Techniki heurystycznego uczenia warstw ukrytych w sieci głębokiej

inż. Jacek Miszczak
inż. Filip Malczak

3.06.2015

Streszczenie

Your abstract goes here... ...

Spis treści

Rozdział 1

Wstęp

Niniejszy raport jest podsumowaniem projektu którego temat brzmiał „Techniki heurystycznego uczenia warstw ukrytych w sieci głębokiej". Od czytelnika oczekuje się podstawowej znajomości dziedziny sieci neuronowych i obliczeń miękkich. Bardziej szczegółowy opis zastosowanych technik znajduje się w rozdziale 2.

Pierwotnym celem projektu było zbadanie alternatywnego podejścia do uczenia sieci głębokich. Miało ono polegać na zbudowaniu stosu autokoderów (opisanych w sekcji 2.1), uczonych w dwóch krokach. Pierwszym z nich miała być zastosowanie popularnej metody propagacji wstecznej. Niestandardową częścią uczenia był krok drugi, polegający na wykorzystaniu heurystyk optymalizacyjnych (konkretnie algorytmów ewolucyjnych i rojowych) w celu poprawienia jakości działania wstępnie wyuczonej sieci.

Zamierzenia tego nie udało się zrealizować, czego powody zostały opisane w rozdziale 3. Aby w ramach tego projektu odkryć nową wiedzę nt. sieci neuronowych i ich uczenia, zdecydowano się na zmianę tematu.

Nowym celem projektu zostało zbadanie jakości uczenia pojedynczego autokodera przy użyciu podejścia heurystycznego i mieszanego. Wyniki dla alternatywnych metod nauki porównano z klasyczną metodą propagacji wstecznej.

Rozdział 2

Wprowadzenie

W rozdziale tym opisano zagadnienia i podejścia wykorzystywane podczas realizacji projektu, którego podsumowaniem jest ten raport. Przyjęto założenie, że czytelnik jest zaznajomiony z podstawowymi pojęciami dotyczącymi zagadnienia sieci neuronowych, takimi jak pojęcie warstw wejściowych, ukrytych i wyjściowych, algorytm propagacji wstecznej bądź wagi pomiędzy neuronami. Skupiono się tutaj na zagadnieniach stanowiących nowości w dziedzinie oraz takich spoza dziedziny obliczeń neuronowych.

2.1 Autokodery i stosy autokoderów

Autokodery (*ang. autoencoders*) stanowią jedną z istniejących architektur głębokich sieci neuronowych. Znajdują one zastosowanie zarówno w redukcji wymiarowości danych jak i w odnajdywaniu ich wewnętrznej, ukrytej reprezentacji.

Pojedynczy autokoder jest trójwarstwową siecią neuronową, której celem jest odtwarzanie na wyjściu danych zadanych na wejściu [4]. W związku z tym rozmiar warstwy wyjściowej jest równy rozmiarowi warstwy wejściowej. Rozmiar warstwy ukrytej zależy od przeznaczenia projektowanego autokodera – wykorzystanie warstwy mniejszej od warstwy wejściowej skutkuje otrzymaniem sieci potrafiącej zredukować wymiarowość danych, co może być także wykorzystane do zadania kompresji. Gdy warstwa ukryta jest większa, to sieć taka powinna nauczyć się wewnętrznej reprezentacji danych.

Wykorzystanie pojedynczego autokodera do znacznej redukcji wymiarowości danych jest jednak nieefektywne. Gdy różnica pomiędzy rozmiarami warstwy wejściowej a warstwy ukrytej jest zbyt duża, to nauczenie sieci poprawnego odtwarzania tak mocno zredukowanych wzorców jest trudne. Rozwiązaniem tego problemu jest idea stosu autokoderów. Polega ona na zbudowaniu sieci neuronowej z większa liczbą warstw ukrytych, gdzie kolejne warstwy mają coraz mniejsze rozmiary. Różnica rozmiarów pomiędzy kolejnymi warstwami jest tym samym mniejsza.

