Laborator 4

Lucian M. Sasu

1 Regresie logistică pentru două clase

1. (**Recunoaşterea 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/ şi disponibil în directorul `./date`. Setul MNIST conţine imagini gri ale cifrelor 0–9 — Figura 1.

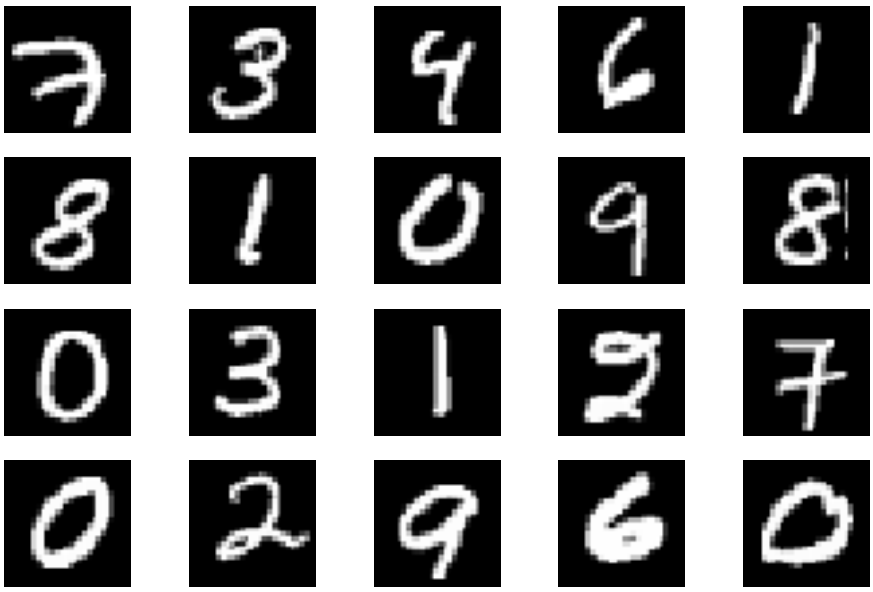

Figura 1: 20 de cifre din setul MNIST

Setul de antrenare este format din două fişiere in format CSV:

(a) `mnist_train_input.csv` – conţine reprezentarile matriceale pentru 55000 desene de cifre, datele de intrare pentru setul de antrenare; fiecare cifră e codificată ca o matrice de 28x28 de numere în virgulă mobilă, liniarizată (flattened);

(b) `mnist_train_labels.csv` — conţine 55000 de randuri, iar pe fiecare rând se găseşte codificarea one-hot a clasei din care face parte imaginea corespunzătoare (adică de pe acelaşi rând) din fişierul `mnist_train_input.csv`, reprezentând etichetele corespunzătoare pentru setul de intrare; prima matrice de 28x28 din fişierul de la punctul anterior este desenul pentru cifra 7, pe prima linie din fisierul `mnist_train_labels.csv` avem şirul de 10 numere 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 (1 pe indicele 7, restul 0). Pentru detalii despre codificarea one-hot a se vedea secţiunea 3.1.

Setul de testare este structurat la fel, în fişiere al căror nume conţine particula `_test_` în loc de `_train_`, dar numărul de desene de cifre este de 10000.

Paşii de lucru sunt:

(a) Scrierea de funcţii auxiliare[1]:

i. funcţie care citeşte datele din fişier şi returnează: matricea X_train de 55000×784 – din fisierul `mnist_train_input.csv`, vectorul y_train de 55000 valori intregi $0 \ldots 9$ (unica poziţie pe care se află valoarea 1 pe rândul acela: 0, 0, 0, 0, 0, 0, 0, 1, 0, 0 $\rightarrow$ 7) şi similar X_test şi y_test din cele două fişiere cu date de testare; funcţia returnează aceste 4 matrice;

ii. funcţie 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 învăţ doar cifrele 0 şi 1, returnez doar liniile corespunzând desenelor pentru 0 şi 1 din matricea X_* si doar etichete 0 şi 1 din vectorul y_*). Formataţi perechea de valori returnate în mod convenabil (matrice, liste de vectori etc.); cifrele vor fi alese de student;

iii. funcţie care adaugă la o matrice dată o primă coloană plină cu 1 şi returnează noua matrice;

iv. funcţie pentru implementarea modelului de predicţie $h_{\boldsymbol{\theta}}$:

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

[1]Pentru notă maximă: funcţii vectorizate.

v. funcţie de calcul a erorii J, incluzând regularizare pentru elementele lui $\boldsymbol{\theta}$ mai puţin primul element al vectorului (a se vedea cursul);

vi. funcţie care primind parametrii unui model (în cazul nostru: vectorul $\boldsymbol{\theta}$) şi setul de date de testare (perechea (X_test, y_test) filtrată cu funcţia de la punctul (1(a)ii), va calcula acurateţea de clasificare a modelului. Acurateţea este procentul de cazuri în care datele sunt recunoscute corect: clasa estimată de către model pentru o intrare (vector de 768 numere în virgulă mobilă) coincide cu eticheta cunoscută asociată intrării;

vii. alte funcţii necesare.

Precizare: ordinea aplicării funcţiilor este orientativă.

(b) Calculul prin metoda gradient descent a vectorului optim $\boldsymbol{\theta}$ folosind doar setul de antrenare (acele linii din setul de date X_train şi y_train rezultate după filtrare; setul de testare este de asemenea cel filtrat pentru aceleaşi cifre); se va face regularizare; pentru hiperparametrul de regularizare λ se vor încerca diferite valori, idem pentru hiperparametrul rată de învăţare α. Se va reprezenta grafic evoluţia valorilor funcţiei 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 şi matricea de confuzie – în cazul acesta de 2×2.

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

3. (Opţional: **Regresie logistică, estimarea calităţii hiperprametrilor 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 învăţare) şi λ (coeficientul pentru termenul de regularizare) se determină prin trial and error. Se cere implementare similară cu cea de la punctul (1b), dar estimarea performanţei modelului pentru o anumită valoare α şi λ să se facă prin k–fold cross validation, descris în secţiunea 3.2. Să se calculeze acurateţea de clasificare pentru 12 combinaţii de hiperparametri $(\alpha, \lambda) \in \{0.1, 0.2, 0.3\} \times \{0, 1, 10, 100\}$. Combinaţia de valori pentru care eroarea medie calculată pe cele k folduri este minimă se consideră a fi cea mai bună combinaţie de hiperparametri. Modelul antrenat cu aceste valori α şi λ e în final antrenat pe

setul de antrenare şi testat pe cel de testare (în ambele cazuri: considerând doar datele pentru cifrele alese de student).

Valoarea lui k este 5.

4. (Opţional) 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 aceeaşi ca la punctul anterior.

2 Precizări

- Din punctajul acordat, 1 punct este pentru scrierea codului vectorizat.
- Studenţii se pot consulta pentru rezolvarea temei, dar rezolvările vor fi individuale.
- Predarea temei se va face la laboratorul din săptămâna 26-30 martie. 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 mulţimi de n clase, 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 mulţimea 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 acelui 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 şi raportează performanţa, metoda k–fold cross validation produce

k testări şi prin urmare k valori ale acurateţei modelului; pentru acestea în final se calculează valoarea medie, care reprezintă estimarea performanţei modelului. În practică este metoda preferată de cuantificare a calităţii unui model şi permite compararea relevantă cu alte modele.

Validarea funcţionează astfel: se împarte matricea (setul de date) $\mathbf{X}$ în k perechi de submulţimi disjuncte, de dimensiuni cât mai aporpiate; în paralel se face acelaşi lucru pentru setul $\mathbf{y}$. Rezultă partiţionarea seturilor $\mathbf{X}$ şi $\mathbf{y}$ în k perechi de submulţimi $(\mathbf{X}_i, \mathbf{y}_i)$, $1 \leq i \leq k$.

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

Observaţie: Pentru valori i diferite valorile optime $\boldsymbol{\theta}$ pot diferi; de asemnea, poate diferi numărul de iteraţii efectuate până la oprire. Explicaţia 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 acurateţea a_i, cele k valori se mediază şi acesta este rezultatul final ce se raportează.

Detalii pentru k–fold cross validation se găsesc în:

- Regularization and model selection
- Tutorial 1
- Tutorial 2
- Tutorial video pentru Python