Zadanie nr 1 - klasyfikacja wzorców

Analiza danych złożonych z detekcją wyjątków

Karol Kazusek - 254189, Sebastian Zych - 254264

14.10.2024

1 Cel zadania

Zadanie polegało na przeprowadzeniu analiz porównawczych klasyfikacji wzorców (własnych, wyjątkowych) w strumieniach danych. Do analiz należało wybrać tematykę medycyny lub rozpoznawania faz ruchu aktywności człowieka (systemy HAR)

2 Opis zaproponowanych Klasyfikatorów

Klasyfikatory to algorytmy stosowane w *uczeniu maszynowym*, których celem jest przypisanie danych wejściowych do jednej z wcześniej zdefiniowanych kategorii (klas). Klasyfikacja składa się z dwóch głównych etapów: **trenowania** i **predykcji**.

- **Trenowanie** polega na dostarczeniu algorytmowi zbioru danych treningowych, na podstawie którego algorytm uczy się rozpoznawać wzorce i różnicować między klasami.
- **Predykcja** to proces, w którym przetrenowany model jest wykorzystywany do klasyfikowania nowych, wcześniej niewidzianych danych.

Skuteczność klasyfikatorów ocenia się przy użyciu różnych miar, takich jak:

- **Dokładność** (accuracy) — odsetek poprawnie sklasyfikowanych przypadków,
- **Precyzja** (precision) — odsetek prawdziwych pozytywnych wyników spośród wszystkich przykładów sklasyfikowanych jako pozytywne,
- **Czułość** (recall) — odsetek prawdziwie pozytywnych wyników spośród wszystkich pozytywnych przypadków w rzeczywistości,
- **Specyficzność** (specificity) — odsetek prawdziwych negatywnych wyników spośród wszystkich przypadków rzeczywiście negatywnych. Mierzy zdolność klasyfikatora do prawidłowego rozpoznawania negatywnych przykładów, co jest szczególnie istotne w przypadku, gdy negatywna klasa jest dominująca.

Klasyfikatory znajdują zastosowanie w wielu dziedzinach, takich jak medycyna, rozpoznawanie obrazów, analiza tekstu oraz inne zadania związane z przetwarzaniem danych.

2.1 Metody klasyfikacyjne

2.1.1 K-nn

Metoda **K-Najbliższych Sąsiadów (K-NN)** to jedna z najprostszych technik klasyfikacji. Działa na zasadzie porównywania nowego punktu danych z istniejącymi przykładami treningowymi. Algorytm wybiera K najbliższych sąsiadów (na podstawie odległości euklidesowej lub innych miar), a następnie przypisuje nowy punkt do klasy, która występuje najczęściej wśród tych sąsiadów.

- `n_neighbors`: Liczba sąsiadów (K) do uwzględnienia przy klasyfikacji.
- `weights`: Sposób ważenia sąsiadów. Możliwe wartości to `'uniform'` (wszyscy sąsiedzi mają taką samą wagę) i `'distance'` (bliżsi sąsiedzi mają większy wpływ).
- `metric`: Miara odległości używana do porównania punktów,która w zależności od parametru może reprezentować odległość euklidesową lub Manhattan.

2.1.2 Klasyfikator baysowski

Klasyfikacja Bayesowska opiera się na *twierdzeniu Bayesa*, które opisuje zależność między prawdopodobieństwem wystąpienia klasy a dostarczonymi danymi. W najprostszym przypadku, **naiwny klasyfikator Bayesa** zakłada, że cechy są niezależne od siebie. Model wylicza prawdopodobieństwo każdej klasy na podstawie danych wejściowych, a następnie przypisuje nowy przykład do klasy o największym prawdopodobieństwie.

- `var_smoothing`: Parametr ten reguluje wielkość wygładzania (dodanie małej stałej do wariancji, aby uniknąć dzielenia przez zero). (Wygładzanie Laplace).

2.1.3 Drzewa decyzyjne

Drzewa decyzyjne to metoda klasyfikacji oparta na strukturze drzewa, gdzie każdy węzeł reprezentuje decyzję na podstawie jednej cechy, a każda gałąź odpowiada wynikom tej decyzji. Proces ten powtarza się rekurencyjnie, aż do osiągnięcia końcowych węzłów liściowych, które reprezentują klasy.

- `criterion`: Funkcja oceny podziału. Możliwe wartości to `'gini'` (współczynnik Giniego) i `'entropy'` (entropia).

- `max_depth`: Maksymalna głębokość drzewa. Ograniczenie głębokości może zapobiec nadmiernemu dopasowaniu (overfitting).
- `min_samples_split`: Minimalna liczba próbek potrzebna do podziału w węźle.
- `min_samples_leaf`: Minimalna liczba próbek w liściu. Pozwala na kontrolę wielkości końcowych węzłów.
- `max_features`: Maksymalna liczba cech brana pod uwagę przy każdym podziale. Może to być liczba całkowita, wartość zmiennoprzecinkowa lub `'auto'`, `'sqrt'`, `'log2'`.