Aby wykorzystać stos autokoderów do odtwarzania należy utworzyć sieć zawierająca nieparzystą liczbę warstw ukrytych, których rozmiary są symetryczne. Dzięki temu można wskazać na środkową warstwę ukrytą, najmniejszą ze wszystkich, zawierającą zredukowane dane. Warstwę taką nazywa się warstwą kodu. Warstwy leżące pomiędzy warstwą wejściową, a warstwą kodu, włącznie z wymienionymi, nazywa się koderem (*ang. encoder*). Dekoder (*ang. decoder*) zaś stanowią warstwy od warstwy kodu do warstwy wyjściowej.

Zarówno pojedyncze autokodery jak ich stosy można uczyć za pomocą propagacji wstecznej za błąd przyjmując dowolną funkcję rozbieżności danych wejściowych i wyjściowych, jak na przykład błąd średniokwadratowy. Uczenie sieci z dużą liczbą warstw ukrytych za pomocą propagacji wstecznej jest jednak nieefektywne [4] z powodu dyfuzji gradientu. Aby rozwiązać ten problem stosowane jest podejście znane z sieci DBN (*Deep Belief Network*) [6]. Polega ono na oddzielnym uczeniu pojedynczych, trójwarstwowych, autokoderów.

W każdym etapie takiej nauki autokoder uczony jest odtwarzać dane zadane na wejściu nie zważając na resztę sieci. Wejściem warstwy n jest jednak wyjście warstwy $n-1$. Tym samym pierwsza warstwa ukryta uczona jest rekonstrukcji danych z warstwy wejściowej całej sieci. Aby było to możliwe należy utworzyć jeszcze jedną warstwę o rozmiarze takim samym jak warstwa wejściowa. Po zakończeniu nauki pojedynczej warstwy odłączana jest dodana wcześniej warstwa wykorzystywana podczas nauki. Już nauczone wagi między warstwą wejściową a ukrytą są zapamiętywane. Następnie algorytm zaczyna naukę kolejnej warstwy, która tym razem musi nauczyć się rekonstrukcji danych uzyskanych z wcześniejszej warstwy ukrytej.

Poza kompresją i redukcją wymiarowości danych stosy autokoderów znajdują także zastosowanie w innych zadaniach z dziedziny maszynowego uczenia, takich jak rozpoznawanie czy opisywanie. Czyni się to poprzez eliminację warstw stanowiących dekoder, zamiast nich dołączając po warstwie kodu algorytm odpowiedni do rozważanego zadania. Może to być przykładowo klasyfikator, także w formie perceptronu zachowując tym samym neuronową naturę stosowanego rozwiązania. Algorytm taki operuje na danych o zredukowanej wymiarowości stanowiących swego rodzaju wewnętrzną reprezentację oryginalnych danych.

2.2 Algorytmy genetyczne

Ewolucja to proces zachodzący w naturze odpowiedzialny za dopasowywanie się osobników danego gatunku do środowiska w jakim żyją. Podstawą tego procesu jest przetrwanie lepiej przystosowanych osobników, dziedziczenie i mutacja.

Przetrwanie lepiej przystosowanych osobników to zasada zgodnie z którą

osobniki lepiej dopasowane do środowiska mają większą szansę na przeżycie, a co za tym idzie, na wydanie potomstwa. Oznacza to, że rodzice większości osobników z kolejnego pokolenia będą radzić sobie w tym środowisku lepiej niż pozostałe osobniki z ich pokolenia.

Dziedziczenie to zjawisko przekazywania cech rodziców dzieciom. Odbywa się ono podczas rozmnażania, a więc zachodzi między dwojgiem rodziców, a potomstwem. Kod genetyczny potomstwa tworzony jest przez losowe łączenie odpowiednich części kodu genetycznego rodziców, przez co kolejne pokolenie dzieli ich cechy. W ten sposób losowe osobniki przejmą od rodziców te cechy, które pozwalały im się dopasować do środowiska i w niektórych przypadkach pozwoli im to na jeszcze lepsze dopasowanie się do otoczenia. Część osobników przejmie jednak nie tylko cechy poprawiające ich szansę przetrwania, ale również cechy negatywne, co przełoży się na ich gorsze dopasowanie.

