Zastosowanie algorytmu opartego na agentach programowych do redukcji danych

Ireneusz Czarnowski[1]

Streszczenie: W pracy zaproponowano algorytm redukcji danych w uczeniu maszynowym i eksploracji danych. Proponowany algorytm został zaimplementowanych w środowisku agentów programowych JABAT (**JA**DE **B**ased **A-T**eam), które jest przeznaczone do rozwiązywania trudnych obliczeniowo problemów optymalizacyjnych. JABAT zastosowano do redukcji zbioru danych opartej na przeszukiwaniu wektorów uczących i wyborze wektorów referencyjnych. Prezentowany w pracy problem redukcji danych obejmuje również redukcję liczby atrybutów i usuniecie atrybutów nieistotnych dla klasyfikacji. W pracy przedstawiono także wyniki wybranych eksperymentów obliczeniowych.

Słowa kluczowe: selekcja wektorów referencyjnych, selekcja atrybutów, eksploracja danych, klasyfikacja.

1. Wstęp

Zainteresowanie systemami wspomagania decyzji stymuluje rozwój współczesnej informatyki, która poza gromadzeniem, przechowywaniem, udostępnianiem danych i informacji musi w obliczu wzrostu ich objętości w coraz większym stopniu wspomagać człowieka w procesie ich analizy (Krawiec, 2000). Analiza zgromadzonych danych a następnie wyciąganie użytecznych wniosków są typowymi procesami poprzedzającymi podjęcie decyzji. Narzędzia analizy danych są wykorzystywane w instytucjach finansowych, firmach telekomunikacyjnych czy w korporacjach handlowych. Następstwem takiego stanu rzeczy jest rosnące zainteresowanie narzędziami wspomagania decyzji wykorzystującymi techniki sztucznej inteligencji. Pośród nich szczególne miejsce zajmują metody rozpoznawania wzorców (ang.: *pattern recognition*). Sukces algorytmów rozpoznawania wzorców leżał u podstaw powstania nowego działu informatyki związanego z odkrywaniem wiedzy drogą analizy danych gromadzonych w bazach danych. Celem odkrywania wiedzy z danych, a więc procesu eksploracji danych, jest wspieranie procesu podejmowania decyzji. Jednym z ważniejszych problemów eksploracji danych jest klasyfikacja, której celem jest prawidłowa klasyfikacja obiektu na podstawie analizy wartości jego atrybutów. W większości przypadków odkrywanie wiedzy w bazach danych realizowane jest przy użyciu metod uczenia maszynowego.

Wszystkie algorytmy uczenia maszynowego wymagają zbioru danych treningowych. W przypadku klasyfikacji zbiór treningowy zawiera przykłady zwane również wektorami uczącymi lub obiektami, w skład których wchodzą wektory wejściowe oraz wartości wyjściowe. Wektory wejściowe najczęściej opisywane są przy użyciu

[1] Katedra Systemów Informacyjnych, Akademia Morska w Gdyni, Morska 83, 81-225 Gdynia
e-mail: irek@am.gdynia.pl

notacji atrybut-wartość (Krawiec, 2003). Zgodnie z tą notacją każdy wektor wejściowy jest opisany za pomocą ustalonego zbioru atrybutów. Wartość wyjściowa, odpowiednia dla danego wektora wejściowego, jest określana wartością atrybutu decyzyjnego.

Zwiększenie efektywności metod uczenia maszynowego może łączyć się z pozostawieniem w zbiorze danych treningowych tzw. wektorów referencyjnych i wyeliminowanie wektorów nadmiarowych i zawierających błędy lub szumy. Podawanie dużej ilości wektorów referencyjnych w procesie uczenia nie warunkuje wysokiej jakości klasyfikacji, a często jedynie spowalnia proces uczenia (Wilson, 2000). Redukcja rozmiaru zbioru treningowego prowadzi do skrócenia czasu potrzebnego na przeprowadzenie klasyfikacji oraz zmniejszenia wymagań, co do zasobów obliczeniowych. W efekcie, redukcja danych treningowych może przyspieszyć proces uczenia przy jednoczesnym zachowaniu pożądanego poziomu jakości klasyfikacji, a nawet polepszeniu jej jakości. W związku z tym proces redukcji danych uczących jest istotnym elementem odkrywania wiedzy w bazach danych i pozostaje ciągle aktywnym polem badań (Wilson, 2000; Czarnowski, 2004).

