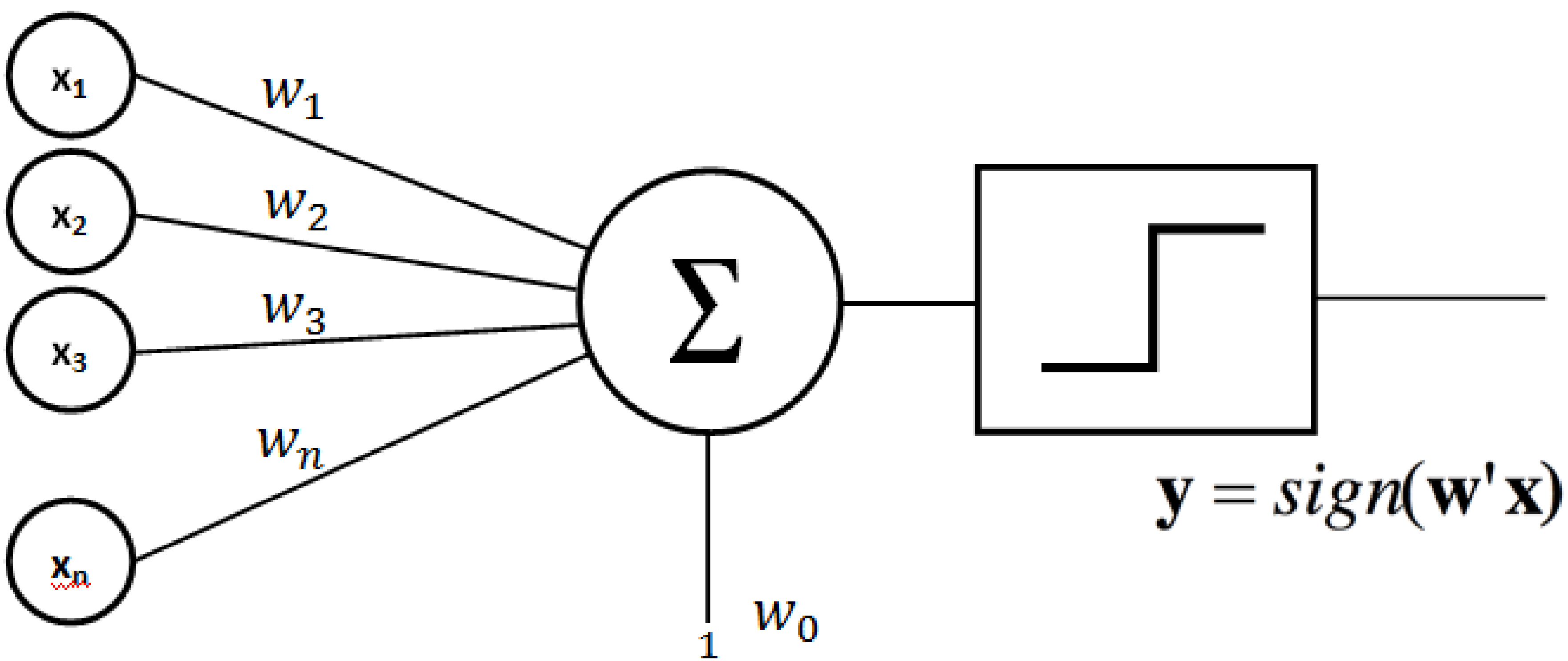
Metode kernel.
Regresia Ridge.
Mașini cu Vectori Suport.

Prof. Dr. Radu Ionescu
raducu.ionescu@gmail.com
Facultatea de Matematică și Informatică
Universitatea din București

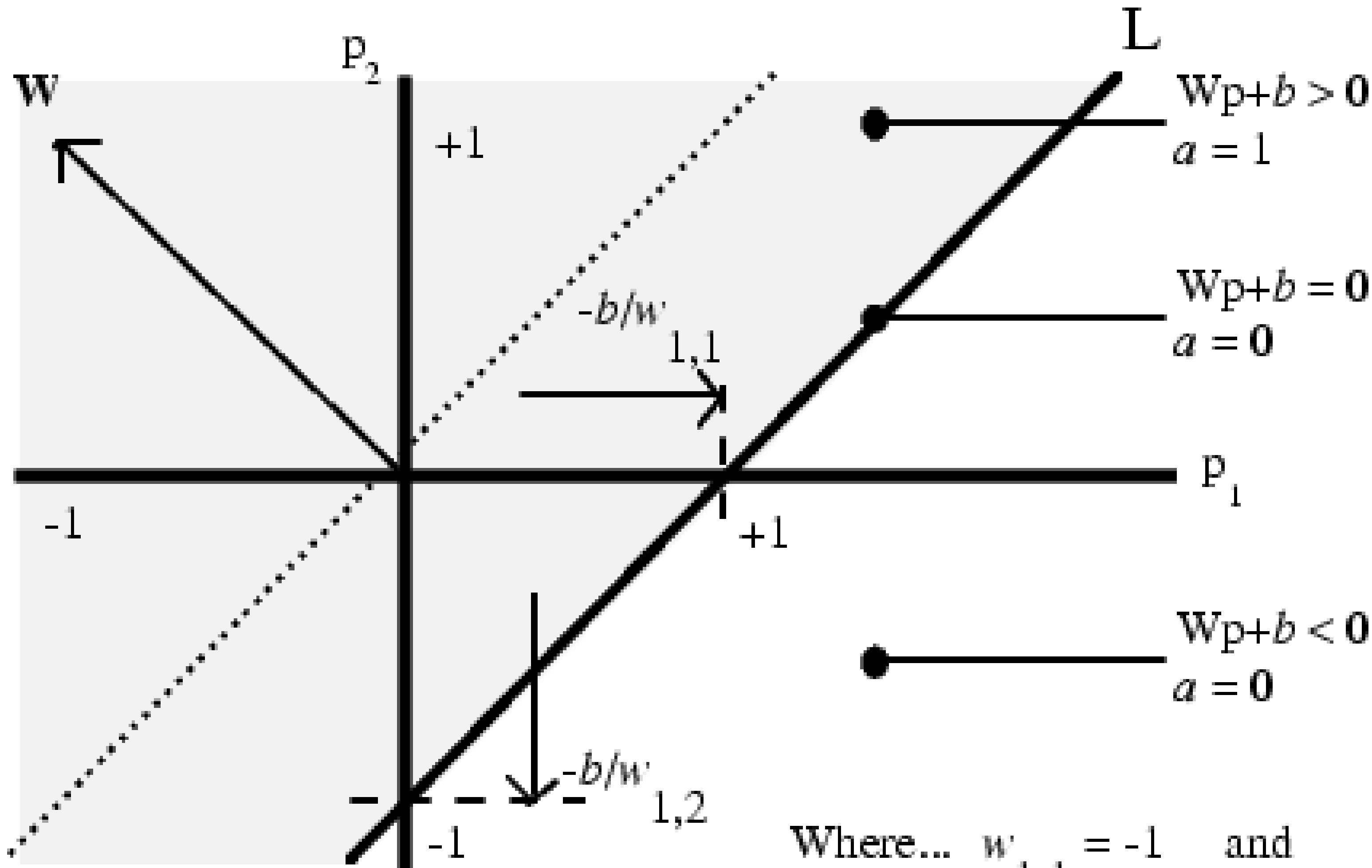
Evoluția metodelor de învățare automată

- Anii 1950: este introdus perceptronul (Rosenblatt, 1957)
- Anii 1980: este introdus algoritmul de backpropagare pentru rețele neuronale multistrat (Hinton, 1986)
- Anii 1990: apar metodele kernel (nucleu)

