

**Uniwersytet Warszawski**  
Wydział Matematyki, Informatyki i Mechaniki

**Grzegorz Szpak**

Nr albumu: 319400

# **Klasyfikacja wielowymiarowych szeregów czasowych przy ewoluujących pojęciach**

Praca magisterska  
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem  
**dra Andrzeja Janusza**  
Instytut Informatyki

Grudzień 2016

## **Oświadczenie kierującego pracą**

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

## **Oświadczenie autora (autorów) pracy**

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora (autorów) pracy

## **Streszczenie**

W pracy przedstawiono sposoby wydajnej klasyfikacji szeregów czasowych dla danych pochodzących ze źródła o zmiennym rozkładzie. Opisane zostały metody ekstrakcji cech z wielowymiarowego szeregu czasowego. Autor opisuje także metody wyboru przestrzeni atrybutów odpornej na zmiany rozkładu źródła.

## **Słowa kluczowe**

Eksploracja danych, wielowymiarowy szereg czasowy, ewoluujące pojęcia, dopasowanie dziedziny, ekstrakcja cech, selekcja cech, lasy losowe, regresja logistyczna, maszyna wektorów wspierających

## **Dziedzina pracy (kody wg programu Socrates-Erasmus)**

11.4 Sztuczna inteligencja TODO

## **Klasyfikacja tematyczna**

D. Software TODO

## **Tytuł pracy w języku angielskim**

Classification of multivariate time series in the presence of concept drift



# Spis treści

<b>1. Wprowadzenie</b>	5
1.1. Uczenie z nadzorem a <i>concept drift</i>	5
1.2. Formalizacja problemu	6
1.2.1. Paradygmaty uczenia się	6
1.2.2. Ewolucja pojęć a <i>overfitting</i>	7
<b>2. Opis przeprowadzanego eksperymentu</b>	9
2.1. Przedstawienie badanego problemu	9
2.1.1. Opis zbioru danych	9
2.1.2. Ewaluacja jakości klasyfikatora	11
2.2. Opis użytych klasyfikatorów	11
2.2.1. Klasyfikator liniowy	11
2.2.2. Lasy losowe	13
2.3. Przykład Ewolucji pojęć w analizowanym zbiorze	14
<b>3. Ekstrakcja cech z szeregów czasowych</b>	15
3.1. Metody klasyfikacji szeregów czasowych	15
3.2. Proces ekstrakcji cech	15
3.2.1. Reprezentacje pochodne szeregu czasowego	16
3.2.2. Statystyki wyliczane z reprezentacji szeregu	19
3.2.3. Pozostałe cechy	23
<b>4. Redukcja <i>concept drift</i> poprzez selekcję cech</b>	25
4.1. Sprowadzenie problemu adaptacji dziedziny do problemu klasyfikacji	25
4.2. Miary jakości cech	27
4.3. Wykrywanie ewolucji pojęć między zbiorem treningowym a testowym	29
4.3.1. Wyniki eksperymentów	29
4.4. Detekcja zmiany dziedziny przy użyciu klasteryzacji	36
4.4.1. Algorytmy klasteryzacji	37
4.4.2. Opis algorytmu detekcji dziedziny	37
4.5. Wnioski z przeprowadzonych eksperymentów	37
<b>5. Inne metody adaptacji dziedziny</b>	39
5.1. Uczenie iteracyjne	39
5.1.1. Opis algorytmu	39
5.1.2. Wyniki i wnioski	40
5.2. Powiększenie przestrzeni cech	40
5.2.1. Opis algorytmu	40
5.2.2. Wyniki i wnioski	41

<b>6. Podsumowanie</b> . . . . .	43
<b>Bibliografia</b> . . . . .	45

# Rozdział 1

## Wprowadzenie

### 1.1. Uczenie z nadzorem a *concept drift*

Podstawowym problemem rozważanym w teorii uczenia maszynowego jest problem uczenia z nadzorem (ang. *supervised learning*). Niech dane będą:

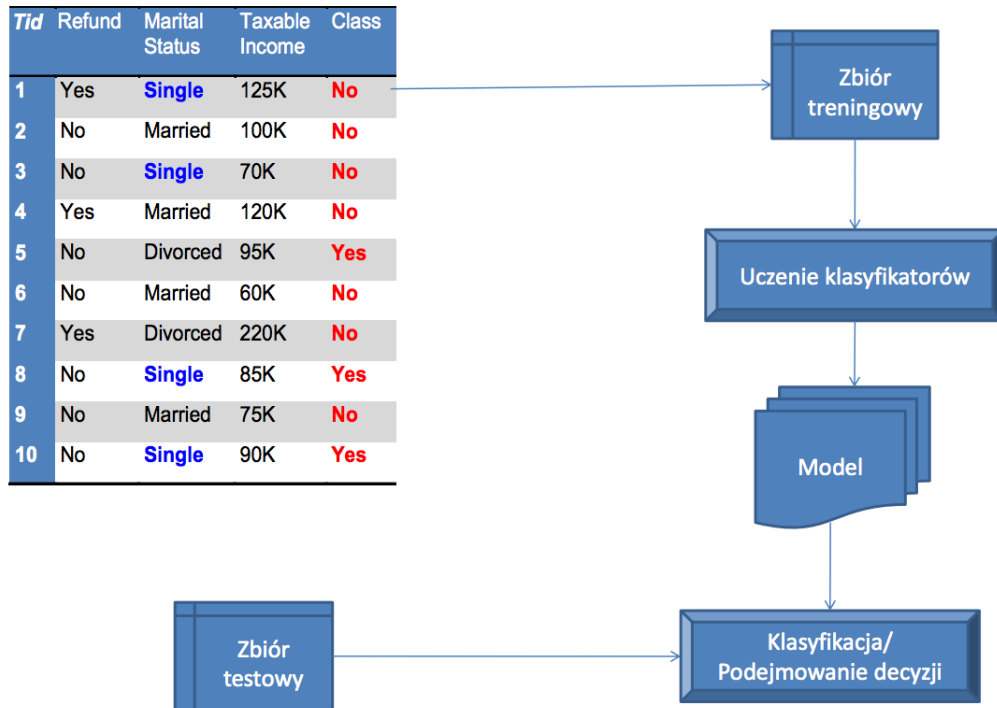
- zbiór  $X$  przykładów
- zbiór  $Y$  decyzji
- funkcja  $f : X \rightarrow Y$
- Parę zbiorów ( $X_{train} \subseteq X, Y_{train} \subseteq Y$ ) instancji  $x_1, x_2, \dots, x_n$  oraz odpowiadających im decyzji  $f(x_1), f(x_2), \dots, f(x_n)$ , zwaną *zbiorem treningowym*

Zadanie uczenia z nadzorem polega na wyznaczeniu na podstawie zbioru treningowego oraz przy użyciu pewnego algorytmu uczącego (*klasyfikatora*) takiej funkcji  $c : X \rightarrow Y$  (zwanej *modelem*) będącej dobrą aproksymacją funkcji  $f$ . Jakość modelu  $c$  określa się, porównując jego wartości dla elementów skończonego zbioru *testowego*  $X_{test}$  z rzeczywistymi wartościami funkcji  $f$  dla tych elementów. Istotne jest przy tym założenie, że elementy zbiorów  $X_{train}$  oraz  $X_{test}$  losowane są ze zbioru  $X$  według tego samego rozkładu prawdopodobieństwa  $D$ . Schemat rozwiązywania problemu uczenia z nadzorem przedstawia rys. 1.1.

Na przestrzeni ostatnich dziesięcioleci opracowanych zostało wiele algorytmów uczących wykazujących się dużą skutecznością w przeróżnych dziedzinach: od rozpoznawania obrazów, przez klasyfikację tekstów, rekomendację produktów, wykrywanie spamu, po przewidywanie zmian na giełdzie czy diagnostykę medyczną.

Sytuacja zmienia się diametralnie, gdy pominiemy założenie o równości rozkładów dla zbiorów treningowych i testowych. Wiele klasyfikatorów cierpi wtedy na znaczny spadek jakości. Problem ten nazywa się *ewolucją pojęć* (ang. *concept drift*) lub *dopasowaniem dziedziny* (ang. *domain adaptation*). Jest on szczególnie widoczny w zadaniach przetwarzania języka naturalnego (ang. *natural language processing*, w skrócie NLP). Rozpatrzmy dla przykładu problem rozpoznawania nazw własnych (ang. *named entity recognition*). Załóżmy, że klasyfikator uczony jest na podstawie danych encyklopedycznych oraz testowany na danych pochodzących z komunikatora internetowego. Obydwa zbiory, jakkolwiek powiązane, różnią się w znaczący sposób - przykładowo, szukanie wielkich liter może być bardzo pomocne w pierwszej dziedzinie, a nieść znacznie mniej informacji w wiadomościach z komunikatora.

Stąd też właśnie w dziedzinie NLP powstało najwięcej metod mających rozwiązać problem



Rysunek 1.1: Schemat uczenia z nadzorem

ewoluujących pojęć. Przykładami takich metod są algorytm *structural correspondence learning* opisywany w [1] czy metoda odpowiedniego dopasowania przestrzeni parametrów zaproponowana przez Daume w [2].

Problem *domain adaptation* nie jest jednak często poruszany w przypadku klasyfikacji szeregów czasowych. W poniższej pracy autor przedstawia sposoby radzenia sobie z *concept drift* podczas klasyfikacji szeregów czasowych oraz wykonuje studium przypadku na wybranym zbiorze danych.

## 1.2. Formalizacja problemu

### 1.2.1. Paradygmaty uczenia się

Przyjmijmy definicje jak na początku sekcji 1.1. W zależności od rozkładów  $D_{train}$ ,  $D_{test}$  oraz od dostępności zbiorów  $Y_{train}$ ,  $X_{test}$ ,  $Y_{test}$ , można (za [3]) zdefiniować inne paradygmaty uczenia.

I tak, jeśli zbiór  $Y_{train}$  jest nieznany w momencie tworzenia modelu, mamy do czynienia z *uczeniem bez nadzoru* (ang. *unsupervised learning*).

Gdy zbiór  $X_{test}$  nie jest znany podczas uczenia, mowa o *uczeniu indukcyjnym* (ang. *inductive learning*). W przeciwnym razie takie uczenie nazywa się *uczeniem transdukcyjnym* (ang. *transductive learning*).

W powyższym przykładach istotne jest założenie, iż zbiory  $X_{train}$ ,  $X_{test}$  pochodzą z tego samego rozkładu  $D$ . Odwrotna sytuacja rozpatrywana jest w paradygmacie *uczenia z przeniesieniem wiedzy* (ang. *transfer learning*). Przyjmuje się w nim, że dane są dwa różne rozkłady



$D^{source}$  i  $D^{target}$ . Model wyuczony na danych treningowych  $X_{train}^{source}, Y_{train}^{source}$  wykorzystywany jest zatem do klasyfikacji zbioru testowego  $X_{test}^{target}, Y_{test}^{target}$  pochodzących z rozkładu  $D^{target}$ . W poniższej pracy autor skupia się na problemie *dopasowania dziedziny*, który zakłada, że zbiór dostępnych klas  $Y$  jest ten sam dla  $D^{source}$  i  $D^{target}$ . Przeciwnieństwem dopasowania dziedziny jest zadanie *uczenia wielozadaniowego* (ang. *multi-task learning*, więcej między innymi w [4]), gdzie zbiory  $X_{train}, X_{test}$  pochodzą z tego samego rozkładu, natomiast zbiory  $Y_{train}, Y_{test}$  są różne.

Powyższe rozważania podsumowuje tabela 1.1.

Tabela 1.1: Paradygmaty uczenia w teorii uczenia maszynowego.

We wszystkich przypadkach zakładamy, że zbiór  $X_{train}$  jest dostępny podczas uczenia, podczas gdy zbiór  $Y_{test}$  nie jest znany.

Paradygmat	$Y_{train}$ dostępny?	$X_{test}$ dostępny?	Rozkład danych testowych
Indukcyjne uczenie bez nadzoru	Nie	Nie	$D^{source}$
Transdukcyjne uczenie bez nadzoru	Nie	Tak	$D^{source}$
Indukcyjne uczenie z nadzorem	Tak	Nie	$D^{source}$
Transdukcyjne uczenie z nadzorem	Tak	Tak	$D^{source}$
Indukcyjne uczenie bez nadzoru z przeniesieniem wiedzy	Nie	Nie	$D^{target}$
Transdukcyjne uczenie bez nadzoru z przeniesieniem wiedzy	Nie	Tak	$D^{target}$
Indukcyjne uczenie z nadzorem z przeniesieniem wiedzy	Tak	Nie	$D^{target}$
Transdukcyjne uczenie z nadzorem z przeniesieniem wiedzy	Tak	Tak	$D^{target}$

W poniższej pracy autor skupi się na problemie transdukcyjnego uczenia z nadzorem z przeniesieniem wiedzy. Przedstawione zostaną algorytmy, które wykorzystują dostępny zbiór  $X_{test}$  do znalezienia reprezentacji odpornej na zmiany rozkładu, co skutkować będzie zwiększoną jakością klasyfikacji w stosunku do standardowego podejścia opisanego w 1.1.

