

Calcul numeric

Metode nucleu (kernel)

Paul Irofti
Andrei Pătrașcu
Cristian Rusu

Departmentul de Informatică
Facultatea de Matematică și Informatică
Universitatea din București
Email: `prenume.nume@fmi.unibuc.ro`



- ▶ probleme de învățare automată
- ▶ modele liniare
- ▶ tipuri de modele nelineare
- ▶ metode kernel
- ▶ exemple
- ▶ aplicații



Probleme de învățare automată

- ▶ probleme supervizate
- ▶ primim un set de date cu N elemente, sub formă de perechi:

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\} \quad (1)$$

unde proprietățile $\mathbf{x}_i \in \mathbb{R}^d$ și eticheta $y_i \in \mathbb{R}$ sau $y_i \in \{0, 1\}$

- ▶ tipuri de probleme supervizate
 - ▶ regresie, eticheta este $y \in \mathbb{R}$
 - ▶ clasificare, eticheta este $y \in \{0, 1\}$
- ▶ exemplu: $\mathbf{x}_i \in \mathbb{R}^d$ poate să fie o imagine (dacă imaginea este 1024×1024 pixeli atunci $d = 1024^2$) iar $y_i \in \{\text{cat}, \text{dog}\}$
- ▶ exemplu: $\mathbf{x}_i \in \mathbb{R}^d$ sunt proprietățile unei case de vânzare iar $y_i \in \mathbb{R}_+$ este prețul



Probleme de învățare automată

- ▶ probleme nesupervizate
- ▶ primim un set de date cu N elemente:

$$\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \quad (2)$$

unde proprietățile $\mathbf{x}_i \in \mathbb{R}^d$, dar de data aceasta fără eticheta y_i

- ▶ tipuri de probleme supervizate
 - ▶ reducere dimensională
 - ▶ detectarea anomaliilor
- ▶ exemplu: $\mathbf{x}_i \in \mathbb{R}^d$ poate să fie o imagine (dacă imaginea este 1024×1024 pixeli atunci $d = 1024^2$) problema poate să fie să reducem dimensiunea datelor dacă 1024^2 este prea mare



- ▶ probleme supervizate: regresie
- ▶ primim un set de date cu N elemente, sub formă de perechi:

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$$

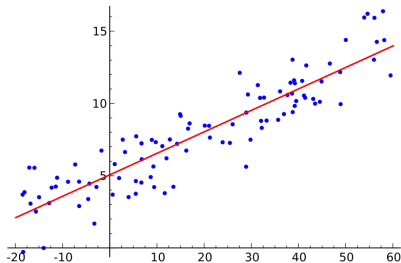
unde $\mathbf{x}_i \in \mathbb{R}^d$ și eticheta $y_i \in \mathbb{R}$

- ▶ ce înțelegem bine, sunt **problemele și modele liniare**
 - ▶ adică avem o funcție $f_w(\mathbf{x}_i) \approx y_i$ pentru toți $i = 1, \dots, N$
 - ▶ dacă f_w este liniară atunci $f_w(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$
 - ▶ adică avem o funcție care face o combinație liniară a proprietăților



Modele liniare

- ▶ ce înțelegem bine, sunt **problemele și modele liniare**
 - ▶ adică avem o funcție $f_w(\mathbf{x}_i) \approx y_i$ pentru toți $i = 1, \dots, N$
 - ▶ dacă f_w este liniară atunci $f_w(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$
 - ▶ adică avem o funcție care face o combinație liniară a proprietăților

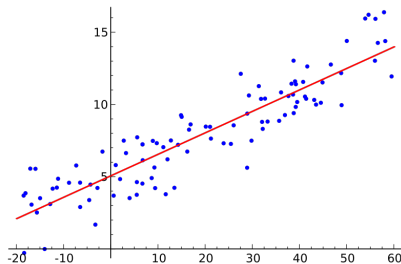


- ▶ aici $d = 1$ deci x_i sunt numere (nu vectori) – și sunt pe axa ox
- ▶ y_i este valoare pe axa oy
- ▶ exemplu: x_i poate să fie câți bani are în cont persoana i , y_i este câți bani ar mai putea împrumuta de la o bancă