Perceptronul



Granița de separare liniară

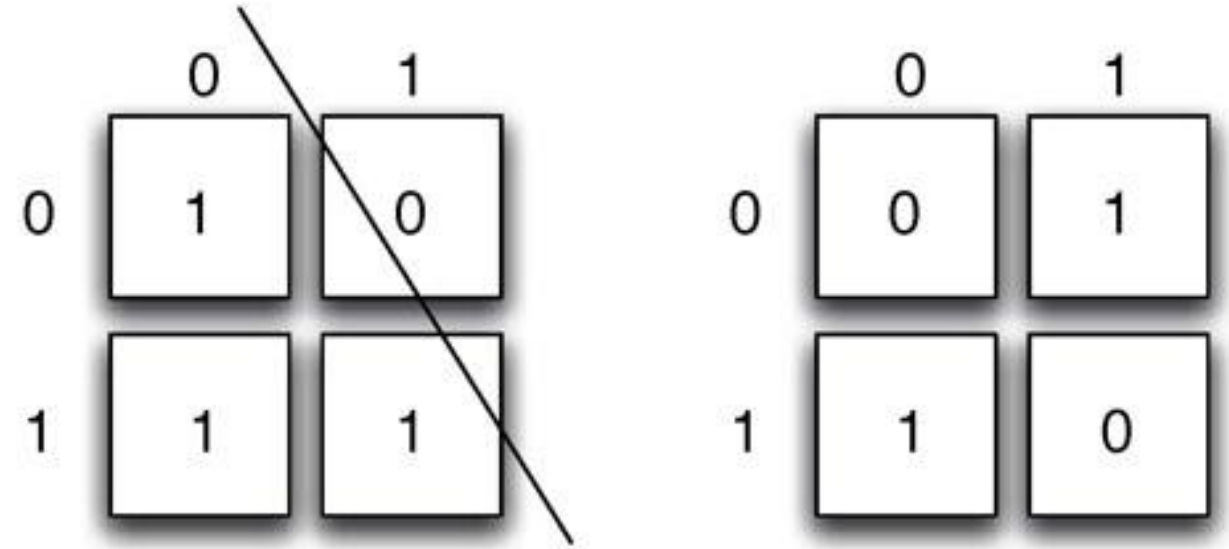


Where... $w_{1,1} = -1$ and $b = +1$

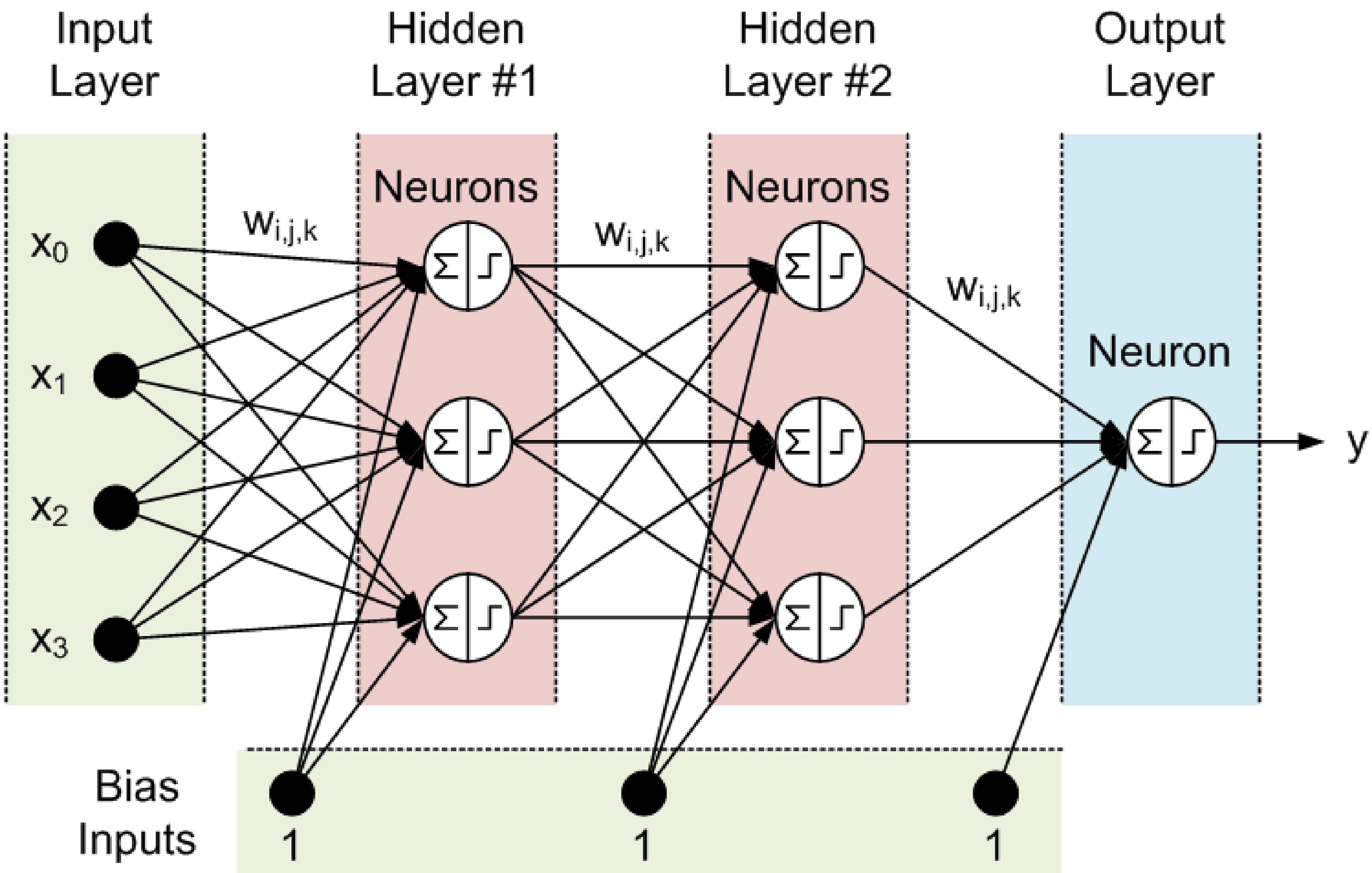
$w_{1,2} = +1$

XOR (Minsky și Papert, 1969)