### 1.2.2. Ewolucja pojęć a *overfitting*

Mówiąc o problemie ewoluujących pojęć, należy wspomnieć o zagadnieniu przeuczenia (ang. *overfitting*). Polega on na zbudowaniu nadmiernie skomplikowanego modelu, co skutkuje słabą jego jakością.

Obydwa pojęcia mogą być mylone przy niewłaściwym sposobie walidacji modelu. Jeśli jakość klasyfikacji sprawdzana jest wyłącznie na zbiorach treningowym i testowym, zarówno *concept drift*, jak i *overfitting* dają podobne objawy - wysoki wynik na zbiorze treningowym

oraz niski na zbiorze testowym. W przypadku przeuczenia jest to spowodowane nadmiernym dopasowaniem modelu do danych treningowych i jego niską zdolnością do uogólniania. Jeśli mamy do czynienia z ewoluującymi pojęciami, słaba jakość modelu jest spowodowana innym rozkładem dla zbioru testowego.

W rozróżnieniu obydwu sytuacji pomagać może zastosowanie *walidacji krzyżowej* (ang. *cross-validation*) na zbiorze treningowym. Walidacja krzyżowa polega na podziale zbioru treningowego na  $n$  równolicznych części. Następnie budowane jest  $n$  modeli, przy czym  $n - 1$  części tworzy zbiór treningowy, natomiast pozostała część - zbiór testowy. Ostateczny wynik jest średnim wynikiem powstałych  $n$  modeli.

Nietrudno zauważyć, że w przypadku *concept drift* nie powinno się zauważyć znacznego spadku jakości modelu przy wykonaniu walidacji krzyżowej - w tym przypadku zbiór testowy pochodzi z tej samej dziedziny co treningowy. Inaczej będzie w przypadku przeuczenia - tu wynik walidacji krzyżowej będzie wyraźnie niższy niż wynik na zbiorze treningowym (tabela 1.2).

Tabela 1.2: *Concept drift* a *overfitting* - obniżony wynik modelu 1. przy walidacji krzyżowej świadczy o przeuczeniu, a nie występowaniu ewoluujących pojęć. W drugim przypadku model mimo dobrej umiejętności klasyfikacji elementów pochodzących z rozkładu  $D^{source}$ , cierpi na spadek jakości przy ewaluacji na zbiorze pochodzącym z rozkładu  $D^{target}$ .

	Wynik na $X_{train}$	Wynik CV	Wynik na $X_{test}$
Model 1	0.98	0.73	0.68
Model 2	0.97	0.92	0.76

## Rozdział 2

# Opis przeprowadzanego eksperymentu

### 2.1. Przedstawienie badanego problemu

#### 2.1.1. Opis zbioru danych

Dane, na których sprawdzana była jakość analizowanych algorytmów, pochodzą z konkursu *AAIA '15 Data Mining Competition: Tagging Firefighter Activities at a Fire Scene*<sup>1</sup> organizowanego przez Uniwersytet Warszawski oraz Szkołę Główną Służby Pożarniczej w Warszawie. Na potrzeby konkursu zebrano odczyty pochodzące z "inteligentnego kombinezonu", który monitoruje ruchy oraz funkcje życiowe strażaka. W skład kombinezonu wchodziło po siedem akcelerometrów i żyroskopów umieszczonych na: tułowi, ramionach, dłoniach i nogach (rys. 2.1).



Rysunek 2.1: Rozmieszczenie czujników na ciele strażaka

Każda instancja w zbiorze danych składała się z zebranych w krótkich odcinkach czasu (około 1.8 s) odczytów z zamontowanych sensorów. Dla każdego akcelerometru oraz żyroskopu zebrano po 400 odczytów wzdłuż osi  $x$ ,  $y$ ,  $z$ , wraz ze względny czas wykonania odczytu. Dla każdego wiersza daje to 42 - wymiarowy szereg czasowy o długości 400. Jak łatwo wywnioskować, średni odstęp między kolejnymi odczytami wynosił około 4.5 ms. Zestaw

<sup>1</sup><https://knowledgepit.fedcsis.org/contest/view.php?id=106>

atrybutów uzupełniały dodatkowo 42 agregaty ze wskazań urządzeń monitorujących funkcje życiowe strażaka (takie jak EKG, częstotliwość oddechu, temperatura skóry). Nieprzetworzone dane zawierały więc  $400 * 43 + 42 = 17242$  atrybuty. Zarówno zbiór treningowy, jak i testowy składały się z 20000 przykładów. Bardzo istotny dla dla analizy tego zbioru danych okazał się fakt, że dane treningowe i testowe pochodziły od różnych czteroosobowych grup strażaków.

Zadaniem uczestników konkursu było przypisanie każdej instancji w zbiorze postury strażaka w danym momencie oraz wykonywanej przez niego czynności. Zastosowane algorytmy miały pomóc w stworzeniu systemu monitorującego bezpieczeństwo strażaka podczas akcji.

Jako że problem klasyfikacji wieloetykietowej nie jest tematem niniejszej pracy, autor postanowił skupić się na problemie przewidywania czynności wykonywanej przez strażaka. Zbiór klas liczył więc 16 elementów. Postawiony problem utrudniał dodatkowo fakt, że klasy były wysoce niezbalansowane - najczęstsza klasa (*manipulating*) wystąpiła w zbiorze treningowym 6349 razy, podczas gdy najrzadsza (*signal\_hose\_pullback*) - jedynie 98 razy. Rozkład klas przedstawia rys. 2.2.



Rysunek 2.2: Rozkład klas w zbiorze treningowym

### 2.1.2. Ewaluacja jakości klasyfikatora

Standardową miarą jakości jest precyzja (ang. *accuracy*):

$$ACC(c) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}[c(\mathbf{x}_i) = y_i]$$

W opisywanym konkursie zastosowano modyfikację tej miary zwaną *balanced accuracy*, zdefiniowaną następująco:

$$ACC_i(c) = \frac{|j : c(\mathbf{x}_j) = y_j = i|}{|j : y_j = i|} \quad (2.1)$$

$$BAC(c) = \frac{\sum_{i=1}^l ACC_i(c)}{l} \quad (2.2)$$

Nietrudno zauważyć, że miara *BAC* jest bardziej wrażliwa na błędną klasyfikację rzadkich klas niż standardowa miara *ACC*.

## 2.2. Opis użytych klasyfikatorów

W kolejnych rozdziałach przedstawione zostaną metody mające na celu zwiększenie jakości modelu budowanego na danych, w których obecny jest problem ewoluujących pojęć. Użyteczność tych metod sprawdzana będzie dla trzech różnych algorytmów uczenia: klasyfikatorze opartym na regresji logistycznej, maszynie wektorów nośnych (ang. *support vector machine*, SVM) oraz drzewach decyzyjnych.

### 2.2.1. Klasyfikator liniowy

Regresja logistyczna i maszyna wektorów wspierających należą do grupy klasyfikatorów liniowych. Przyjmijmy oznaczenia jak w sekcji 1.1. Załóżmy, że zbiór klas  $Y$  jest dwuelementowy - dla uproszczenia niech  $Y = \{+1, -1\}$ . Niech dalej  $y \in Y$  będzie decyzją dla elementu  $\mathbf{x} = (x_1, x_2, \dots, x_m) \in X \subseteq \mathbb{R}^m$ . Klasyfikatorem liniowym nazywamy  $m - 1$  - wymiarową hiperpłaszczyznę rozdzielającą punkty z  $X$ . Niech  $\mathbf{w} = (w_1, w_2, \dots, w_m) \in \mathbb{R}^m$  będzie wektorem współczynników tej hiperpłaszczyzny. Uczenie klasyfikatora liniowego zwykle polega na minimalizacji wartości pewnej funkcji błędu  $l(\mathbf{w})$  na zbiorze treningowym.

#### Klasyfikacja oparta na regresji logistycznej (*logit*)

Niech

$$g(z) = \frac{1}{1 + e^{-z}}$$

zwana będzie funkcją logistyczną. Jej wykres przedstawia rysunek 2.3.

Funkcja  $g$  jest oczywiście ciągła. Jednocześnie  $\lim_{z \rightarrow -\infty} g(z) = 0$  oraz  $\lim_{z \rightarrow \infty} g(z) = 1$ . Dzięki tym właściwościom nadaje się ona do modelowania prawdopodobieństwa jakiegoś zjawiska. Klasyfikator bazujący na regresji logistycznej modeluje prawdopodobieństwo należenia do klasy pozytywnej przez funkcję:

$$p_w(\mathbf{x}) = g(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}},$$

przy czym klasyfikacja dokonywana jest przez:



Rysunek 2.3: Wykres funkcji sigmoidalnej

$$c_w(\mathbf{x}) = 1 \iff p_w(\mathbf{x}) \geq \frac{1}{2}.$$

Uczenie klasyfikatora opartego na regresji logistycznej polega więc na znalezieniu hiperpłaszczyzny  $\mathbf{w}$ , która minimalizuje następującą funkcję straty:

$$l_{\log\_loss}(\mathbf{w}) = \sum_{i=1}^n \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}).$$

### Maszyna wektorów nośnych

Innym typem klasyfikatora liniowego jest (ang. *support vector machine*, SVM). Uczenie SVM polega na znalezieniu hiperpłaszczyzny o największej odległości od punktów obydwu klas (rys. 2.4) - z tego powodu SVM nazywany jest klasyfikatorem maksymalnego marginesu (ang. *max margin classifier*).



Rysunek 2.4: Płaszczyzna  $H_1$  nie rozdziela klas.  
Płaszczyzna  $H_2$  rozdziela je, ale z niewielkim marginesem.  
Płaszczyzna  $H_3$  rozdziela je z maksymalnym marginesem.

Założenie o liniowej separowalności klas jest jednak rzadko spełnione. Uczenie klasyfikatora SVM polega więc na znalezieniu hiperpłaszczyzny  $\mathbf{w}$ , która minimalizuje funkcję straty

daną jako

$$l_{\text{hinge\_loss}}(\mathbf{w}) = \sum_{i=1}^n \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i).$$

### Klasyfikator "jeden przeciwko wszystkim"

Jak wspomniano wcześniej, zarówno regresja logistyczna, jak i SVM są klasyfikatorami binarnymi - zakładają, że zbiór klas  $Y$  jest dwuelementowy. Aby użyć ich do klasyfikacji danego zbioru, potrzebna była metoda klasyfikacji wieloklasowej. Zastosowano podejście zwane "jeden przeciw wszystkim" (ang. *one-vs.-all*, OvA, bądź *one-vs.-rest*, OvR).

Niech  $Y = \{1, 2, \dots, k\}$ . Klasyfikacja OvA polega na nauczaniu  $k$  binarnych klasyfikatorów  $c_1, c_2, \dots, c_k$ . Klasa dla  $j$ -tego klasyfikatora zdefiniowana jest następująco:

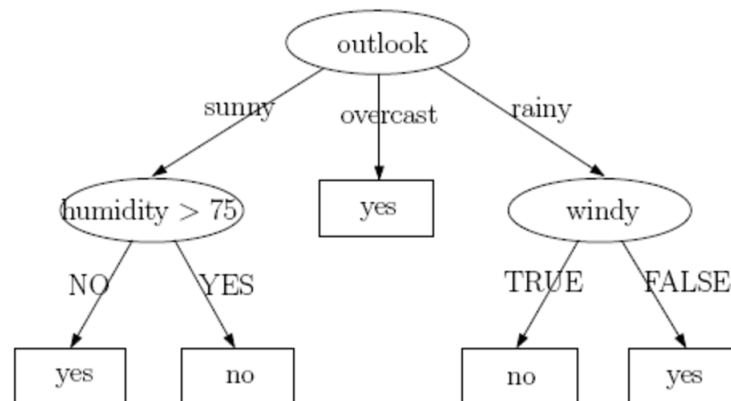
$$y_i^j = 1 \iff y_i = j$$

Ostatecznie elementowi  $x$  przypisywana jest klasa, której klasyfikator daje najmniejszą wartość funkcji błędu:

$$c(\mathbf{x}) = \arg \min_{j \in 1 \dots k} l_j(\mathbf{x})$$

### 2.2.2. Lasy losowe

Innym typem klasyfikatora jest las losowy. Jest to klasyfikator wykorzystujący prostszy klasyfikator - drzewo decyzyjne. Drzewo decyzyjne to etykietowane drzewo, którego każdy węzeł odpowiada przeprowadzeniu pewnego testu na wartościach atrybutów. Z węzła wewnętrznego wychodzi tyle gałęzi, ile jest możliwych wyników testu odpowiadającego temu węzłowi. Każdy liść zawiera decyzję o klasyfikacji obiektu. Przykładowe drzewo decyzyjne przedstawia rys. 2.5.