Modele liniare

- ▶ ce înțelegem bine, sunt **problemele și modele liniare**
 - ▶ adică avem o funcție $f_w(\mathbf{x}_i) \approx y_i$ pentru totii $i = 1, \dots, N$
 - ▶ dacă f_w este liniară atunci $f_w(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$
 - ▶ adică avem o funcție care face o combinație liniară a proprietăților



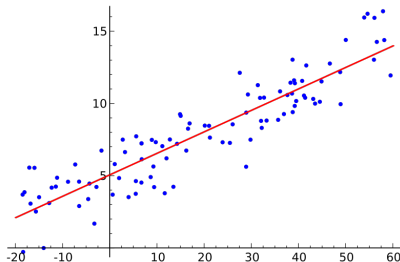
- ▶ atunci $f_w(x_i) = y_i$ și avem

$$\mathbf{w}^T \mathbf{x}_i = \begin{bmatrix} w_0 & w_1 \end{bmatrix} \begin{bmatrix} 1 \\ x_i \end{bmatrix} = w_0 + w_1 x_i = y_i$$

(3)



Modele liniare



- ▶ atunci $f_w(x_i) = y_i$ și avem

$$\mathbf{w}^T \mathbf{x}_i = \begin{bmatrix} w_0 & w_1 \end{bmatrix} \begin{bmatrix} 1 \\ x_i \end{bmatrix} = w_0 + w_1 x_i = y_i \quad (4)$$

- ▶ trebuie să îi aflăm pe w_0 și w_1
- ▶ rezolvăm:

$$\underset{\mathbf{w} \in \mathbb{R}^2}{\text{minimizează}} \sum_{i=1}^N \ell(f_w(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w}\|_2^2 \quad (5)$$



- rezolvăm:

$$\text{minimizează } \sum_{i=1}^N \ell(f_w(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w}\|_2^2 \quad (6)$$

- pierderea (*loss function*) se definește
 $\ell(f_w(\mathbf{x}_i), y_i) = (f_w(\mathbf{x}_i) - y_i)^2 = (\mathbf{w}^T \mathbf{x}_i - y_i)^2 = (w_0 + w_1 x_i - y_i)^2$
- problema devine:

$$\text{minimizează}_{\mathbf{w} \in \mathbb{R}^2} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad (7)$$

- de minim câte puncte N avem nevoie?



- rezolvăm:

$$\text{minimizează } \sum_{i=1}^N \ell(f_w(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w}\|_2^2 \quad (6)$$

- pierderea (*loss function*) se definește
 $\ell(f_w(\mathbf{x}_i), y_i) = (f_w(\mathbf{x}_i) - y_i)^2 = (\mathbf{w}^T \mathbf{x}_i - y_i)^2 = (w_0 + w_1 x_i - y_i)^2$
- problema devine:

$$\text{minimizează}_{\mathbf{w} \in \mathbb{R}^2} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad (7)$$

- de minim câte puncte N avem nevoie? $N = 2$ pentru că avem doi parametri de aflat, avem nevoie de două constrângeri



- ▶ problema devine:

$$\underset{\mathbf{w} \in \mathbb{R}^2}{\text{minimizează}} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad (8)$$

- ▶ de minim câte puncte N avem nevoie? $N = 2$ pentru că avem doi parametri de aflat, avem nevoie de două constrângeri
- ▶ de ce avem nevoie de mai multe puncte? pentru că în general $f_w(\mathbf{x}_i) = y_i + \epsilon_i$ (se adaugă zgomot, deci măsurătorile nu sunt perfecte - cu mai multe măsurători mitigăm efectul zgomotului)
- ▶ fie $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \end{bmatrix}$ și $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_N \end{bmatrix}^T$
- ▶ dacă $N = 2$ cum aflăm \mathbf{w} ?
- ▶ dacă $N > 2$ cum aflăm \mathbf{w} ?



