AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI

Praca dyplomowa magisterska

Algorytm sterowania wykorzystujący sztuczne sieci neuronowe dla bezzałogowego statku latającego typu TRICOPTER

Autor: *Rafał Włodarz*
Kierunek studiów: *Automatyka i robotyka*
Opiekun pracy: *dr hab. Adam Piłat*

Kraków, 2015

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wstęp

1.1. Cele pracy

1.2. Zawartość pracy

2. Sieci neuronowe

Rozdział ten zawiera informacje na temat historii rozwoju sieci neuronowych, ich architektury, zasady działania oraz algorytmów uczenia.

2.1. Początki sztucznych sieci neuronowych

Początki prac nad poznaniem procesów zachodzących w mózgu datuje się na rok 1943. W pracy McCulloch'a oraz Pitts'a przedstawiono matematyczny model neuronu, który zapoczątkował badania związane z tym tematem. W 1949 roku Donald Hebb odkrył, iż informacje przechowywane w sieci neuronowej są reprezentowane jako wartości wag pomiędzy poszczególnymi neuronami. Na podstawie tych informacji zaproponował on pierwszy algorytm uczenia sieci neuronowej, który został nazwany regułą Hebba. Już wtedy odkryto, iż bardzo dużą zaletą sieci jest równoległy sposób przetwarzania informacji oraz metodologia uczenia, która zastępuje tradycyjny proces programowania.

2.2. Sztuczne sieci neuronowe

Sztuczne sieci neuronowe są często określane jako początki sztucznej inteligencji ze względu na ich sposób działania, który przypomina działanie ludzkiego mózgu. Korzystanie z sieci neuronowych wymaga przejścia przez dwa podstawowe etapy, pierwszy z nich określany jest jako uczenie sieci, a drugi testowanie, które jest używane do oceny poprawności działania sieci po jej wytrenowaniu. Istnieją różne algorytmy uczenia sieci neuronowej, najpopularniejszy z nich nazywany jest perceptron wielowarstwowy (ang. multilayer perceptron). Polega on na propagacji danych w przód, a następnie przy użyciu algorytmu wstecznej propagacji, modyfikacji odpowiednich wag neuronów. Sztuczne sieci neuronowe są uznawane jako skuteczna metoda do rozpoznawania wzorców. W ich skład wchodzą połączenia między neuronami jaki same neurony, które równolegle przetwarzają dane wejściowe. Podejście to zostało zainspirowane z biologicznego systemu nerwowego.

2.3. Biologiczne neurony

Ludzki mózg składa się z milionów neuronów, które są połączone między sobą przez prawie 10 miliardów synaps. Architektura ta pozwala na równoległe przetwarzanie informacji. Biologiczny neuron

został przedstawiony na rysunku

Podstawową funkcję neuronu jest transportowanie przetworzonej informacji w postaci impulsu nerwowego, które są reprezentowane przez krótkotrwałą zmianę potencjału. Są one przewodzone od aksonu do synapsy która znajduje się na jego zakończeniach.

2.4. Sztuczny neuron

Sposób działania sztucznego neuronu jest ściśle oparty na działaniu biologicznego neuronu. Posiada on wiele wejść oraz jedno wyjście. Dla lepsze zrozumienia funkcjonowania takiego neuronu wskażmy jego różnice w stosunku do biologicznego. Współczesne komputery posiadają bardzo dużą moc obliczeniową skupiającej się w pojedynczych procesorach taktowanych wysoką częstotliwością. W odróżnieniu ludzki mózg posiada miliardy neuronów, które przetwarzają informacje wolniej niż współczesne procesory. Informacja przenoszona przez bodźce w ludzkim mózgu są reprezentowane przez logikę binarną o określonym progu aktywacji. W przypadku sztucznych neuronów funkcjonalność ta jest realizowane przez funkcje aktywacji, która na podstawie dobranego progu przypisuje wartość logiczną zbliżoną do "1"dla wartości powyżej oraz "0"dla pozostałych. W sztucznych sieciach neuronowych po wykonaniu obliczeń w konkretnym neuronie jego wartość wyjściowa przekazywana jest wzdłuż łańcucha sieci do kolejnych neuronów. W celu lepszego zobrazowania działania sztucznych neuronów często porównywane sa one do przekaźników. Synapsy występujące w ludzkim mózgu są odpowiednikami wag dla poszczególnych wejść neuronów. Jak pokazano na rysunku wartości na każdym z wejść są przemnażane przez odpowiadające temu wejściu wagi. Jedno wejście posiada statyczną wartość 1, zabieg ten umożliwia sieci neuronowej lepiej przystosowywać się do wzorców. Neuron odpowiada za sumowanie wszystkich wartości, a następnie suma podawana jest na wejście funkcji aktywacji, której wyjście jest jednoznaczne z końcową wartością uzyskaną po przetworzeniu danego wektora wartości wejściowych.