Równie ważnym problemem jest redukcja liczby atrybutów i usunięcie ze zbioru danych atrybutów redundantnych oraz nieistotnych dla klasyfikacji. W rzeczywistych sytuacjach atrybuty istotne z punktu analizowanego problemu są często nieznane a priori. Intuicyjnie, gromadzenie danych o dużej liczbie atrybutów jest próbą jak najlepszego opisu obiektów z danego problemu (Dash, 1997). Jednak użycie wektorów uczących składających się nieistotnych atrybutów może przynieść nieoczekiwane skutki w postaci niskiej jakości klasyfikacji. Natomiast wykorzystanie atrybutów redundantnych nie wnosi żadnej dodatkowej informacji. Ponadto Langley (1993) pokazał, że liczba obiektów niezbędna do uzyskania danej jakości klasyfikacji rośnie wykładniczo z liczbą nieistotnych atrybutów. I choć istnieje wiele metod selekcji atrybutów, na przykład wywodzących się ze statystyki, teorii informacji czy sztucznej inteligencji, problem pozostaje nadal przedmiotem intensywnych badań (Dash, 1997; Raman, 2003).

Interesującym problemem badawczym jest eliminacja danych nadmiarowych, nieistotnych i zaszumionych w każdym z wymiarów, to jest zarówno w przestrzeni atrybutów jak i przestrzeni przykładów. Problem jednoczesnej selekcji wektorów referencyjnych jak i atrybutów w literaturze niekiedy postrzegany jest jako typowy problem redukcji danych w każdym z wymiarów (Cano, 2004) lub jako problem rewizji przekonań w adaptacyjnych systemach klasyfikujących (Wróblewski, 2001).

W pracy do redukcji danych zaproponowano algorytm zaimplementowany w środowisku agentów programowych JABAT. JABAT jest uniwersalnym środowiskiem przeznaczonym do rozwiązaywania trudnych złożonych probemów optymalizacyjnych. Krótką charakterystykę proponowanych algorytmów redukcji danych zawarto w części 2. W części 3 przedstawiono charakterystykę środowiska JABAT oraz implementację proponowanego w pracy algorytmu redukcji danych. Sposób przeprowadzenia eksperymentów obliczeniowych oraz uzyskane wyniki przedstawiono w części 4. Ostatnia część pracy zawiera wnioski i propozycje kierunku dalszych badań.

2. Algorytmy redukcji danych

2.1. Selekcja wektorów referencyjnych

Problem redukcji danych związany z selekcją wektorów referencyjnych prowadzi do pozostawienia pewnej liczby przypadków z oryginalnego zbioru danych treningowych T i utworzenie zredukowanego zbioru treningowego S. Niech, zatem N jest liczbą przypadków w zbiorze T, n - jest liczbą atrybutów wektora wejściowego oraz $X = \{x_{ij}\}$ (gdzie $i = 1, \ldots, N$, $j = 1, ..., n+1$) jest macierzą o $n+1$ kolumnach i N wierszach zawierającą wszystkie wektory wejściowe wraz z wartością wyjściową z T ($n+1$ element tablicy jest wartością wyjściową dla danego wektora wejściowego).

Znane algorytmy redukcji danych wybierają wektory referencyjne obliczając odległość pomiędzy wektorami w zbiorze danych treningowych. Przypadkami referencyjnymi stają się wówczas wektory leżące w okolicach centrów skupień tworzonych przez wektory podobne. Algorytmy te wykorzystują mechanizm grupowania (klasteryzacji) (Duch, 2000; Salzberg, 1991; Wilson, 2000). Wariantem metod klasteryzacji stosowanych w redukcji danych wejściowych jest zmniejszanie tzw. rozdzielczości danych (Duch, 2000). Inna grupa metod należąca do metod bazujących na podobieństwie usuwa ze zbioru treningowego k najbliższych sąsiadów z danej klasy wektorów zakładając, że wszystkie wektory z sąsiedztwa będą i tak jednoznacznie klasyfikowane (Duch, 2000; Wilson, 2000). Inna grupa algorytmów eliminuje wektory treningowe testując klasyfikator i redukując sukcesywnie zbiór danych wejściowych (Duch, 2000). W literaturze problem selekcji wektorów referencyjnych, nazywany jest również problem selekcji prototypów, rozwiązywany jest także metodami ewolucyjnymi oraz heurystykami opartymi na mechanizmach losowych (Caono, 2004; Skalak, 1994; Rozsypal, 2003).

Proponowany w pracy algorytm redukcji danych treningowych dla przestrzeni przykładów został oparty na założeniu, iż wektory referencyjne będą wybierane z klastrów, które wyznacza się na podstawie następującej procedury:
Krok 1: Normalizacja każdej wartości $x_{ij} \in X$ ($i = 1, \ldots, N$, $j = 1, \ldots, n$) do przedziału $[0, 1]$ oraz zaokrąglenie x_{ij} do najbliższej wartości całkowitej.
Krok 2: Obliczenie dla każdego przykładu współczynnika podobieństwa I_i:

$$I_i = \sum_{j=1}^{n+1} x_{ij} s_j, \ i = 1, \ldots, N, \tag{1}$$

gdzie

$$s_j = \sum_{i=1}^{N} x_{ij}, \ j = 1, \ldots, n+1. \tag{2}$$

Krok 2: Grupowanie wektorów z X w t klastrów Y_v ($v = 1, \ldots, t$) zawierających wektory o identycznych wartościach współczynnika I_i, gdzie t jest liczbą różnych wartości I.
Krok 4: Wybór wektorów referencyjnych i utworzenie zbioru S. Jeżeli przez $|Y_v|$ oznaczymy liczbę wektorów w klastrze v ($v = 1, \ldots, t$), to wybór wektorów referencyjnych przebiega następująco:

- Jeżeli $|Y_v| = 1$ to $S = S \cup Y_v$,
- Jeżeli $|Y_v| > 1$ to $S = S \cup \{x_v\}$, gdzie x_v jest wektorem referencyjnym z Y_v wybranym przez algorytm oparty na agentach programowych.

