

Laborator 4

Lucian M. Sasu

1 Regresie logistică pentru două clase

1. (**Recunoașterea a două cifre**) Să se implementeze algoritmul de regresie logistică pentru a face clasificarea a două cifre. Setul de date este MNIST, descris la <http://yann.lecun.com/exdb/mnist/> și disponibil în arhiva date MNIST - format CSV.zip. Setul MNIST conține imagini gri ale cifrelor 0–9 — figura 1.

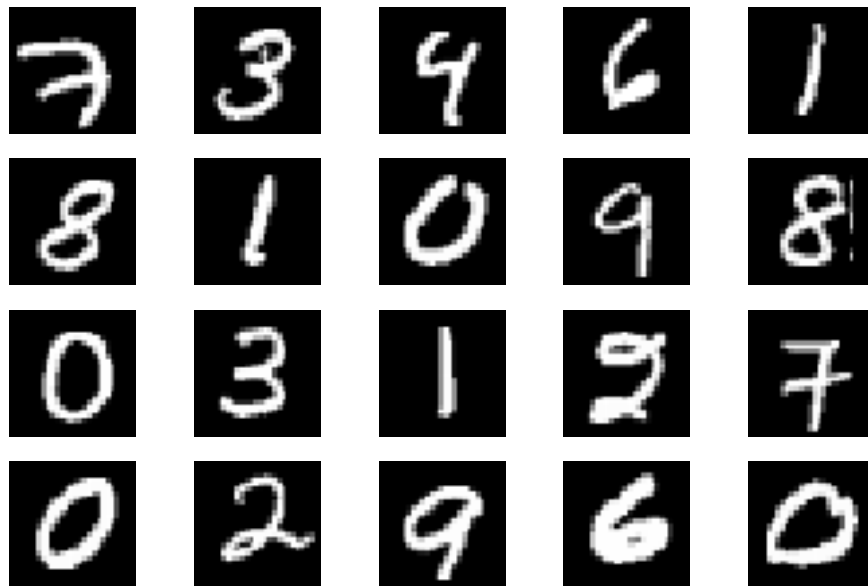


Figura 1: 20 de cifre din setul MNIST

Setul de antrenare este format din două fișiere în format CSV:

- (a) `mnist_train.csv` – conține 60000 de randuri. Pe fiecare rand, primul număr este o cifră de la 0 la 9, reprezentând chiar clasa cifrei respective și sunt considerate etichete de ieșire; următoarele 28*28 numere de pe aceeași linie sunt valorile pixelilor desenelor cifrelor, datele de intrare pentru setul de antrenare; dacă se transformă aceste 28*28 valori de pe linie într-o matrice de 28 de linii și tot atâtea coloane, se obțin reprezentările grafice din figura 1.
- (b) `mnist_test.csv` – conține 10000 de randuri, cu aceeași codificare ca la setul de antrenare. Datele din acest fișier sunt păstrate pentru testarea acurateții modelului.

Pașii de lucru sunt:

- (a) Scrierea de funcții auxiliare¹:
 - i. funcție care citește datele din fișier și returnează: matricea `X_train` de 60000×784 – din fișierul `mnist_train.csv`, vectorul `y_train` de 60000 valori întregi 0...9 și similar `X_test` și `y_test` din fișierul cu date de testare; funcția returnează aceste 4 matrice;
 - ii. funcție de filtrare: pentru două cifre diferite specificate prin argumente, se vor returna din matricele `X_*`, `y_*` acele cazuri (linii) care corespund cifrelor indicate (de exemplu, dacă vreau să învăț doar cifrele 0 și 1, returnez doar liniile corespunzând desenelor pentru 0 și 1 din matricea `X_*` și doar etichete 0 și 1 din vectorul `y_*`). Formatați perechea de valori returnate în mod convenabil (matrice, liste de vectori etc.); cifrele vor fi alese de student;
 - iii. funcție care adaugă la o matrice dată o primă coloană plină cu 1 și returnează noua matrice;
 - iv. funcție pentru implementarea modelului de predicție h_{θ} :

$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + \exp(-\theta^t \cdot \mathbf{x})} \quad (1)$$

- v. funcție de calcul a erorii J , incluzând regularizare pentru elementele lui θ mai puțin primul element al vectorului (a se vedea cursul);
- vi. funcție care primind parametrii unui model (în cazul nostru: vectorul θ) și setul de date de testare (perechea (`X_test`,

¹Pentru notă maximă: funcții vectorizate.

`y_test`) filtrată cu funcția de la punctul (1(a)ii), va calcula acuratețea de clasificare a modelului. Acuratețea este procentul de cazuri în care datele sunt recunoscute corect: clasa estimată de către model pentru o intrare (vector de 784 numere) coincide cu eticheta cunoscută asociată intrării;

vii. alte funcții auxiliare.

Precizare: ordinea aplicării funcțiilor este orientativă.

(b) Calculul prin metoda gradient descent a vectorului optim θ folosind doar setul de antrenare (acele linii din setul de date `X_train` și `y_train` rezultate după filtrare; setul de testare este de asemenea cel filtrat pentru aceleași cifre); se va face regularizare; pentru hiperparametrul de regularizare λ se vor încerca diferite valori, idem pentru hiperparametrul rată de învățare α . Se va reprezenta grafic evoluția valorilor funcției de eroare J , urmărindu-se dacă aceste valori scad pe măsură ce se iterează.