Mutacja to zjawisko zachodzenia losowych zmian w kodzie genetycznym osobnika, przez które ma on szansę zyskać nowe cechy, które w niektórych przypadkach doprowadzą do lepszego dopasowania. Osobniki z przypadkowymi zmianami, które poprawiają ich dopasowanie mają większe szanse na przeżycie i wydanie potomstwa, *ergo* przypadkowe pozytywne zmiany powinny zostać rozpropagowane wśród osobników przyszłych pokoleń.

Algorytmy ewolucyjne to rodzina heurystyk naśladujących proces ewolucji w celu optymalizacji [2]. Pojedynczy punkt w przestrzeni rozwiązań jest w nich nazywany osobnikiem. Osobniki możemy między sobą porównywać pod względem wartości optymalizowanej funkcji dla nich, a relacja mniejszości (dla problemów minimalizacji) lub większości (dla problemów maksymalizacji) reprezentuje relację bycia lepiej przystosowanym do środowiska. Ponadto, na osobnikach określone są operatory mutacji i krzyżowania, które mają na celu kolejno symulację losowych zmian w osobniku i tworzenie nowych osobników na podstawie starych. Heurystyka polega na wielokrotnym przetworzeniu populacji (czyli zbioru osobników) poprzez zastosowanie każdego z operatorów z pewnym prawdopodobieństwem. W każdym kroku (nazywanym w nomeklaturze algorytmów ewolucyjnych pokoleniem) do dotychczasowej populacji dołączane są wyniki działania tych operatorów (czyli zbiory osobników zmutowanych i potomstwa), a następnie wybierana jest nowa populacja, używana w kolejnym kroku. Aby odwzorować zasadę przetrwania najlepiej dopasowanych osobników do kolejnej populacji wybierane są z wyższym prawdopodobieństwem osobniki lepiej przystosowane.

2.3 Particle Swarm Optimization

Kolejna heurystyką wzorowaną na naturze jest optymalizacja za pomocą roju cząsteczek (*ang. Particle Swarm Optimization, PSO*). Zainspirowana została ona przez obserwację zachowań stad zwierząt, w szczególności pta-

ków i owadów. Jej celem jest osiągnięcie inteligencji obliczeniowej poprzez wykorzystanie analogii interakcji społecznych w przeciwieństwie do odwzorowywania zdolności kognitywnych jednostki [8].

W metodzie PSO pewna liczba bytów (cząsteczek) jest umieszczana w D-wymiarowej przestrzeni rozwiązań rozważanego problemu. Każda z takich pozycji reprezentuje jedno z możliwych rozwiązań. Następnie każda cząsteczka ewaluuje optymalizowaną funkcję celu w swojej obecnej pozycji. W kolejnym, kluczowym, kroku działania heurystyki dla wszystkich cząsteczek obliczany jest wektor przemieszczenia na podstawie rozwiązania obecnego, najlepszych rozwiązań znalezionych zarówno przez obecnie rozważany byt jak i przez cały rój oraz pewnego czynnika losowego. Po przemieszczeniu wszystkich cząsteczek rozpoczyna się kolejna iteracja. W miarę upływu czasu cały rój, analogicznie do stada ptaków wspólnie poszukujących pożywienia, zbliży się do globalnego optimum funkcji celu. Metoda kończy swoje działanie gdy osiągnięty zostanie warunek stopu, w najprostszym przypadku reprezentowany przez limit liczby iteracji.

Każda cząsteczka modelowana jest jako 3 wektory D-wymiarowe, gdzie D jest liczbą wymiarów przeszukiwanej przestrzeni rozwiązań:

- $\vec{x}_i$ - obecna pozycja,
- $\vec{p}_i$ - historycznie najlepsza pozycja,
- $\vec{v}_i$ - prędkość.

Obecna pozycja $\vec{x}_i$ może być rozumiana jako współrzędne reprezentujące pozycję w przestrzeni. W każdej iteracji algorytmu obecna pozycja jest ewaluowana jako możliwe rozwiązanie problemu. Jeśli ta pozycja okaże się lepsza od wszystkich znalezionych wcześniej, to jej koordynaty zapisywane są w wektorze $\vec{p}_i$, a wartość funkcji celu w zmiennej $pbest_i$. Celem jest odszukiwanie coraz lepszych pozycji i aktualizowanie $\vec{p}_i$ oraz $pbest_i$. Nowa pozycja obliczana jest poprzez dodanie wektora prędkości $\vec{v}_i$ do pozycji $\vec{x}_i$ i algorytm działa poprzez modyfikację $\vec{v}_i$.