- O metodă de clasificare liniară nu poate învăța funcția XOR

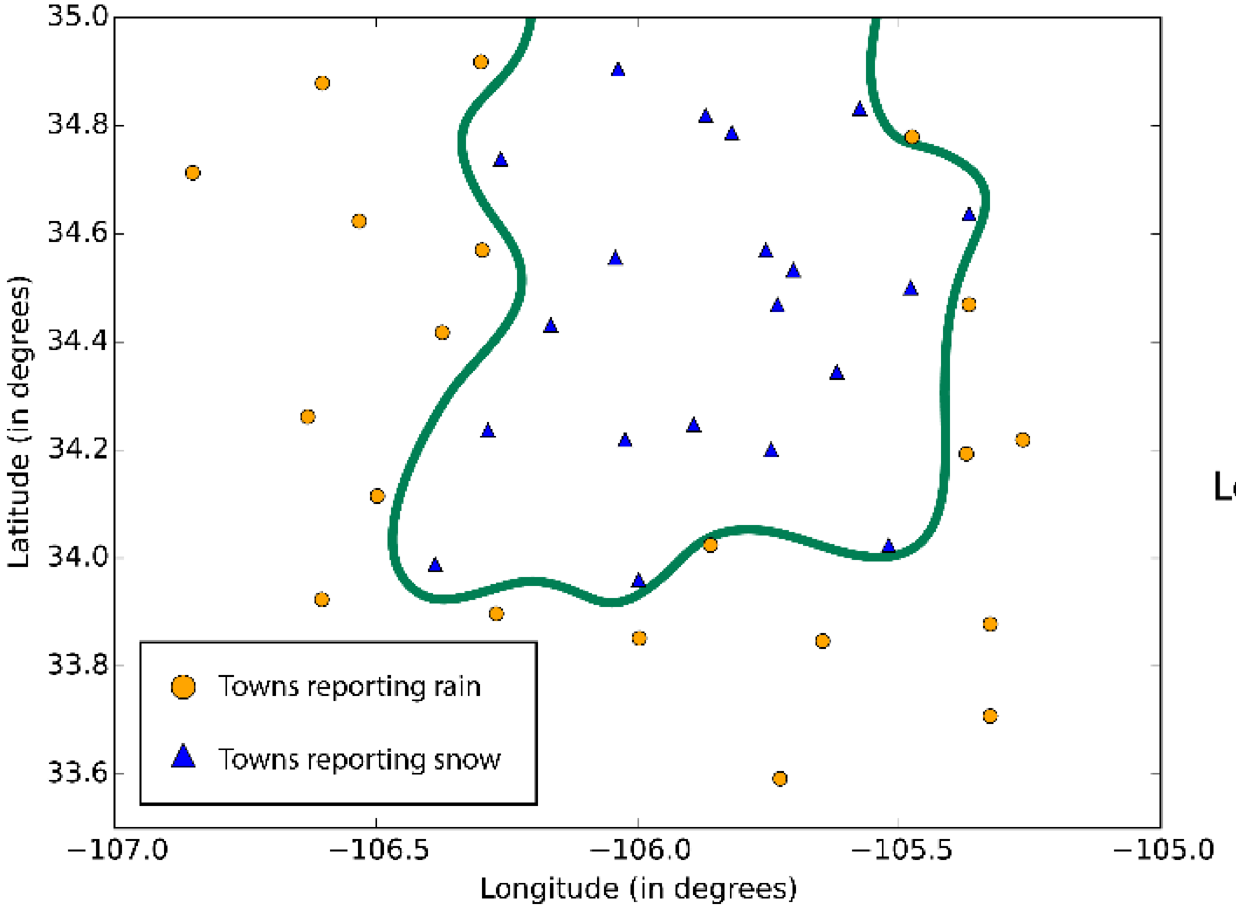


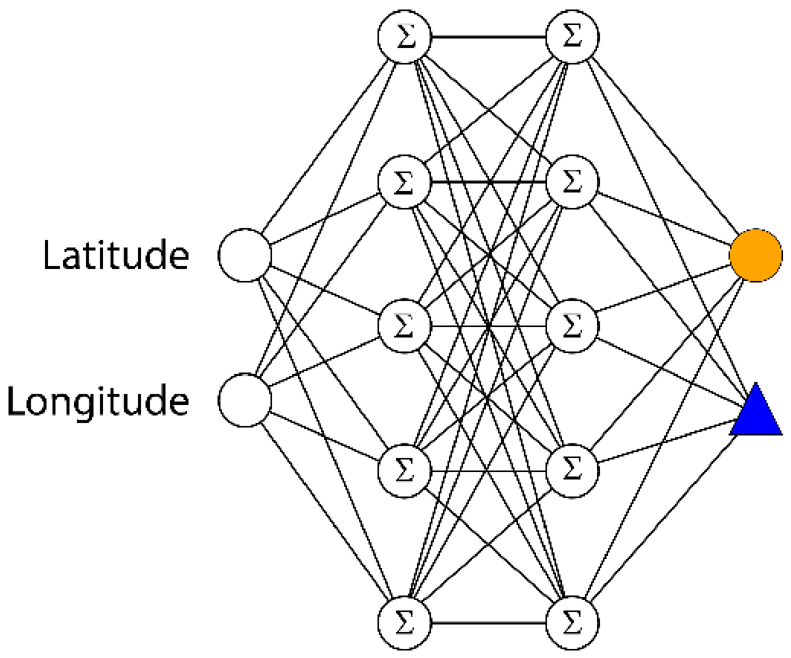
Soluția 1: Rețelele neuronale



Soluția 1: Rețelele neuronale

- Granița de decizie devine non-liniară



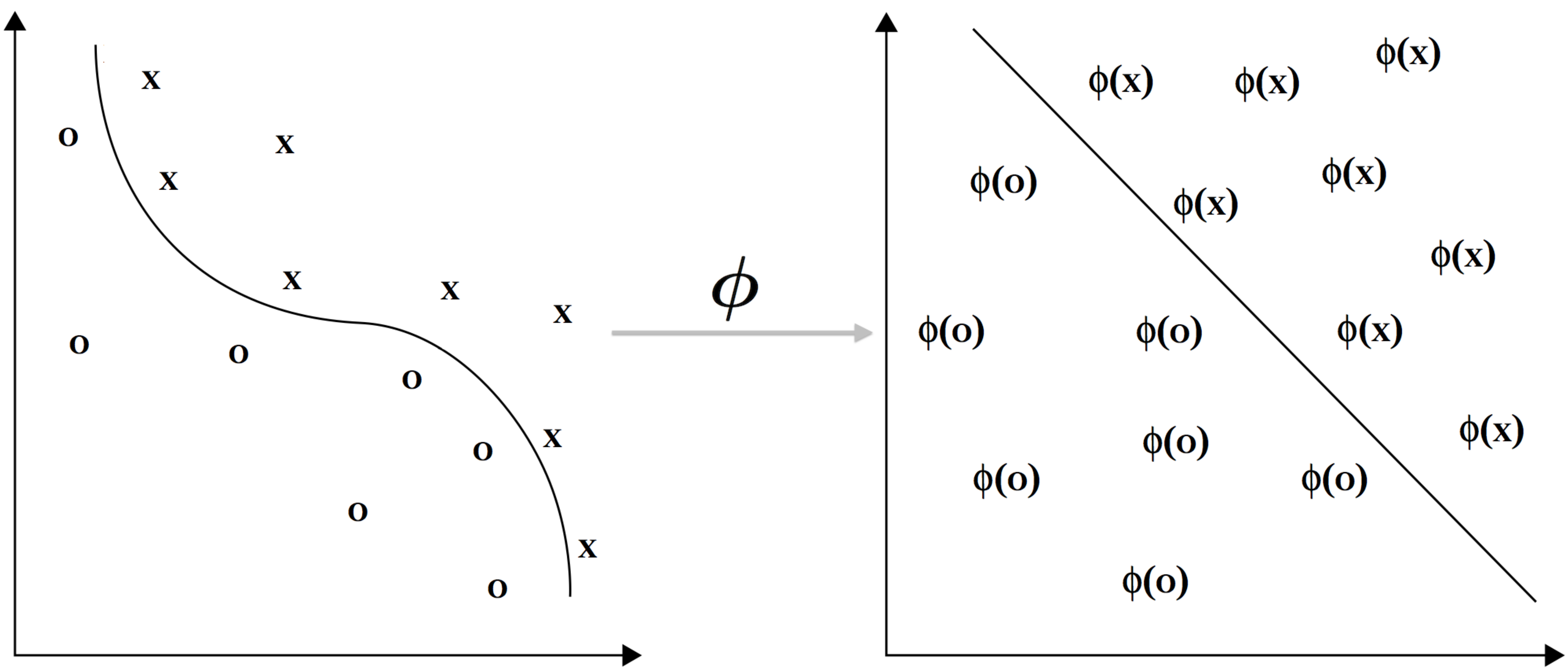


Soluția 2: Metode kernel

- Metodele kernel funcționează prin următorii doi pași:
  - 1. Datele sunt scufundate într-un spațiu (Hilbert) cu mai multe dimensiuni
  - 2. Relațiile liniare sunt căutate în acest spațiu
- Scufundarea datelor se realizează implicit, prin specificarea produsului scalar între exemple

Scufundarea datelor cu o funcție kernel

- Relațiile neliniare din spațiul original sunt transformate în relații liniare prin scufundare



Metode kernel

- Algoritmii sunt implementați (în forma duală) astfel încât coordontale punctelor scufundate nu sunt necesare, fiind suficientă specificarea produsului scalar între perechi de puncte
- “Kernel trick”: Produsul scalar poate fi înlocuit cu orice funcție de similaritate, numită și funcție kernel (funcție nucleu)

Forma primală

Features: $f_1, f_2, f_3, f_4, f_5, f_6, f_7$

Train samples: x_1, x_2, x_3, x_4

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
x_1	4	0	2	5	3	0	1
x_2	0	0	1	3	4	0	2
x_3	2	1	0	0	1	2	5
x_4	1	3	0	1	0	1	2

= X

l_1	1
l_2	1
l_3	-1
l_4	-1

= L

Linear classifier: $C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b)$ such that $\text{sign}(X * W' + b) = L$

Test samples: y_1, y_2, y_3

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
y_1	1	0	2	4	2	0	2
y_2	1	2	0	1	2	2	1
y_3	3	1	0	0	4	1	1

= Y

p_1	?
p_2	?
p_3	?

= P

Apply C to obtain predictions: $P = \text{sign}(Y * W' + b)$

Forma duală

Kernel type: **linear**

Train samples:
x_1, x_2, x_3, x_4

	x_1	x_2	x_3	x_4
x_1	55	31	16	11
x_2	31	30	14	7
x_3	16	14	35	17
x_4	11	7	17	16

$= X * X' = K_X$

l_1	1
l_2	1
l_3	-1
l_4	-1

$= L$

Linear classifier: $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$ such that $\text{sign}(K_X * \alpha' + b) = L$

Test samples:
y_1, y_2, y_3

	x_1	x_2	x_3	x_4
y_1	36	26	14	9
y_2	16	13	15	12
y_3	25	18	18	9

$= Y * X' = K_Y$

p_1	?
p_2	?
p_3	?

$= P$

Apply C to obtain predictions: $P = \text{sign}(K_Y * \alpha' + b)$

Regresia liniară

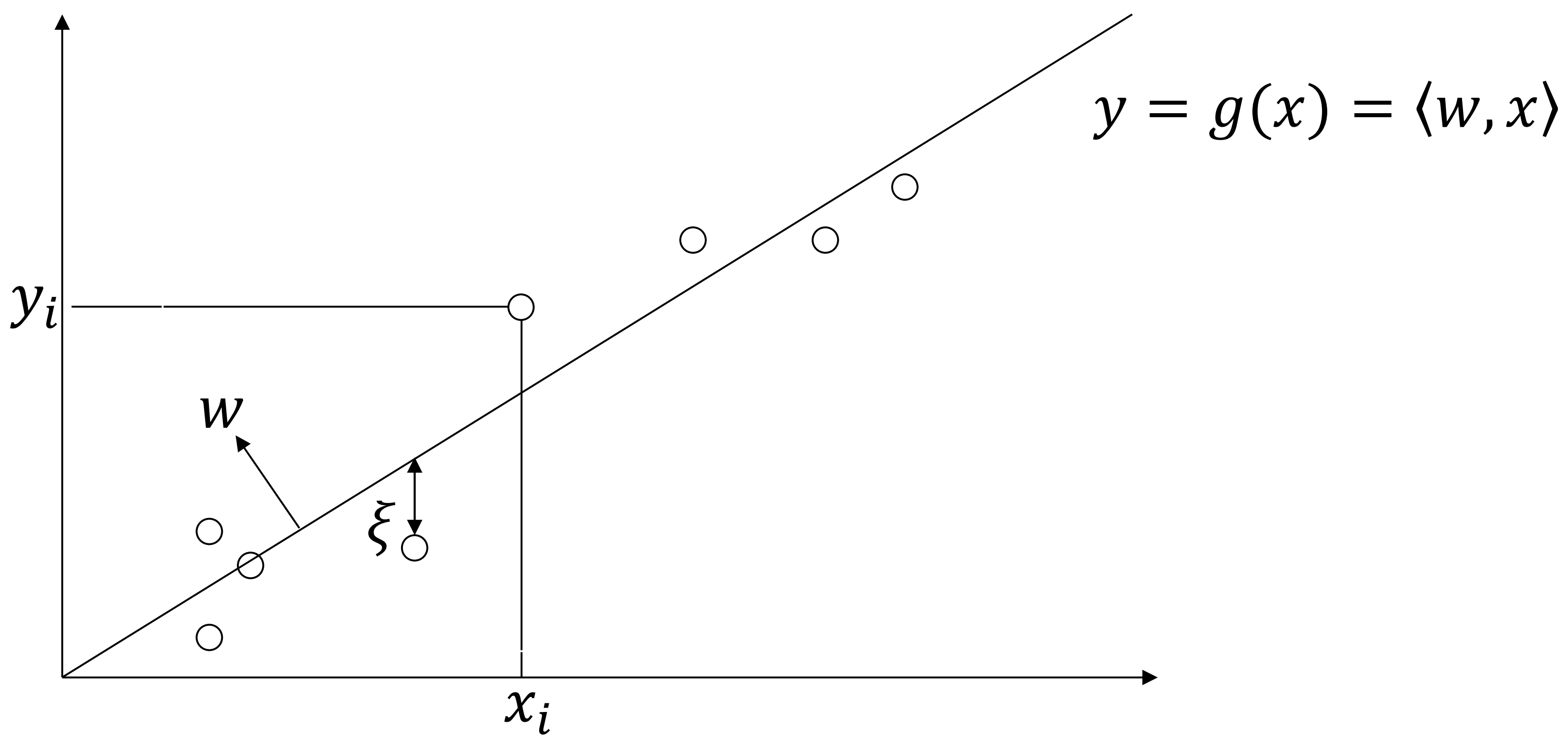
- Problema găsirii funcției g de forma:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

- care interpolează cel mai bine un set de exemple:

$$S = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

Regresia liniară



Regresia liniară

- Eroarea funcției liniare pe un exemplu particular:

$$\xi = (y - g(\mathbf{x}))$$

- Funcția de pierdere pe toate exemplele:

$$\mathcal{L}(g, S) = \mathcal{L}(\mathbf{w}, S) = \sum_{i=1}^{\ell} (y_i - g(\mathbf{x_i}))^2 =$$

$$= \sum_{i=1}^{\ell} \xi^2 = \sum_{i=1}^{\ell} \mathcal{L}(g, (\mathbf{x_i}, y_i))$$

