STUDIA INFORMATICA 2015
Volume 36 Number 1 (119)

Marcin BLACHNIK, Tadeusz WIECZOREK
Politechnika Śląska, Katedra Informatyki Przemysłowej

PRZEGLĄD METOD UCZENIA INKREMENTACYJNEGO

Streszczenie. Klasyczne metody uczenia modeli inteligencji obliczeniowej opierają się na budowaniu bazy wiedzy, przy założeniu że dostępny jest cały, skończony zbiór przypadków uczących. Założenie to nie zawsze jest spełnione, dlatego też w artykule dokonano przeglądu różnych metod uczenia z możliwością douczania modelu predykcyjnego w miarę napływu nowych danych uczących. Omówiono także metody z rodziny: wnioskowania na podstawie przypadków, modeli bazujących na funkcjach jądrowych oraz systemów regułowych.

Słowa kluczowe: uczenie inkrementacyjne, douczanie, uczenie maszynowe, inteligencja obliczeniowa

SURVAY OF INCREMENTAL LEARNING METHODS

Summary. Classical training methods of computational intelligence models are based on building a knowledge base, assuming that the entire, complete set of learning vectors is available. This assumption is not always met, particularly in issues related to the industry. In the paper we provide an overview of a broad group of algorithms supporting incremental learning which includes: case based on reasoning, kernel methods, and incremental induction of rule-based systems.

Keywords: incremental learning, on-line learning, machine learning, computational intelligence

1. Wstęp

Klasyczne metody uczenia systemów inteligencji obliczeniowej (ang. *Computational Intelligence*, CI) koncentrują się głównie na problemie zbudowania pewnego modelu wiedzy w sytuacji, gdy dostępny jest skończony zbiór przypadków. Innymi słowy, zakłada się, iż budowany model matematyczny (np. sieć neuronowa) powstanie na bazie pewnej skończonej liczby danych empirycznych, pochodzących z badanego zjawiska, przy założeniu że całość

danych dostępna jest jednocześnie. Z matematycznego punktu widzenia na podstawie zbioru danych **X,** składającego się z *n* par $\mathbf{X} = [[\mathbf{x}_1 y_1], [\mathbf{x}_2 y_2], ..., [\mathbf{x}_n y_n]]^T$, gdzie pojedynczy wektor $\mathbf{x}_i = [x_1, x_2, ..., x_m]$, zdefiniowany jest w *m*-wymiarowej przestrzeni, a y_i jest stowarzyszoną z nim etykietą, stawia się hipotezę *H* bez hipotez pośrednich jako funkcję *f* [9, 6] o postaci

$$H = f(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) \tag{1}$$

Założenie o dostępności całości zbioru danych podczas uczenia nie zawsze może być spełnione. Typowym tego przykładem są problemy środowiskowe, gdzie w trakcie swojego działania stopniowo pozyskuje się nowe dane, na podstawie których system musi uczyć się w sposób ciągły. Wówczas, ze względu na brak całości zbioru uczącego, trening takiego systemu polega na ciągłym douczaniu, co powoduje, że po prezentacji *i*-tego wektora danych $\mathbf{x}_i$ system tworzy nową hipotezę H_i, na podstawie hipotezy H_{i-1} oraz owego nowego przypadku $\mathbf{x}_i$.

$$H_i = f_i(H_{i-1}, \mathbf{x}_i) \tag{2}$$

Dodatkowo, w zależności od sposobu realizacji douczania, możliwa jest jeszcze inna opcja, w której hipoteza H_i tworzona jest z wykorzystaniem całego dostępnego zbioru przypadków, co można zapisać jako:

$$H_i = f_i(H_{i-1}, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_i) \tag{3}$$

Należy zauważyć, iż w przypadku metod douczania nie definiuje się wartości *n*.

W literaturze zależność (2) związana jest z tak zwanymi systemami uczenia typu online, natomiast zależność (3) z uczeniem inkrementacyjnym [1].

Innym zagadnieniem, gdzie również wykorzystuje się metody douczania, jest uczenie dużych zbiorów danych, w których liczba wektorów oraz zmiennych nie pozwala na dokonanie uczenia na całym zbiorze danych ze względu na moce obliczeniowe stosowanych komputerów. Równie poważnym wyzwaniem jest problem uczenia w rozproszonych bazach danych. Zostało to między innymi opisane jako jedno z głównych wyzwań stojących przed systemami inteligencji obliczeniowej [5].

Trzecim obszarem, w którym zastosowanie metod douczania może przynieść istotne zyski, są wszelkiego rodzaju zadania związane z gospodarką i przemysłem. Wiąże się to z tak zwanym dryftem koncepcji, czyli ciągle zmieniającymi się warunkami, w których system pracuje. Przykładem tego może być zmieniająca się charakterystyka urządzeń lub całych systemów przemysłowych, związana z ich zużyciem lub zmianami związanymi z eksploatacją. Z tego typu sytuacją mamy do czynienia w budowanym, w ramach projektu PBS1/A2/10/2012, systemem ekspertowym do kompleksowej oceny kopalń węgla kamiennego w Polsce [7]. Wskutek zmian zachodzących w kopalni, głównie związanych z eksploatacją kolejnych pokładów węgla, wbudowana w system baza wiedzy, będąca modelem ko-

palni i stosowanych procesów technologicznych, dezaktualizuje się. Jedną z możliwości, badanych w ramach projektu, jest zastosowanie metod douczania inkrementacyjnego.