Rój cząsteczek stanowi coś więcej niż tylko kolekcję pojedynczych cząsteczek. Pojedynczy byt nie posiada niemalże żadnych możliwości rozwiązania zadanego problemu, rozwój odbywa się poprzez interakcję cząsteczek. Byty ułożone są według pewnej topologii komunikacji w sąsiedztwa [5]. Najprostszym wariantem jest sąsiedztwo globalne, w którym wszystkie cząsteczki mogą się ze sobą komunikować.

W procesie optymalizacji rojem cząsteczek prędkość każdego z bytów jest iteratywnie modyfikowana tak, by cząsteczki stochastycznie oscylowały wokół pozycji $\vec{p}_i$ oraz $\vec{p}_g$, gdzie $\vec{p}_g$ oznacza najlepsze rozwiązanie znalezione przez byty należące do sąsiedztwa rozważanej cząstki i.

Modyfikacja wektora prędkości odbywa się według wzoru przedstawionego w równaniu (2.1) [8]:

$$\vec{v_i} = \vec{rand}(0, \Phi_1) \otimes (\vec{p_i} - \vec{x_i}) + \vec{rand}(0, \Phi_2) \otimes (\vec{p_g} - \vec{x_i}). \tag{2.1}$$

Notacja:

- $\vec{rand}(0, \Phi_i)$ oznacza wektor losowych liczb równomiernie rozłożonych na $[0, \Phi_i]$. Jest on generowany losowo w każdej iteracji i dla każdej cząsteczki,
- $\otimes$ oznacza operację mnożenia wektorów zdefiniowaną w następujący sposób: $[x_0, x_1, \ldots, x_n] \otimes [y_0, y_1, \ldots, y_n] = [x_0 y_0, x_1 y_1, \ldots, x_n y_n]$,
- Φ_i oznacza siłę wpływu czynnika losowego na poruszanie się cząsteczek.

Rozdział 3

Powody zmiany celu

Pierwotnym celem projektu było zbadanie skuteczności stosu autokoderów w problemie opisywania obiektów przy uczeniu z wykorzystaniem heurystyk oraz propagacji wstecznej. Już na wczesnym etapie wykonywania badań stwierdzono jednak, iż problemy występujące w stosowanych rozwiązaniach uniemożliwiają uzyskanie miarodajnych wyników. Zdecydowano się tym samym na zmianę celu wykonywania projektu i idei badań.

W rozdziale tym opisano powody zmiany celu pracy oraz przedstawiono nowy cel.

3.1 Pierwotny cel

Pierwotnym celem było zastosowanie głębokich stosów autokoderów z dodaną warstwą klasyfikującą w postaci perceptronu do problemu opisywania obiektów. Dla przypomnienia: zadanie opisywania, znane także jako klasyfikacja wieloklasowa, polega na przypisaniu obiektowi na podstawie jego obserwacji pewnego podzbioru etykiet ze ściśle ustalonego zbioru. Najbardziej oczywistym przykładem takiego problemu jest zagadnienie opisywania obrazów, czyli określenia, jakie obiekty występują na obrazie i przypisanie mu etykiet odpowiadających tym obiektom.

Uczenie głębokich sieci neuronowych z pomocą algorytmu propagacji wstecznej jest bardzo czasochłonne, szczególnie gdy wykorzystywane jest podejście do zachłannego uczenia poszczególnych warstw opisane w sekcji 2.1. Pierwotnym celem było wykorzystanie podejść heurystycznych oraz mieszanych (wykorzystujących zarówno propagację wsteczną jak i heurystyki) celem przyspieszenia procesu nauki sieci oraz zwiększenia jej skuteczności w problemie rozpoznawania. Skuteczność tak uczonych sieci (reprezentowana przez miarę F) miała zostać porównana z sieciami uczonymi jedynie z pomocą propagacji wstecznej.