Regresia liniară

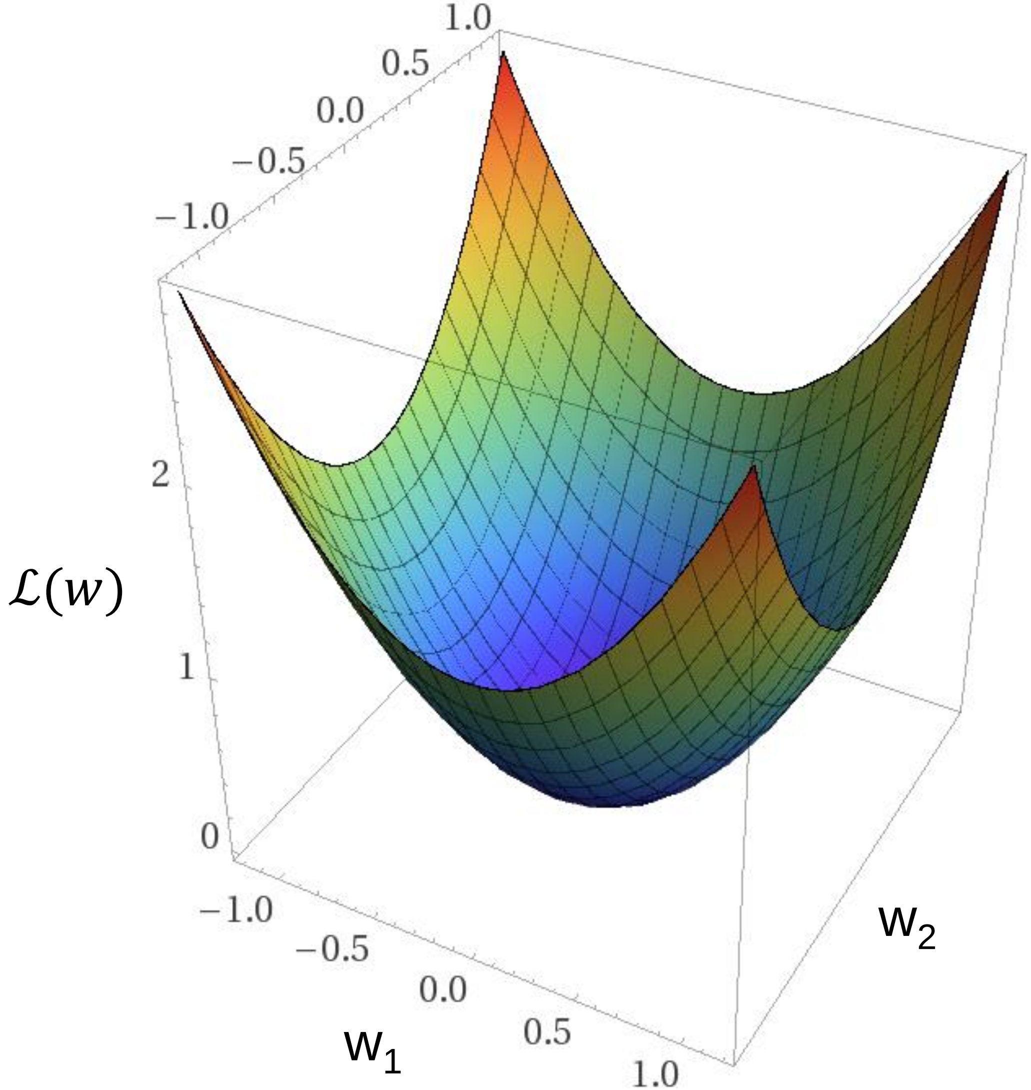
- Funcția de pierdere scrisă vectorial:

$$\boldsymbol{\xi} = \mathbf{y} - \mathbf{X}\mathbf{w}$$

$$\mathcal{L}(\mathbf{w}, S) = \|\boldsymbol{\xi}\|_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- Care este valoarea optimă pentru w?

Regresia liniară



Regresia liniară

- Valoarea optimă pentru w:

$$\frac{\partial \mathcal{L}(\mathbf{w}, S)}{\partial \mathbf{w}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{0}$$

- Ecuația normală devine:

$$\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{X}'\mathbf{y}$$

- De unde îl putem scoate pe w, dacă există inversa:

$$\mathbf{w} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

Regresia Ridge

- Dacă inversa nu există, problema este “prost-pusă” și trebuie să utilizăm regularizarea
- Criteriul de optimizare devine:

$$\min_{\mathbf{w}} \mathcal{L}_{\lambda}(\mathbf{w}, S) = \min_{\mathbf{w}}(\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell}(y_i - g(\mathbf{x_i}))^2)$$

- Iar soluția optimă pentru w este dată de:

$$\frac{\partial \mathcal{L}_{\lambda}(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial(\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell}(y_i - g(\mathbf{x_i}))^2)}{\partial \mathbf{w}} = \mathbf{0}$$

Regresia Ridge

- Soluția optimă este:

$$\frac{\partial \mathcal{L}_\lambda(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial(\lambda \|\mathbf{w}\|^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w}))}{\partial \mathbf{w}} =$$

$$= 2\lambda\mathbf{w} - 2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = \mathbf{0}$$

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_n)\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_n)^{-1}\mathbf{X}'\mathbf{y}$$

Regresia Ridge Duală

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = \lambda^{-1}(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{w}) = \lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- Dar:

$$\mathbf{w} = \mathbf{X}'\boldsymbol{\alpha}$$

- Astfel că:

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

Regresia Ridge Duală

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

$$\lambda\boldsymbol{\alpha} = (\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

$$\mathbf{X}\mathbf{X}'\boldsymbol{\alpha} + \lambda\boldsymbol{\alpha} = \mathbf{y}$$

$$(\mathbf{X}\mathbf{X}' + \lambda\mathbf{I}_\ell)\boldsymbol{\alpha} = \mathbf{y}$$

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda\mathbf{I}_\ell)^{-1}\mathbf{y}$$

- Unde:

$$\mathbf{G} = \mathbf{X}\mathbf{X}'$$

- este matricea Gram:

$$\mathbf{G}_{ij} = \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

Regresia Ridge Duală

- În forma duală, informația din exemplele de antrenare este dată prin matricea Gram ce conține produsul scalar între perechi de puncte:

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda \mathbf{I}_{\ell})^{-1} \mathbf{y}$$

- Funcția de predicție este dată de:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{\mathbf{i}=\mathbf{1}}^{\ell} \alpha_{\mathbf{i}} \mathbf{x}_i , \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i , \mathbf{x} \rangle$$

Forma primală

Features: f_1, f_2, f_3, f_4, f_5, f_6, f_7

Train samples: x_1, x_2, x_3, x_4

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
x_1	4	0	2	5	3	0	1
x_2	0	0	1	3	4	0	2
x_3	2	1	0	0	1	2	5
x_4	1	3	0	1	0	1	2

= X

l_1	1
l_2	1
l_3	-1
l_4	-1

= L

Linear classifier: C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b) such that sign(X * W' + b) = L

Test samples: y_1, y_2, y_3

	f_1	f_2	f_3	f_4	f_5	f_6	f_7
y_1	1	0	2	4	2	0	2
y_2	1	2	0	1	2	2	1
y_3	3	1	0	0	4	1	1

= Y

p_1	?
p_2	?
p_3	?

= P

Apply C to obtain predictions: P = sign(Y * W' + b)

Forma duală

Kernel type: **linear**

Train samples:
x_1, x_2, x_3, x_4

	x_1	x_2	x_3	x_4
x_1	55	31	16	11
x_2	31	30	14	7
x_3	16	14	35	17
x_4	11	7	17	16

$= X * X' = K_X$

l_1	1
l_2	1
l_3	-1
l_4	-1

$= L$

Linear classifier: $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$ such that $\text{sign}(K_X * \alpha' + b) = L$

Test samples:
y_1, y_2, y_3

	x_1	x_2	x_3	x_4
y_1	36	26	14	9
y_2	16	13	15	12
y_3	25	18	18	9

$= Y * X' = K_Y$

p_1	?
p_2	?
p_3	?

$= P$

Apply C to obtain predictions: $P = \text{sign}(K_Y * \alpha' + b)$

Regresia Ridge Kernel

- Aplicăm “kernel trick”, înlocuind produsul scalar cu o funcție kernel:

$$\langle\ \rangle \mapsto k$$

$$\mathbf{G} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_2, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_n \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & \langle \mathbf{x}_n, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{pmatrix} \mapsto \mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Regresia Ridge Kernel

- Ponderile duale se calculează astfel:

$$\boldsymbol{\alpha} = (\mathbf{G} + \lambda \mathbf{I}_\ell)^{-1}\mathbf{y} \rightarrow \boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_\ell)^{-1}\mathbf{y}$$

- Funcția de predicție devine:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{\mathbf{i}=\mathbf{1}}^{\ell} \alpha_{\mathbf{i}} \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

$$\downarrow$$

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

Regresia Ridge Kernel (Python)

```
# Parametrul de regularizare lambda:
lmb = 10 ** -6

# X_train - datele de antrenare (un exemplu pe linie)
# T_train - clasele datelor de antrenare

n = X_train.shape[0]
K = np.matmul(X_train, X_train.T)

# Antrenarea metodei:
alpha = np.matmul(np.linalg.inv(K + lmb * np.eye(n)),
        T_train)

# Prezicerea etichetelor pe datele de antrenare:
Y_train = np.matmul(K, alpha)
Y_train = np.sign(Y_train)

acc_train = (T_train == Y_train).mean())
print('Train accuracy: %.4f' % acc_train)
```

Regresia Ridge Kernel (Python)

```
# X_test - datele de testare (un exemplu pe linie)
# T_test - clasele datelor de testare

K_test = np.matmul(X_test, X_train.T)

# Prezicerea etichetelor pe datele de test:
Y_test = np.matmul(K_test, alpha)
Y_test = np.sign(Y_test)

acc_test = (T_test == Y_test).mean()
print('Test accuracy: %.4f' % acc_test)
```

Funcția kernel

- **Definiție:** O funcție kernel este o funcție

$$k : X \times X \longmapsto \mathbb{R}$$

pentru care există o funcție de scufundare din spațiul X în spațiul Hilbert F

$$\phi : x \in \mathbb{R}^m \longmapsto \phi(x) \in F$$

a.î. pentru orice $x, z \in X$

$$k(x, z) = \langle \phi(x), \phi(z) \rangle$$

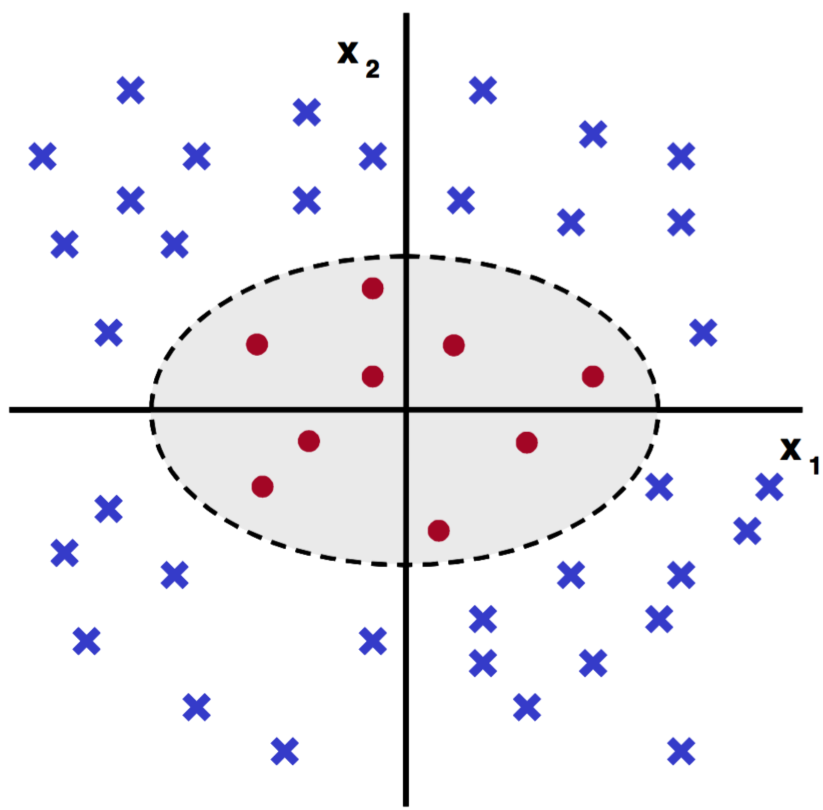
- **Teoremă:** O funcție k este funcție kernel doar dacă este finit pozitiv semi-definită