2.5. Nauka sieci

Nie jest możliwe aby w pełni za modelować pracę ludzkiego mózgu. Nie mniej jednak sztuczne sieci neuronowe pozwalają na rozwiązywanie wiele skomplikowanych zagadnień jak rozpoznawanie wszelkiego rodzaju wzorców oraz wiele innych. Bardzo ciekawą własnością sieci neuronowych jest to, iż nawet przy tej samej architekturze po ówczesnym przygotowaniu są w stanie one rozwiązywać całkowicie różne zagadnienia. Takie przystosowanie określane jest jako nauka sieci (ang. learning),proces ten można porównać do okresu rozwoju noworodka, który na podstawie zdobytych doświadczeń zdobywa nowe umiejętności. Naukę dzieli się głównie na dwie podstawowe kategorie:

– nauczanie z nauczycielem – (nadzorowane) (ang. supervised learning) – podejście to wymaga nadzoru, w większości jest to zbiór oczekiwanych wartości odpowiedzi dla konkretnych wejść. Dokonuje się w nim próby przewidzenia wyników dla znanych danych wejściowych. Najbardziej

znanym algorytmem w tej kategorii jest wsteczna propagacja błędów (ang. backpropagation). Polega ono na uczeniu sieci bazując na błędach. Początkowo wagi połączeń między neuronami są wybierane w sposób losowy, następnie na wejście sici podawany jest wektor wejść z znanymi poszczególnymi wartościami oraz znana jest również wartość oczekiwana dla wyjścia. Jest ona porównywana z aktualnym wyjściem sieci, a następnie na podstawie wielkości tego błędu wyliczana jest wartość korekcji poszczególnych wag każdego z połączeń, tak aby błąd ten został zminimalizowany. Największą wadą algorytmów tego typu jest znajomość dokładnej postaci wektora wyjściowego, która jest ciężka do spełnienia.

– nauczanie bez nauczyciela (nienadzorowane) (ang. unsupervised learning) - podejście to zyskało swoją popularność ze względu na brak konieczności znajomość oczekiwanego wektora danych wyjściowych podczas uczenia sieci. W tym przypadku wyjście sieci nie jest weryfikowane. Do najbardziej znanych algorytmów reprezentujących to podejście zaliczamy sieci Kohonena - samoograniczające się odwzorowania (ang. self-organizing map). Podczas podawania kolejnych danych uczących, to na sieci spoczywa odpowiedzialność za wytworzenie odpowiednich wzorców w zależności od danych wejściowych.

Istnieje możliwość wykorzystania obydwu metod uczenia odpowiednio ze sobą złączonych, które są stosowane oraz dają najlepsze rezultaty dla bardzo złożonych problemów.

2.6. Funkcja aktywacji

W sieciach neuronowych funkcja aktywacji odpowiedzialna jest za przekształcenie sumy powstałej przez dodanie iloczynów poszczególnych wag z odpowiadającymi im sygnałami wejściowymi. Wyróżnia się trzy główne typy takich funkcji: progowa, liniowa, sigmoidalna.

2.6.1. Progowa funkcja aktywacji

Progowa funkcja aktywacji została przedstawiona przez McCullon'a oraz Pits'a w 1943 roku. Najprostrza funkcja tego typu może być określona wzorem:

$$f(x) = \begin{cases} +1 & \text{gdy } x > 0 \\ -1 & \text{gdy } x < 0 \end{cases}$$

Funkcja ogranicza się do przypisania zadanej wartości dla wejścia powyżej progu aktywacji. W pozostałych przypadkach neuron otrzymuje stan świadczący o braku jego aktywności.