2.1.4 Lasy losowe

Lasy losowe to technika zespołowa oparta na wielu drzewach decyzyjnych. Każde drzewo jest trenowane na losowym podzbiorze danych oraz cech. Wyniki klasyfikacji uzyskuje się poprzez głosowanie większościowe wśród wszystkich drzew. Lasy losowe zmniejszają problem nadmiernego dopasowania, który jest powszechny w pojedynczych drzewach decyzyjnych.

- `n_estimators`: Liczba drzew decyzyjnych w lesie. Większa liczba drzew zwiększa stabilność predykcji.
- `criterion`: Kryterium oceny podziału, takie jak `'gini'` lub `'entropy'`, podobnie jak w przypadku drzew decyzyjnych.
- `max_features`: Maksymalna liczba cech brana pod uwagę przy każdym podziale.
- `bootstrap`: Jeśli `True`, to próbki są losowane z zamianą (bootstrap). Jeśli `False`, nie jest stosowana zamiana.
- `max_depth`, `min_samples_split`, `min_samples_leaf`: Parametry te działają podobnie jak w przypadku pojedynczych drzew decyzyjnych.

2.1.5 SVM

Maszyny wektorów nośnych (SVM) to metoda klasyfikacji, która stara się znaleźć optymalną hiperpłaszczyznę, która maksymalnie rozdziela dane pomiędzy dwie klasy. SVM może działać zarówno liniowo, jak i nieliniowo, dzięki zastosowaniu tzw. *jąder* (kernels), które przekształcają dane na wyższe wymiary, aby umożliwić rozdzielenie nieliniowych danych.

- `C`: Parametr regularizacji. Wyższa wartość C sprawia, że model bardziej dopasowuje się do danych treningowych, ale może prowadzić do nadmiernego dopasowania.
- `kernel`: Funkcja jądrowa do przekształcania danych. Możliwe wartości to `'linear'`, `'poly'` (wielomianowa), `'rbf'` (jądro radialne) i `'sigmoid'`.
- `gamma`: Parametr jądra, który kontroluje zakres wpływu pojedynczego przykładu treningowego. Może być ustawiony na `'scale'` lub `'auto'`.
- `probability`: Jeśli `True`, to model będzie zwracał prawdopodobieństwa klas, co wymaga dodatkowego obliczenia.

2.1.6 Perceptron Wielowarstwowy (MLP)

Perceptron wielowarstwowy (MLP) to algorytm uczenia głębokiego bazujący na sztucznych sieciach neuronowych. To jedna z najprostszych form sieci neuronowych, będąca modelem uczenia nadzorowanego. Jego struktura składa się z trzech głównych warstw: warstwy wejściowej, jednej lub więcej warstw ukrytych oraz warstwy wyjściowej. Każda warstwa składa się z neuronów, które są podstawowymi jednostkami przetwarzającymi dane.

- `hidden_layer_sizes`: Liczba neuronów w każdej warstwie ukrytej. Można podać jedną wartość (liczba neuronów w jednej warstwie) lub krotkę definiującą liczbę neuronów w kilku warstwach. Domyślnie: (100,).
- `activation`: Funkcja aktywacji dla neuronów. Możliwe wartości to `'identity'`, `'logistic'`, `'tanh'` i `'relu'`.
- `solver`: Algorytm używany do optymalizacji. Możliwe wartości to `'lbfgs'` (optymalizacja metodą quasi-Newtona), `'sgd'` (stochastyczny gradient prosty) i `'adam'` (optymalizacja adaptacyjna).
- `alpha`: Parametr regularyzacji L_2, który zapobiega nadmiernemu dopasowaniu.
- `learning_rate`: Szybkość uczenia. Możliwe wartości to `'constant'`, `'invscaling'` oraz `'adaptive'`.
- `max_iter`: Maksymalna liczba iteracji podczas treningu.

3 Charakterystyka wybranych do danych

MIT-BIH Arrhythmia Database to szeroko stosowany zestaw danych wykorzystywany w badaniach dotyczących analizy sygnałów elektrokardiograficznych (EKG) oraz automatycznej klasyfikacji arytmii. Zbiór ten zawiera 48 zapisów sygnałów EKG zarejestrowanych u 47 pacjentów, w tym zarówno osób z różnymi typami arytmii, jak i zdrowych. Każdy zapis obejmuje około 30 minut sygnału EKG, który został pobrany w częstotliwości 125 Hz.

