

Odpowiedzi na pytania do egzaminu z APU

Wykłady 1–3

1 Wykład

1. Definicja procesów uczenia

Nie ma jednej definicji procesów uczenia się:

- “Uczenie się oznacza zmiany w systemie, które mają charakter adaptacyjny w tym sensie, że pozwalają systemowi wykonać za następnym razem takie same zadanie lub zadania podobne bardziej efektywnie” - Herbert Simon (1983)
- “System uczący się wykorzystuje zewnętrzne dane empiryczne w celu tworzenia i aktualizacji podstaw dla udoskonalania działania na podobnych danych w przyszłości oraz wyrażania tych podstaw w zrozumiałej i symbolicznej postaci” - Donald Miche (1991)
- “Uczenie się to konstruowanie i zmiana reprezentacji doświadczanych faktów. W ocenie konstruowanych reprezentacji bierze się pod uwagę:
 - (a) wiarygodność - określa stopień w jakim reprezentacja odpowiada rzeczywistości;
 - (b) efektywność - charakteryzuje przydatność reprezentacji do osiągnięcia danego celu;
 - (c) poziom abstrakcji - odpowiada zakresowi szczegółowości i precyzji pojęć używanych w reprezentacji; określa on tzw. moc opisową reprezentacji. Reprezentacja jest rozumiana jako np. opisy symboliczne, algorytmy, modele symulacyjne, plany obrazy.” - Ryszard Michalski (1986)
- Elementem wspólnym tych definicji są: Wejście (dane empiryczne), miara oceny (Zmiany i poprawa działania) oraz postulat zdobywania wiedzy, reprezentowania jej wewnątrz systemu i stosowania jej do wykonania zadania (nacisk na zrozumiałość reprezentacji)

2. Przykłady problemów rozwiązywanych przez systemy uczące się

- Uczenie się rozpoznawania mowy
- Uczenie się kierowania pojazdem (np. ALVINN)
- Uczenie się klasyfikacji obiektów astronomicznych (NASA Sky Survey)
- Uczenie się rozgrywania pewnych gier
- Uczenie się rozpoznawania chorób na podstawie symptomów
- Uczenie się rozpoznawanie pisma na podstawie przykładów
- Uczenie się klasyfikowania tekstów do grup tematycznych
- Uczenie się aproksymacji nieznanej funkcji na podstawie próbek
- Uczenie się odnajdowania drogi w nieznanym środowisku
- Automatyczne odkrywanie zależności funkcyjnych w danych
- Przewidywanie trendów w danych finansowych

3. Motywacje dla budowy systemów uczących się

- Zadania eksploracji i analizy danych, gdzie duże rozmiary zbiorów danych uniemożliwiają ich analizę w sposób nieautomatyczny (np. ekonomiczne lub medyczne bazy danych)
- Środowiska gdzie system musi się dynamicznie dostosowywać do zmieniających się warunków (np. systemy sterowania)
- Problemy które są złożone, trudne do opisu i często nie posiadają wystarczających modeli teoretycznych albo ich uzyskanie jest bardzo kosztowne lub mało wiarygodne.

4. Klasyfikacja metod maszynowego uczenia się

- Uczenie indukcyjne - na podstawie znanych faktów i obserwacji tworzona jest uogólnienie, które próbuje dopasować wszystkie znane fakty do hipotezy
- "Nabywanie umiejętności" - optymalizacja zastosowania już posiadanej wiedzy w sposób zwiększający jej efektywność

5. Tworzenie modelu uczenia maszynowego

- (a) Przygotowanie danych (zebranie danych, uzupełnienie braków, normalizacja)
- (b) Zdefiniowanie zadania (regresja, klasyfikacja, grupowanie, inne)
- (c) Wybór metody (regresja liniowa, logistyczna, drzewa decyzyjne, sieci neuronowe)
- (d) Strojenie parametrów
- (e) Ocena modelu

6. Język R. Wykonywanie instrukcji

Sekwencyjne, możliwe wpisywanie kolejnych poleceń z konsoli lub z pliku skryptu o rozszerzeniu *.R. W konsoli występuje znak zachęty >. Komentarz tworzony jest przez #. Zapisywanie ma postać `zmienna <- wartość`.