Wykorzystane miało zostać podejście oparte o uczenie zachłanne – pojedyncze autokodery były uczone indywidualnie, ale po pewnej liczbie epok

propagacji wstecznej (także zerowej) wykorzystywano podejście oparte o heurystykę do dalszej optymalizacji wag. W analogiczny sposób uczona była warstwa ostatnia, klasyfikująca.

Przykład opisywania obrazów został wcześniej podany nie bez powodu – zamierzano skorzystać ze zbioru danych *corel5k* [1] reprezentującego właśnie problem opisywania obrazów.

3.2 Napotkane problemy

Zamierzano wykorzystać zbiór danych zawierający jedynie atrybuty o wartościach binarnych, tj należących do zbioru $\{0, 1\}$. Wybrano tym samym zbiór *corel5k* zawierający 5000 obiektów reprezentowanych przez 499 cech binarnych i opisanych z pomocą 374-elementowego zbioru etykiet. Szybko natknięto się jednak na ograniczenie natury technicznej – w wykorzystywanej bibliotece do obliczeń neuronowych *encog* liczba wag pomiędzy warstwami zapisywana była w zmiennej typu *Integer*. Dla tak dużej liczby cech oraz połączeń typu każdy-z-każdym realna liczba wag przekraczała dopuszczalny zakres dla zmiennej tego typu.

Zdecydowano się tym samym wykorzystać zbiór danych, w którym instancje opisane są z pomocą mniejszej liczby cech. Nie znaleziono jednak zbioru zawierającego mniej cech binarnych. Zrezygnowano tym samym z ograniczenia na typ cech i wykorzystano zbiór *emotions* [1] zawierający 593 obiekty reprezentowane przez 72 cechy rzeczywistoliczbowe, i opisane z pomocą 6 etykiet. Przed wykorzystaniem tych danych w sieci neuronowej wymagane było wykonanie normalizacji wartości cech, czyli takie ich przekształcenie, by mieściły się w zakresie (0, 1). Było to możliwe dzięki podanym w zbiorze danych dziedzinom kolejnych cech.

Wymóg operowania na znormalizowanych danych bierze się z tego, iż w autokoderach używane były sigmoidalne funkcje aktywacji których przeciwdziedziną jest przedział (0, 1). Autokoder taki nie jest w stanie odtworzyć wartości spoza tego zakresu.

Zmiana wykorzystywanego zbioru danych poskutkowała jednak tym, iż wykorzystywane autokodery miały za zadanie odtwarzać wartości rzeczywiste, co jest zadaniem trudniejszym niż odtwarzanie wartości pochodzących z dwuelementowego zbioru. Wstępne badania pokazały, że skuteczność odtwarzania takich wartości nie jest zadowalająca przy wykorzystaniu dotychczasowej architektury. Aby rozwiązać ten problem zdecydowano się wykorzystać metodę nauki paczkami (*ang. batch learning*) [7]. Dane używane podczas nauki podzielono na paczki zawierające nie więcej niż 50 instancji i proces nauki z wykorzystaniem propagacji wstecznej powtarzano osobno dla każdej paczki. Pozwoliło to poprawić wyniki i uzyskać zadowalającą dokładność rekonstrukcji dla pojedynczego autokodera.

Niestety podczas układania autokoderów w stosy na jaw wyszedł kolejny

problem. O ile pojedynczy autokoder odtwarzał zadane wartości rzeczywiste z niewielkim błędem, to w dużej mierze była to zasługa precyzyjnie dobranych wag w warstwie dekodującej. Zaobserwowano, że dla jednej zadanej instancji wszystkie neurony w warstwie kodu zapalały się dokładnie w tym samym stopniu – ich funkcje aktywacji przyjmowały takie same wartości. Pomiędzy różnymi wektorami wartości wejściowych wartości aktywacji się od siebie różniły, ale zawsze wewnątrz jednego przykładu do rekonstrukcji wszystkie neurony przyjmowały takie same wartości.

Tym samym każdy z przykładów uczących był efektywnie reprezentowany tylko przez jedną liczbę. O ile możliwe jest nauczenie klasyfikatora korzystając z takiej reprezentacji by w miarę poprawnie opisywał obiekty należące do zbioru danych uczących, to nie oferowało to zdolności generalizacji na nieznane obiekty, czyli te wchodzące w skład zbioru testowego.