(c) Pentru setul de testare se va raporta: procentul de clasificare corectă și **matricea de confuzie** – în cazul acesta de 2×2 .

2. (**Regresie logistică multinomială pentru recunoașterea tuturor celor 10 cifre**) Se va folosi întregul set de date de antrenare și algoritmul de regresie multinomială descris în curs; testarea se va face pe toate cele 10000 de cazuri din setul de testare.

3. (Opțional: **Regresie logistică, estimarea calității hiperparametriilor prin k-fold cross validation peste setul de antrenare filtrat pe cele 2 cifre alese de student**). De regulă, valorile pentru hiperparametrii α (rata de învățare) și λ (coeficientul pentru termenul de regularizare) se determină prin trial and error. Se cere implementare similară cu cea de la punctul (1b), dar estimarea performanței modelului pentru o anumită valoare α și λ să se facă prin k-fold cross validation, descris în secțiunea 3.2. Să se calculeze acuratețea de clasificare pentru 12 combinații de hiperparametri $(\alpha, \lambda) \in \{0.1, 0.2, 0.3\} \times \{0, 1, 10, 100\}$. Combinația de valori pentru care eroarea medie calculată pe cele k fold-uri este minimă se consideră a fi cea mai bună combinație de hiperparametri. Modelul antrenat cu aceste valori α și λ e în final antrenat pe setul de antrenare și testat pe cel de testare (în ambele cazuri: considerând doar datele pentru cifrele alese de student).

Valoarea lui k este 5.

4. (Opțional) Să se aplice k-fold cross validation pentru regresia logistică multinomială, cazul celor 10 cifre. Să se încerce mai multe variante de

valori pentru hiperparametrii α , λ . Strategia de lucru e aceeași ca la punctul anterior.

2 Precizări

- Din punctajul acordat, 1 punct este pentru scrierea codului vectorizat.
- Studenții se pot consulta pentru rezolvarea temei, dar rezolvările vor fi individuale.
- Predarea temei se va face la laboratorul din săptămâna 8–12 aprilie 2019. 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.

3 Appendix

3.1 Codificarea “one-hot”

Pentru cazul în care se cere codificarea numerică a unei mulțimi de n clase – precum se cere la regresia logisitcămulinomială – următoarea variantă este populară: fiecare clasă i , $0 \leq i < n$, se codifică sub forma unui vector cu n elemente, având componenta pe indicele i setată la 1 iar restul 0.

Exemplu: dacă mulțimea de 5 clase este {mere, pere, portocale, nuci, struguri}, atunci pentru clasa “mere” vom avea codificarea vectorul $(1, 0, 0, 0, 0)^t$, pentru “pere” $(0, 1, 0, 0, 0)^t$ etc., pentru “struguri” $(0, 0, 0, 0, 1)^t$. Vectorii de codificare one-hot sunt “rari”, în sensul https://en.wikipedia.org/wiki/Sparse_matrix.

Decodificarea unui astfel de vector înseamnă găsirea celui indice i din vector care are valoarea 1; acest i este clasa codificată de vector.

3.2 Metoda de evaluare k-fold cross validation

Spre deosebire de metoda care testează un model pe un singur set de testare și raportează performanța, metoda k -fold cross validation produce k testări și prin urmare k valori ale acurateții modelului; pentru acestea în final se calculează valoarea medie, care reprezintă estimarea performanței modelului. În practică este metoda preferată de cuantificare a calității unui model și permite compararea relevantă cu alte modele.

Validarea funcționează astfel: se împarte matricea (setul de date) \mathbf{X} în k perechi de submulțimi disjuncte, de dimensiuni cât mai apropiate; în paralel

se face același lucru pentru setul \mathbf{y} . Rezultă partiționarea seturilor \mathbf{X} și \mathbf{y} în k perechi de submulțimi $(\mathbf{X}_i, \mathbf{y}_i)$, $1 \leq i \leq k$.

Pentru i de la 1 la k se procedează astfel: subseturile $(\mathbf{X}_i, \mathbf{y}_i)$ sunt păstrate strict pentru testare; subseturile $(\mathbf{X}_1, \mathbf{y}_1), \dots, (\mathbf{X}_{i-1}, \mathbf{y}_{i-1}), (\mathbf{X}_{i+1}, \mathbf{y}_{i+1}), \dots, (\mathbf{X}_k, \mathbf{y}_k)$ sunt folosite doar pentru antrenare; se face antrenarea folosind toate datele din toate cele $k - 1$ subseturi de antrenare și se testează pe $(\mathbf{X}_i, \mathbf{y}_i)$. Rezultă o valoare de acuratețe a_i .

Observație: Pentru valori i diferite valorile optime θ pot diferi; de asemenea, poate diferi numărul de iterații efectuate până la oprire. Explicația este dată de faptul că datele de antrenare diferă de la o valoare a lui i la alta.

După ce fiecare $(\mathbf{X}_i, \mathbf{y}_i)$ produce acuratețea a_i , cele k valori se mediază și acesta este rezultatul final ce se raportează.

Detalii pentru k -fold cross validation se găsesc în:

- [Regularization and model selection](#)
- [Tutorial 1](#)
- [Tutorial 2](#)