W artykule omówiono różne algorytmy, pozwalające na douczanie modeli matematycznych, należące do trzech różnych rodzin metod inteligencji obliczeniowej:

- metod wnioskowania opartych na przypadkach,
- metod bazujących na funkcjach jądrowych, w tym ze szczególnym uwzględnieniem algorytmu maszyny wektorów wsparcia (SVM),
- metod wyodrębniania reguł, w tym algorytmy: drzew decyzji oraz sekwencyjnego pokrywania.

Tak szeroki przegląd różnych metod pozwala na dobór odpowiedniej rodziny do stawianego problemu lub charakteru zadania.

2. Wnioskowanie oparte na przypadkach

CBR (ang. *Case Base Reasoning*) powszechnie stosowany jest do rozwiązywania różnych problemów analizy przypadków, gdzie pojedyncze sytuacje odpowiadają wystąpieniu określonych zjawisk. Idea ich działania opiera się na zasadzie, iż podobne zadania powinny być rozwiązywane w podobny sposób, dlatego też systemy te działają gromadząc w bazie wiedzy charakterystyczne przypadki. Innymi słowy, CBR bazuje na rozwiązywaniu nowych problemów na podstawie podobieństwa do już rozwiązanych zagadnień zgromadzonych w bazie wiedzy. CBR między innymi używany jest w analizie przepisów prawa precedensowego, wiele wdrożeń można również zaobserwować w medycynie, np. [15] oraz przemyśle. W systemach CBR główny nacisk położony jest również na maksymalizację dokładności predykcji, jednakże poszczególne przypadki traktowane są jako pojedyncze reguły postępowania, gdzie baza wiedzy jest inkrementacyjnie rozbudowywana. Związane jest to z procesem eksploracji danych online, gdzie każdorazowy sukces lub porażka predykcji zakończony jest umieszczeniem tego przypadku w bazie wiedzy z odpowiednio zmodyfikowaną etykietą. Innymi słowy, douczenie polega na zapisaniu nowego przypadku (wektora) w bazie wiedzy, jest więc to najszybszy i najprostszy sposób douczania modeli.

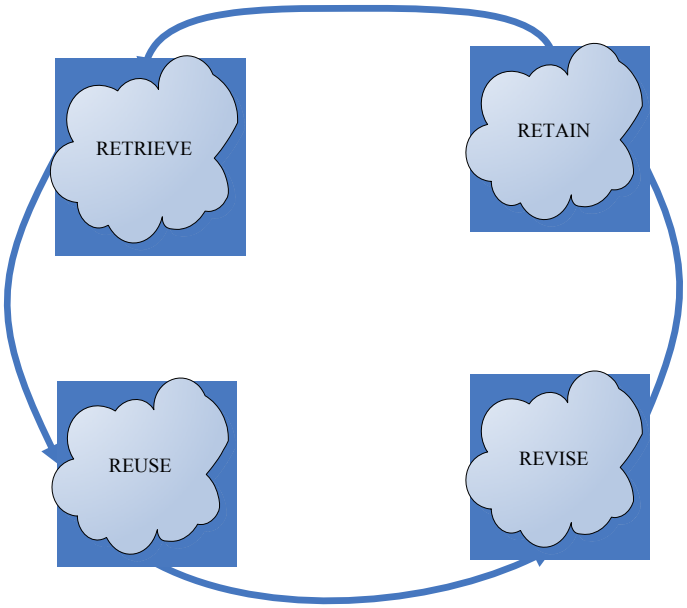
Do metod uczenia poprzez analizę przypadków można zaliczyć: wnioskowanie pamięciowe zaproponowane przez Stanfilla i Waltza (ang. *Memory Based Reasoning*, MBR) [16], uczenie oparte na instancji autorstwa Aha i innych (ang. *Instance Based Learning*, IBL) [1]. Wszystkie one bazują na modyfikacjach algorytmu *k* najbliższych sąsiadów (kNN). Określa go zależność:

$$y(\mathbf{z}) = \text{mean}\left(y\left(\forall i \in [0...k-1] \arg\min_{j=[1...n]/i} \left(D\left(\mathbf{x}_j, \mathbf{z}\right)\right)\right)\right) \tag{4}$$

gdzie **z** oznacza badany przypadek, $\mathbf{x}_j$ to j-ty element bazy wiedzy, k to liczba najbliższych sąsiadów, n – liczba przypadków zgromadzonych w bazie wiedzy, natomiast $D(\mathbf{x}_j, \mathbf{z})$ jest miarą odległości dobieraną odpowiednio do stosowanych atrybutów (jakościowych, ilościowych lub mieszanych). Powyższą zależność należy interpretować jako znalezienie w bazie wiedzy k najbardziej podobnych przypadków do badanego wektora **z** oraz wyznaczenie odpowiedzi poprzez średnią wartość spośród k-sąsiadów. W praktyce systemów CBR często stosuje się ważenie odpowiednich decyzji, tak iż najbardziej podobne przypadki zgromadzone w bazie wiedzy posiadają najwyższy współczynnik istotności przy podejmowaniu decyzji. W takich sytuacjach zamiast średniej stosuje się operator średniej ważonej.