- ▶ problema devine:

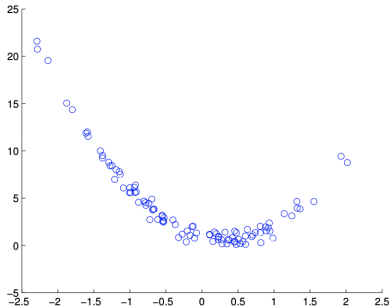
$$\underset{\mathbf{w} \in \mathbb{R}^2}{\text{minimizează}} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - y_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad (9)$$

- ▶ de minim câte puncte N avem nevoie? $N = 2$ pentru că avem doi parametri de aflat, avem nevoie de două constrângeri
- ▶ de ce avem nevoie de mai multe puncte? pentru că în general $f_w(\mathbf{x}_i) = y_i + \epsilon_i$ (se adaugă zgomot, deci măsurătorile nu sunt perfecte - cu mai multe măsurători mitigăm efectul zgomotului)

- ▶ fie $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \end{bmatrix}$ și $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_N \end{bmatrix}^T$
- ▶ dacă $N = 2$ cum aflăm \mathbf{w} ? rezolvăm un sistem $\mathbf{X}\mathbf{w} = \mathbf{y}$ și atunci $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$
- ▶ dacă $N > 2$ cum aflăm \mathbf{w} ? cele mai mici pătrate $(\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I}_2)\mathbf{w} = \mathbf{X}\mathbf{y}$ și atunci $\mathbf{w} = (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I}_2)^{-1}\mathbf{X}\mathbf{y}$



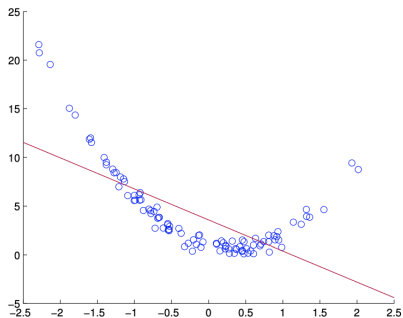
Modele liniare



► ce facem acum? cum arată soluția liniară acum?



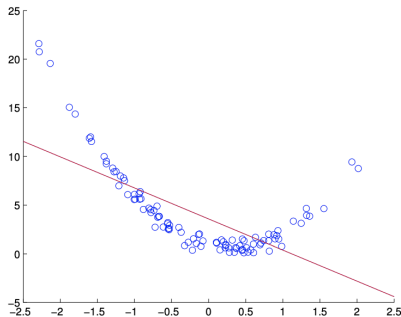
Tipuri de modele nelineare



- ▶ soluția liniară nu corespunde realității
- ▶ cum adăugăm nelineritate în problema/soluția noastră?
 - ▶ **metode nucleu (kernel)**
 - ▶ rețele neuronale



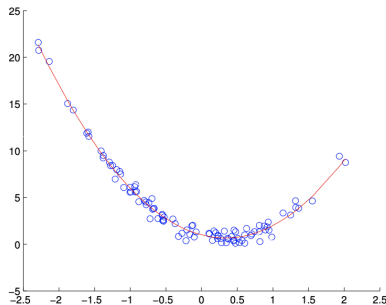
Tipuri de modele nelineare



- ▶ soluția liniară nu corespunde realității
- ▶ cum adăugăm nelineritate în problema/soluția noastră?
 - ▶ **metode nucleu (kernel)**
- ▶ avem x_i și putem genera noi:
 $x_i^2, x_i^3, \dots, \cos(x_i), \sin(x_i), \dots, e^{x_i}, \dots$ pe care le putem considera proprietăți *noi*



Tipuri de modele nelineare



- acum $f_w(x_i) = w_0 + w_1x_i + w_2x_i^2 = \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix}^T \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix}$
- observați că în necunoscutele \mathbf{w} problema este tot liniară
- nelinearitatea este în felul în care am ales indicatori noi (am ridicat la pătrat x_i)