Idea przedstawionej powyżej selekcji wektorów referencyjnych była wcześniej prezentowana w pracy (Czarnowski, 2004). Wyniki eksperymentów obliczeniowych zaprezentowane w pracy (Czarnowski, 2004) pokazały, że selekcja wektorów referencyjnych spośród grup wektorów o identycznych wartościach współczynnika podobieństwa może prowadzić do redukcji liczby przykładów w zbiorze treningowym przy jednoczesnym zagwarantowaniu odpowiedniej jakości klasyfikacji. Wyniki eksperymentów pokazały również, że redukcja zbioru treningowego może przyczynić się do podniesienia efektywności algorytmów uczenia maszynowego.

2.2. Redukcja danych w przestrzeni atrybutów

Równie ważnym problemem, jak selekcja wektorów referencyjnych, jest redukcja liczby atrybutów i usunięcie ze zbioru danych atrybutów redundantnych oraz nieistotnych dla klasyfikacji. Ponadto problem znalezienia optymalnego i minimalnego podzbioru atrybutów jest problemem NP-trudnym (Wróblewski, 2001). Dokładne metody selekcji atrybutów dokonują przeszukiwania wszystkich możliwych ich podzbiorów. Takie podejście jest jednak zbyt kosztowne obliczeniowo. Metody wywodzące się ze statystyki oceniają pojedyncze atrybuty. Inne metody dokonują oceny podzbiorów atrybutów, które tworzone są przy użyciu różnych technik opartych na przeszukiwaniu losowym, heurystycznym bądź ewolucyjnym. Przeglądu metod selekcji atrybutów dokonano między innymi w pracach (Dash, 1997; Raman, 2003; Zongker, 1996).

3. Algorytm redukcji danych oparty na agentach programowych

3.1. Charakterystyka środowiska JABAT

Prezentowany w pracy problem redukcji danych rozwiązano przy użyciu algorytmu, który zaimplementowano w środowisku JABAT. JABAT jest uniwersalnym środowiskiem oparym na agentach programowych, które kooperują i współdzielą pamięć celem rozwiązania problemu optymalizacyjnego. We współnej pamięci jest zapisana populacja składająca się z potencjalnych rozwiązań problemu. JABAT, wzorowany na koncepcji A-Team (Talukdar, 1998), jest środowiskiem, w którym rozwiązywanie problemu opiera się na następujących założeniach:

- do rozwiązania każdego problemu optymalizacyjnego należy użyć zbioru agentów programowych wykonujących procedury poprawy,
- celem przeszukiwania przestrzeni rozwiązań oraz unikania optimów lokalnych należy wygenerować populację rozwiązań (osobników), które w trakcie obliczeń będą optymalizowane przez niezależnych agentów.

JABAT organizuje i prowadzi obliczenia przez generowanie populacji początkowej rozwiązań (zazwyczaj z wykorzystaniem mechanizmów losowych), optymalizację rozwiązań odczytywanych z pamięci i ponowne zapisywanie ich do pamięci, oraz przez kontynuowanie cyklu czytania-poprawiania-zapisywania, aż do spełnienia kryterium zatrzymania.

Realizacja powyższych kroków opiera się na wykorzystaniu dwóch klas agentów: *OptiAgent* - dotyczy agenta z implementowanym algorytmem optymalizacji, *SolutionManager* - dotyczy agenta zajmującego się zarządzaniem populacją rozwiązań. *SolutionManager* posiada pełne informacje o typie i charakterze rozwiązywanego problemu. Obie klasy agentów kooperują, a współpraca jest wynikiem negocjacji i skojarzenia instancji rozwiązywanego problemu z oferowaną przez agenta optymalizacyjnego usługą.

Cechą JABATu jest jego uniwersalność i niezależność od rozwiązywanych problemów oraz algorytmów poprawy. W JABAT zdefiniowania problemu i rozwiązania dokonuje się w oparciu o klasy bazowe *Task* i *Solution* oraz odpowiednie dla nich ontologie *TaskOntology* i *SolutionOntology* dostarczające definicji dla rozwiązywanego problemu oraz opisu rozwiązania. Definiowanie problemu do rozwiązania, sposoby wymiany rozwiązań oraz przykłady zastosowań JABAT szerzej zostały przedstawione w pracy (Jędrzejowicz, 2006).