Dalsza redukcja wymiarowości danych mijała się tym samym z celem. Już drugi autokoder w stosie miał bardzo proste zadanie – odtwarzał efektywnie tylko jedną wartość zamiast wektora różnych wartości. W związku z tym aktywacje neuronów w drugiej warstwie kodu były zbliżone do zera i niezmiennie wszystkie przyjmowały takie same wartości dla jednego przypadku uczącego.

Ostateczne opisywanie w warstwie klasyfikującej wykorzystując tak „nauczony" stos autokoderów okazało się gorsze od losowego – zależnie od losowego czynnika sterującego początkowymi wagami sieci wszystkie instancje były opisywane albo wszystkimi etykietami, albo żadną z nich. Wyniki te były stałe niezależnie od użytych parametrów liczby epok propagacji wstecznej, rozmiarów oraz liczby warstw ukrytych.

Dalsze przeprowadzanie eksperymentów było tym samym bezcelowe – na dobrą sprawę wyniki były znane jeszcze przed uruchomieniem eksperymentu.

Aby praca ta jednak wnosiła wartość dodaną i dostarczała nowej wiedzy zdecydowano się na modyfikację jej celu.

3.3 Nowy cel pracy

Za nowy cel pracy przyjęto zbadanie efektywności metod heurystycznych oraz mieszanych w uczeniu pojedynczego autokodera mającego za zadanie odtwarzać wartości ciągłe z zakresu $(0, 1)$. Aby nie operować na wygenerowanych danych losowych skorzystano tutaj z danych stanowiących znormalizowane cechy obiektów z opisanego wcześniej zbioru *emotions*, co pozwala ocenić skuteczność autokodera w kompresji i rekonstrukcji realnych danych.

Skupiono się tym samym na zadaniu kompresji poprzez redukcję wymiarowości. Rozmiar warstwy ukrytej ustawiono na $0,75$ rozmiaru warstwy wejściowej. Dla wykorzystanego zbioru danych rozmiar ten wynosi tym samym:

$$\lfloor 0,75 \times 72 \rfloor = 54. \tag{3.1}$$

Autokoder uczony jest z pomocą propagacji wstecznej z zastosowaniem regularyzacji L2 i uczenia za pomocą paczek oraz poprzez zastosowanie podejść heurystycznych. Użyte w badaniach heurystyki to algorytm genetyczny oraz optymalizacja roju cząsteczek. Zostały one opisane w rozdziale 2.

W pierwszej kolejności autokoder jest uczony za pomocą algorytmu propagacji wstecznej przez pewną liczbę epok uczenia, która jest jednym z badanych parametrów. Następnie wszystkie wagi takiej sieci, zarówno te pomiędzy warstwą wejściową i ukrytą, jak i te między warstwą ukrytą a wyjściową, zbierane są do pojedynczego wektora wartości liczbowych. Wektor taki jest następnie wykorzystywany w heurystykach jako genom, czyli reprezentacja osobnika. Ocena konkretnego rozwiązania podczas optymalizacji heurystycznej jest dokonywana przez zbudowanie sieci na podstawie zadanych wag, wykorzystanie jej do kompresji i rekonstrukcji danych treningowych oraz obliczenie błędu średniokwadratowego uzyskanej rekonstrukcji.

Rozdział 4

Badania

4.1 Protokół badawczy

W kolejnych rozdziałach przedstawiono

4.1.1 Zbiór danych i walidacja krzyżowa

Autorzy zbioru *emotions* [1] prócz tego, że udostępnili go w całości, dostarczyli również 5 par jego podzbiorów (zbioru treningowego i testowego), podzielonych zgodnie z zasadą walidacji krzyżowej. Każdy z opisywanych poniżej eksperymentów został wykonany dla każdej z tych par.

4.1.2 Zbadane parametry

W tabeli 4.1 zostały przedstawione zakresy badanych parametrów.

W przypadku korzystania z algorytmu ewolucyjnego używano prawdopodobieństwa mutacji dobieranego automatycznie przez bibliotekę *Opt4J*.

Eksperymenty wykonano dla każdej kombinacji wykorzystywanych parametrów.