Rysunek 2.5: Przykładowe drzewo decyzyjne

Uczenie lasu losowego polega na uczeniu określonej liczby drzew decyzyjnych - każdego na losowej podprzestrzeni atrybutów. Finalna klasa jest modą klas wyznaczonych przez poszczególne drzewa. Lasy losowe, podobnie jak drzewa decyzyjne, wspierają klasyfikację wieloklasową.

## 2.3. Przykład Ewolucji pojęć w analizowanym zbiorze

Okazuje się, że głównym wyzwaniem w konkursie opisywanym w sekcji 2.1.1 są różnice w wykonywaniu poszczególnych czynności między strażakami. Aby się o tym przekonać, nauczono las losowy na 42 atrybutach opisujących funkcje życiowe, następnie zmierzono jego jakość (używając miary  $BAC$ ) na zbiorach treningowym i testowym. Do pomiaru jakości na zbiorze treningowym zastosowano trójwarstwową walidację krzyżową. Problem *concept drift* spowodował drastyczne obniżenie jakości klasyfikatora, która spadła z 0.993 na 0.079.



## Rozdział 3

# Ekstrakcja cech z szeregów czasowych

### 3.1. Metody klasyfikacji szeregów czasowych

Szeregi czasowe obecne są w wielu różnych dziedzinach życia - od medycyny, przez finanse, i wyszukiwanie informacji po przetwarzanie sygnałów oraz prognozę pogody. W ostatnich latach opisanych zostało wiele metod do wykrywania wzorców czasowych w takich zadaniach jak: określanie stanu zdrowia pacjenta na podstawie odczytów elektrokardiografu, przewidywanie wahań na giełdzie, analiza mowy czy prognozowanie temperatur.

Wśród metod zaproponowanych do klasyfikacji szeregów czasowych znaleźć można większość najbardziej znanych algorytmów uczenia się: metodę  $k$ -najbliższych sąsiadów (ang. *k-nearest neighbours*), w skrócie  $k$ -NN - więcej w [6]), sieci neuronowe ([7]) czy ukryty model Markowa ([8]). Wciąż jednak najszerzej używane (i uznawane za najbardziej skuteczne) są warianty algorytmu  $k$ -NN z użyciem różnych metryk, takich jak odległość euklidesowa czy Dynamic Time Warping (w skrócie DTW) - więcej o DTW przeczytać można między innymi w [9].

W poniższej pracy zastosowano nieco odmienne podejście. Autor koncentruje się na ekstrakcji cech opartych na statystycznych własnościach szeregów. Cechy te wyliczane są z różnych reprezentacji danego szeregu. Następnie na tak przekształconych danych uczone były klasyfikatory opisane w sekcji 2.2. Jak pokazują niektóre badania (przykładowo - [8], [9]), takie podejście daje często lepsze wyniki niż algorytmy bazujące na podobieństwie szeregów.

### 3.2. Proces ekstrakcji cech

Zacznijmy od formalnego zdefiniowania *szeregu czasowego*.

**Definicja 3.2.1** Jednowymiarowym szeregiem czasowym  $T_n$  nazwiemy skończony ciąg par

$$(t_1, x_1), (t_2, x_2), \dots, (t_n, x_n)$$

o długości  $n$ , gdzie  $x_k$  jest wartością szeregu w czasie  $t_k$ .

Proces ekstrakcji cech dla szeregu czasowego przebiegał następująco:

1. wyznaczenie pochodnych reprezentacji szeregu czasowego (więcej w sekcji 3.2.1)
2. Wyliczenie statystyk z wyznaczonych reprezentacji

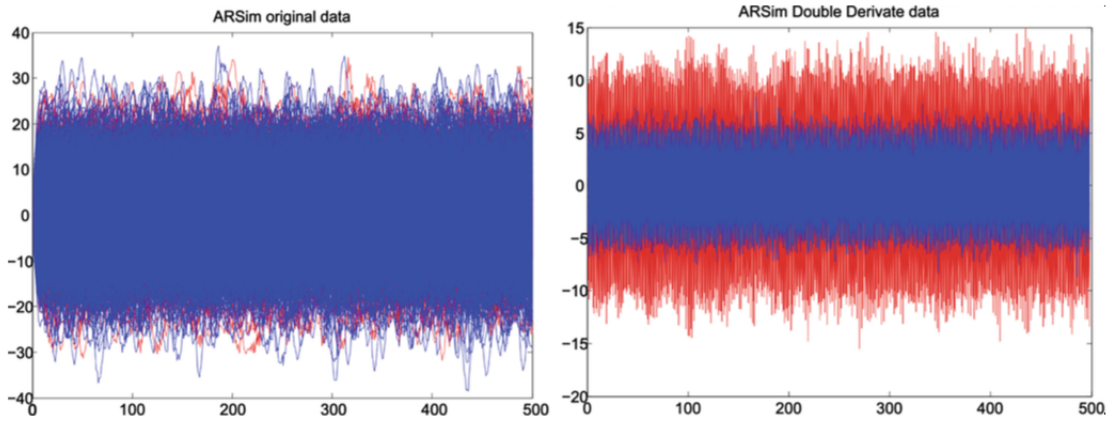
3. Dodanie pozostałych cech (bazujących między innymi na korelacji między szeregami)

W kolejnych sekcjach zostanie omówiony każdy z powyższych kroków.

### 3.2.1. Reprezentacje pochodne szeregu czasowego

Wiele przekształceń i reprezentacji szeregów czasowych zostało opisanych w literaturze - ich szerszy przegląd można znaleźć między innymi w TODO. Celem takich przekształceń jest wykrycie wzorców charakterystycznych dla klas wynikowych, które nie są wykrywalne przy użyciu podstawowej reprezentacji  $T_n$ .

Gay i in. w TODO podają przykład, kiedy to badanie drugiej pochodnej szeregu po czasie znacznie zwiększa jakość klasyfikacji (rys. 3.1). Działając na oryginalnych danych, trudno jest rozróżnić dwie klasy. Policzenie drugiej pochodnej sprawia, że klasyfikacja staje się niemalże trywialna.



Rysunek 3.1: Po lewej widać oryginalny zbiór danych, natomiast po prawej - drugą pochodną po czasie

Autor, poza reprezentacją oryginalną  $T_n$ , zdecydował się wykorzystać w procesie ekstrakcji cech cztery inne reprezentacje:

- pochodną szeregu po czasie  $der_{T_n}$
- całkę szeregu po czasie  $int_{T_n}$
- dyskretną transformatę Fouriera  $fourier_{T_n}$
- dyskretną transformatę falkową Haara  $wavelet_{T_n}$

#### Pochodna szeregu po czasie

Dyskretna pochodna szeregu  $T_n = (t_1, x_1), (t_2, x_2), \dots, (t_n, x_n)$  po czasie wyliczana była w sposób następujący:

$$der_{T_n}(i) = \frac{x_i - x_{i-1}}{t_i - t_{i-1}} \quad (3.1)$$

Takie przekształcenie pozwala na badanie lokalnej zmienności (monotoniczności i jej tempa) danego szeregu. Jako że badane dane (opisywane szerzej w sekcji 2.1.1) składały się z

odczytów z sensorów (akcelerometrów i żyroskopów), pochodna po czasie może być również interpretowana w sensie fizycznym. Przykładowo, miara zmienności przyspieszenia w czasie zwana jest zrywem<sup>1</sup>.

### Całka szeregu po czasie

Analogicznie do pochodnej zdefiniować można dyskretną całkę z szeregu  $T_n = (t_1, x_1), (t_2, x_2), \dots, (t_n, x_n)$ , wyliczaną metodą trapezów:

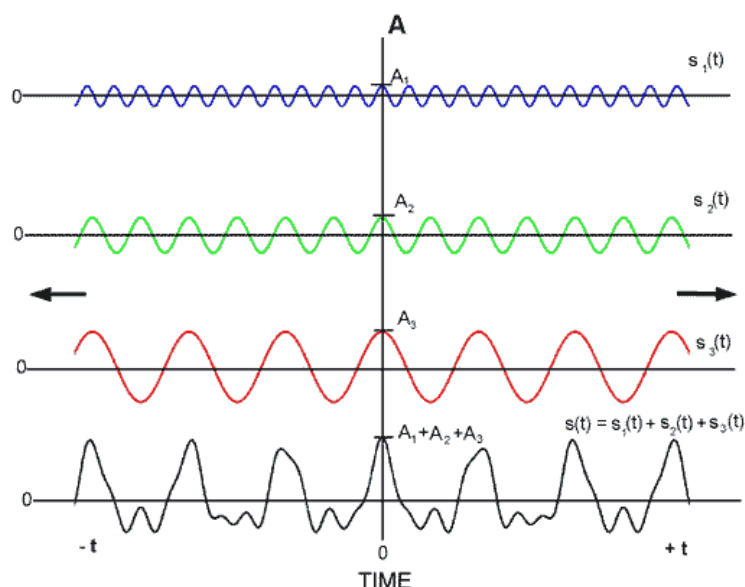
$$int_{T_n}(i) = \sum_{j=1}^i \frac{(x_j + x_{j-1})(t_i - t_{i-1})}{2} \quad (3.2)$$

Tak wyliczonej całce także można przypisać interpretację fizyczną - dla przykładu, wartość  $int_{T_n}(i)$  dla odczytu  $T_n$  z akcelerometru umieszczonego na lewej ręce strażaka oznacza prędkość, z jaką poruszała się ta ręka w czasie  $t_i$ .

Pozostałe dwa przekształcenia (transformata Fouriera oraz transformata falkowa) wyrażają szereg czasowy w bazie pewnej przestrzeni funkcyjnych.

### Transformacja Fouriera

Transformacja Fouriera jest jednym z najważniejszych przekształceń używanych do analizy sygnałów. Bazuje ona na odkryciu, że każdy szereg czasowy może być rozbity na sumę prostych, okresowych sygnałów - sinusów i cosinusów o zmieniających się amplitudzie i fazie (rys. 3.2).



Rysunek 3.2: Cosinusowe fale  $s_1$ ,  $s_2$ ,  $s_3$  dają razem bardziej złożony sygnał

<sup>1</sup><https://pl.wikipedia.org/wiki/Zryw>

Formalnie współczynniki dyskretnej transformaty Fouriera zdefiniowane są następująco:

$$fourier_{T_n}(k) = \sum_{i=0}^{n-1} x_i \cdot \left( \cos\left(\frac{-2\pi ki}{n}\right) + i \sin\left(\frac{-2\pi ki}{n}\right) \right) \quad (3.3)$$

Współczynniki  $fourier_{T_n}(k)$  są liczbami zespolonymi i reprezentują amplitudy oraz fazy sinusoidalnych składowych sygnału. Dla szeregów o wartościach rzeczywistych,  $fourier_{T_n}(i)$  jest zespolonym sprzężeniem  $fourier_{T_n}(n - i + 1)$  (dla  $i = 2, 3, \dots, \frac{n}{2}$ ). Istnieje algorytm (zwany *szybką transformacją Fouriera*, ang. *fast Fourier Transform*), obliczający współczynniki transformaty Fouriera w czasie  $\mathcal{O}(n \log n)$ .

### Transformacja falkowa

Przekształceniem podobnym do transformaty Fouriera jest transformacja falkowa. Jej główną przewagą nad transformacją opracowaną przez francuskiego matematyka jest zachowywanie informacji zarówno o częstotliwości, jak i o czasie. Podczas gdy przekształcenie Fouriera rzutuje sygnał z dziedziny czasu na dziedzinę częstotliwości, transformata falkowa mierzy częstotliwość w różnych momentach czasu - sygnał jest więc rzutowany na płaszczyznę czasu i częstotliwości.