7. Korzystanie z pomocy R

- `help(max)` - klasyczna pomoc
- `?max` - skrócona wersja
- `example(max)` - przykłady użycia
- `RSiteSearch("max function")` - przeszukiwanie forum
- `apropos("max", mode = "function")` - wyszukiwanie z funkcji z nazwą "max"
- `data()` - nazwy obiektów w pakiecie
- `vignette()` - pdf z dokumentacją

8. Zarządzanie obszarem roboczym R

- `getwd()` - ścieżka do aktualnego katalogu roboczego
- `setwd("..")` - ustawia nową ścieżkę do katalogu roboczego
- `ls()` - wyświetla zmienne obszaru roboczego
- `remove(a, b, c)` - usuwa zmienne a, b, c z obszaru roboczego
- `history(3)` - 3 ostatnio użyte instrukcje
- `savehistory("plik.txt")` - zapisuje historię do pliku
- `loadhistory("plik.txt")` - odczyt historii z pliku
- `save.image("workspace.RData")` - zapis obszaru roboczego do pliku
- `save(a, b, c, file = "workspace.RData")` - zapis tylko kilku zmiennych do pliku
- `load("workspace.RData")` - odczyt obszaru roboczego z pliku

9. Pakiety rozszerzające R

- `installed.packages()` - lista zainstalowanych paczek
- `search()` - wyszukiwanie w załadowanych paczkach
- `install.packages("RWeka")` - instalowanie paczki RWeka
- `library(RWeka)` - ładowanie paczki RWeka

10. Skalary i wektory R

Skalary obejmują pojedyncze liczby, łańcuchy znaków i wartości logiczne. Typ każdej zmiennej i wyrażenia można sprawdzić funkcją `class(obj)`. Np. `class(2)` zwróci `[1] "numeric"`. Każda liczba (bez względu czy double czy int) to numeric.

Operacje na liczbach:

- `+`, `-`, `*`, `/` (dodawanie, odejmowanie, mnożenie, dzielenie)
- `%%` (modulo)
- `^` (potęga)
- `sqrt(n)` (pierwiastek kwadratowy)

Dla znaków łączenie nie działa z `+` tylko z funkcją `paste("a", "b", "c", sep=" ")`. Dla typów logicznych operacje porównania są takie same jak w C++ czy Csharp.

Wektor obejmuje wiele skalarów tego samego typu. Są jednowymiarowe. Do utworzenia wektora używa się funkcji `c(obj, ...)`. Próba utworzenia wektora o różnych typach spowoduje konwersję typów.

Można utworzyć sekwencję, która będzie również zachowywała się jak wektor. Tworzy się ją przy użyciu operatora `<liczba>:<liczba>` lub funkcji `seq(<liczba>, <liczba>, <krok>)`. Wektory indeksujemy od 1, a nie od 0. Ujemne wartości w indeksie to pobranie wszystkiego **oprócz** ostatnich N. W indeksie można użyć wektora indeksów. Zwróci on wtedy wektor z wartościami o indeksach jakie zostały wskazane. Można też użyć sekwencji do pobierania tych wartości.

Do sprawdzenia długości wektora używa się `length(vect)`.

11. Ramka danych R

Ramka danych to powiązanie dwóch wektorów o tej samej długości. Np. imię i wiek. Do jej utworzenia używa się `data.frame(label = vector, label = vector)`. `names(ramka)` pozwala na wyświetlenie nazw kolumn. Konstrukcja `names(ramka) <- c("abc", "cde")` umożliwia nadpisanie nazw w ramce. Indeksując ramkę indeksujemy wiersze, ale możemy użyć konstrukcji `frame[1, 1]` co zwróci nam dane tylko z pierwszej kolumny. Korzystając z operatora `ramka$abc` można pobrać kolumnę po nazwie. Można indeksować podając warunek: `ramka[ramka$abc < 10]`. Dodawanie nowego wiersza możliwe jest przy użyciu funkcji `rbind(old, new)`. Odpowiednio dodawanie kolumny wymaga funkcji `cbind(old, new)`.