Dane zostały ręcznie oznakowane przez kardiologów, co pozwala na identyfikację różnych typów arytmii, takich jak np. skurcze dodatkowe, migotanie przedsionków czy blokady serca. MIT-BIH Arrhythmia Database jest często wykorzystywany w algorytmach sztucznej inteligencji do trenowania systemów do automatycznej diagnozy arytmii.
Kroki używane do ekstrakcji uderzeń z sygnału EKG według autorów [1] były następujące:

1. Podział ciągłego sygnału EKG na okna 10s i wybór jednego okna 10s z sygnału EKG.
2. Normalizacja wartości amplitudy do zakresu od zera do jeden.
3. Znalezienie zbioru wszystkich lokalnych maksimów na podstawie zerokrosów pierwszej pochodnej.
4. Znalezienie zbioru kandydatów na szczyty R EKG poprzez zastosowanie progu 0.9 na znormalizowanej wartości lokalnych maksimów.
5. Znalezienie mediany interwałów czasowych R-R jako nominalnego okresu bicia serca dla danego okna (T).
6. Dla każdego szczytu R wybór części sygnału o długości równej 1.2T.
7. Uzupełnienie każdej wybranej części zerami, aby jej długość była równa zdefiniowanej stałej długości.

Liczba próbek: 109446
Liczba kategorii: 5
Częstotliwość próbkowania: 125Hz
Klasy:

Normalne uderzenie	0
Przedwczesne uderzenie nadkomorowe	1
Przedwczesny skurcz komorowy	2
Fuzja skurczu komorowego i normalnego uderzenia	3
Niezdyscyplinowane uderzenie	4

4 Eksperymenty i wyniki

4.1 Eksperyment nr 1

4.1.1 Założenia

Analiza wektora wag neuronu w kontekście wielokrotnego zastosowania algorytmu treningowego, w którym liczba wag neuronu N jest mniejsza niż liczba próbek treningowych M (czyli $N < M$).

Tabela 1: Założenia parametrów wyjściowych - eksperyment nr 1

Parametr	Wartość
Liczba wag neuronu (N)	5
Liczba wzorców treningowych (M)	10
Zakres wartości wag neuronu	[-1, 1]
Liczba epok (K)	14000
Krok treningowy	0.8

Tabela 2: Założenia wag dla eksperymentu nr 1

Neruron	Wagi początkowe neuronu
1	[0.7022602 0.01750665 0.79466921 0.92162914 0.90279926]
2	[0.41036501 0.54634851 0.1133011 0.26291431 0.10229738]
3	[0.35029543 0.53505635 0.51316328 0.62997295 0.84010926]
4	[0.09378692 0.3522956 0.81628097 0.48155047 0.33005786]
5	[0.97057748 0.33640186 0.21091255 0.69016781 0.7327417]

4.1.2 Przebieg

Algorytm treningowy dla neuronu liniowego został uruchomiony, korzystając z parametrów przedstawionych w Tabeli 1. Za każdym razem wykorzystano ten sam zbiór danych treningowych wygenerowane losowo. Wartości wag były inicjowane losowo przy uruchomieniu.

4.1.3 Rezultat

Tabela 3: Rezultaty eksperymentu nr 1

Neruron	Wagi końcowe neuronu
1	[1.10044389 5.19383826 0.27373934 1.73954117 -4.72341669]
2	[1.10044389 5.19383826 0.27373934 1.73954117 -4.72341669]
3	[1.10044389 5.19383826 0.27373934 1.73954117 -4.72341669]
4	[1.10044389 5.19383826 0.27373934 1.73954117 -4.72341669]
5	[1.10044389 5.19383826 0.27373934 1.73954117 -4.72341669]

Tabela 4: Wyniki 1 neuronu z wykorzystaniem zbioru treningowego jako wektory wejściowe neuronu wytrenowanego:

Numer przypadku	Klasyfikacja neurona	R. klasyfikacja
1	1.62866269	1
2	0.75101212	1
3	0.88028287	0
4	2.23147244	1
5	1.40843858	1
6	2.43287648	1
7	2.78033059	0
8	4.54720541	1
9	4.44872991	0
10	3.10839639	1

4.2 Eksperyment nr 2

Analiza wektora wag neuronu w kontekście wielokrotnego zastosowania algorytmu treningowego, w którym liczba wag neuronu N jest równa liczbie próbek treningowych M (czyli $N = M$).