Tablica 4.1: Zakresy badanych parametrów

Parametr	**Zakres wartości**	Znaczenie parametru
Liczba epok	$[0, 1000, 2000, 3000]$	Łączna liczba epok propagacji wstecznej[1]. Jeśli wartość tego parametru wynosiła 0, to technika ta w ogóle nie była stosowana.
Liczba iteracji	$[0, 100, 200, 500]$	Ilość pokoleń w algorytmie ewolucyjnym lub iteracji w algorytmie rojowym. Jeśli wartość tego parametru wynosiła 0, to heurystyka w ogóle nie była używana w procesie uczenia.
Rozmiar populacji	$[100, 200, 500]$	Ilość rozwiązań przetwarzanych przez heurystykę w ramach jednej iteracji. Zarówno PSO jak i EA są heurystykami populacyjnymi, więc mimo różnego nazewnictwa, obie są parametryzowane tą wartością. Jeśli liczba iteracji wynosiła 0, to ten parametr nie był używany.
Prawdopodobieństwo krzyżowania	$[0.5, 0.75, 0.9]$	Parametr używany jedynie w sytuacji, w której korzystaliśmy z algorytmu ewolucyjnego. Określa prawdopodobieństwo tego, że losowy osobnik z populacji weźmie udział w operacji krzyżowania.

[1]Naukę propagacją wsteczną prowadzono paczkami po 50 instancji uczących, co dla zbioru uczącego dawało 10 paczek. Wartość tego parametru to liczba epok wykonana dla wszystkich paczek. Innymi słowy, przy użyciu każdej z nich wykonywano $1/10$ wszystkich epok.

4.2 Realizacja technik uczenia

Przy korzystaniu z propagacji wstecznej korzystano z regularyzacji L2 [3] ze współczynnikiem 0.1. Miało to na celu wymuszenie kodowania rzadkiego w autokoderze [7].

Obie heurystyki służyły do dobrania jak najlepszych wag w pojedynczym autokoderze. Zestawy wag były reprezentowane jako wektory liczb zmiennoprzecinkowych, będące połączonymi wektorami wag między warstwą ukrytą, a wejściową oraz wyjściową, a ukrytą. Wagi były uporządkowane w wektorze w następujący sposób:

$$[h_{0,0}, h_{0,1}, \ldots, h_{1,0}, h_{1,1}, \ldots, o_{0,0}, o_{0,1}, \ldots, o_{1,0}, o_{1,1}, \ldots]$$

gdzie $h_{i,j}$ oznacza wagę połączenia między $i$tym neuronem warstwy ukrytej i $j$tym neuronem warstwy wejściowej, a $o_{i,j}$ analogiczną wagę między warstwą wyjściową i ukrytą.

Początkowy zbiór rozwiązań dla każdej z heurystyk uzyskiwano poprzez losową modyfikację rozwiązania uzyskanego przez zastosowanie propagacji wstecznej. Jeśli technika ta nie była stosowana, to rozwiązania były generowane losowo, przez wybranie losowych wag z przedziału $\langle -\infty, \infty \rangle$. Sama modyfikacja polegała na przemnożeniu każdej wagi przez losowy współczynnik z zakresu $\langle 0.8, 1.2 \rangle$, wybierany według rozkładu równomiernego.

Użyto standardowych operatorów genetycznych dla osobników reprezentowanych przez wektory liczbowe. Operator mutacji modyfikował losowe pozycje w wektorze przez operację mnożenia i dodawania. Operator krzyżowania został zaimplementowany jako proste krzyżowanie jednopunktowe.

Funkcją oceny używaną w heurystykach optymalizacyjnych był błąd średniokwadratowy rekonstrukcji całego zbioru treningowego.

4.3 Wyniki

Najważniejsze

Rozdział 5

Podsumowanie

Wnioski - jest chujnia dla floatów

Bibliografia

[1] J Alcalá, A Fernández, J Luengo, J Derrac, S García, L Sánchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17(255-287):11, 2010.

[2] Lawrence Davis et al. *Handbook of genetic algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.

[3] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[4] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[5] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[6] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

[7] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272, 2011.

[8] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

Dodatek A

Pełne wyniki badań(F)

uber tabele