*Dyskretna transformacja falkowa* również polega na przedstawieniu sygnału jako sumy sygnałów podstawowych. Podczas gdy transformata Fouriera używała funkcji sinusoidalnych, tutaj funkcję podstawową nazywa się *falką*. Funkcje falkowe są postaci:

$$\Psi_{j,k}(t) = 2^{\frac{j}{2}} \Psi(2^j t - k),$$

gdzie  $\Psi$  jest *macierzystą funkcją falkową*. Wtedy każda funkcja  $f \in L^2(\mathbb{R})$  może być przedstawiona w tej bazie jako

$$f(t) = \sum_{j,k} c_{j,k} \Psi_{j,k}(t),$$

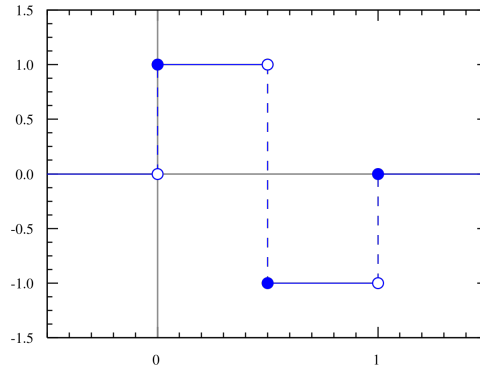
a  $c_{j,k} = \langle \Psi_{j,k}(t), f(t) \rangle$  są współczynnikami dyskretnej transformaty falkowej.

Najbardziej znaną macierzystą funkcją falkową jest falka Haara, zdefiniowana następująco:

$$\Psi_{Haar}(t) = \begin{cases} 0 & t < 0 \\ 1 & 0 \leq t < 0,5 \\ -1 & 0,5 \leq t < 1 \\ 0 & t \geq 1 \end{cases}$$

Wykres falki Haara przedstawia rys. 3.3. Funkcja węgierskiego matematyka została użyta przez autora w procesie ekstrakcji cech.

Podobnie jak w przypadku dyskretnej transformaty Fouriera, tak i tu istnieje szybki algorytm obliczający współczynniki dyskretnej transformaty falkowej (działający w czasie  $\mathcal{O}(n)$  - TODO bibliografia). Czyny to obydwie przekształcenia niezwykle użyteczne w praktyce.



Rysunek 3.3: Wykres falki Haara

### 3.2.2. Statystyki wyliczane z reprezentacji szeregu

Kolejnym krokiem w ekstrakcji cech było opisanie reprezentacji (przedstawionych w poprzedniej sekcji) za pomocą zbioru statystyk, mających jak najlepiej oddać charakter danego szeregu. Poniżej znajduje się lista statystyk (wraz z formalnym opisem) wykorzystanych przez autora.

- **Minimum**

Minimalna wartość przyjmowana w szeregu czasowym:

$$\min(T_n) = \min_{1 \leq i \leq n} x_i \quad (3.4)$$

- **Maksimum**

Maksymalna wartość przyjmowana w szeregu czasowym:

$$\max(T_n) = \max_{1 \leq i \leq n} x_i \quad (3.5)$$

- **Średnia arytmetyczna**

Średnia arytmetyczna wartości przyjmowanych w szeregu czasowym:

$$\text{mean}(T_n) = \frac{\sum_{i=1}^n x_i}{n} \quad (3.6)$$

- **Suma**

Suma wartości przyjmowanych w szeregu czasowym:

$$\text{sum}(T_n) = \sum_{i=1}^n x_i \quad (3.7)$$

- **Odchylenie standardowe**

Odchylenie wartości przyjmowanych w szeregu czasowym:

$$\text{std}(T_n) = \sqrt{\frac{\sum_{i=1}^n (x_i - \text{mean}(T_n))^2}{n}} \quad (3.8)$$

- **Kwantyle**

Kwantylem  $q_p(T_n)$  rzędu  $p$  nazwiemy taki element  $x_k$ , że dokładnie  $p$  elementów szeregu jest od niego mniejszych. W procesie ekstrakcji cech użyto siedmiu kwantyli, rzędów kolejno: 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875.

Kwantyl rzędu 0.5 oznaczany będzie dalej przez  $median(T_n)$ .

- **Wariancja**

Wariancja wartości w szeregu czasowym:

$$var(T_n) = std^2(T_n) \quad (3.9)$$

- **Błąd standardowy**

Błąd standardowy określa odchylenie standardowe dla rozkładu średniej z próby. Zdefiniowany jest jako:

$$sem(T_n) = \frac{std(T_n)}{\sqrt{n}} \quad (3.10)$$

- **Indeks pierwszego maksimum**

Indeks pierwszego maksimum szeregu (normalizowany przez długość szeregu):

$$first\_argmax(T_n) = \frac{\min(\{i : x_i = \max(T_n)\})}{n} \quad (3.11)$$

- **Indeks pierwszego minimum**

Indeks pierwszego minimum szeregu (normalizowany przez długość szeregu):

$$first\_argmin(T_n) = \frac{\min(\{i : x_i = \min(T_n)\})}{n} \quad (3.12)$$

- **Współczynnik skośności**

Współczynnik skośności rozkładu to miara asymetrii rozkładu. Przyjmuje on wartość zero dla rozkładu dla rozkładu symetrycznego, wartości ujemne dla rozkładów o lewostronnej asymetrii (wydłużone lewe ramię rozkładu) i wartości dodatnie dla rozkładów o prawostronnej asymetrii (wydłużone prawe ramię rozkładu) (rys. 3.4).

Traktując wartości szeregu jako wartości pewnej próbki statystycznej, można wyznaczyć jego współczynnik skośności, zdefiniowany jako:

$$skew(T_n) = \frac{3(mean(T_n) - median(T_n))}{std(T_n)} \quad (3.13)$$

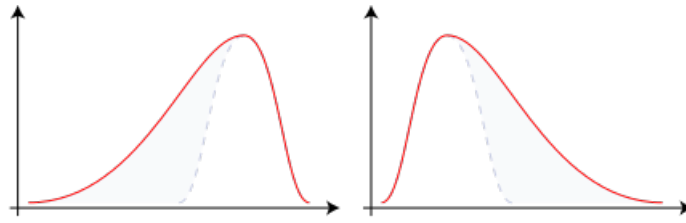
- **Kurtoza**

Kurtoza to miara koncentracji wyników wokół wartości centralnej. Jest to druga, obok skośności, miara kształtu rozkładu.

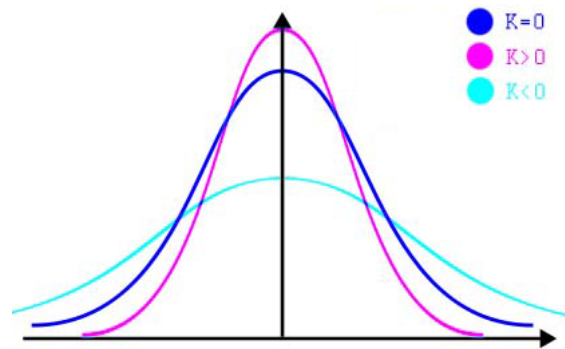
Kurtoza rozkładu normalnego wynosi 0. Jeśli wartość tej statystyki jest dodatnia, mamy do czynienia z rozkładem leptokurtycznym (wysmukłym). Jeśli zaś jest ujemna, rozkład jest rozkładem platykurtycznym (spłaszczonym) (rys. 3.5)

Formalnie kurtozę definiuje się następująco:

$$kurt(T_n) = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - mean(T_n))^4}{std^4(T_n)} - 3 \quad (3.14)$$



Rysunek 3.4: Rozkłady o ujemnym (pierwszy wykres) i dodatnim (drugi wykres) współczynniku skośności



Rysunek 3.5: Kształt rozkładu w zależności od wartości kurtozy

- **Średnia ważona liniowo**

Średnia ważona wartości w szeregu, przy czym wagi rosną liniowo wraz z indeksem. W ten sposób nadaje się największą wagę obserwacjom wykonanym najpóźniej (użycie tej statystyki dla szeregów czasowych opisują Wiens i in. w [11]):

$$linear\_weighted\_mean(T_n) = \frac{2}{n(n+1)} \sum_{i=1}^n ix_i \quad (3.15)$$

- **Średnia ważona kwadratowo**

Jak wyżej - z tą różnicą, że wagi rosną kwadratowo wraz z indeksem:

$$quadratic\_weighted\_mean(T_n) = \frac{6}{n(n+1)(2n+1)} \sum_{i=1}^n i^2 x_i \quad (3.16)$$

- **Średnie odchylenie bezwzględne od średniej**

Średnie odchylenie bezwzględne to kolejna, obok odchylenia standardowego i wariancji, miara rozrzutu próbki. Definiuje się je następująco:

$$mean\_absolute\_deviation(T_n) = \frac{\sum_{i=1}^n |x_i - mean(T_n)|}{n} \quad (3.17)$$

- **Mediana bezwzględnego odchylenia**

Jeszcze inną miarą rozrzutu jest mediana bezwzględnego odchylenia (ang. *median absolute deviation*, MAD). Jest to mediana ciągu bezwzględnych odchyleń od mediany:

$$median\_absolute\_deviation(T_n) = median_{1 \leq i \leq n} (|x_i - median(T_n)|) \quad (3.18)$$

- **Całkowita energia**

Całkowitą energię szeregu definiuje się jako sumę kwadratów jego wartości:

$$E(T_n) = \sum_{i=1}^n x_i^2 \quad (3.19)$$

- **Liczba elementów mniejszych od średniej**

Zdefiniowana w oczywisty sposób:

$$\text{count\_below\_mean}(T_n) = |\{i : x_i < \text{mean}(T_n)\}| \quad (3.20)$$

- **Liczba elementów większych od średniej**

Analogicznie do powyższego:

$$\text{count\_above\_mean}(T_n) = |\{i : x_i > \text{mean}(T_n)\}| \quad (3.21)$$

- **Długość najdłuższego podciągu o wartościach poniżej średniej**

Zdefiniowana formalnie:

$$\text{strike\_below\_mean}(T_n) = \max(\{j - i \mid 1 \leq i \leq j \leq n, \forall_{i \leq k \leq j} x_k < \text{mean}(T_n)\}) \quad (3.22)$$

- **Długość najdłuższego podciągu o wartościach powyżej średniej**

Analogicznie do powyższego:

$$\text{strike\_above\_mean}(T_n) = \max(\{j - i \mid 1 \leq i \leq j \leq n, \forall_{i \leq k \leq j} x_k > \text{mean}(T_n)\}) \quad (3.23)$$

- **Średnia autokorelacja**

**Definicja 3.2.2** Współczynnik korelacji Pearsona *wektorów prób losowych*  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$  zdefiniowany jest jako

$$\text{pearson\_corr}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}},$$

gdzie  $\bar{x}, \bar{y}$  oznaczają wartości średnie z tych prób, tj.  $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$ ,  $\bar{y} = \frac{\sum_{i=1}^n y_i}{n}$ .

Współczynnik Pearsona określa poziom zależności liniowej między zmiennymi losowymi i przyjmuje on wartości z przedziału  $[-1, 1]$ . Wartości bliskie 1 oznaczają silną zależność liniową między próbkami, wartości bliskie zera - brak liniowej zależności, natomiast wartości bliskie  $-1$  - ujemną liniową zależność (rys. 3.6).

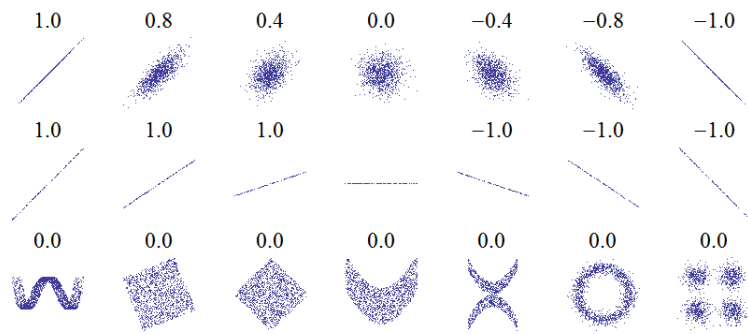
Korelacja Pearsona wykorzystywana jest przy wyliczaniu autokorelacji. Autokorelacja jest funkcją, która argumentowi naturalnemu  $k$  przypisuje wartość współczynnika korelacji pomiędzy szeregiem czasowym a tym samym szeregiem cofniętym o  $k$  jednostek czasu:

$$\text{autocorr}_k(T_n) = \frac{1}{(n-k)\text{std}^2(T_n)} \sum_{i=1}^{n-k} (x_i - \text{mean}(T_n))(x_{i+k} - \text{mean}(T_n))$$

Jako cecha używana była średnia wartość autokorelacji obliczanych dla wszystkich przesunięć  $k$  (od 1 do  $n$ ):

$$\text{mean\_autocorr}(T_n) = \frac{\sum_{k=1}^n \text{autocorr}_k(T_n)}{n} \quad (3.24)$$





Rysunek 3.6: Przykładowe wykresy danych  $(x, y)$  i odpowiadające im wartości współczynnika korelacji liniowej Pearsona

- **Asymetryczność odwracalna w czasie**

Ta statystyka (ang. *time-reversal asymmetry statistic*) została zaproponowana przez Fulchera i Jonesa w TODO. Zdefiniowana jest w zależności od parametru  $k$  przez:

$$TRAS_k(T_n) = \frac{1}{n-2k} \sum_{i=0}^{n-2k} x_{i+2k}^2 \cdot x_{i+k} - x_{i+k} \cdot x_i^2 \quad (3.25)$$

Jako cechy wykorzystane zostały wartości  $TRAS$  dla  $k = 25, 50, 100, 200$ .