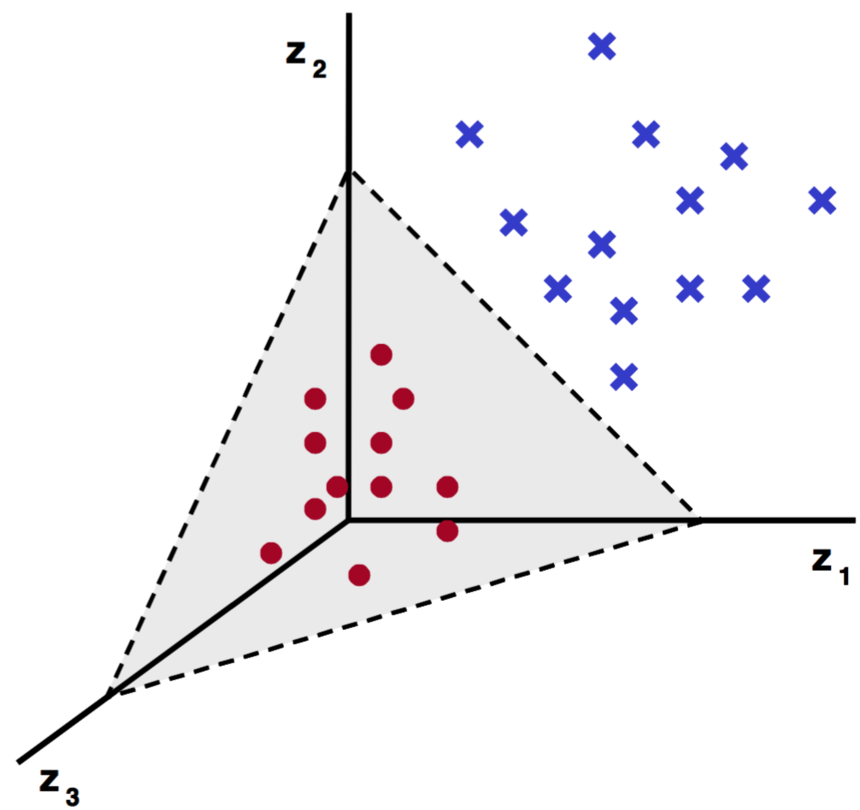
Exemple de funcții kernel

- Prin definirea explicită a funcției de scufundare

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = \left(x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)$$





Exemple de funcții kernel

- Funcția kernel din exemplul anterior:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = \left\langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \right\rangle$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1x_2z_1z_2$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = (x_1z_1 + x_2z_2)^2$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = \langle \mathbf{x}, \mathbf{z} \rangle^2$$

$$k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$$

- Aceeași funcție kernel corespunde scufundării:

$$\phi : \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, x_1x_2, x_2x_1)$$

Funcția kernel polinomială

- Pentru o constantă reală pozitivă c și un număr natural d:

$$k(\mathbf{x},\mathbf{z}) = (\langle \mathbf{x},\mathbf{z} \rangle + c)^d$$

- Constanta c permite controlul gradului de influență al polinoamelor de diverse grade

Funcția kernel Gaussiană (RBF)

- Pentru $x = (1, 2, 4, 1)$ și $z = (5, 1, 2, 3)$ din $\mathbb{R}^4$:

$$\begin{aligned}
k(x, z) &= \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \\
&= \exp\left(-\frac{\sqrt{(1-5)^2 + (2-1)^2 + (4-2)^2 + (1-3)^2}}{2 \cdot 1^2}\right) \\
&= \exp\left(-\frac{\sqrt{16+1+4+4}}{2}\right) \\
&= \exp\left(-\frac{5}{2}\right) \\
&\approx 0.0821.
\end{aligned}$$

Funcția kernel intersecție

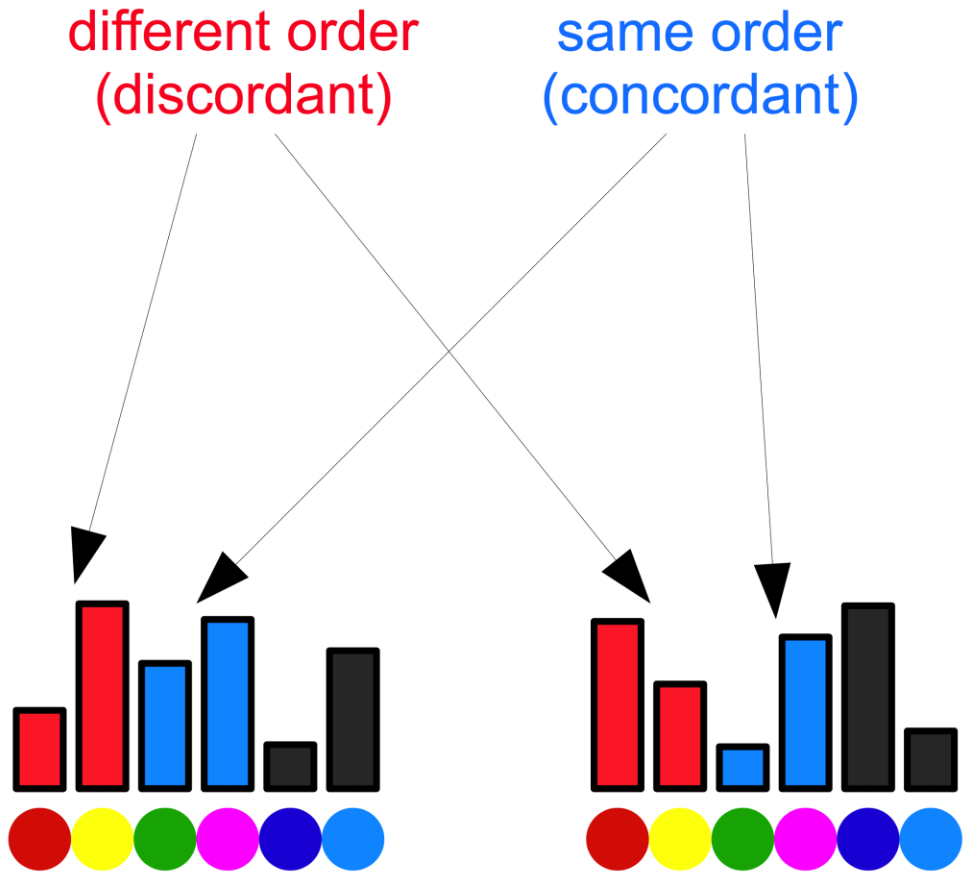
- Pentru $x = (1, 2, 4, 1)$ și $z = (5, 1, 2, 3)$ din $\mathbb{R}^4$:

$$
\begin{aligned}
k(x, z) &= \sum_i \min\{x_i, z_i\} \\
&= \min\{1, 5\} + \min\{2, 1\} + \min\{4, 2\} + \min\{1, 3\} \\
&= 1 + 1 + 2 + 1 \\
&= 5.
\end{aligned}
$$

Alte funcții kernel

- Funcția kernel Hellinger:

$$k(x, z) = \sum_i \sqrt{x_i \cdot z_i}$$



- Funcția kernel PQ:

$$k_{PQ}(X, Y) = 2(P - Q)$$

$$P = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\}|$$

$$Q = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\}|$$

String kernels

- String kernels măsoară similaritatea între perechi de șiruri de caractere, prin numărarea subsecvențelor (n-grame) de character comune dintre cele două șiruri
- Textele pot fi interpretate ca șiruri de caractere
- Avantaje:

➢ Nu trebuie să delimităm cuvintele

➢ Metoda este independentă de limbă

String kernels

- Exemplu:

Fiind date s = “pineapple pi” și t = “apple pie” peste un alfabet Σ, și lungimea n-gramelor p = 2,

construim tabele hash S and T care conțin perechi <key>:<value> de tipul

<2-gram>:<număr de apariții> în s și t:

- S = {pi:2, in:1, ne:1, ea:1, ap:1, pp:1, pl:1, le:1, e_:1, _p:1},
- T = {ap:1, pp:1, pl:1, le:1, e_:1, _p:1, pi:1, ie:1}

String kernel bazat pe biți de presență

- Funcția string kernel bazată pe biți de prezență este definită astfel:

$$k_2^{0/1}(s,t) = \sum_{v \in \Sigma^p} S^{0/1}(v) \cdot T^{0/1}(v)$$

- Exemplu (continuare):

S = {pi:2, in:1, ne:1, ea:1, ap:1, pp:1, pl:1, le:1, e_:1, _p:1},

T = {ap:1, pp:1, pl:1, le:1, e_:1, _p:1, pi:1, ie:1}

$$k_2^{0/1}(s,t) = 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1$$

$$= 1 + 1 + 1 + 1 + 1 + 1 + 1$$

$$= 7$$

De ce metode kernel?

- Obțin rezultate state-of-the-art în anumite probleme:
  - ➢ Native Language Identification [Ionescu & Popescu, BEA17]
  - ➢ Arabic Dialect Identification [Butnaru & Ionescu, VarDial18]
  - ➢ Romanian Dialect Identification [Butnaru & Ionescu, ACL19]

- Utile pentru obținerea unei reprezentări mai compacte în cazul în care:

numărul de exemple << numărul de trăsături

- Numărul de n-grame unice în setul de date TOEFL11:

4,662,520

- … versus numărul de exemple de antrenare:

11,000

De ce metode kernel?