3.2. Implementacja algorytmu redukcji danych

Proponowany w pracy algorytm redukcji danych oraz wszystkie niezbędne do jego użycia klasy zostały zdefiniowane w ramach autorskiego pakietu *IFS* (*instance and feature selection*). *IFS* zawiera dwie podstawowe dla rozwiązywanego problemu klasy: *IFS_ Task* i *IFS_ Solution*, będące odpowiednio podklasami *Task* i *Solution*. *IFS_ Task* definiuje problem i zawiera między innymi nazwę oraz lokalizację zbioru danych. Zadaniem *IFS_ Task* jest również utworzenie klastrów dla potencjalnych wektorów referencyjnych. *IFS_ Task* dostarcza w tym przypadku informacji o tym, które wektory z oryginalnego zbioru treningowego znajdują się w danym klastrze. Tworzenie poszczególnych klastrów przebiega zgodnie z procedurą przedstawioną w części 2.1. Klasa *IFS_ Solution* zawiera opis rozwiązania problemu. Każde rozwiązanie reprezentowane jest przez dwie listy zawierające odpowiednio:

- wektory referencyjne, tj. numery przykładów wybranych z oryginalnego zbioru treningowego. Lista na $|Y_v|$ pierwszych pozycjach zawiera informację o mocy poszczególnych zbiorów (klastrów),

- wybrane atrybuty, a w szczególności numery atrybutów.

IFS_ Solution dostarcza również informacji o wartości funkcji celu. W tym przypadku jest to wartość jakości klasyfikacji uzyskanej przez klasyfikator skonstruowany w oparciu o zredukowany zbiór danych. W prezentowanym w pracy podejściu do oceny rozwiązań wykorzystano algorytm C 4.5 (Quinlan, 1993) choć jest ono niezależne od algorytmu klasyfikacji. Użycie algorytmu C 4.5 pozwoliło na uzyskanie dodatkowych informacji takich, jak rozmiar drzewa decyzyjnego i liczba

reguł, i w konsekwencji na szersze porównanie uzyskanych wyników. *IFS_ Solution* dostarcza również dodatkowych informacji takich, jak: poziom kompresji dla przestrzeni przykładów i atrybutów, liczba reguł oraz rozmiar drzewa.

Do wymiany informacji pomiędzy agentami optymalizacyjnymi a klasą *SolutionManager* zdefiniowano, odpowiednio, klasy *IFS_ TaskOntology* oraz *IFS_ SolutionOntology*. *IFS_ TaskOntology* umożliwia przesyłanie pomiędzy agentami parametrów zadania oraz informacji o numerach przykładów należących do poszczególnych klastrów i reprezentujących potencjalne wektory referencyjne. Natomiast *IFS_ SolutionOntology* umożliwia przesyłanie potencjalnych rozwiązań.

W proponowanym algorytmie każdy z agentów optymalizacyjnych dokonuje poprawy rozwiązania wybranego losowo z populacji przez *SolutionManagera*. Zadaniem każdego z agentów optymalizacyjnych jest poprawa jakości rozwiązania. Każdy z agentów optymalizacyjnych poprawia rozwiązanie przez określoną liczbę iteracji, po czym odsyła je do *SolutionManagera*. SolutionManager aktualizuje wspólną pamięć (populację) przez nadpisanie rozwiązania losowo wybranego z populacji rozwiązaniem poprawionym i odebranym od agenta optymalizacyjnego. *SolutionManager* zarządza populacją rozwiązań, która w momencie rozpoczęcia obliczeń generowana jest losowo. Oznacza to, że populacja początkowa zawiera rozwiązania o różnej liczbie wektorów referencyjnych w poszczególnych klastrach oraz różnej liczbie atrybutów.

Do rozwiązania problemu redukcji danych zaimplementowano, jako podklasy *OptiAgent*, cztery typy agentów reprezentujących różne procedury poprawy. Dwie z zaimplementowanych procedur zostały przeznaczone do optymalizacji otrzymanego rozwiązania przez modyfikację i zmianę wektorów referencyjnych w poszczególnych klastrach. Trzecia procedura poprawia rozwiązanie przez modyfikację listy odnoszącej się do numerów atrybutów. Czwarta z procedur poprawy dedykowana jest poprawie rozwiązania przez jego modyfikację zarówno dla przestrzeni wektorów jak i atrybutów.

Pierwszy agent optymalizacyjny - lokalne przeszukiwanie przestrzeni wektorów z listą ruchów zabronionych - tabu (Glover, 1990), opiera się modyfikacji rozwiązania przez usunięcie losowo wybranego numeru wektora referencyjnego w losowo wybranym klastrze i dodaniu innego aktualnie nie reprezentowanego w poprawianym rozwiązaniu. Modyfikacja ta poprzedzona jest sprawdzeniem, czy losowo wybrany numer wektora nie znajdują się liście tabu. Oznacza to, że w kolejnych iteracjach procedury modyfikowane są klastry, dla których numery wektorów referencyjnych nie znajdują się na liście tabu. W przypadku, gdy nastąpi modyfikacja rozwiązania każde nowo dodane numery wektorów referencyjnych zapisywane są na liście tabu i pozostają tam przez określoną liczbę iteracji. W tym jednak przypadku czas przetrzymywania na liście zależy od maksymalnej możliwej do uzyskania mocy danego klastra. W konsekwencji oznacza to, że czas przetrzymywania numerów wektorów z klastrów o mniejszej mocy jest któtszy. Ostatecznie rozwiązanie zmodyfikowane zastępuje rozwiązanie poprawiane, jeśli jest lepsze przy uwzględnieniu przyjętej funkcji celu, w przeciwnym razie rozwiązanie zmodyfikowane jest odrzucane.