W systemach z rodziny CBR jednym z istotnych problemów jest szybkość podejmowania decyzji przez system, co związane jest z dużą liczbą przypadków zgromadzonych w bazie wiedzy, tym bardziej że z upływem czasu rozmiar bazy wiedzy stopniowo powiększa się. Problem ten powoduje zmniejszenie szybkości działania oraz duże zapotrzebowanie na zasoby systemowe. Dlatego konieczne jest używanie metod, mających na celu przyspieszenie procesu znajdywania najbliższych sąsiadów. Do typowych rozwiązań należą tutaj algorytmy reprezentujące proces poszukiwania najbliższych sąsiadów w postaci drzew, takie jak Kd-tree [19] czy też BallTree [14] lub też metody drzew decyzji jak C4.5, które wykorzystywane są do wstępnej indeksacji i selekcji przypadków [10], zapewniając logarytmiczny czas klasyfikacji badanego przypadku.

Najistotniejszym jednak elementem systemów CBR, który ma szczególne zastosowanie w przypadku aplikacji przemysłowych i powinien być zawsze stosowany, jest schemat przepływu danych. Definiuje się tutaj zamknięty obieg danych, jak pokazano na rys. 1.



Rys. 1. Typowy przebieg procesu w modelach CBR
Fig. 1. Standard process flow for CBR methods

W przypadku języka angielskiego definiuje się go jako 4xR:

- RETRIEVE – pozyskaj najbardziej podobny przypadek lub przypadki,
- REUSE – wykorzystaj wiedzę uzyskaną przez ten przypadek (przypadki) do rozwiązania problemu,
- REVISE – zweryfikuj zaproponowane rozwiązanie,
- RETAIN – zachowaj najbardziej użyteczne części rozwiązania, by w przyszłości rozwiązać podobne problemy.

Tak zdefiniowany obieg informacji można opisać słownie – rozwiązując dane zadanie szukamy w bazie wiedzy najbardziej podobnego przypadku (przypadków), który został już rozwiązany; na podstawie wiedzy o dotychczasowych rozwiązaniach udziel odpowiedzi na nowe zadanie; zweryfikuj uzyskaną odpowiedź i sprawdź jej poprawność; popraw uzyskane rozwiązanie pochodzące z modelu i dokonaj aktualizacji bazy wiedzy o nowo udzieloną odpowiedź.

Na szczególną uwagę zasługują tutaj ostatnie dwa kroki, gdzie weryfikuje się, czy system wymaga douczenia i jeśli tak, następuje jego douczenie w taki sposób, by w przyszłości przewidywał z większą dokładnością. Dodatkowo podczas realizacji tych dwóch kroków następuje sprawdzenie, czym była spowodowana błędna odpowiedź systemu (jeśli była błędna). Ma to istotne znaczenie w warunkach przemysłowych, gdzie niepoprawne rozwiązanie uzyskane z fizycznego systemu mogło być spowodowane bardzo nietypowymi warunkami pracy, sytuacją ekstremalną, wówczas douczenie systemu – umieszczenie w bazie takiego przypadku może być niebezpieczne i prowadzić do utraty stabilności jego działania.

3. Douczanie modeli bazujących na funkcjach jądrowych

Od ponad dekady jednym z modeli predykcyjnych o ugruntowanej renomie jest algorytm maszyny wektorów wspierających, zwany w skrócie jako SVM (ang. *Support Vector Machine*) [3]. Posiada on kilka istotnych cech – zwykle możliwa jest jego interpretacja jako sieci typu RBF (ang. *Radial BasisFinction network*), jednakże przez dobór odpowiedniej funkcji jądrowej, jak np. funkcja sigmoidalna, może on być również interpretowany jako sieć typu MLP (ang. *Multi Layer Perceptron*) z pojedynczą warstwą ukrytą oraz liniowym neuronem wyjściowym. Algorytm SVM opiera się na funkcji decyzyjnej o postaci:

$$h(\mathbf{x}) = sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{5}$$

w której α_i są poszukiwanymi w procesie optymalizacji kwadratowej współczynnikami (wagami funkcji liniowej), $K : R^m \to R_{\geq 0}$ jest funkcją jądrową spełniającą warunek Mercera,

natomiast *b* jest wyrazem wolnym (ang. *bias*) równania liniowego. Proces optymalizacji opiera się na funkcji celu zdefiniowanej jako

$$\Psi(\mathbf{w},\varepsilon)=\frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right)+C\sum_{i=1}^{n}\varepsilon_i \tag{6}$$

gdzie $\mathbf{w}=\sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x},\mathbf{x}_i)$, ε_i jest nieujemną zmienną pomocniczą wynikającą z wprowadzonych ograniczeń, wynikających z możliwości niepoprawnej klasyfikacji przypadków należących do zbioru danych, C – jest stałą definiowaną przez użytkownika.

