Proiect 2 - Clasificare folosind Rețele Neuronale

Constantin Matei

Grupa 324AA

1 Sarcina de învățare

Sarcina de învățare abordată în acest proiect este o problemă de clasificare binară. Obiectivul este de a construi un model capabil să prezică șansele de supraviețuire ale pasagerilor de pe nava Titanic, în funcție de caracteristicile lor.

2 Baza de date

Baza de date folosită este fișierul "train.csv" obținut de pe platforma Kaggle (`https://www.kaggle.com/competitions/titanic/overview`). Acesta conține date despre supraviețuitorii din dezastrul Titanicului și este utilizată ca set de antrenare într-o competiție de predicție a supraviețuirii pasagerilor pe baza anumitor caracteristici precum vârsta, sexul și clasa.

Fișierul "train.csv" conține 891 de exemple, din care am folosit 4 caracteristici relevante: Vârstă, Sex, Clasa (Pclass) și un label binar Supraviețuit (0 sau 1). Caracteristicile conțin și alte informații precum numele pasagerilor, dar acestea nu au fost utilizate în modelul meu. Datele au fost împărțite în set de antrenare (80%) și set de testare (20%) pentru evaluarea modelului.

3 Algoritmi de optimizare implementați

În cadrul proiectului, au fost implementați și evaluați doi algoritmi de optimizare: Descendent pe Panta (SGD Clasic) și Adaptive Moment Estimation (ADAM). Ambii algoritmi au fost utilizați pentru antrenarea unei rețele neuronale cu doua straturi ascunse, pentru rezolvarea sarcinii de clasificare.

3.1 SGD Clasic

Algoritmul Descendent pe Panta este o metodă iterativă de optimizare care actualizează parametrii modelului în direcția inversă a gradientului funcției obiectiv. În cazul nostru, funcția obiectiv este funcția de pierdere (loss function) care măsoară eroarea de predicție a modelului.

3.2 ADAM

ADAM este un algoritm de optimizare bazat pe gradienți, similar cu SGD, dar care utilizează medii mobile ale gradientului și pătratului gradientului pentru a accelera convergența și a oferi o performanță mai bună în cazuri complexe.

4 Rezultate și comentarii

Rezultatele obținute pentru fiecare algoritm de optimizare sunt prezentate mai jos:

Metrica	SGD	ADAM
Timp de antrenare (secunde)	108.48	139.48
Acuratețe	0.79	0.81
Precizie	0.85	0.82
Recall	0.81	0.91
Scor F1	0.835	0.864

Observăm că algoritmul ADAM a obținut rezultate superioare în termeni de acuratețe, precizie, recall și scor F1, în comparație cu SGD. Cu toate acestea, SGD a avut un timp de antrenare mai rapid.

De asemenea, scaderea funcțiilor de pierdere a fost reprezentată grafic pentru ambii algoritmi, oferind o perspectivă vizuală asupra procesului de optimizare.

În general, rezultatele obținute sunt promițătoare și demonstrează capacitatea algoritmilor de optimizare de a antrena modele de inteligență artificială pentru sarcini de clasificare.

5 Anexă: Cod Matlab

Codul Matlab care implementează algoritmii de optimizare și modelul de învățare este:

```
function main()
    % Citirea datelor din fișierul CSV
    train = readtable('train.csv');

    % Extragerea coloanelor relevante din tabelul de date
    age = train{:,"Age"};
    sex = train{:,"Sex"};
    class = train{:,"Pclass"};
    labels = train{:,"Survived"};

    % Codificarea categorică a variabilei 'Sex'
    [~, ~, sex] = unique(sex);
    sex = sex - 1;
```

```
% Tratarea valorilor lipsă (NaN) pentru 'Age' prin înlocuirea cu media
mean_age = mean(age(~isnan(age)));
age(isnan(age)) = mean_age;

% Normalizarea variabilei 'Age'
age = age/100;

% Determinarea dimensiunii setului de date
[n,~]=size(train);

% Împărțirea setului de date în set de antrenare și set de testare (80/20)
train_size=floor(0.8*n);
sex_train = sex(1:train_size);
age_train = age(1:train_size);
class_train = class(1:train_size);
labels_train = labels(1:train_size);
sex_test = sex(train_size+1:end);
age_test = age(train_size+1:end);
class_test = class(train_size+1:end);
labels_test = labels(train_size+1:end);

% Construirea matricei de intrare X pentru setul de antrenare
x = [sex_train age_train class_train];

% Setarea parametrilor de antrenare
learning_rate = 0.001;
hiden_layers_neurons=50;
nr_iteratii=5000;

% Antrenarea și evaluarea modelului cu metoda SGD
disp('Antrenarea modelului cu metoda SGD:');
tic; % Pornirea cronometrului
[w1_sgd, w2_sgd, w3_sgd, b1_sgd, b2_sgd, loss_values_sgd] = SGD(x, labels_train,learning
sgd_time = toc; % Oprirea cronometrului

% Evaluarea modelului SGD pe setul de testare
xx_sgd = [sex_test age_test class_test];
[accuracy_sgd, precision_sgd, recall_sgd, f1_score_sgd] = evaluate_model(w1_sgd, w2_sgd,

% Afișarea rezultatelor SGD
disp('Rezultate pentru metoda SGD:');
disp(['Timp de antrenare: ', num2str(sgd_time), ' secunde']);
disp(['Acuratețe: ', num2str(accuracy_sgd)]);
disp(['Precizie: ', num2str(precision_sgd)]);
disp(['Recall: ', num2str(recall_sgd)]);
disp(['Scor F1: ', num2str(f1_score_sgd)]);
```

```
    % Antrenarea și evaluarea modelului cu metoda ADAM
    disp('Antrenarea modelului cu metoda ADAM:');
    tic; % Pornirea cronometrului
    [w1_adam, w2_adam, w3_adam, b1_adam, b2_adam, loss_values_adam] = ADAM(x, labels_train,l
    adam_time = toc; % Oprirea cronometrului

    % Evaluarea modelului ADAM pe setul de testare
    xx_adam = [sex_test age_test class_test];
    [accuracy_adam, precision_adam, recall_adam, f1_score_adam] = evaluate_model(w1_adam, w2

    % Afișarea rezultatelor ADAM
    disp('Rezultate pentru metoda ADAM:');
    disp(['Timp de antrenare: ', num2str(adam_time), ' secunde']);
    disp(['Acuratețe: ', num2str(accuracy_adam)]);
    disp(['Precizie: ', num2str(precision_adam)]);
    disp(['Recall: ', num2str(recall_adam)]);
    disp(['Scor F1: ', num2str(f1_score_adam)]);

    % Plotarea evoluției funcției obiectiv
    figure;
    plot(loss_values_sgd, 'r-', 'LineWidth', 2);
    hold on;
    plot(loss_values_adam, 'b-', 'LineWidth', 2);
    hold off;
    xlabel('Iterații');
    ylabel('Funcția obiectiv');
    title('Evoluția funcției obiectiv');
    legend('SGD', 'ADAM');
end
```