Drugi agent optymalizacyjny - lokalne przeszukiwanie, opiera się na modyfikacji rozwiązania przez usunięcie losowo wybranego numeru wektora referencyjnego

w losowo wybranym klastrze i dodaniu innego aktualnie nie reprezentowanego w poprawianym rozwiązaniu. W tym przypadku modyfikacja nie wprowadza zmiany w liczebności poszczególnych klastrów. Dodatkowo, co określoną parametrem procedury liczbę iteracji, rozwiązanie jest modyfikowane przez dodanie do losowo wybranego klastra nowego wektora referencyjnego aktualnie nie reprezentowanego przez poprawiane rozwiązanie lub usunięcie losowo wybranego wektora referencyjnego. Proces dekrementacji i inkrementacji mocy klastrów prowadzany jest z jednakowym prawdopodobieństwem, przy czym zmodyfikowane rozwiązanie zastępuje rozwiązanie poprawiane, jeśli jest lepsze przy zadanym kryterium jakościowym, w przeciwnym razie jest ono odrzucane.

Trzeci agent optymalizacyjny - lokalne przeszukiwanie przestrzeni atrybutów z listą ruchów zabronionych - tabu, stanowi pewną analogię do pierwszego z agentów optymalizacyjnych. W tym jednak przypadku modyfikacja i poprawa rozwiązań prowadzona jest odpowiednio w przestrzeni atrybutów. Trzeci agent optymalizuje rozwiązanie przez usunięcie losowo wybranego numeru atrybutu i dodanie innego aktualnie nie reprezentowanego w poprawianym rozwiązaniu. Modyfikacja ta poprzedzona jest sprawdzeniem, czy losowo wybrany numer atrybutu nie znajdują się na liście tabu. Oznacza to, że w kolejnych iteracjach procedury modyfikowane są numery atrybutów, które nie znajdują się na liście tabu. W przypadku, gdy nastąpi modyfikacja rozwiązania, każdy nowo dodany atrybut zostaje zapisywany na liście tabu i pozostaje tam przez określoną liczbę iteracji. Ostatecznie rozwiązanie zmodyfikowane zastępuje rozwiązanie poprawiane, jeśli jest lepsze przy uwzględnieniu przyjętej funkcji celu, w przeciwnym razie rozwiązanie zmodyfikowane jest odrzucane.

Czwarty agent optymalizacyjny - lokalne przeszukiwanie zbioru danych, modyfikuje i poprawia rozwiązanie przez usunięcie losowo wybranego numeru wektora referencyjnego w losowo wybranym klastrze i dodaniu innego aktualnie nie reprezentowanego w poprawianym rozwiązaniu nie wprowadzając przy tym zmian w liczebności poszczególnych klastrów. Dodatkowo, co określoną parametrem procedury liczbę iteracji rozwiązanie jest modyfikowane przez usunięcie losowo wybranego numeru atrybutu, bądź dodanie do rozwiązania nowego numeru atrybutu aktualnie nie reprezentowanego w poprawianym rozwiązaniu. Dodawanie i usuwanie atrybutów prowadzone jest z jednakowym prawdopodobieństwem. Zmodyfikowane rozwiązanie zastępuje rozwiązanie poprawiane, gdy jest ono lepsze przy uwzględnieniu przyjętej funkcji celu, w przeciwnym razie jest ono odrzucane.

4. Wyniki eksperymentów obliczeniowych

Proponowany algorytm poddano weryfikacji na drodze eksperymentów obliczeniowych. Celem eksperymentów była ocena efektywności i użyteczności proponowanego algorytmu, opartego na agentach programowych, do redukcji danych. W oparciu o przeprowadzone eksperymenty przedstawiono i porównano jakość klasyfikacji uzyskanej przy użyciu algorytmu klasyfikującego skonstruowanego przy użyciu zredukowanego zbioru danych z wynikami uzyskanymi przy wykorzystaniu oryginalnego zbioru treningowego.

Eksperymenty obliczeniowe przeprowadzono dla wybranych danych benchmarkowych (Merz, 1998): Cleveland heart disease (303 obiekty, 13 atrybutów, 2 klasy), credit approval (690, 15, 2), Wisconsin breast cancer (699, 9, 2) oraz sonar (208, 60,2).