Un exemplu

- ▶ ni se dă un set de date
 $\mathcal{S} = \{(-1.1, 2), (-0.4, 1), (0.1, 1.4), (0.8, -5)\}$
- ▶ presupunem că modelul corect este un polinom de grad 3
 $f_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3$
- ▶ atunci noi avem $f_w(-1.1) = 2$, $f_w(-0.4) = 1$ ș.a.m.d
- ▶ toate aceste relații se pot scrie astfel

$$\begin{bmatrix} 1 & -1.1 & (-1.1)^2 & (-1.1)^3 \\ 1 & -0.4 & (-0.4)^2 & (-0.4)^3 \\ 1 & 0.1 & 0.1^2 & 0.1^3 \\ 1 & 0.8 & 0.8^2 & 0.8^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1.4 \\ -5 \end{bmatrix} \quad (10)$$



Un exemplu

- ▶ toate aceste relații se pot scrie astfel

$$\begin{bmatrix} 1 & -1.1 & (-1.1)^2 & (-1.1)^3 \\ 1 & -0.4 & (-0.4)^2 & (-0.4)^3 \\ 1 & 0.1 & 0.1^2 & 0.1^3 \\ 1 & 0.8 & 0.8^2 & 0.8^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1.4 \\ -5 \end{bmatrix} \quad (11)$$

- ▶ dacă $x_i < 1$ atunci puterile tot mai mari fac ca x_i^k să fie tot mai mic, invers dacă $x_i > 1$ atunci x_i^k va fi tot mai mare
- ▶ deși matricea va fi inversabilă (dacă \mathcal{S} nu are dubluri pe punctele \mathbf{x}_i) va avea numărul de condiționare $\kappa(\mathbf{X})$ foarte mare (prost)
- ▶ când $\kappa(\mathbf{X})$ este mare atunci pățim în \mathbf{w} să avem elemente de genul 10^{10} și -10^{11} (numere mari care aproape se anulează, soluția nu este stabilă)



Un exemplu

- ▶ toate aceste relații se pot scrie astfel

$$\begin{bmatrix} 1 & -1.1 & (-1.1)^2 & (-1.1)^3 \\ 1 & -0.4 & (-0.4)^2 & (-0.4)^3 \\ 1 & 0.1 & 0.1^2 & 0.1^3 \\ 1 & 0.8 & 0.8^2 & 0.8^3 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1.4 \\ -5 \end{bmatrix} \quad (12)$$

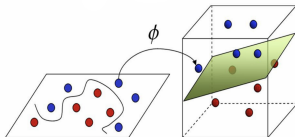
- ▶ când $\kappa(\mathbf{X})$ este mare atunci pățim în \mathbf{w} să avem elemente de genul 10^{10} și -10^{11} (numere mari care aproape se anulează, soluția nu este stabilă)
- ▶ soluția: adăugăm $+\lambda\|\mathbf{w}\|_2^2$ pentru că din cauza lui în loc să inversăm \mathbf{X} , inversăm $\mathbf{X} + \lambda\mathbf{I}$ care are κ mult mai bun (mult mai mic) și în \mathbf{w} nu mai apar numere mari cu semne opuse
- ▶ $+\lambda\|\mathbf{w}\|_2^2$ se numește **regularizare**



Metode nucleu (kernel)

- ▶ pornim de la un \mathbf{x}_i care este in dimensiune \mathbb{R}^d și ajungem într-o altă dimensiune \mathbb{R}^D cu $D \gg d$ (calculăm tot felul de funcții cu \mathbf{x}_i)

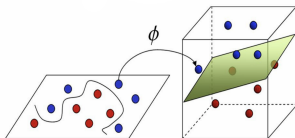
- ▶ **ideea de bază:** $\mathbf{x}_i \xrightarrow{\phi}$
- $$\begin{bmatrix} 1 \\ \mathbf{x}_i^2 \\ \mathbf{x}_i^{(1)} \times \mathbf{x}_i^{(2)} - \mathbf{x}_i^{(3)} \\ \cos(\mathbf{x}_i) \\ \vdots \\ e^{\mathbf{x}_i} \end{bmatrix}$$



Metode nucleu (kernel)