- **Informacje o skokach**

(TODO Jak sensownie przetłumaczyć "peak"?????)

**Definicja 3.2.3** Skokiem o wsparciu  $n$  nazwiemy podciąg szeregu  $T_n$ , w którym istnieje wartość  $x$  będąca większa niż  $n$  wartości występujących bezpośrednio przed nią oraz  $n$  wartości występujących bezpośrednio po niej.

Przykładowo w ciągu  $(3, 0, 0, 4, 0, 0, 13)$  wartość 4 jest skokiem o wsparciu 1 i 2, ale nie jest skokiem o wsparciu 3.

Skoki wykorzystano przy ekstrakcji pięciu kolejnych cech, które oznaczały (kolejno) liczbę skoków o wsparciu: 5, 10, 25, 50, 100.

Funkcje *min*, *max*, *mean*, *std*, *quantiles* nazywane będą *statystykami podstawowymi*. Z reprezentacji pochodnych (pochodnej, całki, transformaty Fouriera, transformaty falkowej) wyliczane były tylko statystyki podstawowe. Wartości wszystkich opisanych wyżej statystyk obliczono tylko dla bazowej reprezentacji szeregu.

W ten sposób z 42 szeregów obecnych w zbiorze danych otrzymano około 4500 cech.

### 3.2.3. Pozostałe cechy

Moerchen udowadnia w (TODO biblio), że użyteczne przy klasyfikacji szeregów czasowych jest  $k$  największych współczynników transformaty Fouriera oraz transformaty falkowej. Autor zdecydował się więc dodać po 10 największych współczynników dla obydwu transformat wyliczonych dla każdego z szeregów.

Kolejne cechy oparte były na korelacji między dwoma szeregami czasowymi. Analogicznie do

autokorelacji, korelację między szeregami  $T_n = (t_1, x_1), \dots, (t_n, x_n)$  i  $T'_n = (t_1, x'_1), \dots, (t_n, x'_n)$  definiuje się jako korelację Pearsona (def 3.2.2) między wektorami  $(x_1, \dots, x_n)$  oraz  $(x'_1, \dots, x'_n)$ .

Ostatecznie otrzymano około 6000 cech (przypomnijmy - początkowa reprezentacja zbioru danych zawierała 17242 atrybuty).

Warto zwrócić uwagę, jak elastyczna jest powyższa metoda ekstrakcji cech - można zastosować ją do praktycznie dowolnego szeregu. Inną zaletą tej metody jest jej niedługi czas działania. Przykładowo, przekształcenie zbioru treningowego (składającego się z 20000 instancji) zajęło około dwóch godzin na komputerze wyposażonym w dwurdzeniowy procesor Intel Core i5 o taktowaniu 2,7 GHz.

## Rozdział 4

# Redukcja *concept drift* poprzez selekcję cech

### 4.1. Sprowadzenie problemu adaptacji dziedziny do problemu klasyfikacji

Zacznijmy od sprawdzenia jakości klasyfikatorów przedstawionych w rozdziale 2 na reprezentacji opisanej w poprzednim rozdziale przy użyciu metryki *BAC*. Wynik na zbiorze treningowym wyliczany był za pomocą trójwarstwowej walidacji krzyżowej. Rezultaty przedstawia tabela 4.1.

Tabela 4.1: Jakość klasyfikacji na wyekstrahowanych cechach

	Wynik na $X_{train}$	Wynik na $X_{test}$
Regresja logistyczna	0.95	0.53
SVM	0.96	0.53
Lasy losowe	0.99	0.71

Ponownie widoczny jest problem ewoluujących pojęć. Mimo rewelacyjnych wyników na zbiorze treningowym (klasyfikator bazujący na lasach losowych był niemalże bezbłędny), jakość predyktorów testowanych na innej grupie strażaków okazała się dużo niższa - wynik pogorszył się o ok. 0.3 w przypadku lasów losowych oraz o prawie 0.5 dla klasyfikatorów liniowych.

Celem tego rozdziału będzie przedstawienie metody wyboru takiej przestrzeni cech, która będzie spełniać dwa wymogi:

- będzie umożliwiać zbudowanie wydajnego klasyfikatora,
- będzie możliwie odporna na zmiany dziedziny

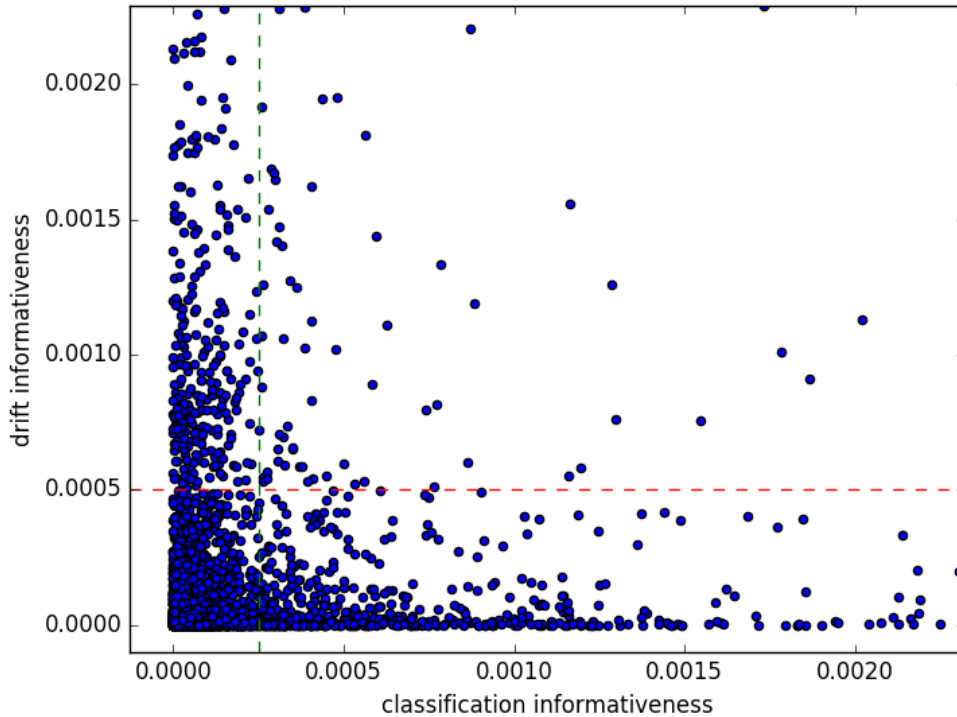
Opisywane podejście stanowi rozszerzenie pomysłu zaproponowanego przez M. Boule w (TODO bib) oraz (TODO bib). Intuicja jest następująca: założmy, że jesteśmy w stanie sprowadzić problem wykrywania *concept drift* do problemu klasyfikacji (zostanie to opisane szerzej w kolejnych podrozdziałach). Założmy też, że mamy daną funkcję  $f$ , która dla danego atrybutu określa jego jakość w odniesieniu do ustalonego problemu klasyfikacji (przykładem takich funkcji jest test  $\chi^2$ ).

Celem jest wybranie tych cech, które jednocześnie:

- dają najwyższy wynik dla oryginalnego problemu klasyfikacji,
- dają najniższy wynik dla problemu wykrywania ewoluujących pojęć.

Przy takim wyborze w oczywisty sposób zostaną spełnione postawione wcześniej wymagania. Intuicyjnie, jeśli będziemy wstanie wybrać reprezentację, która daje dobrą jakość klasyfikacji, ale słabo wykrywa zmianę dziedziny, to nasz klasyfikator będzie mniej czuły na ewolucję pojęć, co skutkować będzie zmniejszeniem różnic między zbiorami treningowym a testowym.

Rysunek 4.1 przedstawia rozkład jakości cech wraz z przykładowym wyborem intuicyjnie ”dobrej” reprezentacji.



Rysunek 4.1: Przykładowy rozkład jakości cech dla problemu klasyfikacji oraz wykrywania ewoluujących pojęć

Oś  $x$  przedstawia jakość cech dla bazowego problemu klasyfikacji, natomiast oś  $y$  - jakość dla problemu detekcji *concept drift*. Przykładowym wyborem jest selekcja cech poniżej czerwonej linii (atributy słabo rozróżniające różne dziedziny) oraz na prawo od linii zielonej (cechy o wysokim wyniku dla postawionego problemu klasyfikacji). Oczywiście nie zawsze taki wybór będzie możliwy, co pokażą kolejne podrozdziały.

Głównym wyzwaniem w opisywanym podejściu jest sprowadzenie problemu wykrywania zmiennej dziedziny do problemu klasyfikacji. Zanim jednak poruszona zostanie ta kwestia, autor zaprezentuje używane najczęściej miary jakości cech.

## 4.2. Miary jakości cech

Podstawowe miary jakości atrybutów pochodzą ze statystyki oraz teorii informacji. Do najbardziej znanych należą:

- współczynnik korelacji Pearsona
- test  $\chi^2$
- informacja wzajemna
- jednoczynnikowa analiza wariancji (ang. *ANOVA F-value*)
- zysk Giniego

W bieżącej sekcji zostanie przedstawiony opis każdej z tych miar.

### Współczynnik korelacji Pearsona

Zdefiniowany w 3.2.2 współczynnik korelacji liniowej Pearsona może być też używany do mierzenia jakości atrybutu. Niech przewidywana zmienna  $y$  będzie ciągła. Dla danego ciągłego atrybutu  $x$  wartość  $pearson\_corr(x, y)$  określa poziom liniowej zależności między tym atrybutem a  $y$ . Badanie korelacji jest często wykorzystywane przy analizie mikromacierzowej DNA (TODO bib).

### Test $\chi^2$

W statystyce testu  $\chi^2$  używa się do badania niezależności dwóch zdarzeń  $A$  i  $B$ , gdzie dwa zdarzenia są uznawane za niezależne, jeśli  $P(A \cap B) = P(A)P(B)$ , bądź równoważnie:  $P(A|B) = P(A)$  i analogicznie  $P(B|A) = P(B)$ . W uczeniu maszynowym test  $\chi^2$  stosowany jest do badania niezależności pomiędzy atrybutami a klasą wynikową. Za wartościowe uważane są te cechy, które nie są niezależne od przewidywanej zmiennej.

Test ten może być zastosowany tylko w przypadku gdy zarówno atrybut, jak i klasa wynikowa są zmiennymi kategorycznymi. W przypadku zmiennych ciągłych stosuje się dyskretyzację.

### Informacja wzajemna

Z punktu widzenia teorii informacji klasyfikator może być traktowany jako narzędzie redukujące poziom *niepewności* odnośnie przewidywanej zmiennej. Klasyfikator, "konsumując" informację niesioną przez kolejne instancje zbioru treningowego, stopniowo zmniejsza początkową niepewność zmiennej wynikowej. W przypadku klasyfikatora idealnego (to jest takiego, który zawsze dobrze przewiduje klasę wynikową) końcowa niepewność będzie wynosiła 0.

W teorii informacji początkową niepewność definiuje się formalnie przez pojęcie *entropii*.