W poszczególnych eksperymentach, z wykluczeniem problemu sonar, do wyznaczenia jakości klasyfikacji posłużono się testem 10 - krotnej walidacji skrośnej. W teście tym każdy ze zbiorów danych podzielony został na 10 równych (o ile to możliwe) części i algorytm redukcji danych był użyty na zbiorze treningowym T składającym się z 9 części, z których otrzymano zredukowany zbiór S. Pozostała 10-ta część posłużyła do testowania. Tak skonstruowany zbiór S był użyty do zbudowania klasyfikatora opartego na algorytmie C 4.5. Procedurę powtarzano 10-krotnie tak, aby każda z części oryginalnego zbioru treningowego wystąpiła w roli zbioru testowego. W przypadku problemu rozpoznawania celów sonarowych eksperyment przeprowadzono w oparciu o oryginalne zbiory: treningowy i testowy, a redukcji danych poddano zbiór treningowy.

Każdy z eksperymentów powtórzono 30 krotnie a uzyskane wyniki zostały uśrednione. Liczba iteracji dla każdej z procedur poprawy wynosiła 100. W przypadku JABAT wielkość pamięci wspólnej (populacji) wyniosła 100. Wartości podanych parametrów dla proponowanego algorytmu zostały określone eksperymentalnie.

Wyniki eksperymentów obliczeniowych pokazano w Tabeli 1. Wyniki te przedstawiono odpowiednio dla dwóch wariantów algorytmu C 4.5: bez przycinania drzewa oraz z przycinaniem drzewa. Ponadto celem zobrazowania efektywności redukcji danych, realizowanej przy użyciu prezentowanego w pracy algorytmu, w Tabeli 1 podano wartości jakości klasyfikacji otrzymane dla następujących przypadków:
A - jakość klasyfikatora skonstruowanego przy użyciu pełnego, oryginalnego zbioru danych;
B - jakość klasyfikatora skonstruowanego przy wykorzystaniu zbioru treningowego opisanego na przestrzeni atrybutów wybranych przy użyciu techniki *wrapper* (w podejściu wrapper do oceny poszczególnych podzbiorów atrybutów wykorzystano algorytm C4.5) (Dash, 1997);
C - jakość klasyfikatora skonstruowanego przy użyciu zbioru treningowego składającego się z wektorów referencyjnych (do zbudowania zbioru treningowego wykorzystano wariant prezentowanego w pracy algorytmu, który został oparty wyłącznie na agentach optymalizacyjnych przeznaczonych do wyboru wektorów referencyjnych);
D - jakość klasyfikatora skonstruowanego przy użyciu zbioru danych zbudowanego zgodnie z procedurą dla przypadku C rozszerzoną o selekcję atrybutów opartą na technice *wrapper*;
E - jakość klasyfikatora opartego na zbiorze danych zbudowanym przy użyciu algorytmu redukcji danych prezentowanego w niniejszej pracy.

W Tabeli 1 przedstawiano również średni rozmiar drzewa decyzyjnego algorytmu C 4.5 oraz średnią liczbę reguł. Te dodatkowe wielkości zostały przedstawione celem pełniejszego zobrazowania wpływu redukcji danych na jakość uzyskiwanych wyników. W tym przypadku podane wyniki pozwalają na ocenę redukcji

Tabela 1. Średnia jakość klasyfikacji (w %) algorytmu C 4.5 oraz średnia liczba reguł i średni rozmiar drzewa decyzyjnego

	Przycinanie drzewa					Brak przycinania drzewa				
	Średnia jakość klasyfikacji (%)									
Problem	A	B	C	D	E	A	B	C	D	E
credit	84.9	77.2	90.7	81.3	92.6	83.2	74.8	90.4	76.7	90.9
cancer	94.6	94.4	97.4	95.0	98.1	95.0	94.4	98.4	86.5	97.9
heart	77.9	79.9	91.2	81.7	93.0	76.9	79.5	92.4	81.3	92.2
sonar	74.0	72.1	83.7	78.4	87.5	74.0	71.2	83.7	76.4	88.8
średnio	82.9	80.9	90.8	84.1	92.8	82.3	80.0	91.2	80.2	92.5
Średnia liczba reguł										
credit	12.0	36.0	16.4	15.5	10.5	54.0	75.0	24.5	15.4	13.9
cancer	15.0	8.0	8.2	2.3	6.5	20.0	19.0	12.8	2.7	7.7
heart	17.0	11.0	17.6	8.7	11.5	44.0	26.0	23.8	8.4	14.3
sonar	8.0	10.0	9.0	16.0	11.0	8.0	12.0	10.0	14.0	11.4
Średni rozmiar drzewa										
credit	23.0	71.0	31.8	30.0	20.0	107.0	149.0	48.0	29.8	26.8
cancer	29.0	15.0	15.4	3.6	12.0	39.0	37.0	24.6	4.4	14.3
heart	33.0	21.0	34.3	16.4	22.0	87.0	35.0	46.6	15.8	27.6
sonar	15.0	20.0	17.0	19.0	21.0	15.0	25.0	19.0	17.0	21.8

danych z perspektywy reprezentacji wiedzy.