2.6.2. Liniowa funkcja aktywacji

Funkcja liniowa odpowiada za liniowe przekazanie wartości wejściowej na wyjściową po przemnożeniu jej przez odpowiedni współczynnik. Dana jest ona wzorem:

$$f(x) = a * x + b$$

2.6.3. Sigmoidalna funkcja aktywacji

Funkcja sigmoidalna jest najczęściej używana w sztucznych sieciach neuronowych ze względu na jej różniczkowalność oraz zachowanie zgodne z głównym własnościami sieci neuronowych. Sigmoidalna funkcja unipolarna charakteryzuje się ona nieliniowym narastaniem w zakresie 0 do 1. Opisana jest wzorem:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Bardzo często spotyka się również jej rozszerzoną wersję - bipolarną, która opisana jest wzorem tangensa hiperbolicznego:

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

2.7. Perceptron

Perceptronem określa się sieć neuronową najprostszego typu, składającej się z neuronów McCullocha-Pittsa. Ich działania oparte jest na klasyfikacji sygnału wejściowego przez ustawieniu odpowiadający mu poziom wartości wyjściowej. Poprawne działanie zarówno tej jak i innych sieci neuronowych wymaga jej wytrenowanie.

Perceptrony posiadają zdolność klasyfikacji danych na charakterystyczne zbiory, które są liniowo separowalne. Własność ta pozwala tworzyć sieci w których aktywność poszczególnego neuronu oznacza przynależność do danego zbioru. Cecha ta uniemożliwia również wytrenowanie sieci, która rozpoznawała by zbyt dużą liczbę zbiorów przy użyciu kilku neuronów np. sieć wykonująca operację logiczną typu and, or, xor wymaga architektury posiadającej więcej niż jeden neuron.

Ze względu na tą własność pojedynczy neuron nie jest szczególnie użyteczny. Dopiero łączenie neurony równolegle w warstwy, a następnie szeregowe ustawienie kilku warstw powoduje zdecydowany wzrost możliwości danej sieci neuronowej.

3. Architektura statku latającego typu tricopter

Poniższy rozdział przedstawia zbiór podstawowych zagadnień związanych z konstrukcją wirnikowca typu tricoper oraz zawiera informacje na temat zasad sterowania układem.

3.1. Konstrukcja tricopter-a

3.2. Budowa modelu

4. Aplikacja sterująca

W niniejszym rozdziale przedstawiono informacje na temat aplikacji sterującej. Zagadnienia teoretyczne konieczne do zrozumienia problemu, poszczególne etapy przygotowania platformy sprzętowej, oraz proces tworzenia systemu czasu rzeczywistego.

4.1. System czasu rzeczywistego

System czasu rzeczywistego (ang. real-time system) to system, który przetwarza każdy rodzaj informacji i który musi reagować na sygnały wejściowe - bodźce generowane z zewnątrz w skończonym i określonym czasie. Jego poprawne działanie zależy zarówno od prawidłowych rezultatów logicznych, jak również od czasu reakcji. Na podstawie tych kryteriów są one dzielone na:

– Systemy o ostrych wymaganiach czasowych (ang. hard real-time) - wymagania czasowe muszą być skrupulatnie przestrzegane, naruszenie ram czasowych może wpłynąć na życie ludzkie, środowisko czy też sam system,

– Systemy o słabych wymaganiach czasowych (ang. soft real-time) – głównym kryterium oceny tych jest średni czas odpowiedzi. Sporadyczne opóźnienie nie powoduje zagrożenia lecz jedynie wpływa negatywnie na ocenę całego systemu,

– Systemy o solidnych wymaganiach czasowych (ang. firm real-time) - są one kombinacją systemów o wymaganiach ostrych oraz słabych. Naruszenie kryterium czasowych może pojawiać się okazjonalnie. Często dla lepszej oceny systemu stosuje się ograniczenia czasowe o charakterze słabym- krótsze, których przekroczenie nie powoduje katastrofy oraz óstrym- dłuższe, których naruszenie oznacza nieprawidłowe działanie systemu.

Bez względu na to do której grupy systemów zalicza się aplikację musi ona charakteryzować się następującymi cechami:

– Ciągłość działania - powinny działać nieprzerwanie w okresie od uruchomienia systemu do jego wycofania,

– Zależność od otoczenia - zachowanie opiera rozpatruje się w kontekście otoczenia. Prowadzone obliczenia zależą od zdarzeń oraz danych pochodzących z zewnątrz układu,

– Współbieżność - struktura systemu narzuca, aby jednoczesne zdarzenia były obsługiwane równocześnie przez szereg procesów,

– Przewidywalność - zdarzenia i dane generowane przez otoczenie pojawiają się przypadkowo co nie narusza deterministycznego zachowania systemu,

– Punktualność - odpowiedź systemu na bodźce zewnętrzne powinna być dostarczona w odpowiednich momentach - wymaganych ramach czasowych.