**Definicja 4.2.1** *Jeśli przewidywana zmienna  $C$  jest zmienną dyskretną przyjmującą  $m$  wartości  $c_1, c_2, \dots, c_m$ , entropia definiowana jest jako*

$$H(C) = - \sum_{i=1}^m P_C(c_i) \log P_C(c_i)$$

Niech teraz atrybut  $F$  będzie atrybutem dyskretnym przyjmującym  $k$  wartości  $f_1, f_2, \dots, f_k$ . Wtedy niepewność po poznaniu wartości cechy  $F$  nazywa się *entropią warunkową*:

$$\begin{aligned} H(C|F) &= \sum_{i=1}^k P_C(f_i) H(C|F = f_i) = \\ &= - \sum_{i=1}^k P_F(f_i) \left( \sum_{j=1}^m P_{C|F}(c_j|f_i) \log P_{C|F}(c_j|f_i) \right) \end{aligned}$$

Zachodzi wtedy:

$$H(C|F) \leq H(C),$$

czyli entropia warunkowa jest nie większa niż entropia początkowa. Równość zachodzi wtedy i tylko wtedy, gdy zmienne  $C$  i  $F$  są niezależne.

*Informację wzajemną*  $I(C; F)$  definiuje się wtedy jako wartość, o jaką początkowa niepewność odnośnie klasy wynikowej została pomniejszona po poznaniu wartości cechy  $F$ , czyli formalnie:

$$I(C; F) = H(C) - H(C|F) \quad (4.1)$$

Po nietrudnych przekształceniach można otrzymać zwarty wzór na informację wzajemną:

$$I(C; F) = \sum_{i=1}^k \sum_{j=1}^m P_{CF}(c_j, f_i) \log \left( \frac{P_{CF}(c_j, f_i)}{P_C(c_j) P_F(f_i)} \right) \quad (4.2)$$

### Jednoczynnikowa analiza wariancji

Jednoczynnikowa analiza wariancji używa testu statystycznego  $F$  do mierzenia jakości cechy. Test  $F$  określa, czy wartości oczekiwane danej cechy  $X$  wewnątrz  $m$  określonych grup (odpowiadających  $m$  wartościom klasy wynikowej) różnią się między sobą. Wartość  $F$ -testu zdefiniowana jest jako:

$$F = \frac{MS_M}{MS_W}$$

$MS_M$  określa wariancję między grupami i jest zdefiniowana jako:

$$MS_M = \frac{\sum_{i=1}^m n_i (\bar{x}_i - \bar{x})^2}{m - 1},$$

gdzie  $n_i$  oznacza liczbę obserwacji w  $i$ -tej grupie,  $\bar{x}_i$  oznacza średnią w  $i$ -tej grupie, a  $\bar{x}$  - średnią z całej próbki.  $MS_W$  określa z kolei wariancję wewnątrz grup, zdefiniowaną jako:

$$MS_W = \frac{\sum_{ij} (x_{ij} - \bar{x}_i)^2}{n - m},$$

gdzie  $x_{ij}$  oznacza  $j$ -tą obserwację  $i$ -tej grupy.

### Zysk Giniego

Niech ponownie zmienna przewidywana  $C$  będzie zmienną dyskretną przyjmującą  $m$  wartości  $c_1, c_2, \dots, c_m$ .

**Definicja 4.2.2** Indeks *em* Giniego zmiennej  $C$  nazywa się *wartość*

$$Gini(C) = 1 - \sum_{i=1}^m P_C^2(c_i)$$

Podobnie jak entropia, indeks Giniego jest miarą *niejednorodności* zbioru. Przykładowo, dla jednej klasy indeks Giniego - niejednorodność zbioru wynosi  $1 - 1^2 = 0$ . Jeśli rozkład klas jest jednorodny (czyli  $P_C(c_i) = \frac{1}{m}$ ), indeks Giniego osiąga maksimum.

Ze względu na podobieństwo między indeksem Giniego a entropią, *zysk Giniego* definiuje się analogicznie do informacji wzajemnej. Dla atrybutu dyskretnego  $F$  przyjmującego wartości  $f_1, f_2, \dots, f_k$  zysk Giniego wyraża, o ile zmalała niejednorodność zbioru (jego indeks Giniego) po pozaniu wartości cechy  $F$ :

$$GiniGain(C; F) = Gini(C) - Gini(C|F) = Gini(C) - \sum_{i=1}^k P_F(f_i) Gini(C|F = f_i) \quad (4.3)$$

W dalszych eksperymentach autor wykorzystał trzy miary jakości cech: informację wzajemną, jednoczynnikową analizę wariancji oraz zysk Giniego.

### 4.3. Wykrywanie ewolucji pojęć między zbiorem treningowym a testowym

Wróćmy do problemu wykrywania ewolucji pojęć.

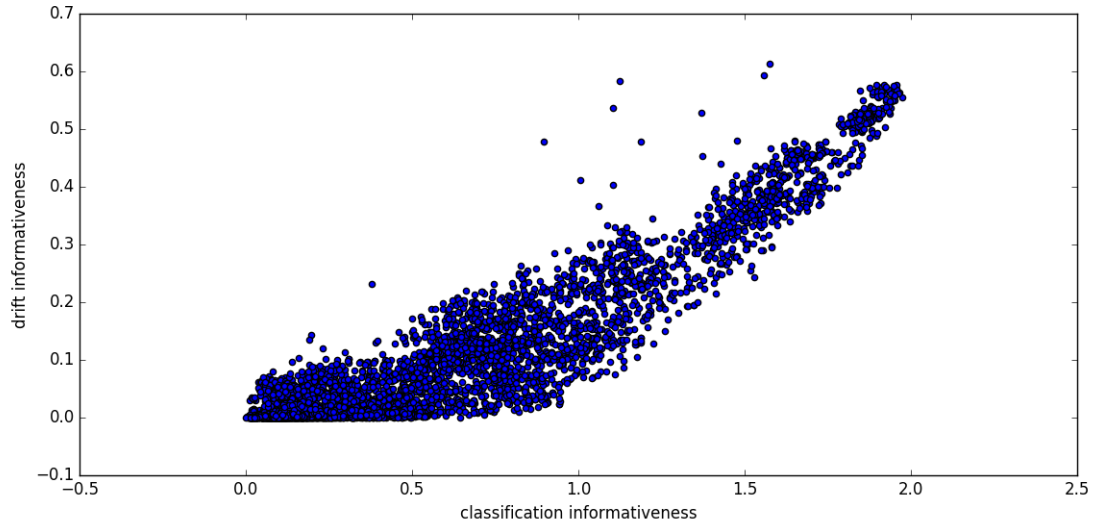
W celu wykrycia *concept drift* między zbiorem treningowym a zbiorem testowym wykorzystany został pomysł opisywany w (TODO biblio). Zbiory  $X_{train}$  i  $X_{test}$  zostały połączone w jeden:  $X = X_{train} \cup X_{test}$ . Klasa wynikowa była dwuelementowa i przyjmowała wartość "train" dla obiektów ze zbioru treningowego oraz "test" dla zbioru testowego. Następnie wybierane były te cechy, które dobrze przewidywały czynności wykonywane przez strażaków i jednocześnie słabo rozróżniały obiekty ze zbioru treningowego od obiektów ze zbioru testowego.

#### 4.3.1. Wyniki eksperymentów

Poniżej zaprezentowane zostaną rezultaty selekcji cech przy użyciu trzech miar jakości wspomnianych w poprzedniej sekcji. W opisie każdego z eksperymentów  $threshold_{drift}$  będzie oznaczał górny próg wartości miary przy detekcji *concept drift*, natomiast  $threshold_{class}$  - dolny próg wartości przy klasyfikacji czynności wykonywanej przez strażaka.

#### Informacja wzajemna

Rysunek 4.2 przedstawia rozkład jakości cech przy użyciu informacji wzajemnej jako miary. Widoczna jest wysoka dodatnia korelacja między jakością cechy dla klasyfikacji a jej jakością dla problemu ewolucji pojęć. Utrudnia to w oczywisty sposób dobranie dobrych wartości dla progów  $threshold_{drift}$  oraz  $threshold_{class}$  - eliminując atrybuty "słabe" dla detekcji *concept drift*, eliminujemy jednocześnie cechy "dobre" dla klasyfikacji czynności strażaka. Znajduje to swoje odzwierciedlenie w jakości otrzymanych klasyfikatorów. Tabele 4.2, 4.3, 4.4 prezentują jakość klasyfikacji dla różnych wartości  $threshold_{class}$  i  $threshold_{drift}$  odpowiednio dla klasyfikatorów: regresji logistycznej, SVM i lasów losowych. Pierwszy wiersz każdej tabeli zawiera wyniki bez selekcji atrybutów.



Rysunek 4.2: Rozkład jakości cech przy użyciu informacji wzajemnej

Tabela 4.2: Jakość klasyfikacji dla różnych wartości progowych przy użyciu informacji wzajemnej i regresji logistycznej

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	0.7	5355	0.9341	0.5227
0.0	0.05	2867	0.9116	0.516
0.0	0.1	3559	0.9201	0.509
0.0	0.2	4362	0.9209	0.5492
<b>0.25</b>	<b>0.05</b>	<b>950</b>	<b>0.8494</b>	<b>0.5468</b>
0.25	0.1	1448	0.915	0.4899
0.25	0.2	2246	0.9471	0.5313
0.5	0.05	266	0.7615	0.5008
0.5	0.1	597	0.9292	0.5499
0.5	0.2	1379	0.9662	0.5453
0.5	0.4	2044	0.944	0.5225
1.0	0.1	15	0.1636	0.1358
1.0	0.3	443	0.7416	0.4221
1.0	0.5	920	0.7788	0.4255
1.5	0.7	486	0.6942	0.4673



Tabela 4.3: Jakość klasyfikacji dla różnych wartości progowych przy użyciu informacji wzajemnej i SVM

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	0.7	5355	0.9501	0.53
0.0	0.05	2867	0.9062	0.5668
0.0	0.1	3559	0.9256	0.5374
0.0	0.2	4362	0.9267	0.5379
<b>0.25</b>	<b>0.05</b>	<b>950</b>	<b>0.8594</b>	<b>0.5756</b>
0.25	0.1	1448	0.8985	0.5295
0.25	0.2	2246	0.9342	0.5507
0.5	0.05	266	0.7179	0.3884
0.5	0.1	597	0.9039	0.524
0.5	0.2	1379	0.9114	0.5153
0.5	0.4	2044	0.9213	0.5266
1.0	0.1	15	0.1719	0.1192
1.0	0.3	443	0.7094	0.4488
1.0	0.5	920	0.6926	0.4644
1.5	0.7	486	0.696	0.2582

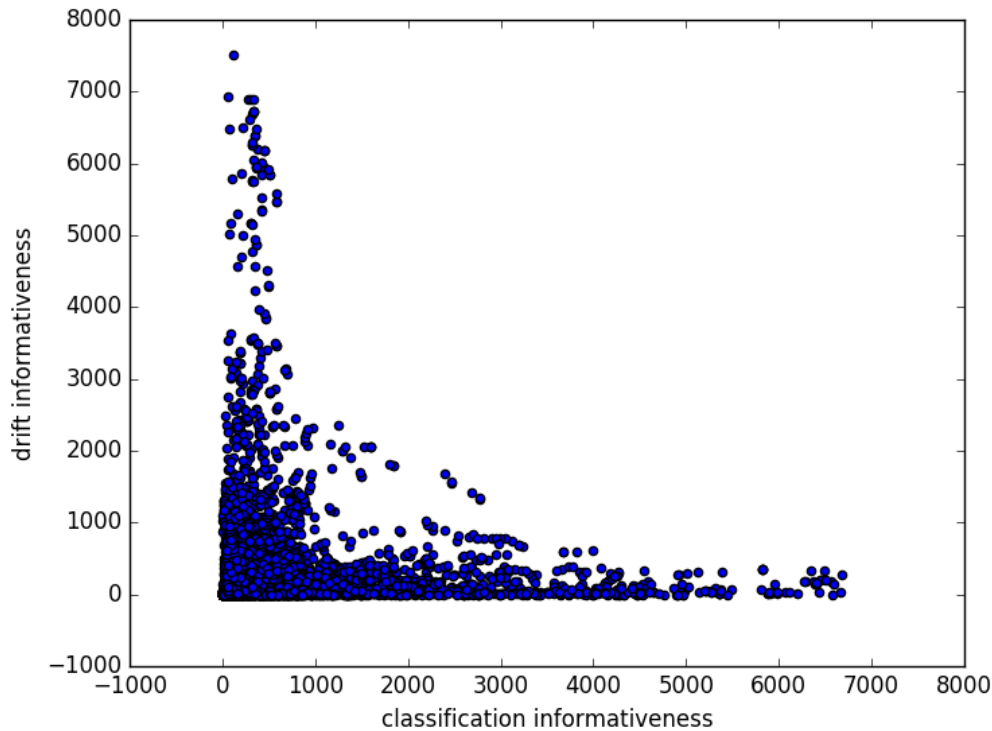
Tabela 4.4: Jakość klasyfikacji dla różnych wartości progowych przy użyciu informacji wzajemnej i lasów losowych

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	0.7	5355	0.9952	0.7118
0.0	0.05	2867	0.9862	0.6932
0.0	0.1	3559	0.992	0.7194
0.0	0.2	4362	0.9933	0.7055
0.25	0.05	950	0.9892	0.7044
0.25	0.1	1448	0.9923	0.7167
<b>0.25</b>	<b>0.2</b>	<b>2246</b>	<b>0.9927</b>	<b>0.7249</b>
0.5	0.05	266	0.9868	0.6625
0.5	0.1	597	0.9924	0.7172
0.5	0.2	1379	0.9935	0.6886
0.5	0.4	2044	0.9954	0.7089
1.0	0.1	15	0.8553	0.2661
1.0	0.3	443	0.9935	0.6888
1.0	0.5	920	0.9951	0.7064
1.5	0.7	486	0.9914	0.6438

Zastosowanie opisanej selekcji cech pomogło poprawić wynik o około 0.045 w najlepszym przypadku. Oczekiwane rezultaty zaobserwować można przykładowo dla wartości  $threshold_{class} = 0.25$  i  $threshold_{drift} = 0.05$  - po wybraniu 950 cech, wynik na zbiorze treningowym obniżył się, podczas gdy wynik na zbiorze testowym wzrósł. Wybrana reprezentacja, mimo że teoretycznie gorsza jeśli chodzi o klasyfikację czynności strażaków, okazała się bardziej odporna na zmianę dziedziny.

