, SciPy

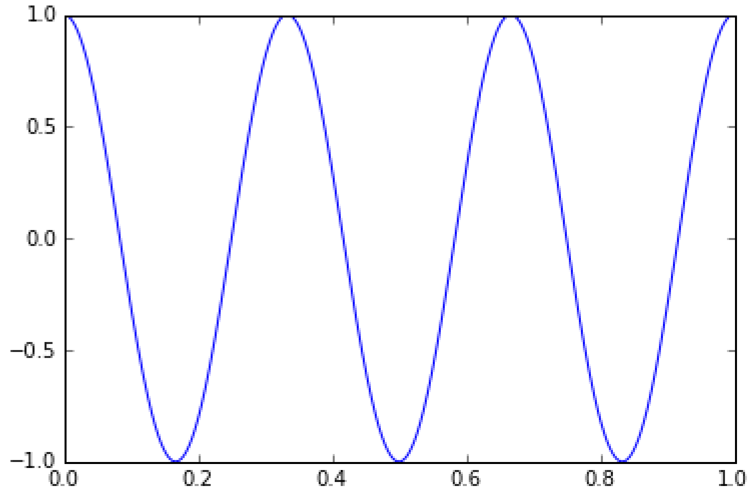
 # Funkcije za brzo procesiranje pojedinacnih elemenata niza

 # "Element-wise processing". U Matlabu se za ovo koristila tacka, npr. ".*"

```
In [19]: x = np.array(range(10))

In [20]: print (x)

[0 1 2 3 4 5 6 7 8 9]

In [21]: np.sqrt(x)

Out[21]: array([ 0.        ,  1.        ,  1.41421356,  1.73205081,  2.        ,
         2.23606798,  2.44948974,  2.64575131,  2.82842712,  3.        ])

In [22]: x

Out[22]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [23]: x = np.sqrt(x)

In [24]: t = np.arange(1000, dtype=float)

In [25]: t = t / 1000.0

In [26]:  x = np.cos(2*np.pi*3*t )
```

Uvod u Ipython:: NumPy, SciPy

```
In [27]: t2 = np.arange(0,1, 0.001)

In [28]: import matplotlib.pyplot as plt

In [29]: plt.plot(t,x)

Out[29]: [<matplotlib.lines.Line2D at 0x8f5e588>]
```