Pierwszą metodę douczania modelu SVM zaproponował Syed i inni w [17]. Jedną z motywacji powstania tej metody był problem uczenia dużych zbiorów danych. Ze względu na złożoność obliczeniową algorytmu SVM, która dla funkcji jądrowej typu Gaussowskiego waha się pomiędzy O(n^2) a O(n^3), zbiór danych losowo dzielony jest na mniejsze podzbiory, na których następnie uczony jest model SVM w sposób inkrementacyjny.

W pierwszym kroku algorytm uczy się na pierwszym podzbiorze. Uczenie drugiego podzbioru odbywa się z dodatkowym wykorzystaniem wektorów wsparcia wyznaczonych w pierwszym podzbiorze. Podejście to bazuje na dwóch właściwościach algorytmu SVM:

- zerowanie się wielu współczynników α podczas uczenia modelu SVM, co powoduje, że uzyskane rozwiązanie ze zbioru danych wybiera jedynie małą – rzadką reprezentację wektorów wsparcia,
- stabilność kształtu granicy decyzji modelu SVM, gdzie w miarę douczania kształt granicy decyzji przestaje się zmieniać i stabilizuje się, tym samym powoduje to, że często wybierane są te same wektory wsparcia, przy założeniu iż dane pochodzą z identycznych i niezależnych rozkładów prawdopodobieństwa (ang. *identically independent distributed*).

Ogólnie więc algorytm ten można przedstawić jako:

1. dokonaj podziału zbioru danych na podzbiory $\mathbf{X}\rightarrow\mathbf{X}^1\cup\mathbf{X}^2\cup,...,\cup\mathbf{X}^k$
2. naucz SVM na pierwszym podzbiorze $\mathbf{X}^1\Rightarrow\mathbf{SVM}^1$
3. oznacz iterację jako i=1
4. wykorzystaj wektory wsparcia $\mathbf{SV}^i$ nauczonego już $\mathbf{SVM}^i$ i dołącz je do podzbioru $\mathbf{X}^{i+1}$: $\bar{\mathbf{X}}^{i+1}=\mathbf{X}^{i+1}\cup\mathbf{SV}^i$
5. naucz $\mathbf{SVM}^{i+1}$na zbiorze $\bar{\mathbf{X}}^{i+1}$
6. i=i+1, przejdź do kroku 4, aż nie zostanie wykorzystany cały zbiór danych

Rys. 2. Prosty algorytm uczenia inkrementacyjnego algorytmu SVM
Fig. 2. Naive algorithm for incremental learning of SVM

Innym rozwiązaniem douczania modelu SVM jest metoda zaproponowana przez d'Alche-Buc oraz L. Ralaivola [4], która stanowi rozwinięcie powyżej opisanego pomysłu. W metodzie tej autorzy analizują wpływ pojawienia się nowego wektora danych na aktualizację wag poszczególnych neuronów. Opisany przez nich model ma zastosowanie w przypadku wszyst-

kich modeli o lokalnym charakterze, jak sieci typu RBF oraz SVM o Gaussowskich funkcjach jądrowych. W przypadku algorytmu SVM określa się wpływ parametrów α_i na podjętą przez system decyzję podczas wyznaczania wartości wyjściowej dla wektora $\mathbf{x}_n$:

$$\frac{\partial h(\mathbf{x}_n)}{\partial \alpha_i} = y_i k(\mathbf{x}_n, \mathbf{x}_i) \tag{7}$$

Z powyższej zależności wynika, że w przestrzeni danych wejściowych wpływ na podjętą decyzję zmienia się w funkcji odległości Euklidesa (dla funkcji jądrowych typu Gaussowskiego) od badanego wektora $\mathbf{x}_n$. Tym samym douczanie modelu można sprowadzić do rozwiązania problemu optymalizacji kwadratowej i doboru odpowiednich wartości α_i w bezpośrednim otoczeniu *k*-najbliższych wektorów do wektora $\mathbf{x}_n$, co redukuje złożoność obliczeniową modelu SVM, przyjmując $n=k$. Opis zaproponowanego algorytmu przedstawia poniższy schemat:

1. jeżeli $y_n h^{n-1}(\mathbf{x}_n) > 1$, nie douczaj systemu, gdyż hipoteza klasyfikująca $\mathbf{x}_n$ spełniona jest w dostatecznym stopniu
2. j = 1
3. powtarzaj
   a. j = j + 1
   b. dodaj do zbioru roboczego *k*-najbliższych sąsiadów wokół wektora $\mathbf{x}_n$
   c. doucz kandydującą hipotezę h_j^n poprzez optymalizację problemu kwadratowego na zbiorze roboczym
4. do spełnienia kryterium stopu $C_n(j)$

Rys. 3. Algorytm uczenia inkrementacyjnego algorytmu SVM
Fig. 3. Algorithm for incremental learning of SVM

Autorzy w swojej pracy rozważali również wpływ doboru wartości liczby sąsiadów *k* na jakość predykcji modelu. Zaowocowało to powstaniem trzech różnych metod określania sąsiedztwa: pierwsza bazująca na manualnym doborze parametru *k*; druga to tzw. metoda wczesnego zatrzymania oraz trzecia wykorzystująca granicę zaufania zdefiniowaną przez Cristianini oraz Shaw-Taylora [3].