### Jednoczynnikowa analiza wariancji

Rozkład jakości atrybutów mierzony przy użyciu jednoczynnikowej analizy wariancji (wykres 4.3) prezentuje się dużo korzystniej niż w poprzednim przypadku. Cechy skupione są przy osiach  $x$  i  $y$ , zatem wybór zmienne odporne na zmianę dziedziny nie powodować spadku jakości klasyfikatora.



Rysunek 4.3: Rozkład jakości cech przy użyciu jednoczynnikowej analizy wariancji

Znajduje to swoje odzwierciedlenie w otrzymanych rezultatach. Największy zysk wydajności zaobserwowano dla klasyfikatorów liniowych. Wynik na zbiorze testowym dla regresji logistycznej i SVM przy odpowiednich wartości progów poprawił się nawet o 0.15 (tabele 4.5 i 4.6), przy jednoczesnym niewielkim spadku  $BAC$  na zbiorze treningowym. Dla lasów losowych poprawa miary jakości wyniosła około 0.05 w najlepszym przypadku (tabela 4.7). Należy przy tym zauważyć, że polepszenie jakości klasyfikatora nastąpiła wraz z redukcją przestrzeni cech - najlepsze wyniki uzyskano dla ok. 1500 atrybutów. Niesie to ze sobą kolejną zaletę, jaką jest obniżenie czasu budowy modelu.

Warto też zwrócić uwagę na wyniki klasyfikatorów dla ustalonej wartości  $threshold_{class}$ . Praktycznie w każdym przypadku eliminowanie większej liczby cech wrażliwych na zmianę dziedziny skutkuje wzrostem jakości.

Tabela 4.5: Jakość klasyfikacji dla różnych wartości progowych przy użyciu analizy wariancji i regresji logistycznej

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	8000	5355	0.9497	0.5348
0.0	250	3813	0.9067	0.5853
0.0	500	4434	0.9295	0.5940
0.0	750	4694	0.9387	0.5931
0.0	1000	4871	0.9445	0.5838
<b>250</b>	<b>250</b>	<b>1890</b>	<b>0.9188</b>	<b>0.6711</b>
250	500	2278	0.9426	0.6333
250	750	2446	0.9538	0.6102
250	1000	2560	0.9559	0.6289
<b>500</b>	<b>250</b>	<b>1309</b>	<b>0.8926</b>	<b>0.6680</b>
500	500	1566	0.9124	0.6050
500	750	1656	0.9171	0.5898
500	1000	1714	0.9310	0.6169
500	2000	1805	0.9214	0.5495
1000	500	873	0.7031	0.5725
1000	1000	927	0.7118	0.5436
1000	2000	946	0.8019	0.4943

Tabela 4.6: Jakość klasyfikacji dla różnych wartości progowych przy użyciu analizy wariancji i SVM

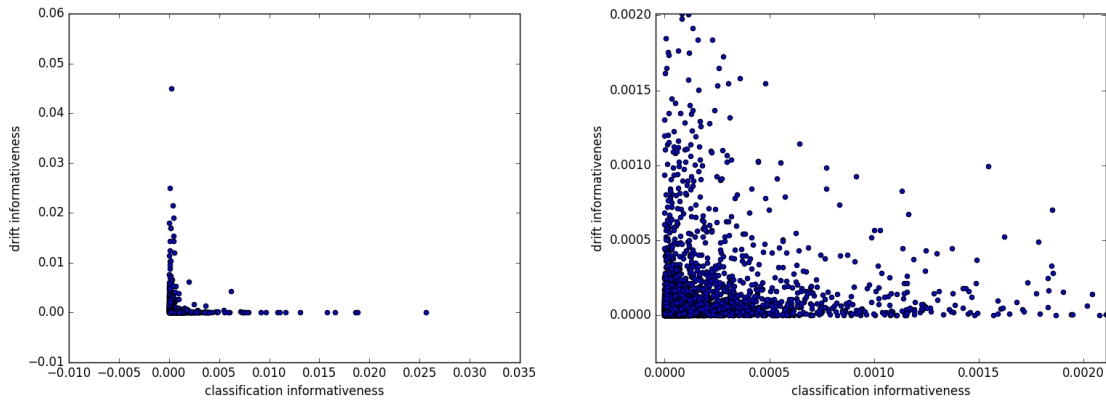
$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	8000	5355	0.9586	0.5346
0.0	250	3813	0.9064	0.5603
0.0	500	4434	0.9334	0.5857
0.0	750	4694	0.9301	0.6035
0.0	1000	4871	0.9342	0.6054
<b>250</b>	<b>250</b>	<b>1890</b>	<b>0.9254</b>	<b>0.6733</b>
250	500	2278	0.9357	0.6492
250	750	2446	0.9510	0.6015
250	1000	2560	0.9578	0.6176
<b>500</b>	<b>250</b>	<b>1309</b>	<b>0.8599</b>	<b>0.6849</b>
500	500	1566	0.9069	0.5904
500	750	1656	0.9135	0.6239
500	1000	1714	0.9188	0.6006
500	2000	1805	0.9128	0.5373
1000	500	873	0.7966	0.4923
1000	1000	927	0.8039	0.5589
1000	2000	946	0.8005	0.5950

Tabela 4.7: Jakość klasyfikacji dla różnych wartości progowych przy użyciu analizy wariancji i lasów losowych

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	8000	5355	0.9948	0.7118
0.0	250	3813	0.9915	0.7401
0.0	500	4434	0.9923	0.7362
0.0	750	4694	0.9922	0.7311
0.0	1000	4871	0.9929	0.7275
<b>250</b>	<b>250</b>	<b>1890</b>	<b>0.9919</b>	<b>0.7659</b>
250	500	2278	0.9933	0.7225
250	750	2446	0.9927	0.7300
250	1000	2560	0.9931	0.7233
500	250	1309	0.9911	0.7171
500	500	1566	0.9921	0.7235
500	750	1656	0.9920	0.7238
500	1000	1714	0.9933	0.7056
500	2000	1805	0.9935	0.7113
1000	500	873	0.9912	0.6692
1000	1000	927	0.9915	0.6685
1000	2000	946	0.9919	0.6679

### Zysk Giniego

Rysunek 4.4 przedstawia rozkład jakości cech przy użyciu zysku Giniego. Podobnie jak w przypadku jednoczynnikowej analizy wariancji, rozkład cech jest korzystny dla opisywanej metody redukcji *concept drift* - atrybuty są skupione przy osiach  $x$  i  $y$ . Otrzymane wyniki przedstawiają tabele 4.8, 4.9 i 4.10.



Rysunek 4.4: Rozkład jakości cech przy użyciu zysku Giniego. Wykres po prawej stronie stanowi przybliżenie wykresu po stronie lewej

Tabela 4.8: Jakość klasyfikacji dla różnych wartości progowych przy  
użyciu zysku Giniego i regresji logistycznej

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	0.05	5355	0.9497	0.5348
0.0	1e-05	2018	0.8548	0.4719
0.0	5e-05	3776	0.9165	0.4847
0.0	0.0001	4386	0.9289	0.5472
0.0	0.001	5185	0.9436	0.5285
5e-05	1e-05	429	0.7764	0.5359
5e-05	5e-05	1053	0.8935	0.5245
5e-05	0.0001	1392	0.9254	0.5516
5e-05	0.0025	1999	0.9610	0.5643
5e-05	0.005	2028	0.9529	0.5484
0.0001	5e-05	620	0.8859	0.5913
0.0001	0.0001	841	0.8947	0.6004
<b>0.0001</b>	<b>0.005</b>	<b>1331</b>	<b>0.9553</b>	<b>0.6176</b>
0.001	0.0001	117	0.6451	0.4872
0.001	0.0025	185	0.6829	0.5600

Tabela 4.9: Jakość klasyfikacji dla różnych wartości progowych przy  
użyciu zysku Giniego i SVM

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	0.05	5355	0.9586	0.5346
0.0	1e-05	2018	0.8562	0.4723
0.0	5e-05	3776	0.9109	0.4935
0.0	0.0001	4386	0.9318	0.5489
0.0	0.001	5185	0.9317	0.5273
5e-05	1e-05	429	0.8063	0.5762
5e-05	5e-05	1053	0.9296	0.4788
5e-05	0.0001	1392	0.9288	0.5371
5e-05	0.0025	1999	0.9532	0.5141
5e-05	0.005	2028	0.9580	0.5358
0.0001	5e-05	620	0.8727	0.5763
<b>0.0001</b>	<b>0.0001</b>	<b>841</b>	<b>0.8693</b>	<b>0.5952</b>
0.0001	0.005	1331	0.9562	0.5876
0.001	0.0001	117	0.6483	0.4387
0.001	0.0025	185	0.6582	0.5136

Tabela 4.10: Jakość klasyfikacji dla różnych wartości progowych przy użyciu zysku Giniego i lasów losowych

$threshold_{class}$	$threshold_{drift}$	Liczba cech	$X_{train} BAC$	$X_{test} BAC$
0.0	0.05	5355	0.9948	0.7118
0.0	1e-05	2018	0.9859	0.7000
0.0	5e-05	3776	0.9893	0.7254
0.0	0.0001	4386	0.9914	0.7271
0.0	0.001	5185	0.9933	0.7163
5e-05	1e-05	429	0.9890	0.7207
5e-05	5e-05	1053	0.9925	0.7393
5e-05	0.0001	1392	0.9925	0.7338
5e-05	0.0025	1999	0.9953	0.7112
5e-05	0.005	2028	0.9962	0.7246
0.0001	5e-05	620	0.9933	0.7281
<b>0.0001</b>	<b>0.0001</b>	<b>841</b>	<b>0.9927</b>	<b>0.7502</b>
0.0001	0.005	1331	0.9960	0.7205
0.001	0.0001	117	0.9918	0.7298
0.001	0.0025	185	0.9928	0.7100

Dla każdego z algorytmów uczących można ponownie zauważyć poprawę jakości klasyfikatora. Wzrost  $BAC$ , chociaż nie tak wyraźny jak w przypadku ANOVA, jest też znaczny i wyniósł kolejno: 0.08 dla regresji logistycznej, 0.06 dla maszyny wektorów nośnych i 0.04 dla lasów losowych.

