zumz3b

April 12, 2018

0.1 Uczenie maszynowe UMZ 2017/2018

1 3. Naiwny klasyfikator bayesowski, drzewa decyzyjne

1.0.1 Cz 2

1.1 3.2. Drzewa decyzyjne

1.1.1 Drzewa decyzyjne – przykad

In [2]:
```
# Przydatne importy

import ipywidgets as widgets
import matplotlib.pyplot as plt
import numpy as np
import pandas

%matplotlib inline
```

In [3]:
```
alldata = pandas.read_csv('tennis.tsv', sep='\t')
alldata
```

Out[3]:

	Day	Outlook	Humidity	Wind	Play
0	1	Sunny	High	Weak	No
1	2	Sunny	High	Strong	No
2	3	Overcast	High	Weak	Yes
3	4	Rain	High	Weak	Yes
4	5	Rain	Normal	Weak	Yes
5	6	Rain	Normal	Strong	No
6	7	Overcast	Normal	Strong	Yes
7	8	Sunny	High	Weak	No
8	9	Sunny	Normal	Weak	Yes
9	10	Rain	Normal	Weak	Yes
10	11	Sunny	Normal	Strong	Yes
11	12	Overcast	High	Strong	Yes
12	13	Overcast	Normal	Weak	Yes
13	14	Rain	High	Strong	No

In [4]:
```
# Dane jako lista sowników
data = alldata.T.to_dict().values()
```

```
        features = ['Outlook', 'Humidity', 'Wind']

        # Moliwe wartoci w poszczególnych kolumnach
        values = {feature: set(row[feature] for row in data)
                  for feature in features}
        values
```

```
Out[4]: {'Humidity': {'High', 'Normal'},
         'Outlook': {'Overcast', 'Rain', 'Sunny'},
         'Wind': {'Strong', 'Weak'}}
```

- Czy John zagra w tenisa, jeeli bdzie pada, przy wysokiej wilgotnoci i silnym wietrze?
- Algorytm drzew decyzyjnych spróbuje *zrozumie* taktyk" Johna.
- Wykorzystamy metod dziel i zwyciaj".

```
In [5]: # Podziel dane
        def split(features, data):
            values = {feature: list(set(row[feature]
                                        for row in data))
                      for feature in features}
            if not features:
                return data
            return {val: split(features[1:],
                               [row for row in data
                                if row[features[0]] == val])
                    for val in values[features[0]]}
```

```
In [6]: split_data = split(['Outlook'], data)

        for outlook in values['Outlook']:
            print('\n\tOutlook\tHumid\tWind\tPlay')
            for row in split_data[outlook]:
                print('Day {Day}:\t{Outlook}\t{Humidity}\t{Wind}\t{Play}'
                      .format(**row))
```

```
        Outlook        Humid        Wind        Play
Day 3:       Overcast        High        Weak        Yes
Day 7:       Overcast        Normal        Strong        Yes
Day 12:       Overcast        High        Strong        Yes
Day 13:       Overcast        Normal        Weak        Yes

        Outlook        Humid        Wind        Play
Day 1:       Sunny        High        Weak        No
Day 2:       Sunny        High        Strong        No
Day 8:       Sunny        High        Weak        No
Day 9:       Sunny        Normal        Weak        Yes
Day 11:       Sunny        Normal        Strong        Yes
```

```
        Outlook        Humid        Wind        Play
Day 4:        Rain        High        Weak        Yes
Day 5:        Rain        Normal        Weak        Yes
Day 6:        Rain        Normal        Strong        No
Day 10:        Rain        Normal        Weak        Yes
Day 14:        Rain        High        Strong        No
```