Rozszerzenie metod douczania o uwzględnienie dryftu koncepcji (ang. *concept drift*) w przypadku modelu SVM zaproponował S. Ruping [13]. Wyszedł on z założenia, iż zbiór wektorów wsparcia uzyskanych po nauczeniu modelu SVM reprezentuje nie zbiór danych, lecz funkcję decyzyjną definiującą kształt granicy pomiędzy różnymi klasami. Powoduje to, iż prezentacja nowego przypadku uczącego powinna uwzględniać inny sposób ważenia błędu – inną wagę popełnianej pomyłki niż w przypadku pozostałych wektorów. Jak wspomniano wcześniej, związane jest to z tak zwanym dryftem koncepcji, czyli zmianą kształtu funkcji decyzyjnej związaną z upływem czasu i zmianą parametrów badanego obiektu fizycznego,

którego właściwości system uczy się. W tym celu Ruping zaproponował modyfikację funkcji kosztu klasyfikatora SVM do postaci:

$$\Psi\left(\mathbf{w},\varepsilon,\varepsilon^*\right)=\frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right)+C\left(\sum_{i\in\mathbf{I}}\varepsilon_i+L\sum_{i\in\mathbf{S}}\varepsilon_i\right) \tag{8}$$

gdzie **S** stanowi zbiór dotychczasowych wektorów wsparcia, natomiast **I** to nowy wektor prezentowany podczas uczenia. Wartość *L* przyjmowana jest jako:

$$L=\frac{\text{liczba przypadków}}{\text{liczba wektorów wsparcia}} \tag{9}$$

Niezależna od pozostałych koncepcja douczania modelu SVM została zaproponowana przez Cauwenberghsa oraz T. Poggio [11]. W odróżnieniu od pozostałych metod autorzy skoncentrowali się na zapewnieniu spełnienia warunków Kuhn-Tuckera (KT), które są wymagane podczas realizacji procesu optymalizacji. Oznaczając *g(.)* jako warunek KT, **S** jako zbiór wektorów tworzących margines, dokładnie leżących na marginesie ($y_i h\left(\mathbf{x}_i\right)=1$), zbiór **E** – wskazujący błędne wektory wsparcia rozszerzające margines oraz **R** jako zbiór zignorowanych wektorów wewnątrz marginesu, zaproponowany algorytm można przedstawić jako:

1. inicjalizuj α_n jako 0 (współczynnik α nowo prezentowanego wektora)
2. jeżeli $g\left(\mathbf{x}_\mathrm{n}\right)>0$, przerwij
3. jeżeli $g\left(\mathbf{x}_\mathrm{n}\right)\le 0$, zastosuj możliwie największy przyrost α_n aż do spełnienia jednego z warunków:
   a. $g\left(\mathbf{x}_\mathrm{n}\right)=0$
   b. $\alpha_n=0$
   c. dokonaj migracji elementów zbioru treningowego $\left\{\mathbf{X}\setminus\mathbf{x}_n\right\}$ pomiędzy **S**, **R**, **E**
4. powtarzaj, jeśli konieczne

Rys. 4. Algorytm inkrementacyjnego uczenia modelu SVM ze wsparciem dla dryftu koncepcji
Fig. 4. Incremental and stream learning SVM with concept drift support

Zaproponowana przez autorów metoda pozwala również na oduczanie klasyfikatora, co przykładowo może zostać wykorzystane przy estymacji jego dokładności o postaci testu jeden pozostaw (ang. *leave one out*), lub do oduczania systemu w przypadku zaprezentowania mu w procesie uczenia wektorów odstających.

4. Inkrementacyjne uczenie systemów regułowych

Podstawową wadą sieci neuronowych, jak i modeli bazujących na funkcjach jądrowych jest tak zwany problem czarnej skrzynki, który wiąże się z brakiem zdolności interpretacji wiedzy zgromadzonej przez sieć neuronową. Wady tej pozbawione są wszelkiego rodzaju

systemy regułowe, w których wiedza reprezentowana jest w postaci listy reguł bądź w postaci drzewa decyzyjnego. Podobnie jak w przypadku sieci neuronowych, tak i tutaj większość algorytmów projektowana była przy założeniu dostępności całego treningowego zbioru danych, jednakże, jak już powyżej wspomniano, warunek ten często może nie być spełniony.

Do najpopularniejszych klasycznych algorytmów indukcji listy reguł należy algorytm CN2 zaproponowany przez Clarka i Nibletta [2]. Składa się on z dwóch głównych elementów – głównej pętli, w której wywoływana jest procedura poszukiwania reguły oraz obsługa zbioru danych uczących oraz funkcji, która poszukuje najbardziej odpowiedniej reguły bazując na estymacji prawdopodobieństwa z uwzględnieniem poprawki Laplace'a.