Tabela 5: Założenia parametrów wyjściowych - eksperyment nr 2

Parametr	**Wartość**
Liczba wag neuronu (N)	5
Liczba wzorców treningowych (M)	5
Zakres wartości wag neuronu	[-1, 1]
Liczba epok (K)	14000
Krok treningowy	0.8

Tabela 6: Założenia wag dla eksperymentu nr 2

Neruron	**Wagi początkowe neuronu**
1	[0.25718081 0.23160417 0.12374798 0.2100194 0.43299329]
2	[0.9268277 0.03963894 0.59703405 0.13471647 0.06900094]
3	[0.30719827 0.43147593 0.11097351 0.13570207 0.27112304]
4	[0.63129549 0.00481451 0.72005156 0.74442693 0.73818147]
5	[0.23434677 0.5381695 0.76920115 0.57128691 0.6599291]

4.2.1 Przebieg

Algorytm treningowy dla neuronu liniowego został uruchomiony, korzystając z parametrów przedstawionych w Tabeli 5. Za każdym razem wykorzystano ten sam zbiór danych treningowych wygenerowane losowo. Wartości wag były inicjowane losowo przy uruchomieniu.

4.2.2 Rezultat

Tabela 7: Rezultaty eksperymentu nr 2

Neruron	Wagi końcowe neuronu
1	[1.75545992 0.72352631 -0.45735995 0.30204395 -0.62659508]
2	[1.75545992 0.72352631 -0.45735995 0.30204395 -0.62659508]
3	[1.75545992 0.72352631 -0.45735995 0.30204395 -0.62659508]
4	[1.75545992 0.72352631 -0.45735995 0.30204395 -0.62659508]
5	[1.75545992 0.72352631 -0.45735995 0.30204395 -0.62659508]

Tabela 8: Wyniki 1 neuronu z wykorzystaniem zbioru treningowego jako wektory wejściowe neuronu wytrenowanego:

Numer przypadku	Klasyfikacja neurona	R. klasyfikacja
1	1.00000000e+00	1
2	1.00000000e+00	1
3	-4.62567283e-16	0
4	1.00000000e+00	1
5	1.00000000e+00	1

4.3 Eksperyment nr 3

Analiza wektora wag neuronu w kontekście wielokrotnego zastosowania algorytmu treningowego, w którym liczba wag neuronu N jest większa niż liczba próbek treningowych M (czyli $N > M$).

Tabela 9: Założenia parametrów wyjściowych - eksperyment nr 3

Parametr	**Wartość**
Liczba wag neuronu (N)	5
Liczba wzorców treningowych (M)	2
Zakres wartości wag neuronu	[-1, 1]
Liczba epok (K)	14000
Krok treningowy	0.8

Tabela 10: Założenia wag dla eksperymentu nr 3

Neruron	**Wagi początkowe neuronu**
1	[0.042668 0.24044789 0.39155331 0.72852958 0.18479858]
2	[0.82432786 0.08099875 0.74098371 0.63442987 0.18990254]
3	[0.47082526 0.67922145 0.38200882 0.77416749 0.02424662]
4	[0.63795142 0.32744639 0.56587628 0.52658763 0.81397631]
5	[0.44989125 0.83818744 0.62580719 0.2009764 0.86055305]

4.3.1 Przebieg

Algorytm treningowy dla neuronu liniowego został uruchomiony, korzystając z parametrów przedstawionych w Tabeli 9. Za każdym razem wykorzystano ten sam zbiór danych treningowych wygenerowane losowo. Wartości wag były inicjowane losowo przy uruchomieniu.

4.3.2 Rezultat

Tabela 11: Rezultaty eksperymentu nr 3

Neruron	Wagi końcowe neuronu
1	[0.10632304 0.80405649 0.4017791 0.45961974 0.28680895]
2	[0.70692232 0.59012639 0.63854747 0.09975784 0.12163089]
3	[0.38247018 0.85732429 0.31598488 0.5082556 -0.04300429]
4	[0.43267451 0.5126257 0.42481907 0.06093346 0.64003502]
5	[0.27104984 0.54160379 0.52762757 0.10003904 0.67356591]

Tabela 12: Wyniki 1 neuronu z wykorzystaniem zbioru treningowego jako wektory wejściowe neuronu wytrenowanego:

Numer przypadku	Klasyfikacja neurona	R. klasyfikacja
1	1	1
2	1	1

5 Wnioski

Wnioski z przeprowadzonych eksperymentów dowodzą, że

- Na wyjściu dla przypadku N $<$ M wyniki znacznie różniły się od oczekiwanych wartości dla wzorców treningowych. Powód można znaleźć w nie dostatecznej ilości wag.
- Dla przypadku N $=$ M oraz N $>$ M wartość końcowych wag nie są zmienne. Oznacza to brak przeszkód w nauce wzorców oraz przewidywań wartości wag. W przypadku N $<$ M nie jest w stanie skutecznie nauczyć wzorzec. Zwraca ona różne wartości wag.
- Dało się zaobserwować podobieństwo działania neuronu liniowego do problemu rozwiązywania układu równań z wieloma niewiadomymi.

Bibliografia

[1] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. ECG heartbeat classification: A deep transferable representation. *CoRR*, abs/1805.00794, 2018.