Dodatkowe funkcje które pozwalają uzyskać informacje o ramce:

- `length(ramka)` - Liczba kolumn w ramce
- `dim(ramka)` - Wymiary ramki
- `str(ramka)` - Tekstowy opis zawartości ramki
- `head(ramka, n = 2)` - Wyświetlenie pierwszych 2 wierszy w ramce
- `tail(ramka, n = 2)` - Wyświetlenie ostatnich 2 wierszy w ramce

12. Przegląd wykresów

- Histogram

```
hist(vector,  
      breaks = unique(vector),  
      right = FALSE,  
      main = "Main title",  
      xlab = "Label x",  
      ylab = "Label y")
```

- Wykres pudełkowy - wizualizuje minimum, pierwszy kwartył (0.25), medianę, drugi kwartył (0.75) i wartość maksymalną

```
boxplot(iris$Sepal.Width,  
        main = "Podstawowe statystyki dla szerokosci kwiatu sepal")
```

- Wykres kołowy

```
pie(table(new_iris$Species) / length(new_iris$Species))
```

- Wykres wachlarzowy

```
install.packages("plotrix")  
library(plotrix)  
fan.plot(percentage, labels = names(percentage))
```

- Wykres słupkowy

```
count <- table(new_iris$Species)  
colors <- rainbow(3)  
barplot(count,  
        main = "Liczebność gatunków",  
        ylim = c(0, 50),  
        xlab = "Gatunek",  
        ylab = "Ilość",  
        col = colors)
```