► **ideea de bază:** $\mathbf{x}_i \xrightarrow{\phi}$

$$\begin{bmatrix} 1 \\ \mathbf{x}_i^2 \\ \mathbf{x}_i^{(1)} \times \mathbf{x}_i^{(2)} - \mathbf{x}_i^{(3)} \\ \cos(\mathbf{x}_i) \\ \vdots \\ e^{\mathbf{x}_i} \end{bmatrix}$$



- inițial problema nu se putea rezolva liniar, dar când am adăugat proprietăți mai multe atunci da



Metode nucleu (kernel)

► **ideea de bază:** $\mathbf{x}_i \xrightarrow{\phi}$

$$\begin{bmatrix} 1 \\ \mathbf{x}_i^2 \\ \mathbf{x}_i^{(1)} \times \mathbf{x}_i^{(2)} - \mathbf{x}_i^{(3)} \\ \cos(\mathbf{x}_i) \\ \vdots \\ e^{\mathbf{x}_i} \end{bmatrix}$$

- dar cum alegem acest ϕ ? fiecare element din ϕ e o funcție de elemente, de unde scoatem aceste elemente (pot fi funcții polinomiale, trigonometrice sau orice altceva)? și câte sunt? adică cum alegem D ?
- am vrea cât mai multe, intuiția: cu cât sunt mai multe și mai diferite cu atât e mai mare probabilitatea să găsim o formă apropiată de conținutul datelor



Metode nucleu (kernel)

- ▶ ni se dă $\mathbf{x} \in \mathbb{R}^d$
- ▶ transformăm \mathbf{x} în $\phi(\mathbf{x}) = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \phi_D(\mathbf{x}) \end{bmatrix}$ care este un vector în \mathbb{R}^D
- ▶ atunci $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ se numește *feature map* (maparea proprietăților)
- ▶ y nu pățește nimic
- ▶ funcția noastră liniară este $f_w(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) = \sum_{j=1}^D w_j \phi_j(\mathbf{x})$ iar $\mathbf{w} \in \mathbb{R}^D$ trebuie calculat



Metode nucleu (kernel)

- ▶ înainte aveam matricea cu date $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \end{bmatrix}$ de dimensiune $d \times N$
- ▶ acum avem datele transformate $\Phi = \begin{bmatrix} \phi(\mathbf{x}_1) & \phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_N) \end{bmatrix}$ de dimensiune $D \times N$
- ▶ înainte am rezolvat $\mathbf{w} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}_d)^{-1}\mathbf{X}\mathbf{y}$
- ▶ acum rezolvăm $\mathbf{w} = (\Phi\Phi^T + \lambda\mathbf{I}_D)^{-1}\Phi\mathbf{y}$
- ▶ problema acum este de dimensiune $D \times D$ este extrem de mare (vedem imediat ca vrem D să fie enorm)
- ▶ Teorema de reprezentare: \mathbf{w} soluția sistemului cu Φ se scrie

$$\mathbf{w} = \Phi\mathbf{c} = \sum_{i=1}^N \phi(\mathbf{x}_i)c_i. \quad (13)$$

Adică soluția noastră este o combinație liniară a coloanelor din matricea de date transformate Φ (metodă neparametrică).

Iar $\mathbf{c} = (\Phi^T\Phi + \lambda\mathbf{I}_N)^{-1}\mathbf{y}$



Metode nucleu (kernel)

- Teorema de reprezentare: \mathbf{w} soluția sistemului cu Φ se scrie

$$\mathbf{w} = \Phi \mathbf{c} = \sum_{i=1}^N \phi(\mathbf{x}_i) c_i, \text{ iar } \mathbf{c} = (\Phi^T \Phi + \lambda \mathbf{I}_N)^{-1} \mathbf{y}. \quad (14)$$

