NAI

Zadania nr 4

25 marca 2025

I. **Macierz omyłek I:**
System sztucznej inteligencji przeznaczony do automatycznej filtracji wiadomości e-mail pod kątem spamu. System został przetestowany na 120 różnych wiadomościach z różnych kategorii. W wyniku przeprowadzonych testów stwierdzono następujące wyniki:

- System poprawnie rozpoznał 40 wiadomości jako "normalne"
- System poprawnie rozpoznał 50 wiadomości jako spam
- System nie prawidłowo rozpoznał 20 "normalnych"wiadomości jako spam
- System pominął 10 wiadomości spam

a) Narysuj macierz pomyłek dla tego klasyfikatora

b) Oblicz następujące miary wydajności:
   - Precyzję (Precision) - P
   - Czułość (Recall) - R
   - F Miarę - F_1

Pomocne wzory: $P = \frac{TP}{TP+FP}$ $\quad R = \frac{TP}{TP+FN}$ $\quad F_1 = 2 \cdot \frac{P \cdot R}{P+R}$

II. **Macierz omyłek II:**
System sztucznej inteligencji dedykowany identyfikacji zagrożeń pożarowych na podstawie zdjęć z kamer monitoringu miejskiego, został przetestowany na 100 różnych scenach, w tym na terenach miejskich, leśnych i przemysłowych. W wyniku przeprowadzonych testów stwierdzono następujące wyniki:

- System rozpoznał 55 scen z rzeczywistym ryzykiem pożaru
- System błędnie zidentyfikował 10 scen bez ryzyka pożaru jako potencjalnie niebezpieczne
- System pominął 5 scen z rzeczywistym ryzykiem pożaru
- System poprawnie rozpoznał pozostałe sceny jako bez ryzyka pożaru

a) Narysuj macierz pomyłek dla tego klasyfikatora

b) Oblicz następujące miary wydajności:
   - Precyzję (Precision) - P
   - Czułość (Recall) - R
   - F Miarę - F_1

III. **Zadanie na zajęcia**

Zaimplementować klasę **Perceptron** wykorzystującą ciągłą funkcję aktywacji posiadającą następujące pola:

- weights
- α (stała uczenia)
- θ (próg aktywacji)

Dodatkowo należy dostarczyć również następujące metody:

- Konstruktor przyjmujący rozmiar wektora wag oraz α
- `compute(data)` zwracająca wynik [0, 1] dla pojedynczego rekordu/obserwacji
- `update(data, decision)` aktualizująca wektor wag perceptronu i θ zgodnie z tym jak decyzja z `compute(data)` była blisko prawidłowej odpowiedzi
- `info()` zwracającą informację co dany perceptron przewiduje (na potrzeby tego zadania może zostać pusta)

Zaimplementować klasę **Layer** łączącą grupę perceptronów w jedną sieć. Klasa powinna zawierać pojedyncze pole:

- perceptrons (kolekcję obiektów typu **Perceptron**)

Oraz implementować następujące metody:

- `layerCompute(data)` wywołującą `compute(data)` na każdym z perceptronów w warstwie i zwracająca jako wynik kolekcję wartości funkcji aktywacji dla każdego z perceptronów.
- `layerUpdate(data, decision)` aktualizująca wagi wszystkich perceptronów w warstwie na podstawie wyniku metody `layerCompute(data)` oraz kolekcji `decision` składających się odpowiednio z samych 0 i jednej 1. Np: dla warstwy składającej się z 3 perceptronów: `{0, 0, 1}`

Dodatkowe informacje:

Sugerowaną funkcją aktywacji jest funkcja sigmoidalna unipolarna. $y = \frac{1}{1+e^{-net}}$
Do zmiany wag należy obliczyć pochodną funkcji aktywacji w przypadku funkcji unipolarnej sigmoidalnej:

$$f`(x) = f(x) - (1 - f(x))$$

Pochodną tą podstawiamy do wzoru wcześniej wykorzystanego do zmiany wag perceptronu skalarnego:

$$W` = W + (d - y)(f'(net))\alpha X$$

Po odpowiednich podstawieniach otrzymujemy następujący wzór:

$$W` = W + (d - y)y(1 - y)\alpha X$$

gdzie W` jest wektorem zmienionych wag, d jest prawidłowym wynikiem perceptronu (1 lub 0), y jest wynikiem funkcji aktywacji na podstawie wartości net

Podobnie wtedy do zmiany wartości θ można zostać następujący wzór:

$$\theta` = \theta - (d - y)y(1 - y)\alpha$$

Domyślnie wartości w wektorze wag powinny zostać wylosowane (sugerowany zakres [-1, 1]) a θ powinna przyjąć wartość 0

IV. **Zadanie mpp**

W celu zaliczenia projektu mpp należy rozszerzyć program z "Zadania na zajęcia".
Celem zadanie będzie przystosowanie warstwy perceptronów do klasyfikacji różnych języków na podstawie analizy frekwencyjnej. W tym celu należy rozszerzyć kod o poniższe elementy:

- Przygotować plik lub pliki zawierające po kilka (5+) paragrafów w 3 wybranych językach wraz z ich etykietami.
  (Paragrafy powinny być powyżej 200 słów dla dobrych wyników modelu)(Wikipedia jest stosunkowo dobrym źródłem do znalezienia danych)

- Przygotowane dane powinny zostać wczytane podczas uruchomienia programu i powinny zostać z nich usunięte wszystkie znaki interpunkcyjne po za spacjami oraz wszystkie znaki z poza zakresu ASCII. Dodatkowo wszystkie znaki powinny zostać zamienione na ich "lowerCase"odpowiedniki.
- Do klasy **Layer** należy dodać metodę `train(data[], decision[], iterations)`, metoda ta wywoła odpowiednio pozostałe metody `layerUpdate` i `layerCompute` w celu nauczenia (dostosowania wag i θ-y) do prawidłowej klasyfikacji języków.
- Po wytrenowaniu program powinien pozwolić użytkownikowi na wprowadzenie tekstu (przez podanie ścieżki do pliku albo przez wklejenie danych w konsolę) do zaklasyfikowania. Jako wynik program powinien wyświetlić przewidywany język i jego prawdopodobieństwo (wartość funkcji aktywacji).

Sugerowanym jest by języki wybrane do mpp pochodziły z różnych grup językowych