Obserwacja: John lubi gra, gdy jest pochmurnie.
W pozostaych przypadkach podzielmy dane ponownie:

```
In [7]: split_data_sunny = split(['Outlook', 'Humidity'], data)

        for humidity in values['Humidity']:
            print('\n\tOutlook\tHumid\tWind\tPlay')
            for row in split_data_sunny['Sunny'][humidity]:
                print('Day {Day}:\t{Outlook}\t{Humidity}\t{Wind}\t{Play}'
                      .format(**row))
```

```
        Outlook        Humid        Wind        Play
Day 1:        Sunny        High        Weak        No
Day 2:        Sunny        High        Strong        No
Day 8:        Sunny        High        Weak        No

        Outlook        Humid        Wind        Play
Day 9:        Sunny        Normal        Weak        Yes
Day 11:        Sunny        Normal        Strong        Yes
```

```
In [8]: split_data_rain = split(['Outlook', 'Wind'], data)

        for wind in values['Wind']:
            print('\n\tOutlook\tHumid\tWind\tPlay')
            for row in split_data_rain['Rain'][wind]:
                print('Day {Day}:\t{Outlook}\t{Humidity}\t{Wind}\t{Play}'
                      .format(**row))
```

```
        Outlook        Humid        Wind        Play
Day 6:        Rain        Normal        Strong        No
Day 14:        Rain        High        Strong        No

        Outlook        Humid        Wind        Play
Day 4:        Rain        High        Weak        Yes
Day 5:        Rain        Normal        Weak        Yes
Day 10:        Rain        Normal        Weak        Yes
```

- Outlook=
- Overcast
  - Playing
- Sunny
  - Humidity=
  - High
    - Not playing
  - Normal
    - Playing
- Rain
  - Wind=
  - Weak
    - Playing
  - Strong
    - Not playing
- (9/5)
- Outlook=Overcast (4/0)
  - YES
- Outlook=Sunny (2/3)
  - Humidity=High (0/3)
  - NO
  - Humidity=Normal (2/0)
  - YES
- Outlook=Rain (3/2)
  - Wind=Weak (3/0)
  - YES
  - Wind=Strong (0/2)
  - NO

1.1.2 Algorytm ID3

- podziel(wze, zbiór przykadów):
  1. A najlepszy atrybut do podziau zbioru przykadów
  2. Dla kadej wartoci atrybutu A, utwórz nowy wze potomny
  3. Podziel zbiór przykadów na podzbiory wedug wzów potomnych
  4. Dla kadego wz potomnego i podzbioru:
     - jeeli podzbiór jest jednolity: zakocz
     - w przeciwnym przypadku: podziel(wze potomny, podzbiór)

Jak wybra najlepszy atrybut”? * powinien zawiera jednolity podzbiór * albo przynajmniej w miar jednolity”

Skd wzi miar jednolitoci” podzbioru? * miara powinna by symetryczna (4/0 vs. 0/4)

1.1.3 Entropia

$$H(S) = -p_{(+)} \log p_{(+)} - p_{(-)} \log p_{(-)}$$

- S – podzbiór przykadów
- $p_{(+)}$, $p_{(-)}$ – procent pozytywnych/negatywnych przykadów w S

Entropi mona traktowa jako liczb bitów" potrzebn do sprawdzenia, czy losowo wybrany $x \in S$ jest pozytywnym, czy negatywnym przykadem.

Entropia – przykad

- (3 TAK / 3 NIE):
$$H(S) = -\frac{3}{6} \log \frac{3}{6} - \frac{3}{6} \log \frac{3}{6} = 1 \text{ bit}$$
- (4 TAK / 0 NIE):
$$H(S) = -\frac{4}{4} \log \frac{4}{4} - \frac{0}{4} \log \frac{0}{4} = 0 \text{ bitów}$$

1.1.4 *Information gain*

Information gain – rónica midzy entropi przed podziaem a entropi po podziale (podczas podziau entropia zmienia si):

$$\text{Gain}(S, A) = H(S) - \sum_{V \in \text{Values(A)}} \frac{|S_V|}{|S|} H(S_V)$$

***Information gain* – przykad**

$$\text{Gain}(S, Wind) = H(S) - \frac{8}{14} H(S_{Wind=\text{Weak}}) - \frac{6}{14} H(S_{Wind=\text{Strong}}) = \ = 0.94 - \frac{8}{14} \cdot 0.81 - \frac{6}{14} \cdot 1.0 = 0.049$$

- *Information gain* jest cakiem sensown heurystyk wskazujc, który atrybut jest najlepszy do dokonania podziau.
- **Ale**: *information gain* przeszacowuje uyteczno atrybutów, które maj duo rónych wartoci.
- **Przykad**: gdybymy wybrali jako atrybut *dat*, otrzymalibymy bardzo duy *information gain*, poniewa kady podzbiór byby jednolity, a nie byoby to ani troch uyteczne!

1.1.5 *Information gain ratio*

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{-\sum_{V \in \text{Values}(A)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}}$$

- *Information gain ratio* moe by lepszym wyborem heurystyki wskazujcej najuyteczniejszy atrybut, jeeli atrybuty maj wiele rónych wartoci.