PITANJE 2. LINEARNI KLASIFIKATORI

Linearni klasifikator – primer u .ipynb

plot_lda_qda.ipynb

Edit Insert Cell Kernel Help

Code Cell Toolbar: None

In [ ]:
```
%matplotlib inline
```

Linear and Quadratic Discriminant Analysis with confidence ellipsoid

Plot the confidence ellipsoids of each class and decision boundary

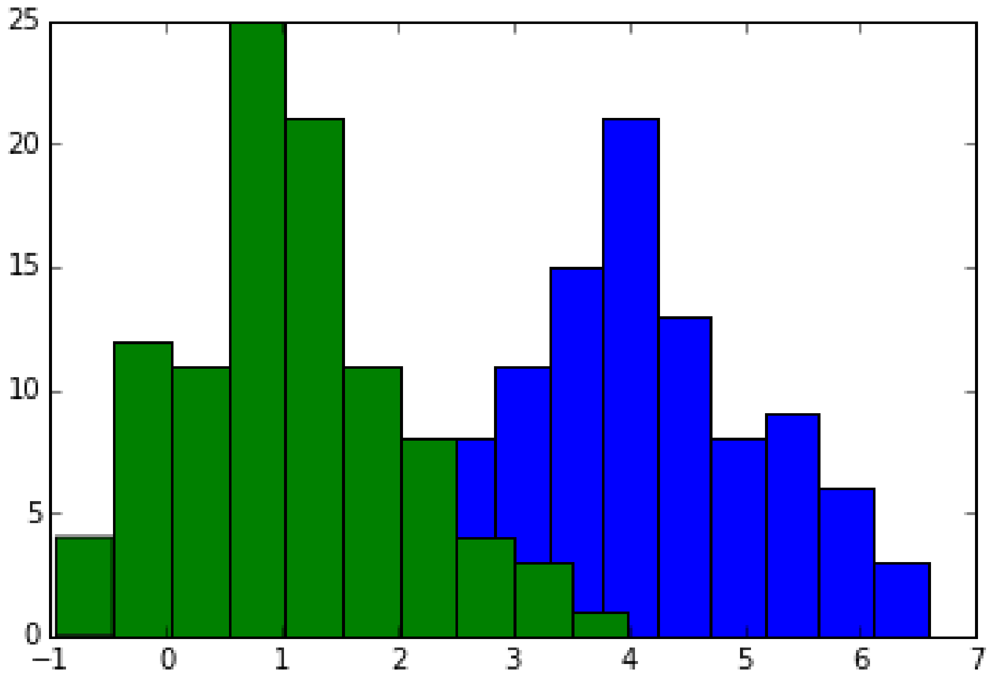
In [ ]:
```
print(__doc__)

from scipy import linalg
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import colors

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```

colormap

In [ ]:
```
cmap = colors.LinearSegmentedColormap(
    'red_blue_classes',
    {'red': [(0, 1, 1), (1, 0.7, 0.7)],
     'green': [(0, 0.7, 0.7), (1, 0.7, 0.7)],
     'blue': [(0, 0.7, 0.7), (1, 1, 1)]})
plt.cm.register_cmap(cmap=cmap)
```

generate datasets

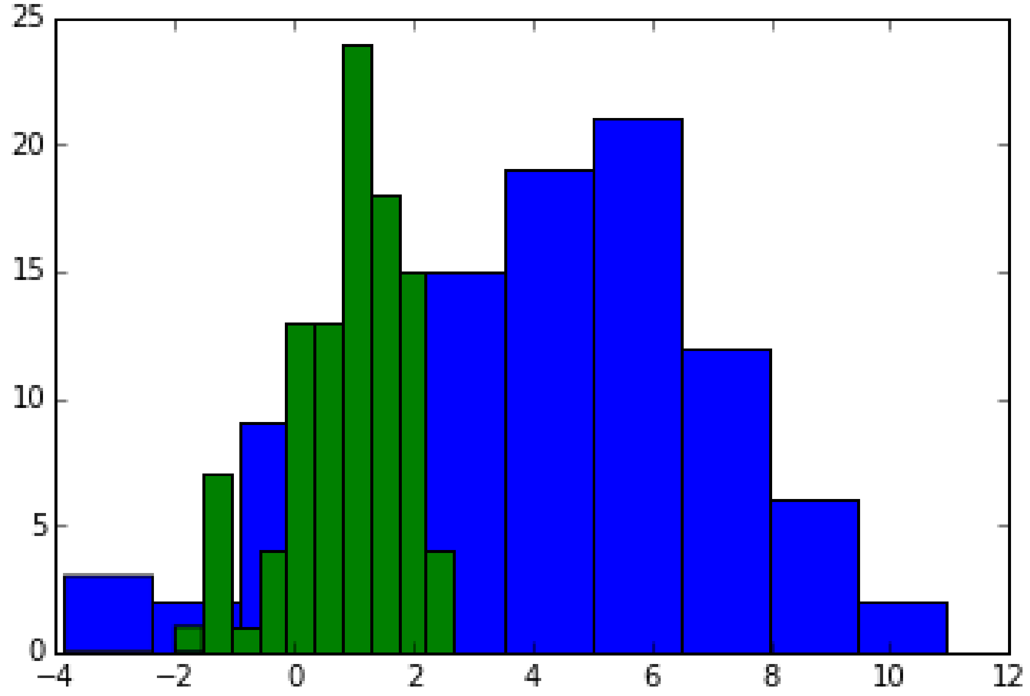
Linearni klasifikator




```
In [1]: import matplotlib.pyplot as plt

In [2]: import scipy as sp

In [36]: x = sp.random.normal(1,1,100)

In [37]: # y = sp.random.normal(4,1,100)
         y = sp.random.normal(4,3,100)