- Dem. (folosim $\Phi = \mathbf{U} \Sigma \mathbf{V}^T$, $\mathbf{I}_D = \mathbf{V}^T \mathbf{V}$ și $\mathbf{I}_D = \mathbf{U} \mathbf{U}^T$):

$$\begin{aligned} \mathbf{w} &= (\Phi \Phi^T + \lambda \mathbf{I}_D)^{-1} \Phi \mathbf{y} \quad (\Phi = \mathbf{U} \Sigma \mathbf{V}^T) \\ &= (\mathbf{U} \Sigma \mathbf{V}^T \mathbf{V} \Sigma \mathbf{U}^T + \lambda \mathbf{I}_D)^{-1} \mathbf{U} \Sigma \mathbf{V}^T \mathbf{y} \quad (\mathbf{V}^T \mathbf{V} = \mathbf{I}_D) \\ &= (\mathbf{U} \Sigma^2 \mathbf{U}^T + \lambda \mathbf{I}_D)^{-1} \mathbf{U} \Sigma \mathbf{V}^T \mathbf{y} \quad (\mathbf{I}_D = \mathbf{U}^T \mathbf{U}) \\ &= (\mathbf{U} \Sigma^2 \mathbf{U}^T + \lambda \mathbf{U} \mathbf{U}^T)^{-1} \mathbf{U} \Sigma \mathbf{V}^T \mathbf{y} \quad (\mathbf{U} \text{ factor și } \mathbf{U}^T \mathbf{U} = \mathbf{I}_D) \\ &= \mathbf{U} (\Sigma^2 + \lambda \mathbf{I}_D)^{-1} \Sigma \mathbf{V}^T \mathbf{y} \quad \text{două matrice diagonale comută} \\ &= \mathbf{U} \Sigma (\Sigma^2 + \lambda \mathbf{I}_D)^{-1} \mathbf{V}^T \mathbf{y} \quad (\mathbf{I}_D = \mathbf{V}^T \mathbf{V}) \\ &= \mathbf{U} \Sigma \mathbf{V}^T \mathbf{V} (\Sigma^2 + \lambda \mathbf{I}_D)^{-1} \mathbf{V}^T \mathbf{y} \quad \text{scriem totul în funcție de } \Phi \\ &= \Phi (\Phi^T \Phi + \lambda \mathbf{I}_N)^{-1} \mathbf{y} \end{aligned}$$



Metode nucleu (kernel)

ce am făcut?

- ▶ am ajuns de la

$$\mathbf{w} = (\Phi\Phi^T + \lambda\mathbf{I}_D)^{-1}\Phi\mathbf{y} \quad (15)$$

o problemă $D \times D$

- ▶ până la

$$\mathbf{w} = \Phi(\Phi^T\Phi + \lambda\mathbf{I}_N)^{-1}\mathbf{y} \quad (16)$$

o problemă $N \times N$

- ▶ de ce ne-am chinuit atât de mult? pentru că D va fi enorm, mult mai mare și decât N (și d). **defapt** D **va fi infinit**. deci nici nu am putea să rezolvăm în forma inițială problema



Metode nucleu (kernel)

- ▶ pentru oricare $\mathbf{x} \in \mathbb{R}^d$ funcția noastră va fi

$$\begin{aligned}f_w(\mathbf{x}) &= \mathbf{w}^T \phi(\mathbf{x})^T \\&= \phi(\mathbf{x})^T \mathbf{w} \\&= \phi(\mathbf{x})^T \Phi \mathbf{c} \\&= \phi(\mathbf{x})^T \Phi (\Phi^T \Phi + \lambda \mathbf{I}_N)^{-1} \mathbf{y} \\&= \mathbf{K}_x^T (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{y} \\&= \mathbf{K}_x^T \mathbf{c}\end{aligned}\tag{17}$$

- ▶ $\mathbf{K} \in \mathbb{R}^{N \times N}$ se numește matricea de kernel, elementul de pe poziția (i, j) este $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
- ▶ \mathbf{K}_x este un vector care conține de poziția j elementul $\phi(\mathbf{x})^T \phi(\mathbf{x}_j)$
- ▶ **idee importantă:** nu avem nevoie să știm $\phi(\mathbf{x})$ avem nevoie doar să știm elemente de tipul $\phi(\mathbf{x})^T \phi(\mathbf{z})$
- ▶ **aceste se numește trucul kernel (the kernel trick)**, nu avem nevoie de niciun ϕ doar de produse scalare între ϕ -uri



