Laborator 3

Lucian M. Sasu

3 martie 2018

1 Regresie liniară

1. Să se scrie în Python o funcţie care să primească la intrare o matrice $\mathbf{X}$ şi să returneze matricea scalată, astfel: dacă matricea $\mathbf{X}$ primită ca parametru este

$$\mathbf{X} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_{n-1}^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \dots & x_{n-1}^{(2)} \\ \dots & \dots & \dots & \dots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_{n-1}^{(m)} \end{pmatrix} \tag{1}$$

atunci:

   (a) pentru fiecare coloană $0 \leq j \leq n-1$ determinăm minimul şi maximul ei:

$$min_j = \min_{i=0,\dots,m-1} x_j^{(i)} \tag{2}$$

   respectiv

$$max_j = \max_{i=0,\dots,m-1} x_j^{(i)} \tag{3}$$

   (b) fiecare element al lui $\mathbf{X}$ se va împărţi la diferenţa dintre maximul şi minimul coloanei pe care se găseşte[1]:

$$y_j^{(i)} = \frac{x_j^{(i)} - min_j}{max_j - min_j} \tag{4}$$

   Note:

[1] Coloanele constante nu se vor scala.

i. După aplicarea în ordine a celor două operaţii de mai sus pe o matrice de intrare $\mathbf{X}$, vom obţine o matrice având componentele în intervalul $[0, 1]$, excepţie fiind eventualele coloane constante.

ii. Se recomandă implementarea folosind cod vectorizat.

Pentru calculul minimului/maximului, a se vedea funcţiile Octave vectorizate `min` şi `max`.

Funcţia pe care o scrieţi va returna tripla: matricea cu elemente scalate şi vectorii $(min_0, \ldots, min_{n-1})$, $(max_1, \ldots, max_{n-1})$, conţinând respectiv minimele şi maximele coloanelor.

2. Să se scrie o a doua funcţie care să primească drept parametru o matrice $\mathbf{X}$ de forma:

$$\mathbf{X} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \ldots & x_{n-1}^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \ldots & x_{n-1}^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ x_0^{(m)} & x_1^{(m)} & \ldots & x_{n-1}^{(m)} \end{pmatrix} \tag{5}$$

si să returneze o matrice cu $n+1$ coloane, prima fiind coloana cu valoare 1 iar restul fiind cele din $\mathbf{X}$:

$$\mathbf{Z} = \begin{pmatrix} 1 & x_0^{(1)} & x_1^{(1)} & \ldots & x_{n-1}^{(1)} \\ 1 & x_0^{(2)} & x_1^{(2)} & \ldots & x_{n-1}^{(2)} \\ 1 & \ldots & \ldots & \ldots & \ldots \\ 1 & x_0^{(m)} & x_1^{(m)} & \ldots & x_{n-1}^{(m)} \end{pmatrix} \tag{6}$$

3. Să se folosească metoda de minimizare folosind căutarea bazată pe gradient, pentru a determina model de predicţie bazat pe regresie liniară. Se va folosi setul de date din secţiunea 2. Se va face în prealabil scalarea datelor, urmată de adăugarea unei coloane de valori 1, folosind funcţiile de la punctele precedente. Se va reprezenta grafic evoluţia funcţiei de eroare J, pentru a permite eventuale ajustări manuale ale ratei de învăţare α (vezi observaţiile din curs).

4. Să se calculeze prin metoda pseudoinversei coeficienţii regresiei liniare pentru setul de date din secţiunea 2.

Coeficienţii funcţiei de regresie se calculează cu formula:

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \ldots \\ \theta_n \end{pmatrix} = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t y \tag{7}$$

unde $\mathbf{X}$ este matricea obţinută după aplicarea funcţiei de la punctul 2. Pentru pseudoniversă se poate folosi funcţia numpy.linalg.pinv. Funcţia de regresie liniară este $h_\theta : \mathbf{R}^{n+1} \to \mathbf{R}$,

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^t \cdot \mathbf{x} \tag{8}$$

Pentru fiecare vector $\mathbf{x}^{(i)}$ din setul de instruire se va afişa valoarea prezisă de (8), împreună cu valoarea efectivă din setul de instruire. Se va calcula şi afişa în final funcţia totală de eroare:

$$J(\theta_0, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(\mathbf{x}^{(i)}) - y_i\right)^2 \tag{9}$$

Observaţie: pentru determinarea vectorului $\boldsymbol{\theta}$ nu e obligatoriu a se face scalarea de la punctul 1 – a se vedea cursul.

5. De regulă, performaţa unui model nu se măsoară pe datele de instruire, ci ne interesează puterea lui de generalizare, *i.e.* capacitatea de a face predicţii pentru date similare cu setul de instruire, dar neutilizate în instruire. Una din metodele acceptate de măsurare a performanţei este:

   - se face o permutare aleatoare a setului de date, se poate folosi funcţia numpy.random.shuffle.
   - se împarte setul de la punctul precedent în două subseturi, primele 70% din date sunt folosite doar pentru antrenare, restul doar pentru testare;
   - pe setul de antrenare se construieşte model de regresie;
   - se calculează eroarea J pentru modelul rezultat şi setul de date de testare.

   Să se urmeze aceşti paşi pentru cuantificarea erorii. Puteţi explica de ce la rulări diferite se raportează rezultate diferite? cum se poate face ca testele să fie reproductibile (să dea de fiecare dată acelaşi rezultat)?[2]

2 Setul de date

Setul de date este "Boston housing" de la adresa:
http://www.dcc.fc.up.pt/~ltorgo/Regression/housing.tar.gz.
O descriere a setului de date se găseşte la

[2]Indicaţie: documentaţi–vă asupra semnificaţiei funcţiei numpy.random.seed.

http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html, alegeţi "Boston Housing" în frame–ul din stânga.

Valoarea ce trebuie prezisă este `MEDV`, trăsăturile de intrare sunt celelalte 13.

3 Precizări

- Se vor scrie funcţii care implementează paşii daţi mai sus.
- Din punctajul acordat, 1 punct este pentru scrierea vectorizată a codului, fără folosirea instrucţiunilor `while`, `for`, `if`, exceptând iterarea peste setul de date în rezolvarea punctului 3 şi afişarea valorilor prezise, conform ecuaţiei (8). Valoarea funcţiei de eroare J din ec. (9) se va calcula fără a folosi instrucţiuni de ciclare.
- Studenţii se pot consulta pentru rezolvarea temei, dar rezolvările vor fi individuale.
- Prezentarea temei se va face la laboratorul din săptămâna 12—16 martie 2018. Pentru o întârziere de cel mult o săptămână se vor scădea 2 puncte din nota cuvenită. Temele predate cu o întârziere mai mare de o săptămână nu vor mai fi luate în considerare.