1.1.6 Drzewa decyzyjne a formuy logiczne

Drzewo decyzyjne mona przeksztaci na formu logiczn w postaci normalnej (DNF):

$$Play = \text{True} \Leftrightarrow (Outlook = \text{Overcast} \vee (Outlook = \text{Rain} \wedge Wind = \text{Weak}) \vee (Outlook = \text{Sunny} \wedge Humidity = \text{No}$$

1.1.7 Klasyfikacja wieloklasowa przy uyciu drzew decyzyjnych

Algorytm przebiega analogicznie, zmienia si jedynie wzór na entropi:

$$H(S) = -\sum_{y \in Y} p_{(y)} \log p_{(y)}$$

1.1.8 Skuteczno algorytmu ID3

- Przyjmujemy, e wród danych uczcych nie ma duplikatów (tj. przykadów, które maj jendakowe cechy x, a mimo to nale do rónych klas y).
- Wówczas algorytm drzew decyzyjnych zawsze znajdzie rozwizanie, poniewa w ostatecznoci bdziemy mieli wzy 1-elementowe na liciach drzewa.

1.1.9 Nadmierne dopasowanie drzew decyzyjnych

- Zauwamy, e w miar postpowania algorytmu dokadno przewidywa drzewa (*accuracy*) liczona na zbiorze uczcym dy do 100% (i w ostatecznoci osiga 100%, nawet kosztem jednoelementowych lici).
- Takie rozwizanie niekoniecznie jest optymalne. Dokadno na zbiorze testowym moe by duo nisza, a to oznacza nadmierne dopasowanie.

1.1.10 Jak zapobiec nadmiernemu dopasowaniu?

Aby zapobiega nadmiernemu dopasowaniu drzew decyzyjnych, naley je przycina (*pruning*).

Mona tego dokonywa na kilka sposobów: * Mona zatrzymywa procedur podziau w pewnym momencie (np. kiedy podzbiory staja si zbyt mae). * Mona najpierw wykona algorytm ID3 w caoci, a nastpnie przyci drzewo, np. kierujc si wynikami uzyskanymi na zbiorze walidacyjnym. * Algorytm *sub-tree replacement pruning* (algorytm zachanny).

Sub-tree replacement pruning

1. Dla kadego wza:
2. Udaj, e usuwasz wze wraz z caym zaczepionym w nim poddrzewem.
3. Dokonaj ewaluacji na zbiorze walidacyjnym.
4. Usu wze, którego usunicie daje najwiksz popraw wyniku.
5. Powtarzaj, dopóki usuwanie wzów poprawia wynik.

1.1.11 Zalety drzew decyzyjnych

- Zasad dziaania drzew decyzyjnych atwo zrozumie czowiekowi.
- Atrybuty, które nie wpywaj na wynik, maj *gain* równy 0, zatem s od razu pomijane przez algorytm.
- Po zbudowaniu, drzewo decyzyjne jest bardzo szybkim klasyfikatorem (zoono $O(d)$, gdzie d jest gbokocia dzewa).

1.1.12 Wady drzew decyzyjnych

- ID3 jest algorytmem zachannym – moe nie wskaza najlepszego drzewa.
- Nie da si otrzyma granic klas (*decision boundaries*), które nie s równolege do osi wykresu.

1.2 3.3. Lasy losowe

1.2.1 Algorytm lasów losowych – idea

- Algorytm lasów losowych jest rozwiniciem algorytmu ID3.
- Jest to bardzo wydajny algorytm klasyfikacji.
- Zamiast jednego, bdziemy budowa k drzew.

1.2.2 Algorytm lasów losowych – budowa lasu

1. We losowy podzbiór S_r zbioru uczcego.
2. Zbuduj pene (tj. bez przycinania) drzewo decyzyjne dla S_r, uywajc algorytmu ID3 z nastpujcymi modyfikacjami:

- podczas podziau uywaj losowego d-elementowego podzbioru atrybutów,
- obliczaj *gain* wzgldem S_r.

1. Powysz procedur powtórz k-krotnie, otrzymujc k drzew $(T_1, T_2, \ldots, T_k)$.

1.2.3 Algorytm lasów losowych – predykcja

1. Sklasyfikuj x wedug kadego z drzew $T_1, T_2, \ldots, T_k$ z osobna.
2. Uyj gosowania wikszociowego: przypisz klas przewidzian przez najwicej drzew.