- Generalizează mai bine decât cuvintele
- Exemple de transfer al limbii native din TOEFL11

German		**French**		**Arabic**		**Hindi**		**Spanish**		**Chinese**	
1	, that	1	indeed	1	alot	2	as compa	1	, is	2	t most
6	german	19	onnal	9	any	9	hence	2	difer	4	chin
11	. but	21	is to	13	them	16	then	13	, but	7	just
13	often	26	franc	16	thier	17	indi	15	, etc	8	still
207	special	28	to concl	19	his	21	towards	17	cesar	14	. take

Italian		**Japanese**		**Korean**		**Telugu**		**Turkish**			
1	ital	1	japan	1	korea	1	i concl	1	i agree.		
3	o beca	15	. if	24	e that	6	days	11	turk		
4	fact	19	i disa	27	. as	7	.the	21	. becau		
9	, for	27	. the	30	soci	11	where as	32	s about		
24	the life	38	. it	36	. also	13	e above	37	being		

Exemple pe cazul French→English

- {onnal}

"…many academics subjects. Additionnally, people always have a subject…"
"I would not be in control of my personnal schedule during the trip."

- {evelopp}

"…and who will have the curiosity to developp research on the disease."
"…be able to do so. Underdevelopped countries are a case in point."

- {n France}

"…studied law in both England and in France, I have had the chance…"
"Numbers have actually shown that in France the number of new cars…"

- {to conc}

"…without a tour guide. To conclude, there are several advantages…"
"…job they will enjoy. To conclude, I think that the best solution is…"

- {exemple}

"…after using them. Onother exemple is my underwear that I bougth…"
"Science is a great exemple of how successful people want to improve…"

Noi funcții kernel din combinații

- Fiind date două funcții kernel k1 și k2, o constantă reală pozitivă a, o funcție f cu valori reale și o matrice simetrică și pozitiv semi-definită B, următoarele funcții sunt kernel:

$$(i)\ k(x,z) = k_1(x,z) + k_2(x,z);$$

$$(ii)\ k(x,z) = ak_1(x,z);$$

$$(iii)\ k(x,z) = k_1(x,y) \cdot k_2(x,z);$$

$$(iv)\ k(x,z) = f(x) \cdot f(z);$$

$$(v)\ k(x,z) = x'Bz.$$

Normalizarea datelor

- În forma primală:

$$x \longmapsto \phi(x) \longmapsto \frac{\phi(x)}{\|\phi(x)\|}$$

- În forma duală:

$$\hat{k}(x_i, x_j) = \frac{k(x_i, x_j)}{\sqrt{k(x_i, x_i) \cdot k(x_j, x_j)}}$$

- Direct pe matricea kernel:

$$\hat{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} \cdot K_{jj}}}$$

Normalizarea datelor (Python)

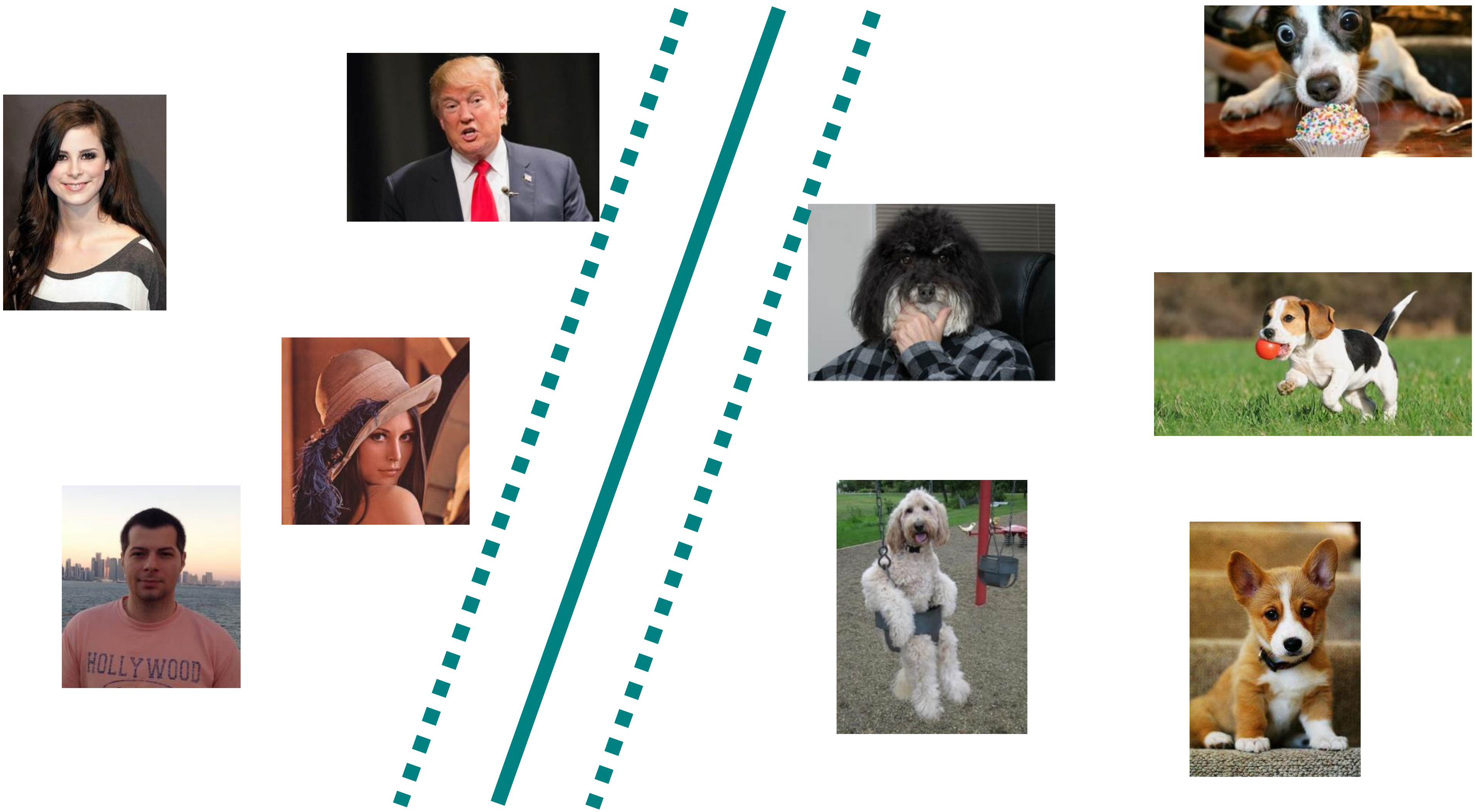
```
% X - datele (un exemplu pe linie)

% Norma L2 în forma primală:
norms = np.linalg.norm(X, axis = 1, keepdims = True)
X = X / norms

% Norma L2 în forma duală:
K = np.matmul(X, X.T)
KNorm = np.sqrt(np.diag(K))
KNorm = KNorm[np.newaxis]
K = K / np.matmul(KNorm.T, KNorm)
```

Cum separăm optim aceste exemple?

Cum separăm optim aceste exemple?

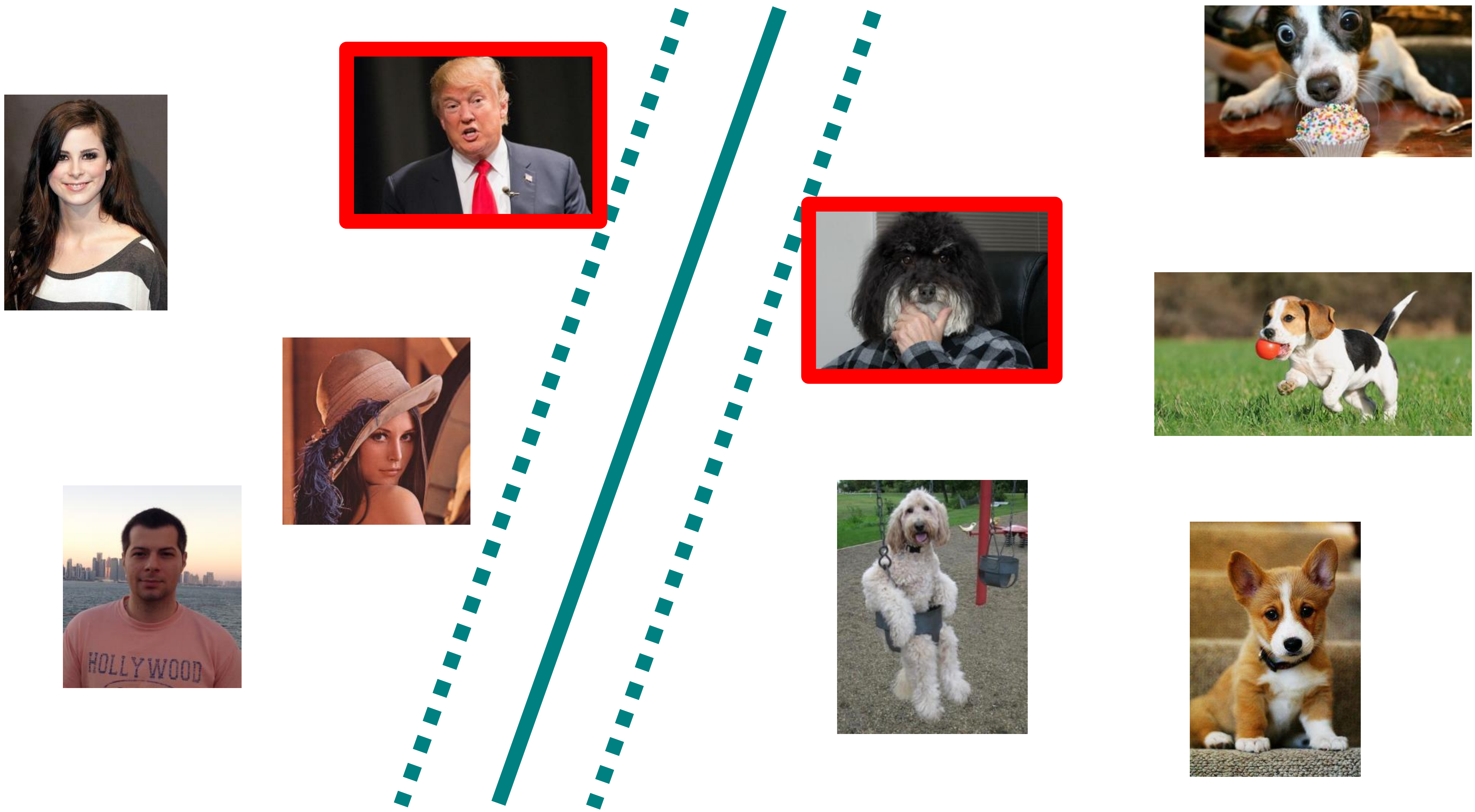
Cum separăm optim aceste exemple?