In [38]: plt.hist(y)
         plt.hist(x)
```

S obzirom da su na slici raspodele dveju rapodela N(1,1) i N(4,1) – Normalna gausova raspodela, intuitivno je jasno da granicu podele treba postaviti na 2.5, kako bi se maksimizovala razlika između centara.

Linearni klasifikator



U ovom slučaju, povećali smo varijansu “plave” raspodele. Ona je sada N(4,3) i vidi se da je raspodela šira u odnosu na prethodni slučaj. Klasifikacija sada deluje “teža“, a da bismo umanjili broj pogrešno klasifikovanih elemenata, potrebno je granicu pomeriti u levu stranu u odnosu na tačku 2.5. Primetimo da je ovde linearna klasifikacija ustvari “prag“ vrednost (engl. threshold).

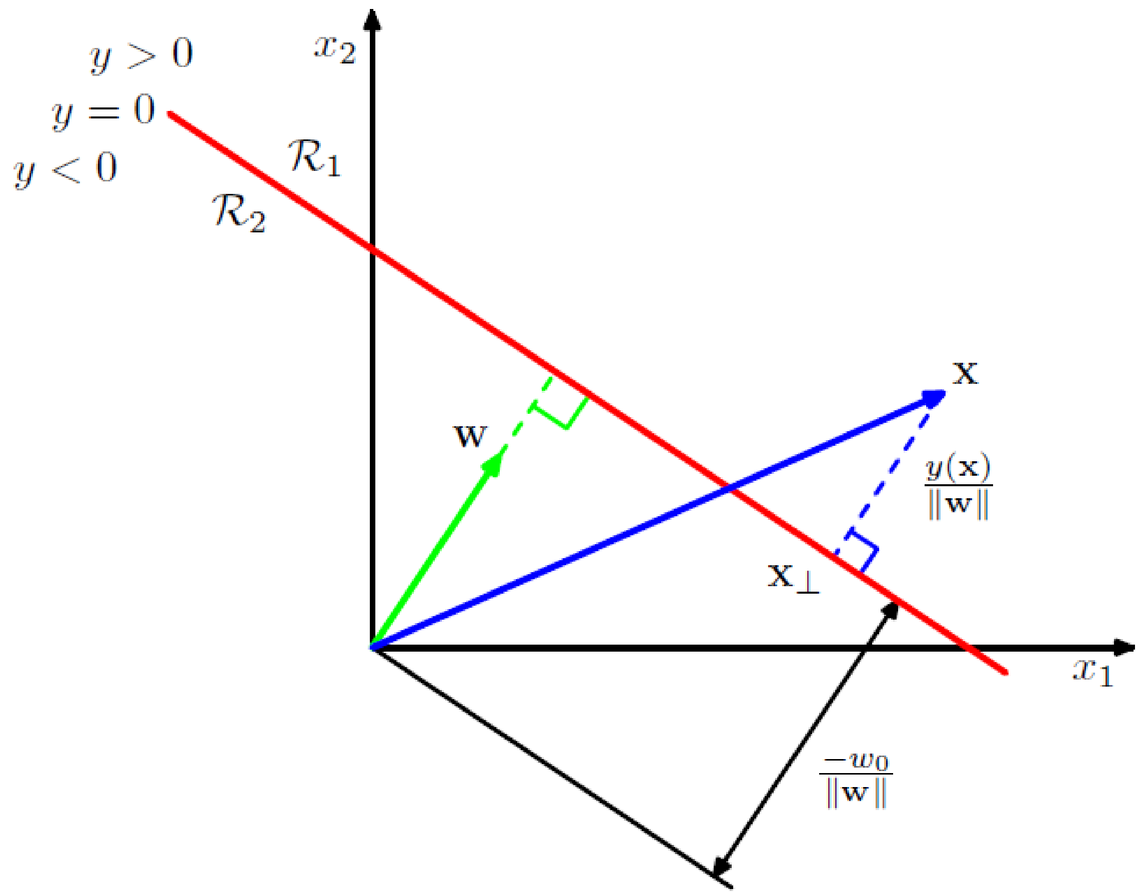
Linearni klasifikator

- Najjednostavniji matematički oblik linearnog klasifikatora dat je sledećom jednačinom koja predstavlja linearnu funkciju ulaznog vektora ***x***:

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

- **w**- se naziva težinski vektor (engl. weight vector), dok se w0 naziva pomeraj (bias) ili prag (threshold). Ulazni vektor pridružuje se klasi C1 ako je y(x)>= 0 i klasi C2 u suprotnom slučaju.

Linearni klasifikator - ilustarcija



Primer

- Neka je linearna diskriminaciona funkcija data relacijom $y(x) = 2*x1 + 3*x2-4$

- Ukoliko su koordinate $(x1,x2) = (2,1)$ ova instanca će pripadati klasi $2*2 + 3*1 -4 = 3 > 0$, dakle klasi 1.

