Relaţie cu valorile proprii

Dacă presupunem disponibilă DVS a matricii A, atunci

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma \underbrace{U^TU}_{I_n}\Sigma V^T$$

$$= V\begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{bmatrix} V^T$$

Se identifică:

$$V^T = P^T$$

$$\sigma_i^2 = \lambda_i$$

Vectorii proprii ai lui A^TA compun vectorii singulari la dreapta a lui A!

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 5 & -4 \\ -4 & 5 \end{bmatrix}$$

urma este suma elementelor de pe diagonală și este egală cu suma valorilor proprii: 2 + 2 (diagonala) = 1 + 3 (val. proprii) = 4

determinantul este produsul valorilor proprii: det(A) = 2x2 - 1 = 3 iar produsul valorilor proprii este 1x3 = 3

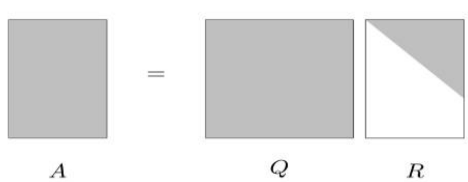
Factorizare QR

Factorizare QR cu reflectori Householder:

$$Ax \overset{CMMP}{=} b \Rightarrow \underbrace{U_1A}_{A_1}x \Rightarrow U_2U_1Ax \Rightarrow$$

$$\cdots \Rightarrow$$

$$\underbrace{U_m\cdots U_1A}_{R}x = \underbrace{U_m\cdots U_1}_{Q^T}b$$



https://www.ics.uci.edu/ xhx/courses/CS206/NLA-QR.pdf

CMMP

Factorizare QR cu reflectori Householder:

$$Ax \overset{CMMP}{=} b \Rightarrow Q^T A = R \Rightarrow A = QR = Q \begin{bmatrix} R' \\ 0 \end{bmatrix}$$

Algoritm:

- 1. Factorizare QR : $Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix}$
- 2. Calcul : $Q^T b = d = \begin{bmatrix} d' \\ d'' \end{bmatrix}$
- 3. Rezolvă: $Rx = d'$

Caz bidiagonal general

Matricea A (superior) bidiagonală:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & \cdots & 0 & 0 \\ 0 & a_{22} & a_{23} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1n-1} & a_{n-1n} \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Algorithm $BDTris(A, b)$:

1. $x := b, x_n = x_n / a_{nn}$
2. **Pentru** $i = n - 1 : -1 : 1$
   1. $x_i = x_i - a_{ii+1} x_{i+1}$
   2. $x_i = x_i / a_{ii}$

Solvabilitate

Notiuni legate de solvabilitatea unui sistem liniar pătratic:

- A singulară = neinversabilă $\rightarrow$ exemplu: $\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$
- A nesingulară = inversabilă $\rightarrow AA^{-1} = A^{-1}A = I_n$
- În alte forme:
  - $\det(A) = 0$
  - coloane liniar dependente
  - Nucleu $Ker(A)$ nenul
- Rang = număr maxim de coloane liniar independente
- Matricile singulare au rang $< n$, *i.e.* fie o infinitate de soluţii, fie fă ră soluţie

Eliminare gaussiană

Pseudocodul algoritmului EG:

Algoritm $G(A)$

1. **Pentru** $k = 1 : n-1$
    1. **Pentru** $i = k+1 : n$
        1. $a_{ik} \leftarrow \mu_{ik} = \frac{a_{ik}}{a_{kk}}$
    2. **Pentru** $j = k+1 : n$
        1. **Pentru** $i = k+1 : n$
            1. $a_{ij} \leftarrow a_{ij} - \mu_{ik} a_{kj}$

Multiplicatorii μ_{ik} se pot memora in triunghiul inferior al matricii A

$$\begin{bmatrix} u_{11} & u_{12} & \dots & u_{1k} & u_{1,k+1} & \dots & u_{1n} \\ \mu_{21} & u_{22} & \dots & u_{2k} & u_{2,k+1} & \dots & u_{2n} \\ & & \dots & & & \dots & \\ \mu_{k1} & \mu_{k2} & \dots & u_{kk} & u_{k,k+1} & \dots & u_{kn} \\ \mu_{k+1,1} & \mu_{k+1,2} & \dots & \mu_{k+1,k} & a^{(k+1)}_{k+1,k+1} & \dots & a^{(k+1)}_{k+1,n} \\ & & \dots & & & \dots & \\ \mu_{n1} & \mu_{n2} & \dots & \mu_{nk} & a^{(k+1)}_{n,k+1} & \dots & a^{(k+1)}_{nn} \end{bmatrix} \qquad \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1k} & \dots & u_{1n} \\ \mu_{21} & u_{22} & \dots & u_{2k} & \dots & u_{2n} \\ & & \dots & & \dots & \\ \mu_{k1} & \mu_{k2} & \dots & u_{kk} & \dots & u_{kn} \\ & & \dots & & \dots & \\ & & \dots & & \dots & \\ \mu_{n1} & \mu_{n2} & \dots & \mu_{nk} & \dots & u_{nn} \end{bmatrix}$$

După pasul k — În final

Activare: liniară

Funcția liniară:

$$f(x) = kx, \ k > 0 \tag{1}$$

folosită de obicei în ultimul strat al rețelelor pentru ieșirea modelului.

Derivata funcției de activare (numită și rata de tragere):

$$f'(x) = k \tag{2}$$

este folositoare în analiză și, mai ales, la algoritmul de retropropagare.

Funcția identitate:

$$f(x) = x \tag{3}$$

folosită pentru neuronii liniari.