Metode nucleu (kernel)

- ▶ deci nu ne trebuie ϕ , ne trebuie doar să știm cum să calculăm

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \quad (18)$$

unde \mathbf{x}_i și \mathbf{x}_j sunt oricare două puncte din setul de date care ne este dat

- ▶ ce proprietăți trebuie să aibă acest nucleu (kernel)?
 - ▶ să fie simetric $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$
 - ▶ toată matricea \mathbf{K} să fie simetrică și pozitiv definită (toate valorile proprii să fie pozitive)



Exemple (câteva)

- ▶ kernel liniar

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \quad (19)$$

- ▶ kernel polinomial

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p \quad (20)$$

- ▶ kernel Gaussian

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp \left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2} \right) \quad (21)$$

- ▶ kernel Gaussian RBF

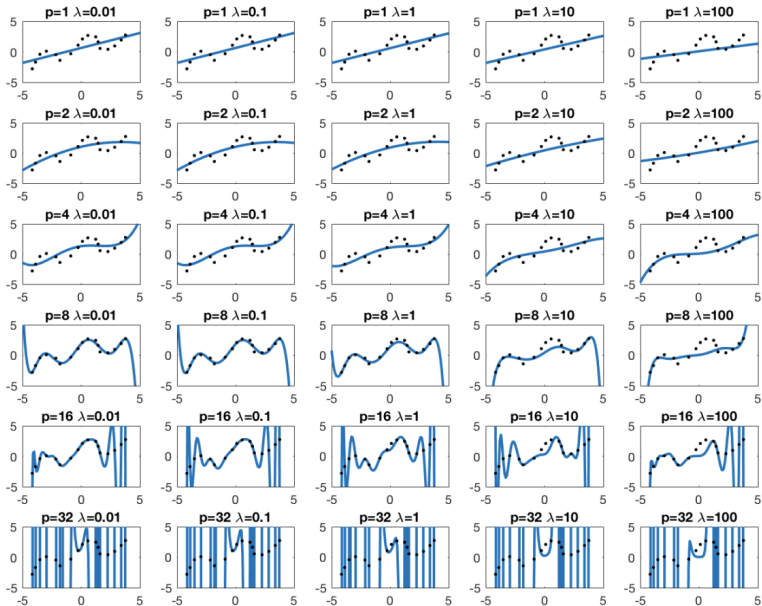
$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp \left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \right) \quad (22)$$

- ▶ kernel sigmoid

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \mathbf{x}_j + \gamma) \quad (23)$$



Exemplu (kernel polinomial)



Cum folosim metode nucleu?

Antrenare:

- ▶ Avem un set de date $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ și alegem un kernel $k(\mathbf{x}_i, \mathbf{x}_j)$.
- ▶ calculăm matricea kernel \mathbf{K} de dimensiune $N \times N$.
- ▶ rezolvăm sistemul $\mathbf{c} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1} \mathbf{y}$, iar $\lambda \in \mathbf{R}_+$ îl alegem noi.

Testare:

- ▶ avem un nou punct \mathbf{x}_{N+1} .
- ▶ calculăm $k(\mathbf{x}_{N+1}, \mathbf{x}_i)$ pentru $i = 1, \dots, N$ (acesta este vectorul notat cu \mathbf{K}_x în curs).
- ▶ rezultatul este $f(\mathbf{x}_{N+1}) = \mathbf{K}_x^T \mathbf{c}$.



- ▶ aplicații ale metodelor de kernel vom vedea la seminar
- ▶ când folosiți multe biblioteci de python veți observa că vă lasă să folosiți un kernel (implicit sau standard este kernel-ul liniar)
- ▶ pe lângă rețele neuronale, sunt principala metodă de a adăuga nelinearități în problemele noastre
- ▶ ce să țineți minte: problema inițială este *ridicată* într-un spațiu dimensional mai mare unde clasificarea liniară este (câteodată, nu mereu) capabilă să rezolve problema
- ▶ tot ce am discutat se află pe un fundament teoretic solid bazat pe spații Hilbert (vectori și matrice infinit dimensionale)