Z pewnością aplikacja sterująca obiektami latającymi powinna spełniać wszystkie powyższe założenia. Gdyby któreś z nich nie zostało spełnione jakiekolwiek próby sterowania zakończyły by się porażką.

Dynamika wielokomórkowców wymaga od aplikacji bardzo szybkiego czasu reakcji na zewnętrzne impulsy. Opóźnienie sterowania w takim przypadku powoduje bardzo negatywne skutki do których zaliczamy: brak kontroli nad obiektem, utrata stabilności w powietrzu oraz niekontrolowane zetknięciu się z przeszkodą. Wszystkie wymienione zachowania mogą powodować bardzo duże zniszczenia dla modelu, jego otoczenia oraz zagrażać zdrowiu operatora oraz innych osób znajdujących się w zasięgu działania modelu.

Na podstawie definicji oraz powyższej analizy skutków określono, iż aplikacja sterującą zalicza się do systemu hard-real time.

4.2. Konfiguracja beaglebone black

Ze względu na kontynuację prac nad modelem bazowa konfiguracja beaglebone black wraz z dodatkowymi czujnikami oraz urządzeniami peryferyjnymi opisano w pracy

W trakcje powstawania niniejszej pracy na rynku dostępne są nowsze czujniki, które mogłyby podnieść jakość oraz częstość pomiarów, jednak ingerencja w konfigurację uniemożliwiła by porównanie algorytmów tradycyjnych z sieciami neuronowym stąd wszystkie testy zostały przeprowadzone na tej samej konfiguracji.

4.3. Przygotowanie systemu operacyjnego

Ze względu na wcześniej wspominaną specyfikę systemu sterującego oraz zastosowanie platformy sprzętowej typu mini PC wraz z systemem operacyjnym typu UNIX, ważne jest, aby wyeliminować wszelkie możliwe przerwania procesora oraz inne aspekty, które wpływają na płynność oraz czas wykonywania się aplikacji sterującej.

Wprowadzono usprawnienie w postaci instalacji systemu operacyjnego wyposażonego w jądro czasu rzeczywistego (ang. real-time kernel). Jądro to określane jest również jako w pełni wywłaszczalne. Oznacza to, iż dopuszczane jest wywłaszczenie procesu działającego w trybie jądra. Cecha ta wraz z wcześniej narzuconymi priorytetami dla poszczególnych procesów gwarantuje, iż aplikacja sterująca, uruchomiona z wysokim priorytetem nie zostanie wywłaszczona na zbyt długi czas przez inne procesy systemowe.

Zabieg ten pozwala nam spełnić podstawową cechę systemów czasu rzeczywistego, którą jest punktualność.

4.4. Tworzenie aplikacji niezależnej od platformy sprzętowej

Dobrze zaprojektowana aplikacja sterująca obiektami latającymi powinna mieć możliwa uruchomienia na różnorodnych platformach sprzętowych, bez względu na architekturę wykorzystanych do ich budowy procesorów.

Podstawowa wersja aplikacji została poddana testom, które porównały wyniki działania sieci neuronowej na 2 całkowicie odmiennych platformach.

4.4.1. Najważniejsze aspekty

Pierwszym, a zarazem najważniejszym aspektem jest problem, który może powstać podczas mnożenia liczb zmiennoprzecinkowych. W przypadku wykonywania tej operacji dwie liczby reprezentowane w postaci binarnej o z góry określonej liczbie dostępnych bitów o określonej liczbie bitów wynik zapisywany jest do zmiennej o określonej liczbie bitów, co wiąże się z uzyskaniem błędu odcięcia w przypadku nie wystarczającej liczbie bitów potrzebnych do reprezentacji części ułamkowej.

Konfiguracja miała na celu uwydatnienie problemu błędu obcięcia liczb zmiennoprzecinkowych

4.4.2. Testy

Do testów wykorzystano komputer stacjonarny z 64 bitowym procesorem Intel i5-4670 3.40 Ghz oraz układ beaglebone black, który został wyposażony w procesor 32 bitowy ARM Cortex-A8 o częstotliwości taktowania 1GHz.

W celu uwydatnienia problemu stworzono sieć neuronową posiadającą po jednej z warstw wejściowej, ukrytej, wyjściowej. Wagi poszczególnych neuronów zostały dobrane w sposób losowy tak, jednak wszystkich z nich wykorzystują maksymalną dostępną dokładność. Następnie na wejście sieci podawana jest stała wartość. Po jej przetworzeniu otrzymujemy wynik

4.5. Architektura systemu sterującego

5. Testy systemu sterującego

6. Podsumowanie

Bibliografia