D. Weber w [20] zaproponował modyfikację tego algorytmu zwaną ICN, która uwzględnia możliwość inkrementacyjnego uczenia. W tym celu zdefiniował on zbiór **RS** zawierający kompletny zbiór reguł, zbiór **GS** (tzw. dobry zbiór) zawierający przypadki poprawnie sklasyfikowane przez przynajmniej jedną regułę z **RS**, oraz zbiór **BS** (tzw. zły zbiór) zawierający przypadki źle sklasyfikowane. Idea algorytmu polega na próbie klasyfikacji każdego nowego wektora treningowego oraz aktualizacji częstości klasy poprawnie klasyfikującej go reguły, a sam wektor, w przypadku kiedy był poprawnie sklasyfikowany, trafia do zbioru **GS**, natomiast jeśli został niepoprawnie sklasyfikowany lub sklasyfikowany przez tzw. regułę domyślną (najbardziej ogólną), trafia do zbioru **BS**. Tworzenie nowej reguły następuje w momencie przekroczenia określonej liczby przypadków z danej klasy w zbiorze **BS**. Wówczas wywoływana jest standardowa procedura szukania wiązką nowej reguły (identyczna z oryginalnym algorytmem CN), która pozwoli na klasyfikację elementów zbioru **BS**, jednocześnie nie psując uzyskanych wyników dla zbioru **GS**. Jeżeli zostanie znaleziona odpowiednia reguła statystycznie istotna (autor wykorzystał w tym celu test χ^2), wówczas dodawana jest ona do zbioru **RS**.

Alternatywne metody douczania systemów regułowych zaproponował Shen w [21], tworząc grupę algorytmów o nazwie CDL1 … CDL4 (ang. *Complementary Discrimination Learning*). W odróżnieniu od algorytmu ICN nowe reguły tworzone są tutaj przez douczanie – uszczegółowienie istniejących reguł. Oznaczając **D** jako listę decyzyjną dla zbioru koncepcji $C_1,\ldots,C_t$, oraz **E** jako zbiór przypadków należących do tej samej klasy co obecny wektor uczący $\boldsymbol{x}$ należący do C_x, procedurę tworzenia reguł według algorytmu CDL4 można przedstawić jako:

1. Jeżeli **D** jest pusty, to stwórz regułę ogólną klasyfikującą wszystkie przypadki do C_x
2. jeśli nie, to
   a. niech $D_j = (f_j, v_j)$ będzie decyzją pokrywającą przypadek **x** (f_j– wartość cechy, v_j – etykieta klasy związana z daną regułą)
   b. jeżeli decyzja jest poprawna, wówczas zapisz **x** jako przypadek z reguły D_j i wyjdź z pętli
   c. jeśli nie, to
      i. znajdź różnice pomiędzy D_j oraz **x** i oznacz jako $G=(g_1,.. g_m)$ (zbiór wartości atrybutów, na których różnią się elementy z D_j oraz **x**)
      ii. zamień regułę $D_j = (f_j, v_j)$ na zbiór reguł $(f_j \wedge g_1, v_j)$, $(f_j \wedge g_2, v_j)$,…, $(f_j \wedge g_m, v_j)$
      iii. rozdziel przypadki klasyfikowane przez D_j pomiędzy nowe reguły
      iv. jeżeli D_j było ostatnią decyzją, wówczas dołącz (prawda,C_x) jako ostatnią decyzję z **D**

Rys. 5. Algorytm inkrementacyjnej indukcji reguł CDL
Fig. 5. CDL incremental algorithm for rule induction

Osobny problem uczenia stanowi zadanie douczania reguł zapisanych w postaci drzew decyzji. Zagadnieniem tym zajmowali się między innymi Schlimmer oraz Fisher, proponując algorytm ID4 wzorowany na drzewie decyzji ID3 opracowanym przez Quinlana [12].

Aby zmniejszyć zapotrzebowanie pamięciowe algorytmu, autorzy ID4 zastosowali metodę, która nie wymaga przechowywania dotychczasowej bazy danych treningowych, gdyż wszystkie niezbędne informacje przechowywane są w węzłach drzewa w postaci częstości występowania określonych wartości atrybutów należących do poszczególnych klas. Zaproponowany przez nich algorytm można opisać jako:

1. po prezentacji nowego wektora treningowego **x**, dla każdego możliwego atrybutu testowego (użytego przy tworzeniu drzewa) zaktualizuj obecny węzeł, zwiększając liczbę pozytywnych lub negatywnych przypadków dla wartości tego atrybutu
2. jeśli wszystkie przypadki znajdujące się w węźle są pozytywne (negatywne), wówczas taka decyzja w drzewie staje się liściem
3. jeśli nie
   a. jeśli obecny węzeł jest liściem, to zmień go na węzeł testowy, zawierający atrybut testowy o największej wartości współczynnika zysku informacyjnego (ang. *Information Gain* IG)
   b. w przeciwnym razie, jeśli obecny węzeł jest węzłem decyzyjnym zawierającym atrybut, który nie maksymalizuje IG
      i. zamień ten atrybut na atrybut maksymalizujący IG
      ii. usuń całe poddrzewo znajdujące się poniżej danego węzła
   c. rekurencyjnie dokonaj aktualizacji drzewa poniżej obecnego węzła decyzyjnego oraz rozbuduj je jeśli trzeba