Na podstawie uzyskanych wyników można sformułować wniosek, że redukcja danych w każdym z wymiarów może prowadzić do uzyskania wyższej jakości klasyfikacji w stosunku do przypadku stosowania w uczeniu klasyfikatora pełnego oryginalnego zbioru danych. Wyniki eksperymentów obliczeniowych pokazały, że podwyższenie jakość klasyfikacji można uzyskać stosując redukcję danych zarówno dla przestrzeni przykładów jak i przestrzeni atrybutów. Dla przykładu, jakość klasyfikacji, dla problemu *credit*, uzyskana przy użyciu zbioru treningowego zbudowanego przy użyciu JABAT wyniosła 92.6% i 90.2%, odpowiednio dla przypadków z przycinaniem i bez przycinania drzewa, a w przypadku użycia oryginalnego zbioru danych algorytm C 4.5 gwarantował uzyskanie jakości klasyfikacji na poziomie 84.9% i 93.2%. W innym przykładzie - problem *sonar*, jakość klasyfikacji uzyskana przy wykorzystaniu zbioru zredukowanego dla obu wymiarów wyniosła 87.5% i 88.8%, odpowiednio dla przypadków z przycinaniem i bez przycinania drzewa, gdy dla przypadku z użyciu oryginalnego zbioru danych jakość wynosi 74% niezależnie od parametryzacji algorytmu C4.5 dotyczącej upraszczania struktury drzewa.

Ponadto wyniki eksperymentów pokazują, że stosowanie redukcji danych wyłącznie dla przestrzeni przykładów również gwarantuje poprawę jakości klasyfikacji. W tym przypadku jakość klasyfikacja przewyższa wyniki dotyczące użycia oryginalnego zbioru danych oraz wyniki związane z użyciem zredukowanego zbioru

Tabela 2. Liczba atrybutów uzyskana w poszczególnych eksperymentach

	Średnia liczba atrybutów					
Problem	A	B	C	D	E	RMHC
credit	15.0	9.0	15.0	8.0	10.5	-
cancer	9.0	5.0	9.0	4.0	7.1	4.8
heart	13.0	7.2	13.0	7.0	9.6	7.6
sonar	60.0	22.5	60.0	27.0	26.1	-

Tabela 3. Średnia jakość klasyfikacji (w %) oraz poziom kompresji (w %) wybranych algorytmów redukcji danych

Problem	cancer		heart		credit		sonar	
Algorytm	Jakość	$\frac{\lvert S\rvert}{\lvert T\rvert}$	Jakość	$\frac{\lvert S\rvert}{\lvert T\rvert}$	Jakość	$\frac{\lvert S\rvert}{\lvert T\rvert}$	Jakość	$\frac{\lvert S\rvert}{\lvert T\rvert}$
JABAT	**98.1**	20	**93.0**	60	**92.6**	30	**88.8**	90
K-NN	96.28	100	81.19	100	84.78	100	58.8*	100
CNN	95.71	7.09	73.95	30.84	77.68	24.22	74.12	32.85
SNN	93.85	8.35	76.25	33.88	81.31	28.38	79.81	28.26
IB2	95.71	7.09	73.96	30.29	78.26	24.15	80.88	33.87
IB3	96.57	3.47	81.16	11.11	85.22	4.78	69.38	12.02
DROP3	96.14	3.58	80.84	12.76	83.91	5.96	78	26.87

* (patrz Rozsypal, 2003)

danych dla przestrzeni atrybutów i uzyskanego techniką *wrapper*. Wyniki pokazują również, że redukcja przestrzeni atrybutów techniką *wrapper* nie gwarantuje istotnego podwyższenia jakości klasyfikacji. Uwzględniając inne kryteria jakościowe - tj. liczba reguł i rozmiar drzewa, wyniki pokazują, że redukcja danych może również przyczynić się uzyskania mniej złożonych, bardziej czytelnych i łatwiejszych do interpretacji opisów zależności występujących w danych.

Dodatkowo w Tabeli 2 zawarto wielkości odnoszące się do średniej liczby atrybutów w zbiorze treningowym odpowiednio dla każdego z eksperymentów. W Tabeli 2 zawarto również średnią liczbę atrybutów uzyskaną w wyniku selekcji prowadzonej przy użyciu algorytmu - RMHC (*random mutation hill climbing*) dedykowanego jednoczesnej selekcji atrybutów i wektorów referencyjnych, oraz walidowanego na tych samym zestawie danych benchmarkowych (Skalak, 1994).

W przypadku redukcji danych w przestrzeni przykładów przedstawiony w pracy algorytm gwarantował uzyskanie kompresji danych na poziomie, odpowiednio: 30% dla problemu *credit*, 20% - *cancer*, 60% - *heart* oraz 90% dla problemu *sonar*. Dla porównania w Tabeli 3 przedstawiono poziom kompresji (kolumna: $|S| \,/\, |T|$) oraz jakość klasyfikacji uzyskaną przy użyciu innych wybranych algorytmów redukcji danych należących do klasy instance-based (Wilson, 2000).