Alegem hiperplanul de margine maximă



Alegem hiperplanul de margine maximă

- Mașini cu vectori suport (SVM)



SVM (Hard Margin)

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

$$\max_{\mathbf{w}, b, \gamma} \gamma$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma$$

$$i = 1, \ldots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$

$$\Longleftrightarrow$$

$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

$$i = 1, \ldots, \ell$$

SVM (Soft Margin)

- În cazul în care exemple nu sunt liniar separabile:

$$\min_{\mathbf{w},b,\gamma,\xi} -\gamma + C\sum_{i=1}^{\ell}\xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \ldots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$

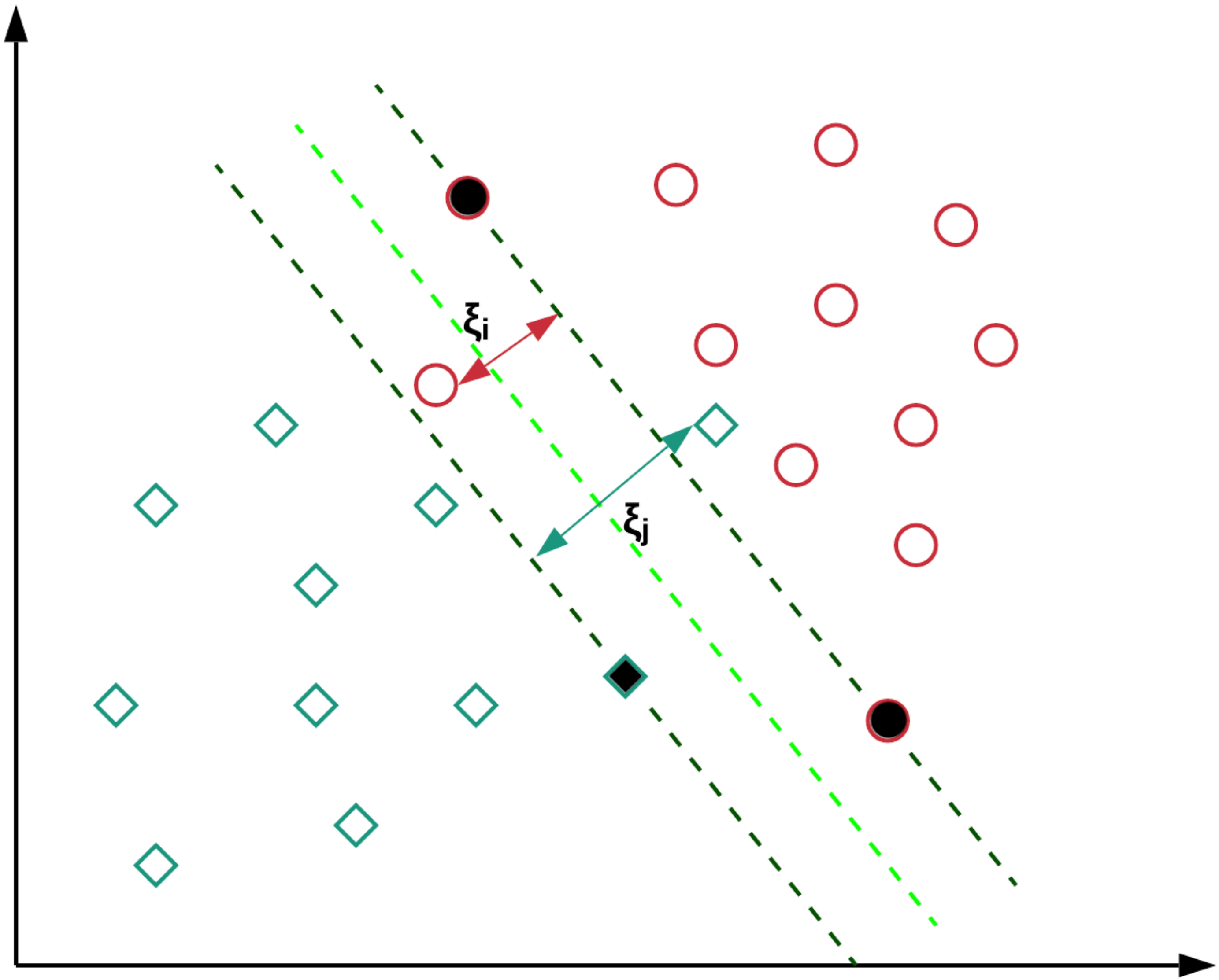
$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0 \quad i = 1, \ldots, \ell$$

SVM (Soft Margin)



SVM (Python)

- Scikit-learn:

https://scikit-learn.org/stable/modules/svm.html#svm-classification

```
from sklearn import svm

clf = svm.SVC(C = 1.0)

clf.fit(X_train, T_train)

Y_test = clf.predict(X_test)
```

- Plus mulți alți clasificatori

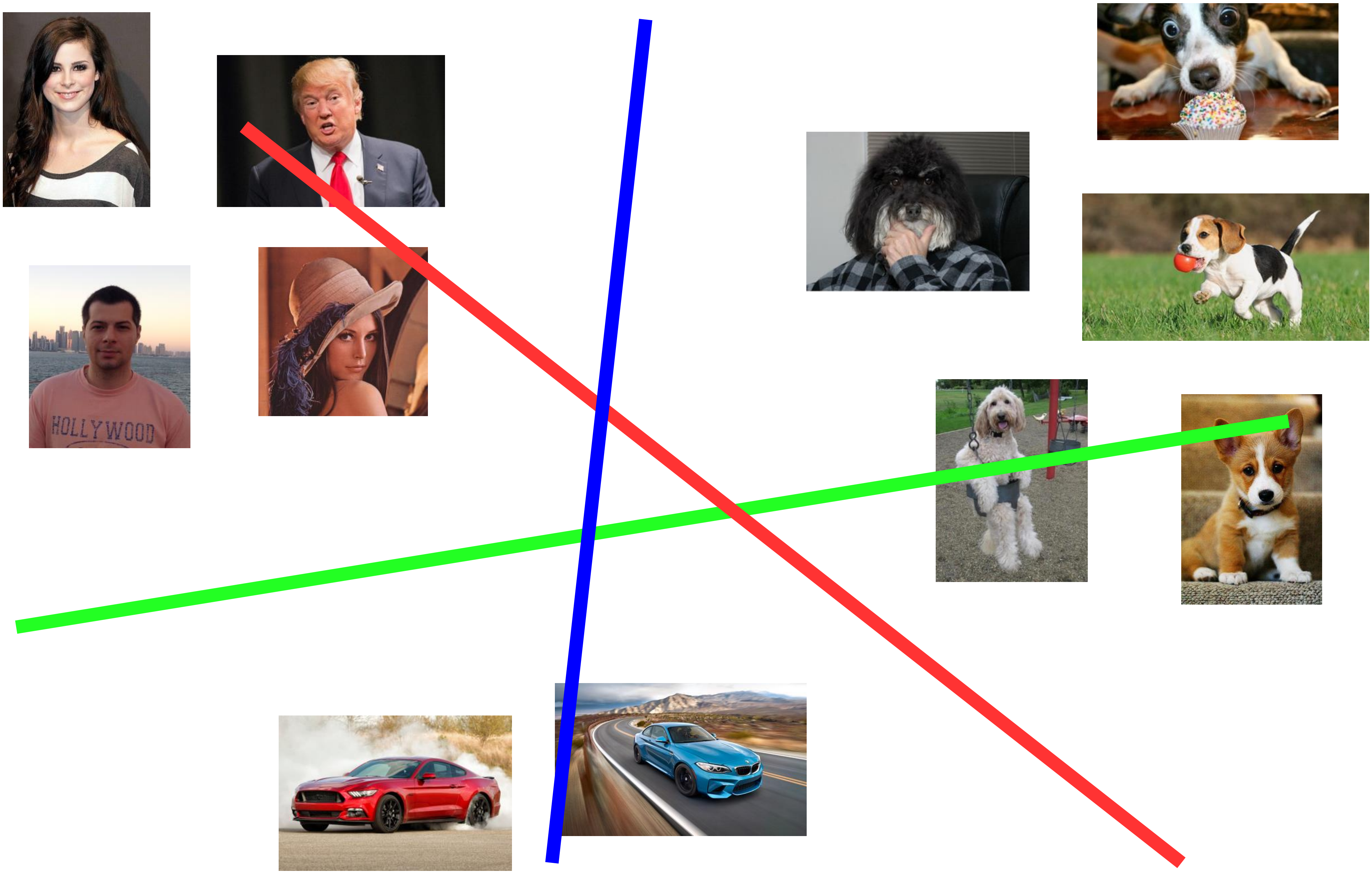
Cum rezolvăm problemele cu mai multe clase?