Rys. 6. Inkrementacyjny algorytm budowy drzewa ID4
Fig. 6. ID4 incremental decision tree algorithm

Dalsze rozwinięcie opisanego algorytmu przedstawił Utgoff [18] pod nazwą ID5R. W zaproponowanej metodzie autor zrezygnował z usuwania fragmentów drzewa poniżej węzła decyzyjnego, który podjął błędną decyzję, jak miało to miejsce w przypadku ID4 na rzecz

reorganizacji drzewa, tak by dany atrybut testowy o największym współczynniku IG znalazł się na miejscu korzenia. Cechą charakterystyczną zaproponowanej metody jest możliwość uzyskania identycznego drzewa z uzyskanym według algorytmu ID3 (identyczny wybór węzłów decyzyjnych oraz ich warunków). Proces manipulacji drzewem podczas reorganizacji pozwala na zmianę struktury drzewa bez utraty jego spójności. Wymaga on jednak posiadania licznika częstości klas w każdym z węzłów oraz dodatkowo w stosunku do ID4 wszystkich przypadków znajdujących się w poszczególnych liściach. Zaproponowany przez niego algorytm można przedstawić jako:

1. jeśli drzewo jest puste, wówczas zdefiniuj je w postaci nierozwiniętej, ustawiając etykietę klasy jako C_x, oraz zbiór przypadków zawierający jedynie ten jeden przypadek
2. w przeciwnym razie, jeśli drzewo jest nierozwinięte oraz nowy przypadek **x** jest z tej samej klasy co etykieta liścia, i dodaj przypadek do listy przechowywanych w tym węźle
3. w przeciwnym przypadku
    a. jeśli drzewo jest nierozwinięte, wówczas rozwiń je o jeden poziom, wybierając atrybut testowy arbitralnie
    b. dla atrybutu testowego oraz dla wszystkich nietekstowych atrybutów w obecnym węźle zaktualizuj liczebność pozytywnych i negatywnych przypadków dla danej wartości tego atrybutu
    c. jeśli obecny węzeł nie zawiera atrybutu testowego o maksymalnej wartości IG, wówczas
        i. zrestrukturyzuj drzewo, tak by atrybut o największym IG stał się korzeniem dla kolejnych węzłów
        ii. rekurencyjnie odtwórz najlepsze atrybuty testowe w każdym poddrzewie, z wyjątkiem tego, który będzie aktualizowany poniżej
        iii. rekurencyjnie zaktualizuj drzewo decyzji poniżej obecnego węzła decyzyjnego, dodaj gałęzie, jeśli konieczne

Rys. 7. Inkrementacyjny algorytm budowy drzewa ID5R
Fig. 7. ID5R incremental decision tree algorithm

5. Podsumowanie

W pracy omówiono i zaprezentowano zbiór różnych rodzin algorytmów cechujących się możliwością ciągłego douczania modeli predykcyjnych. Tak szerokie spektrum analizy wynika z częstego problemu doboru odpowiedniej rodziny algorytmów do stawianego problemu. Dla prostych i łatwych w implementacji rozwiązań najbardziej celowym rozwiązaniem są systemy z rodziny CBR, które w naturalny sposób wspierają możliwość douczania. Zwykle cechuje je jednak nie najwyższa dokładność predykcji, co może determinować konieczność wykorzystania metod z rodziny bazujących na funkcjach jądrowych. Rodzina ta jest jednak dużo bardziej wymagająca pod względem implementacyjnym, a wydobyta przez system wiedza trudna do interpretacji. Dlatego też dla tego typu zadań celowe jest zastosowa-

nie metod reprezentujących wiedzę w postaci zbioru reguł, które pozwalają na jej analizę i weryfikację.

Praca powstała w wyniku prowadzonych badań związanych z możliwością ciągłej adaptacji budowanej bazy wiedzy systemu ekspertowego kopalń w ramach projektu (*PBS1/A2/10/2012).* Przewidywany dalszy kierunek badań obejmuje implementację i empiryczną weryfikację wybranych omówionych w pracy rozwiązań.

Publikacja została opracowana w ramach projektu PBS1/A2/10/2012 „Opracowanie systemu ekspertowego do oceny efektywności środowiskowej, ekonomicznej i społecznej kopalń węgla kamiennego w Polsce" finansowanego przez Narodowe Centrum Badań i Rozwoju w ramach Programu Badań Stosowanych.