5. Zakończenie

W pracy przedstawiono algorytm, oparty na agentach programowych, do redukcji danych. Zaimplementowany w środowisku JABAT algorytm wykorzystano do redukcji danych dla przestrzeni przykładów oraz dla przestrzeni atrybutów. Wyniki przeprowadzonych eksperymentów obliczeniowych wykazały korzystny wpływ redukcji danych w uczeniu maszynowym i eksploracji danych. Użycie proponowanego podejścia redukcji danych zwiększa efektywność uczenia poprzez skuteczne filtrowanie zbioru danych wejściowych i eliminowanie z niego szumów, informacji redundantnych oraz nieistotnych dla klasyfikacji.

Dalsze badania obejmą konstruowanie i weryfikację innych procedur selekcji danych oraz weryfikację proponowanego podejścia pod kątem eksploracji danych w rozproszonym systemie baz danych.

Literatura

Cano J.R., Herrera F., Lozano M. (2004) Stratification for scaling up evolutionary prototype selection. *Pattern Recognition Letters*, Elsevier, (in press)

Czarnowski I., Jędrzejowicz, P. (2004) An approach to instance reduction in supervised learning. W: Coenen F., Preece A. and Macintosh A. (red.), Research and Development in Intelligent Systems XX, Springer, London, 267-282.

Dash, M. Liu H. (1997) Feature selection for classification. *Intelligence Data Analysis* **1**, 3, 131-156.

Duch W., Grudziński K. (2000) Sieci neuronowe i uczenie maszynowe: próba integracji. W: Sieci neuronowe, Duch W., Korbicz J., Rutkowski L., Tadeusiewicz R. (red.), *Biocybernetyka* **6**, 3, 663-690.

Glover F. (1990) Tabu search - part I. *ORSA Journal of Computing* **1**, 190-206.

Jędrzejowicz P., Wierzbowska I. (2006) JADE-based A-team environment. Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 3993, 719-726.

Krawiec K. (2000) Konstruktywna indukcja cech we wspomaganiu decyzji na podstawie informacji obrazowej. Rozprawa doktorska. Instytut Informatyki Politechniki Poznańskiej, Poznań.

Krawiec K., Stefanowski J. (2003) Uczenie maszynowe i sieci neuronowe. Wydawnictwo Politechniki Poznańskiej, Poznań.

Langley, P., Iba, W. (1993) Average-case analysis of a nearest neighbor algorithm. Proceedings of the Thirteenth International Conference on Artificial Intelligence, Chambery, France, 889-894.

Merz C.J., Murphy P.M. (1998) UCI repository of machine learning databases (`http://www.ics.uci.edu/~mlearn/MLRepository.html`)
Irvine, CA: University of California, Department of Information and Computer Science.

Quinlan J.R. (1993) *C 4.5: Programs for Machine Learning.* Morgan Kaufmann, SanMateo, CA.

Raman B., Ioerger T.R. (2003) Enhancing learning using feature and example selection. *Journal of Machine Learning Research*, (in press)

Rozsypal A., Kubat M. (2003) Selecting representative examples and attributes by a genetic algorithm. *Intelligent Data Analysis* **7**, 4, 291-304.

Salzberg S. (1991) A nearest hyperrectangle learning method. *Machine Learning* **6**, 277-309.

Skalak D.B. (1994) Prototype and feature selection by sampling and random mutation hill climbing algorithm. W: International Conference on Machine Learning, 293-301.

Talukdar S., Baeretzen L., Gove A., de Souza P. (1998) Asynchronous teams: cooperation schemes for autonomous agents. *Journal of Heuristics* **4**, 295-321.

Wilson D. R., Martinez T. R. (2000) Reduction techniques for instance-based learning algorithm. *Machine Learning*, Kluwer Academic Publishers, Boston, 33, 3, 257-286.

Wróblewski J. (2001) Adaptacyjne metody klasyfikacji obiektów. Praca doktorska, Uniwersytet Warszawski, Warszawa.

Zongker D., Jain A. (1996) Algorithm for feature selection: an evaluation. W: International Conference on Pattern Recognition, ICPR'96, 18-22.

Implementation of the agent-based algorithm for data reduction

The paper proposes a data reduction algorithm to machine learning and data mining. The proposed algorithm was implemented in JABAT (*JADE Based A-Team*) environment. The JABAT environment is dedicated for solving a variety of computationally hard optimization problems. The JABAT produces solutions to combinational optimization problems using a set of optimizing agents, each representing an improvement algorithm. The proposed algorithm has been used to reduce the original dataset in two dimensions including selection of reference instances and removal of irrelevant attributes. To validate the approach the computational experiment has been carried out. Presentation and discussion of experiment results conclude the paper.