- Scheme de combinare a mai multor clasificatori binari:

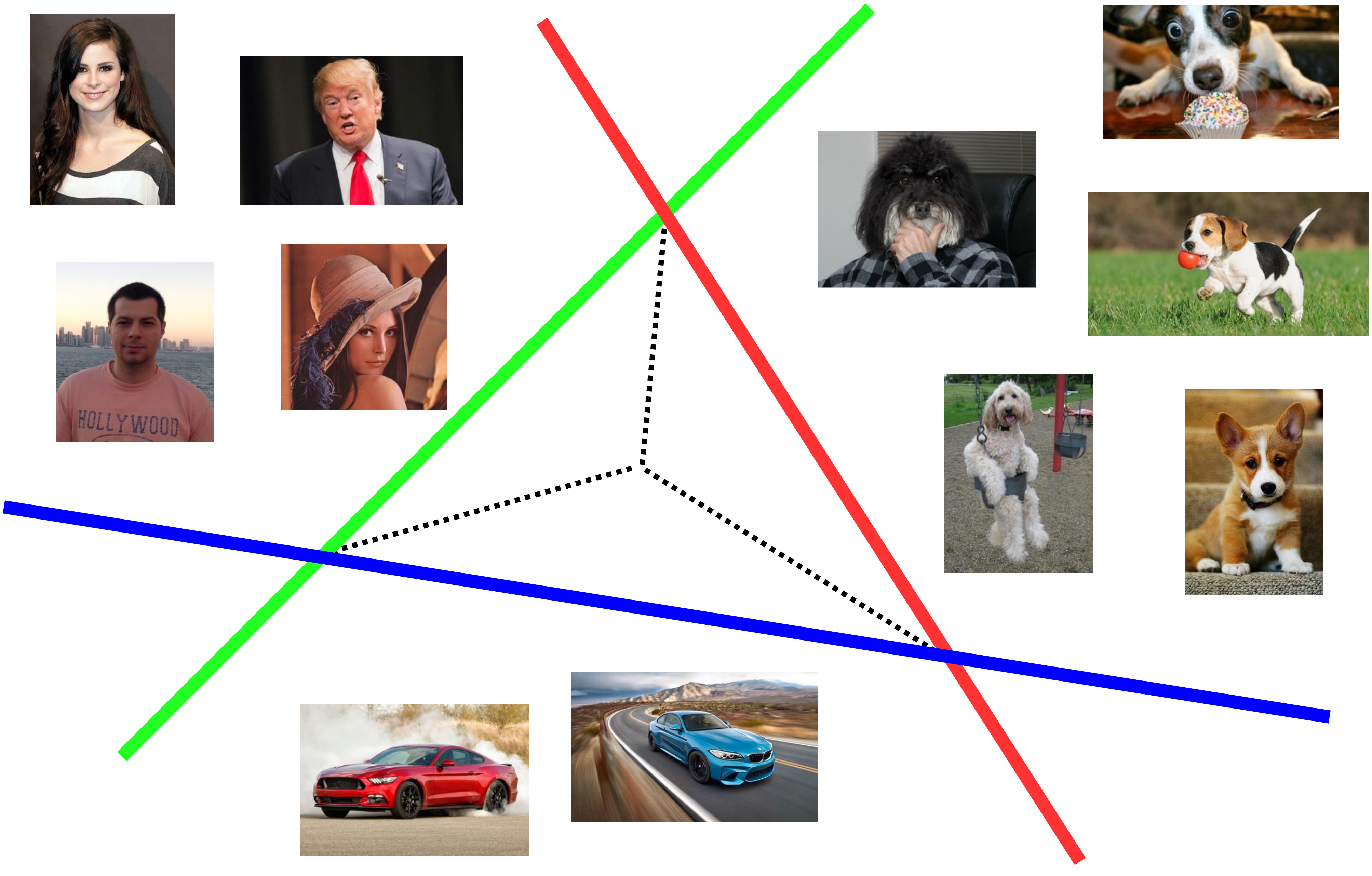
1) One-versus-one

2) One-versus-all

One-versus-one



One-versus-all



Cum rezolvăm problemele cu mai multe clase?

- Utilizarea unor metode de clasifcare capabile să rezolve direct problema:

1) Analiza liniar discriminantă (Fisher)

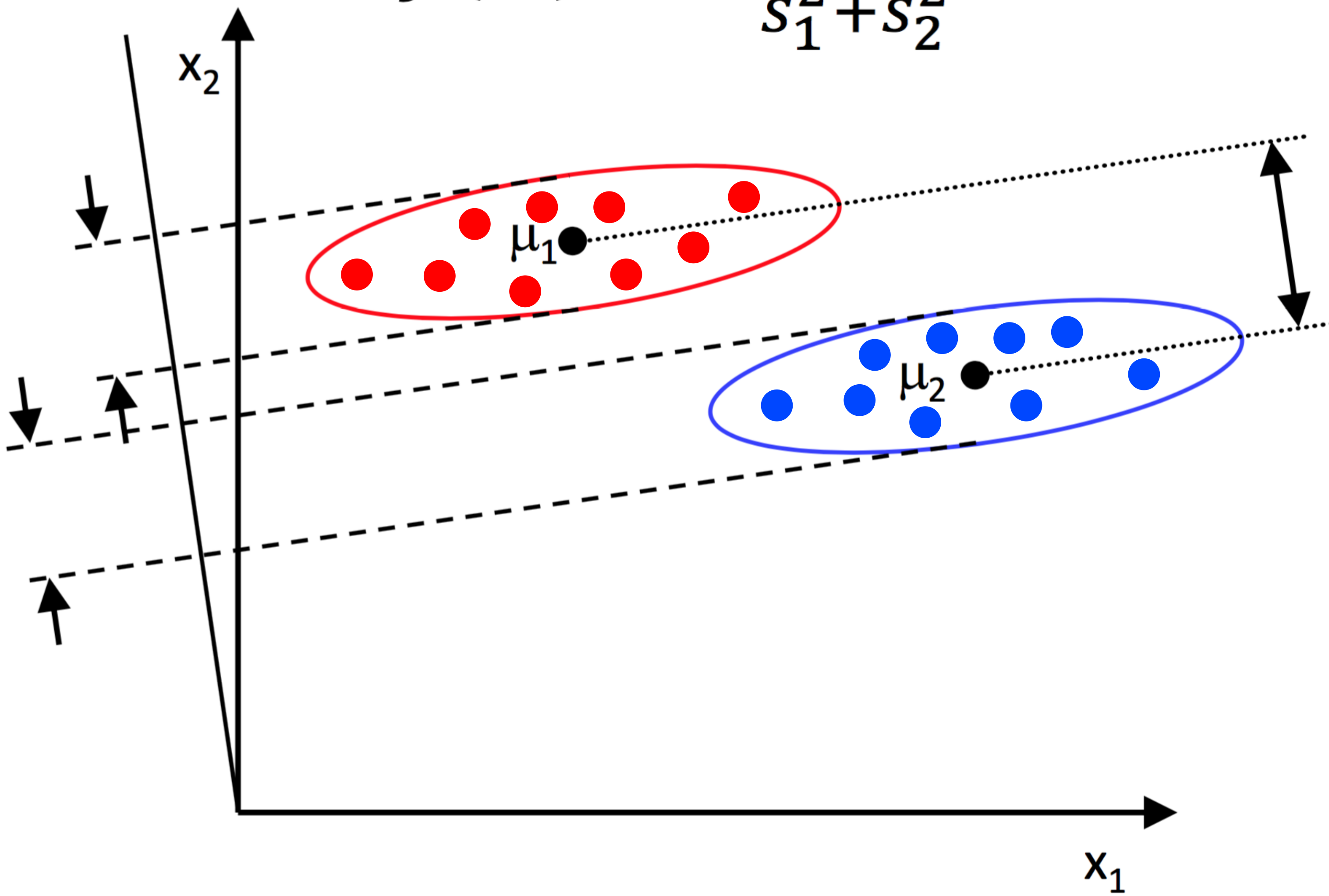
2) Rețele neuronale (cursul următor)

Analiza liniar discriminantă

- Fiecare clasă este aproximată cu o distribuție Gaussiană
- Algoritmul presupune găsirea unui hiperplan pe care se proiectează punctele a.î.:
  - distanța dintre mediile claselor este maximizată
  - dispersia fiecărei clase este minimizată

Analiza liniar discriminantă

$$J(w) = \frac{|\mu_1 - \mu_2|^2}{s_1^2 + s_2^2}$$



Analiza liniar discriminantă (Python)

- Scikit-learn:

https://scikit-learn.org/stable/modules/svm.html#svm-classification

```
from sklearn.discriminant_analysis
     import LinearDiscriminantAnalysis
clf = LinearDiscriminantAnalysis()
clf.fit(X_train, T_train)
Y_test = clf.predict(X_test)
```

Ce metodă de clasificare este cea mai bună?

- **Teorema “No free lunch”:**

Oricare doi algoritmi sunt echivalenți atunci când performanța lor este măsurată (în medie) pe toate problemele posibile

- Rezultă ca nu există nici o scurtătură în alegerea algoritmului potrivit pentru o anumită problemă
- Deobicei încercăm mai mulți algoritmi și vedem care obține rezultate mai bune

Bibliografie

Advances in Computer Vision and Pattern Recognition

A C V P R

Radu Tudor Ionescu
Marius Popescu

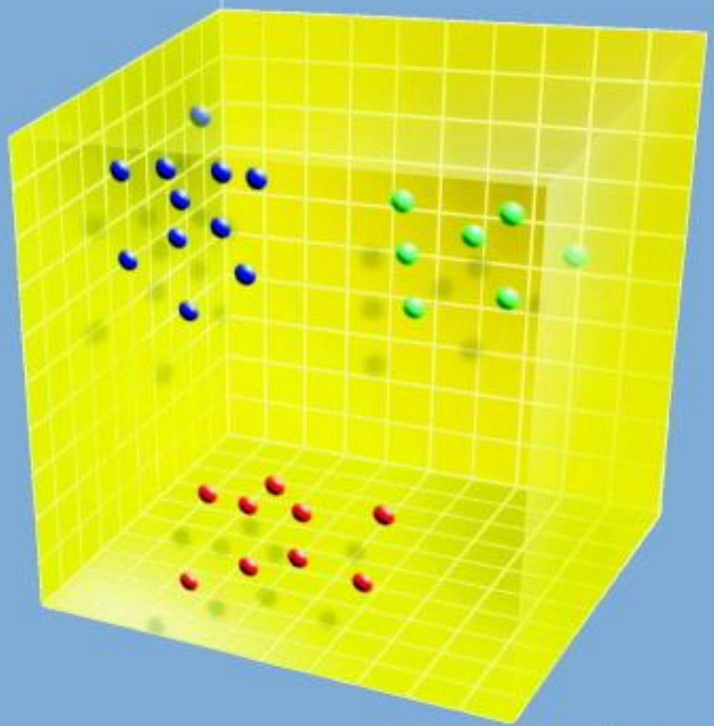
Knowledge Transfer between Computer Vision and Text Mining

Similarity-based Learning Approaches

Springer

John Shawe-Taylor
and Nello Cristianini

Kernel Methods for Pattern Analysis



CAMBRIDGE