Activare: treaptă

Cunoscută drept funcția treaptă, unitate, Heaviside:

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{4}$$

Proprietăți:

- este activat doar când $x \geq 0$
- nu este diferențiabil în $x = 0$
- privită ca o funcție generalizată (ca o distribuție) are derivată:

  $$H'(x) = \delta(x) \tag{5}$$

  unde $\delta(x)$ este funcția Dirac, funcția impuls.

Activare: signum

Cunoscută drept funcția signum, unitate bipolară:

$$S(x) = \begin{cases} -1, \, x < 0 \\ \;\; 1, \, x \geq 0 \end{cases} \tag{6}$$

Proprietăți:

- este activată în ambele ramuri
- în continuare nu este diferențiabilă în $x = 0$
- este legată de treaptă prin $S(x) = 2H(x) - 1$
- privită ca o funcție generalizată (ca o distribuție) are derivată:

$$S'(x) = 2H'(x) = 2\delta(x) \tag{7}$$

Activare: ReLU

Funcțiile de tip băț de hochei (hockey-stick) sunt în formă de L, și în general pleacă de la rectificarea funcției liniare (rectified linear unit) ce au la bază partea pozitivă a argumentului primit.

ReLU (rectified linear unit) este cea mai des folosită și este utilizată pentru crearea unor noi funcții după nevoile aplicației de dedesubt:

$$ReLU(x) = \max\{x, 0\} = \begin{cases} 0, \, x < 0 \\ x, \, x \geq 0 \end{cases} \tag{8}$$

Proprietăți:

- este legată de treaptă prin $ReLU(x) = xH(x)$
- derivabilă cu $ReLU'(x) = H(x)$
- nu ajunge la saturație când folosim gradientul pentru retropropagare

Activare: ReLU parametrizat

PReLU (parametric rectified linear unit) este varianta parametrizată a ReLU:

$$PReLU(x) = \begin{cases} \alpha x, \, x < 0 \\ \;\; x, \, x \geq 0 \end{cases}, \; \alpha > 0 \tag{9}$$

Proprietăți:

- este liniară pe părți
- are rate de activare diferite pentru $x < 0$ și $x > 0$
- cum este legată de celelalte funcții?
- cum derivăm?

Activare: sigmoid logistic

Cunoscută drept funcția logistică sau soft-step cu parametru $c > 0$:

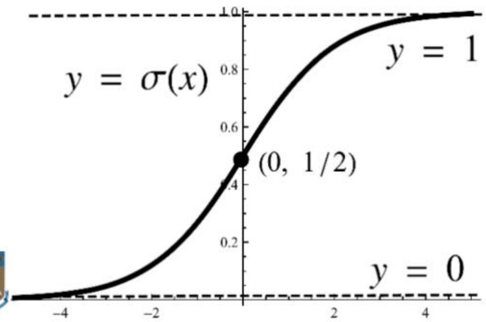
$$\sigma_c(x) = \sigma(c, x) = \frac{1}{1 + e^{-cx}} \quad (10)$$

unde c influențează rata de activare: valoriile mari duc la o schimabre bruscă de la 0 la 1.

Proprietăți:

- când $c \to \infty$ funcția devine $H(x)$ (**demonstrați!**)
- graficul funcției este independent de c la $x = 0$: $\sigma_c(0) = \frac{1}{2}$
- σ_c reprezintă o funcție monotonă ce transformă linia reală în intervalul $(0, 1)$
- derivabilă cu $\sigma_c' = c\sigma_c(1 - \sigma_c)$ (**demonstrați!**)
- pentru $c = 1$ avem funcția standard logistică $\sigma(x)$
- inversa funcției standard, **logit**, este

$$\sigma^{-1}(x) = \log\left(\frac{x}{1 - x}\right) \quad (11)$$



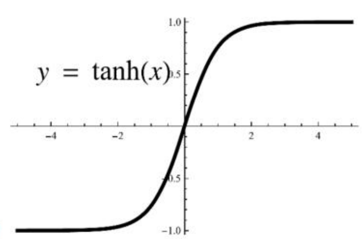
Activare: sigmoid hiperbolic

Cunoscută drept funcția hiperbolic tangentă sau sigmoidă bipolară:

$$t(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (12)$$

Proprietăți:

- t reprezintă o funcție monotonă ce transformă linia reală în intervalul $(0, 1)$
- tinde orizontal asimptotic la ± 1
- este legată de sigmoidul logistic prin $t(x) = 2\sigma_2(x) - 1$ (**demonstrați!**)
- derivabilă cu $t'(x) = 1 - t^2(x)$ (**demonstrați!**)
- graficul funcției trece prin origine și este simetric



Activare: softmax

Funcția softmax este o variantă netedă (smooth) a funcției max:

$$softmax_c(x)_i = \frac{e^{cx_i}}{\|e^{cx}\|_\ell}, \; c > 0 \quad (13)$$

unde x_i reprezintă elementul i al vectorului x, iar norma ℓ este de regulă ℓ_1.

Proprietăți:

- vectorii unitate e_i mai sunt denumiți *one-hot* vectori; atunci pentru $c \to \infty$ avem (**demonstrați**):

$$\lim_{c \to \infty} softmax_c(x) = e_k, \; unde \; k = \arg\max\{x_1, \ldots, x_n\} \quad (14)$$

- folosit adesea ca funcția de activare a ultimului strat din rețeaua neuronală

Exemple (câteva)

- kernel liniar
$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \tag{19}$$
- kernel polinomial
$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p \tag{20}$$
- kernel Gaussian
$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \tag{21}$$
- kernel Gaussian RBF
$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right) \tag{22}$$
- kernel sigmoid
$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\alpha \mathbf{x}_i \mathbf{x}_j + \gamma) \tag{23}$$