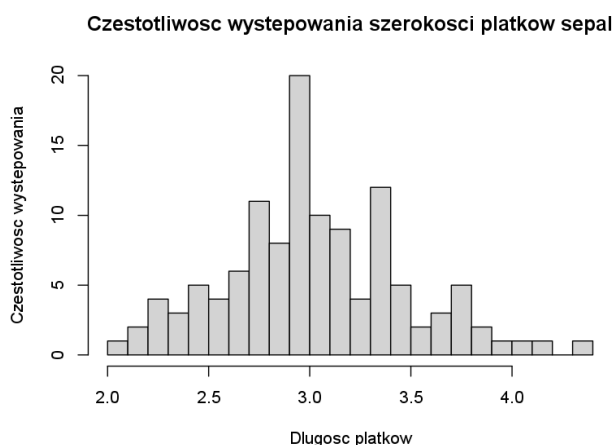


Figure 1: Histogram

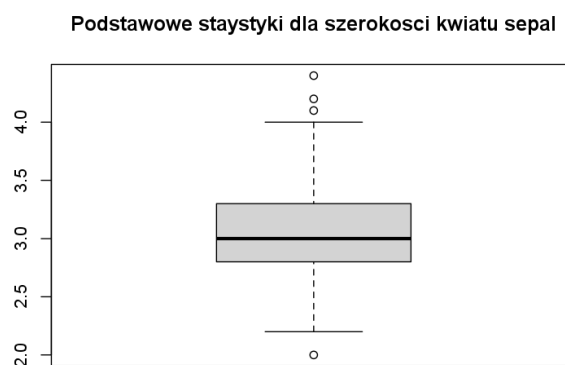


Figure 2: Wykres pudełkowy

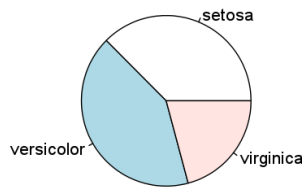


Figure 3: Wykres kołowy

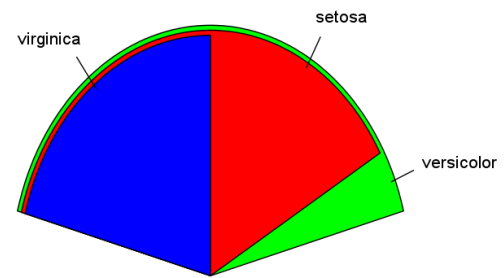


Figure 4: Wykres wachlarzowy

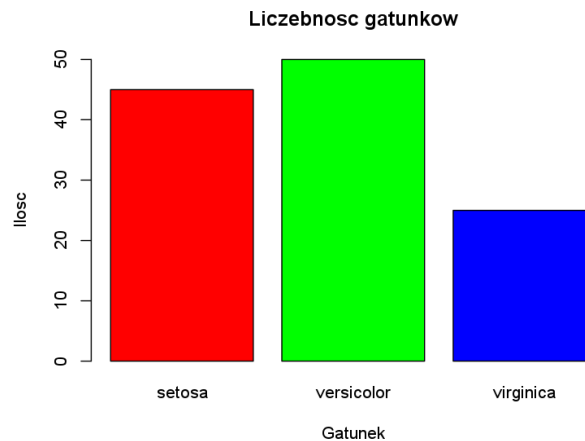


Figure 5: Wykres słupkowy

13. Język R i uczenie maszynowe

- Składniki uczenia maszynowego - szereg konceptów uczenia maszynowego
- Zbiór uczący - doświadczenie w postaci danych. Zbiór może być etykietowany lub nieetykietowany.
- Atrybuty (cechy) - opis obiektów zbioru uczącego
- Instancja - zestaw atrybutów opisujących jeden obiekt
- Model - utworzone na podstawie danych rozwiązanie zadania
 - Geometryczny - oparty o przestrzeń euklidesową. Tworzony przez regresję liniową, logistyczną lub maszynę wektorów wspierających (SVM, SMO)
 - Logiczny - oparty o logiki matematyczne. Zawiera proste reguły w postaci warunków prowadzących do decyzji. Tworzony przez drzewa decyzyjne, JRipper, PART
 - Probabilistyczny - oparty o prawdopodobieństwo warunkowe i regułę Bayesa
 - Hybrydowe
- Ewaluacja - sposoby oceny skuteczności modelu
 - Podział na zbiory uczący i testowy
 - 10-fold cross-validation - 10-krotne powtórzenie podziału zbioru uczącego. Efekt końcowy jest uśrednieniem poprzednich pomiarów.

2 Wykład

1. Analiza eksploracyjna i analiza potwierdzająca
2. Czym są dane w uczeniu maszynowym?
3. Wnioskowanie o typach danych w kolumnach
4. Podsumowania liczbowe w R
5. Średnie, mediany i dominanty w R
6. Kwantyle w R
7. Odchylenia standardowe i wariancje w R
8. Eksploracyjne wizualizacje danych
9. Wizualizowanie powiązań pomiędzy kolumnami
10. Klasyfikacja; Zdefiniowanie zadania
11. Trening i testowanie klasyfikacji
12. Kryteria porównawcze metod klasyfikacji
13. Metody klasyfikacji
14. Drzewa decyzyjne
15. Funkcje testu w celu konstruowania drzew decyzyjnych
16. Konstrukcja drzew decyzyjnych
17. Problem brakujących wartości przy konstruowaniu drzew decyzyjnych
18. Analiza ROC jakości klasyfikacji
19. Krzywe ROC
20. Czułość, a specyficzność klasyfikacji binarnej
21. Konstruowanie krzywych ROC
22. Pakiet ROCR

3 Wykład

1. Wieloatrybutowe problemy decyzyjne
2. Proces analitycznej hierarchizacji problemu decyzyjnego
3. Kroki rozwiązywania problemu AHP
4. Podstawy wieloatrybutowej teorii użyteczności
5. Agregacja ocen z wykorzystaniem macierzy porównań parami
6. Skala preferencji względnej
7. Ocena spójności macierzy porównań parami
8. Krok V – obliczenie priorytetów AHP

9. Obliczanie przybliżonego wektora własnego macierzy porównań parami
10. Inne metody rozwiązywania problemu AHP
11. Przykłady zastosowań metody AHP