Warto też zwrócić uwagę, że mimo korzystnego rozkładu cech, obniżanie  $threshold_{drift}$  (a co za tym idzie - eliminacja większej liczby cech wrażliwych na ewolucję pojęć) nie zawsze niesło ze sobą poprawę jakości klasyfikatora. Najlepiej widać to na przykładzie regresji logistycznej. Przy  $threshold_{class} = 0$ , obniżenie  $threshold_{drift}$  z 0.0001 na 0.00005 spowodowało dość wyraźny spadek jakości klasyfikatora - z 0.5472 na 0.4847. Jest to o tyle zaskakujące, że:

- na podstawie wykresu 4.4 wywnioskować można, że większość wyeliminowanych cech była bezużyteczna dla klasyfikacji czynności strażaka
- Obniżenie progu nie poskutkowało drastycznym spadkiem liczby wybranych cech (z 4386 na 3776)

#### 4.4. Detekcja zmiany dziedziny przy użyciu klasteryzacji

Problem adaptacji dziedziny w kontekście analizowanego zbioru danych dotyczy różnic w wykonywaniu tych samych czynności przez różnych strażaków. Szukając cech odpornych na ewolucję pojęć, naturalne byłoby więc wybieranie tych atrybutów, które najgorzej rozróżniają funkcjonariuszy straży między sobą. Problemem okazał się jednak brak identyfikatora strażaka. W zamian dostępna była inna informacja: liczba osób, od których zbierane były dane w każdym ze zbiorów (gwoili przypomnienia: zbiór treningowy i testowy pochodziły od różnych czteroosobowych grup strażaków). Wykorzystując tę wiedzę, modyfikację metody opisanej na wstępie rozdziału, w której identyfikator funkcjonariusza próbuje się "odzyskać" przez klastrowanie.

#### 4.4.1. Algorytmy klasteryzacji

Klasteryzacja (inaczej analiza skupień) jest metodą uczenia bez nadzoru (por. z sekcją 1.2.1). Jest to metoda grupowania elementów danego zbioru we względnie jednorodne klasy. Podstawą grupowania w większości algorytmów jest podobieństwo między elementami - wyrażone przy pomocy pewnej funkcji podobieństwa. Algorytmy analizy skupień dzieli się na kilka podstawowych kategorii:

- metody hierarchiczne
- grupa metod k-centroidów
- metody rozmytej analizy skupień

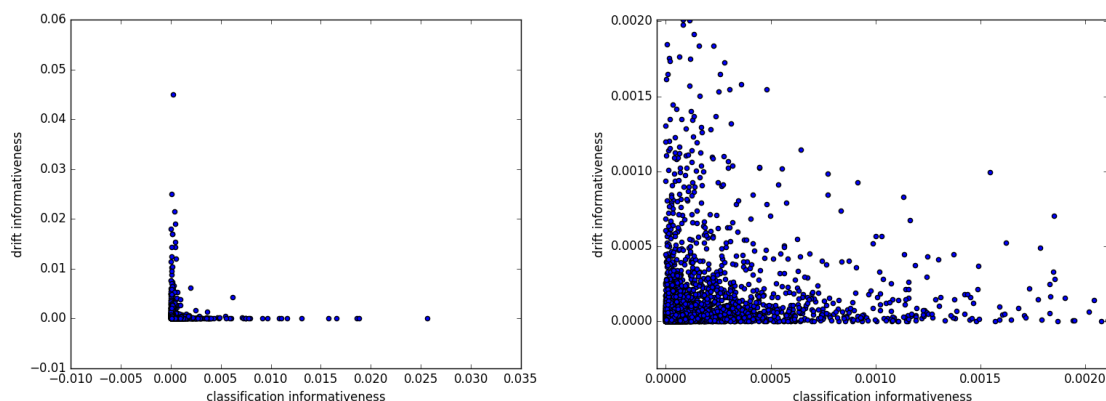
Autor użył implementacji algorytmu k-centroidów z biblioteki scikit-learn<sup>1</sup>.

#### Algorytm k-centroidów

TODO

#### 4.4.2. Opis algorytmu detekcji dziedziny

Mając dany identyfikator strażaka, sprowadzenie problemu wykrycia *concept drift* do problemu klasyfikacji jest trywialne: wystarczy ów identyfikator potraktować jako klasę wynikową.



Rysunek 4.5: Rozkład jakości cech przy użyciu zysku Giniego. Wykres po prawej stronie stanowi przybliżenie wykresu po stronie lewej

### 4.5. Wnioski z przeprowadzonych eksperymentów

---

<sup>1</sup><http://scikit-learn.org>





## Rozdział 5

# Inne metody adaptacji dziedziny

W poniższym rozdziale przedstawione zostaną dwie inne metody adaptacji dziedziny: uczenie iteracyjne oraz metoda zmiany przestrzeni cech zaproponowana przez Daume w (TODO biblio).

### 5.1. Uczenie iteracyjne

#### 5.1.1. Opis algorytmu

Idea uczenia iteracyjnego jest stosunkowo intuicyjna. Polega na iteracyjnym powiększaniu zbioru treningowego na części poprzednio sklasyfikowanych przykładów ze zbioru testowego. Bardziej szczegółowo, uczenie iteracyjne działa według następującego schematu:

1. klasyfikator  $c$  jest uczony na zbiorze treningowym
2.  $c$  klasyfikuje część przykładów ze zbioru testowego
3. zbiór treningowy powiększany jest o uprzednio sklasyfikowane instancje

Poniżej znajduje się pseudokod algorytmu uczenia iteracyjnego.

```
def iterative(self, classifier, X_train, y_train,
              X_test, rows_percentage):
    num_of_rows_per_iteration = rows_percentage * size(X_test)
    classifier.learn(X_train, y_train)
    y_test = []
    while size(X_test) > 0:
        current_sample = sample(X_test, num_of_rows_per_iteration)
        X_test = X_test \ current_sample
        current_y = classifier.classify(current_sample)
        y_test = y_test ∪ current_y
        y_train = y_train ∪ current_y
        X_train = X_train ∪ current_sample
        classifier.learn(X_train, y_train)
    return y_test
```

Argument *rows\_percentage* jest liczbą z przedziału  $(0, 1]$  i określa, jaka część zbioru testowego jest dołączana do zbioru treningowego w każdej iteracji.

### 5.1.2. Wyniki i wnioski

Wyniki klasyfikacji przy zastosowaniu zaprezentowanego wyżej algorytmu opisują tabele 5.1. Pierwszy wiersz każdej z tabel podaje jakość klasyfikacji bez zastosowania uczenia iteracyjnego.

<i>rows_percentage</i>	$X_{train}$ BAC	$X_{test}$ BAC
-	0.9497	0.5348
0.5	0.9541	0.5449
0.34	0.9411	0.5302
0.25	0.9474	0.5274
0.1	0.9507	0.5073

(a) Regresja logistyczna

<i>rows_percentage</i>	$X_{train}$ BAC	$X_{train}$ BAC
-	0.9586	0.5348
0.5	0.9535	0.537
0.34	0.9457	0.5389
0.25	0.9525	0.5282
0.1	0.95127	0.5091

(b) SVM

<i>rows_percentage</i>	$X_{train}$ BAC	$X_{train}$ BAC
-	0.9948	0.7118
0.5	0.9949	0.7249
0.34	0.9945	0.7257
0.25	0.995	0.7273
0.1	0.9948	0.7317

(c) Lasy losowe

Tabela 5.1: Jakość klasyfikatora budowanego przy użyciu uczenia iteracyjnego dla różnych algorytmów uczących i wartości *rows\_percentage*

Zastosowanie *iterative learning* nie miało znaczącego wpływu na jakość klasyfikacji. Można zauważyć, że w przypadku "słabszych" algorytmów (regresji logistycznej i SVM) nastąpiło obniżenie wyniku. Było to najpewniej spowodowane tym, że klasyfikator z każdą iteracją dopasowywał się coraz bardziej do dużej liczby błędnie sklasyfikowanych obiektów. W przypadku lasów losowych można z kolei zauważyć nieznaczny, ale stały wzrost jakości przy obniżaniu wartości *rows\_percentage*.

## 5.2. Powiększenie przestrzeni cech

### 5.2.1. Opis algorytmu

Adaptacja dziedziny przez powiększenie przestrzeni cech została zaproponowana przez Daume w (TODO bibl). Jak większość tego typu metod, pochodzi ona z dziedziny przetwarzania języka naturalnego. Powiększenie przestrzeni cech polega na zastąpieniu każdego atrybutu jego trzema wersjami: wersją ogólną, wersją odpowiadającą dziedzinie źródłowej  $D^{source}$  oraz wersją odpowiadającą dziedzinie docelowej  $D^{target}$ .

Aby zdefiniować tę operację bardziej formalnie, przyjmijmy oznaczenia jak w sekcji 1.1. Niech dla czytelności  $X = \mathbb{R}^F$ , dla pewnego  $F > 0$ . Wtedy powiększona przestrzeń przykładów zdefiniowana jest jako  $\hat{X} = \mathbb{R}^{3F}$ . Następnie definiuje się funkcje:

$$\Phi^{train} : X_{train} \rightarrow \hat{X}, \quad \Phi^{train}(\mathbf{x}) = (\mathbf{x}, \mathbf{x}, \mathbf{0})$$

oraz

$$\Phi^{test} : X_{test} \rightarrow \hat{X}, \quad \Phi^{test}(\mathbf{x}) = (\mathbf{x}, \mathbf{0}, \mathbf{x}),$$

gdzie  $\mathbf{0} = (0, 0, \dots, 0) \in \mathbb{R}^F$ . Funkcje te mapują instance ze zbioru treningowego (pochodzące z założenia z dziedziny  $D^{source}$ ) oraz testowego (pochodzące z  $D^{target}$ ) w rozszerzoną przestrzeń atrybutów  $\hat{X}$ .

Spróbujmy wyjaśnić intuicję stojącą za takim podejściem. Rozważmy problem rozpoznawania części mowy (ang. *Part-of-speech tagging*, w skrócie POS), gdzie zbiór treningowy tworzą artykuły z Wall Street Journal<sup>1</sup>, a zbiór testowy - anglojęzyczne artykuły o sprzęcie komputerowym. Przy tak postawionym problemie, słowo "in" powinno być traktowane w obu dziedzinach jako przyimek, podczas gdy słowo "monitor" powinno być traktowane jako czasownik w dziedzinie treningowej oraz rzeczownik w dziedzinie testowej.

Rozważmy prosty przypadek, w którym  $X = \mathbb{R}^2$ , gdzie wymiar  $x_1$  odpowiada słowu "in", a wymiar  $x_2$  - słowu "monitor". Wtedy, w  $\hat{X}$ ,  $\hat{x}_1$  i  $\hat{x}_2$  odpowiadają "ogólnym" znaczeniom tych słów,  $\hat{x}_3$  i  $\hat{x}_4$  - znaczeniom w dziedzinie artykułów finansowych, a  $\hat{x}_5$  i  $\hat{x}_6$  - znaczeniom w dziedzinie artykułów komputerowych.

Zastanówmy się teraz, co może zrobić algorytm uczący, aby zauważyć, że "in" powinno być traktowane jako przyimek w obydwu dziedzinach, natomiast klasyfikacja słowa "monitor" może się różnić w zależności od dziedziny. W tym przypadku, wektor wag odpowiadający klasie "przyimek" będzie wyglądał następująco:  $(1, 0, 0, 0, 0, 0)$ . Podkreśla to, że "in" jest przyimkiem niezależnie od dziedziny. Z kolei wektor wag dla klasy "rzeczownik" będzie postaci  $(0, 0, 0, 0, 0, 1)$ , co oznacza, że "monitor" jest rzeczownikiem tylko w zbiorze testowym. Analogicznie, wektor dla klasy "czasownik" będzie postaci  $(0, 0, 0, 1, 0, 0)$ .

### 5.2.2. Wyniki i wnioski

---

<sup>1</sup><http://www.wsj.com>



## Rozdział 6

# Podsumowanie



# Bibliografia

- [1] John Blitzer, Ryan McDonald, Fernando Pereira, *Domain adaptation with structural correspondence learning*
- [2] Hall Daume, *Frustratingly easy domain adaptation*
- [3] Andrew Arnold, Ramesh Nallapati, William W. Cohen, *A comparative study of methods for transductive transfer learning*
- [4] R. K. Ando, T. Zhang, *A framework for learning predictive structures from multiple tasks and unlabeled data*
- [5] L. Breiman, *Random forests*, Machine Learning, 45(1), 5-32, 2001.
- [6] Wanpracha Art Chaovalitwongse, Ya-Ju Fan, Rajesh C. Sachdeo, *On the time series k-nearest neighbor classification of abnormal brain activity*
- [7] Alex Nanopoulos, Rob Alcock, Yannis Manolopoulos, *Feature-based classification of time-series data*
- [8] Seyoung Kim, Padhraic Smyth, *Segmental hidden markov models with random effects for waveform modeling*
- [9] Chotirat Ann Ratanamahatana, Eamonn Keogh, *Everything you know about Dynamic Time Warping is wrong*
- [10] Dominique Gay, Romain Guigoures, Marc Boule, Fabrice Clerot *Feature Extraction over Multiple Representations for Time Series Classification*
- [11] Jenna Wiens, John V. Guttag, Eric Horvitz, *Patient Risk Stratification for Hospital-Associated C. diff as a Time-Series Classification Task*