- Slično tome, tačka sa koordinatama $(1,-1)$ će pripadati $2*1 + 3*(-1) -4 =-5 <0$, dakle klasi 2.

- Posmatrajući jednačinu

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$$

- jasno je da je zadatak projektovanja linearnog klasifikatora da se odrede koeficijenti **w** i w0.
- Jedna od metoda je korišćenje metode najmanjih kvadrata.
- Međutim, za razliku od linearne regresije, mnogo efektnija metoda za linearnu klasifikaciju je Linearna Diskriminantna Analiza (LDA). Takođe, zove se još i Fischer-ova analiza po Fischeru koji ju je definisao 1938. godine.

- Primetimo takođe, da u zavisnosti od ulazne dimenzije vektora x, zavisi i funkcija odluke.

- Kada je x dimenzije 2, funkcija odluke je prava. U slučaju da ulazni vektor x ima 3 parametra, recimo (x1, x2 i x3), funkcija odluke biće ravan.

- Dakle, dimenzija diskriminacione funkcije je za jedan manja od dimenzija ulaznih podataka.

- Razmotrimo sada opštiji klasifikacioni problem sa dve klase, gde je dimenzija vektora x jednaka 2. Neka klasi C1 pripada N1 elemenata, dok klasi C2 pripada N2 elementa. Vektori srednjih vrednosti za ove dve klase date su sledećim formulama:

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- Najjednostavniji pristup bi bio da se kao i u prethodnom slučaju maksimizuje rastojanje centara koji su projektovani na osu *y*,