BIBLIOGRAFIA

1. Aha D., Kibler D., Albert M. K.: Instance-Based Learning Algorithms. Machine Learning, Vol. 6, Kluwer Academic Publishers, 1991, s. 37÷66.
2. Clark P., Niblett T.: The CN2 Induction Algorithm. Machine Learning Journal, Vol. 3, No. 4, 1989, s. 261÷283.
3. Cristianini N., Shawe-Taylor J.: An Introduction to Support Vector Machine and other kernel-based methods. Cambridge University Press, 2000.
4. d'Alche-Buc F., Ralaivola L.: Incremental Learning Algorithms for Classification and Regression: local strategies. AIP Conference Proceedings, IOP Institute of Physics Publishing LTD, 2002, s. 320÷329.
5. Duch W., Mandziuk J.: Challenges for Computational Intelligence. Studies in Computational Intelligence Series, Springer, 2007.
6. Giraud-Carrier C.: A Note on the Utility of Incremental Learning. AI Communications, Vol. 13, No. 4, 2000, s. 215÷223.
7. Golak S., Wieczorek T.: Koncepcja systemu ekspertowego do oceny i poprawy ekoefektywności kopalń. Studia Informatica, Vol. 35, No. 2(116), Gliwice 2014.
8. Kwok T. M., Yeung D. Y.: Objective Functions for Training New Hidden Units In Constructive Neural Networks. IEEE Trans. On Neural Networks, Vol. 8, No. 5, 1999, s. 1131÷1148.
9. Lim J., Ross D., Lin R., Yang M.: Incremental Learning for Visual Tracing. Proceedings NIPS'2004, 2004.
10. Plaza E., Aamodt A.: Case-based Reasoning-Fundamental Issues. Methodological Variations, and System Approaches, AICOM, Vol. 7, No. 1, 1994, s. 39÷59.

11. Poggio T., Cauwenberghs G.: Incremental and decremental support vector machine learning. Adv. in NIPS, Vol. 13, MIT Press, 2001, s. 409÷416.
12. Quinlan R.: Induction of decision trees. Machine Learning, Vol. 1, No. 1, 1986, s. 81÷106.
13. Ruping S.: Incremental Learning with Support Vector Machines. Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01), IEEE, 2001, s. 641÷642.
14. Samet H.: The design and analysis of spatial data structures. Addison Wesley Reading, MA 1990.
15. Schmidt R., Pollwein B., Gierl L.: Experiences with Case-Based Reasoning Methods and Prototypes for Medical Knowledge-Based Systems. Lecture Notes in Computer Science, Vol. 1620, Springer Verlag, 1999, s. 124÷132.
16. Stanfill C., Waltz D.: Toward memory-based reasoning. Communications of the ACM, Vol. 29, No. 12, 1986, s. 1213÷1228.
17. Syed N., Liu H., Sung K.: Incremental Learning with Support Vector Machine. Proceedings of the Workshop on Support Vector Machine at the International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden 1999.
18. Utgoff P. E.: Incremental Induction of Decision Trees. Proc. of the Eleventh International Conference on Machine Learning, Morgan-Kaufmann, 1994, s. 318÷325.
19. Wess S., Altho K. D., Derwand G.: Using k-d trees to improve the retrieval step in case-based reasoning. Topics in Case-Based Reasoning, Springer Verlag, Berlin 1994, s. 167÷181.
20. Weber G. D.: Beam Search in Incremental Rule Learning. Proc. of the Fourteenth Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, Ohio 2003, s. 151÷156.
21. Shen W.-M.: Efficient Incremental Induction of Decision Lists – Can Incremental Learning Outperform Non-Incremental Learning? Technical Report, USC-ISI-96-012, Information Sciences Institute, University of Southern California, 1996.
22. Wieczorek T., Blachnik M., Maczka K.: Building a model for time reduction of steel scrap meltdown in the electric arc furnace (EAF). General strategy with a comparison of feature selection methods. LNCS, Springer Verlag, Vol. 5097, 2008.

Abstract

Classical training methods of computational intelligence models are based on building a knowledge base, assuming that the entire, complete set of learning vectors is available. This assumption is not always met, particularly in issues related to the industry. In the paper we

provide an overview of a broad group of algorithms supporting incremental learning. The analysis includes the family of Case Based Reasoning (CBR) which seems to be the most natural approach as every new training instance can be simply added to the set of reference cases. Issues related to the CBR are computational complexity during prediction process, and prediction accuracy. Then we analyze the Support Vector Machine (SVM) and its extensions which support incremental learning. The SVM has proven high prediction accuracy but it lacks comprehensibility. Finally we discuss incremental rule extraction methods based on both: incremental decision trees and incremental sequential covering algorithms. These methods provide high comprehensibility but lack high accuracy.

All of the discussed algorithms were chosen to fulfill and maximize one of the criteria: the prediction model accuracy, the implementation simplicity and comprehensibility of extracted knowledge. By doing this, the prediction system engineer can easily choose and find an appropriate tradeoff between all of these criteria.

Adresy

Marcin BLACHNIK: Katedra Informatyki Przemysłowej, Politechnika Śląska,
ul. Krasińskiego 8, 40-019 Katowice, Polska, mblachnik@polsl.pl.
Tadeusz WIECZOREK: Katedra Informatyki Przemysłowej, Politechnika Śląska,
ul. Krasińskiego 8, 40-019 Katowice, Polska, wieczorek@polsl.pl.