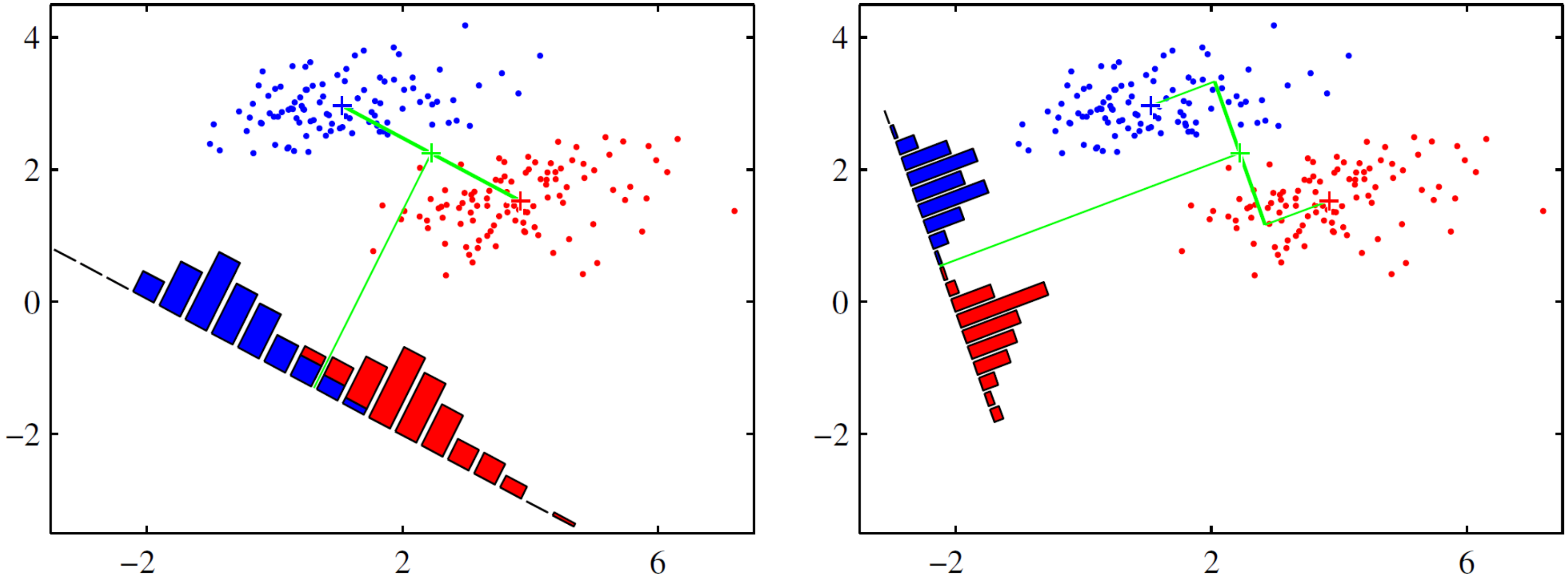
$$m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)$$

$$m_k = \mathbf{w}^{\mathrm{T}}\mathbf{m}_k$$

- >> *mk* – je dakle vrednost centra klastera projektovana na y osu.

- Uvođenjem ograničenja za parametre w, suma(w^2=1) i dalje korišćenjem Lagranževih multiplikatora za rešavanje maksimizacionog problema sa ograničenjem dobijamo:

$$\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$$



Projektovani centar klastera minus vrednost koja ce predstavljati varijansu. želimo da maksimizujemo razliku projektovanih centara.

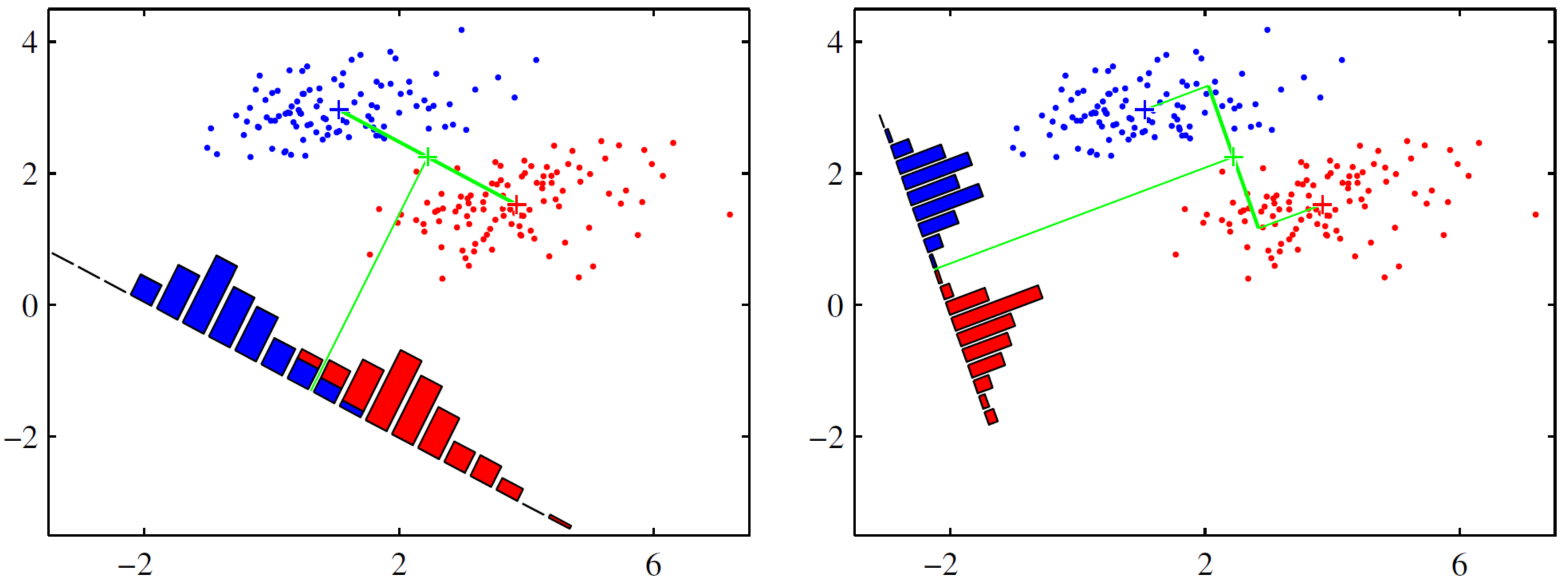
- Neka je projekcija podataka na jednodimenzionu osu *y* data sledećom formulom.

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x}.$$

- Klasifikaciju možemo izvršiti kao i u prvobitnom primeru uvođenjem praga. Ukoliko je *y* veće od neke vrednosti *w0*, element pripada prvoj klasi i obrnuto.
- Unutar klasna varijansa (rasejanje), data je onda sledećom formulom:

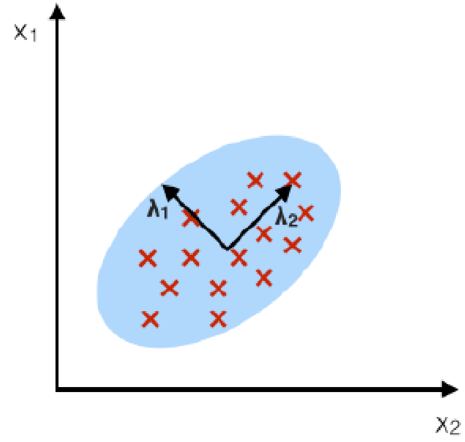
$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \qquad y_n = \mathbf{w}^{\mathrm{T}}\mathbf{x}_n \qquad s_1^2 + s_2^2.$$

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$



PCA vs. LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation