Klasyfikacja danych za pomoca sieci neuronowej

Damian Tobiczyk, Teleinformatyka sem. zimowy 2020/2021

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choc nie cakowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajduj si klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje si lista wszystkich plików z ich krótkimi opisami:

Rozdział 4

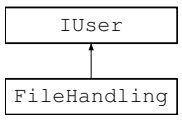
Dokumentacja klas

4.1 Dokumentacja klasy FileHandling

Klasa sluzaca do obslugi programu. Dziedziczy publicznie po klasie IUser.

`#include <FileHandling.h>`

Diagram dziedziczenia dla FileHandling



Metody publiczne

- void checkArguments (int argc, char ∗argv[ ])

  Metoda odpowiedzialna za sprawdzanie poprawnosci przeslanych argumentow programu.
- std::pair< int, int > buildNetwork ()

  Metoda, ktora wypelnia pola dotyczace wielkosci pliku, nastepnie te informacje sa uzywane do budowy sieci neuronowej.
- void runNetwork ()

  Metoda, w ktorej tworzona jest siec neuronowa oraz wywolywane sa odpowiedni funkcje dotycace procesu dzialania sieci neuronowej.
- void showHelp () const override
- void argErrorMess () const override

Statyczne metody publiczne

- static void checkFile (std::string &file)

Atrybuty publiczne

- std::vector< std::string > teachData

  Wektor nazw plikow uczacych.
- std::string forRecognise

  Nazwa pliku ze wzorcem do rozpoznania.

Atrybuty prywatne

- int fileYSize = 0

  pole zawierajace informacje o ilosci wierszy w pliku
- int fileXSize = 0

  pole zawierajace informacje o ilosci znakow w wierwszu pliku
- int size = 0

  pole zawierajace informacje o ilosci znakow w pliku

4.1.1 Opis szczegóowy

Klasa sluzaca do obslugi programu. Dziedziczy publicznie po klasie IUser.

W klasie tej przechowywane sa informacje o plikach podanych przez uzytkownika, sprawdzanie poprawnosci podanych plikow oraz argumentow programu. Dodatkowo metoda buildNetwork() posrednio uczestniczy w budowaniu sieci neuronowej wskazujac odpowiednie argumenty do utworzenia sieci

4.1.2 Dokumentacja funkcji skadowych

4.1.2.1 argErrorMess()

```
void FileHandling::argErrorMess ( ) const  [override], [virtual]
```

Implementacja metody z klasy IUser

Zobacz równie

IUser::argErrorMess()

Implementuje IUser.

4.1.2.2 buildNetwork()

```
pair< int, int > FileHandling::buildNetwork ( )
```

Metoda, ktora wypelnia pola dotyczace wielkosci pliku, nastepnie te informacje sa uzywane do budowy sieci neuronowej.

Zwraca

std::pair<int,int> - metoda zwraca pare liczb z informacjami o wielkosci pliku

4.1.2.3 checkArguments()

```
void FileHandling::checkArguments (
            int argc,
            char * argv[] )
```

Metoda odpowiedzialna za sprawdzanie poprawnosci przeslanych argumentow programu.

Parametry

argc	- ilosc przeslanych argumentow
argv[]	- tablica argumentow

4.1.2.4 checkFile()

```
void FileHandling::checkFile (
            std::string & file )  [static]
```

4.1.2.5 runNetwork()

```
void FileHandling::runNetwork ( )
```

Metoda, w ktorej tworzona jest siec neuronowa oraz wywolywane sa odpowiedni funkcje dotycace procesu dzialania sieci neuronowej.

Zobacz równie

Network::teach(), Network::recognize()

4.1.2.6 showHelp()

```
void FileHandling::showHelp ( ) const  [override], [virtual]
```

Implementacja metody z klasy IUser

Zobacz równie

IUser::showHelp()

Implementuje IUser.

4.1.3 Dokumentacja atrybutów skadowych

4.1.3.1 fileXSize

```
int FileHandling::fileXSize = 0  [private]
```

pole zawierajace informacje o ilosci znakow w wierwszu pliku

4.1.3.2 fileYSize

```
int FileHandling::fileYSize = 0  [private]
```

pole zawierajace informacje o ilosci wierszy w pliku

4.1.3.3 forRecognise

```
std::string FileHandling::forRecognise
```

Nazwa pliku ze wzorcem do rozpoznania.

4.1.3.4 size

```
int FileHandling::size = 0  [private]
```

pole zawierajace informacje o ilosci znakow w pliku

4.1.3.5 teachData

```
std::vector<std::string> FileHandling::teachData
```

Wektor nazw plikow uczacych.

Dokumentacja dla tej klasy zostaa wygenerowana z plików:

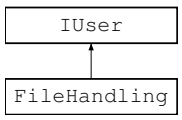
- FileHandling.h
- FileHandling.cpp

4.2 Dokumentacja klasy IUser

Klasa abstrakcyjna opisujaca interakcje programu z uzytkownikiem.

```
#include <IUser.h>
```

Diagram dziedziczenia dla IUser



Metody publiczne

- virtual ~IUser ()

  Wirtualny destruktor.
- virtual void argErrorMess () const =0

  Czysto wirtualna metoda dotyczaca wyswietlania komunikatow zwiazanych z blednymi argumentami wejsciowymi.
- virtual void showHelp () const =0

  Czysto wirtualna metoda dotyczaca wyswietlania okna pomocy programu.

Statyczne metody publiczne

- static void fileErrorMess (int opt)

  Statyczna metoda, w ktorej zostaly zaimplementowane komunikaty odnosnie bledow zwiazanych w przeslanymi plikami.

4.2.1 Opis szczegóowy

Klasa abstrakcyjna opisujaca interakcje programu z uzytkownikiem.

ź

4.2.2 Dokumentacja konstruktora i destruktora

4.2.2.1 ~IUser()

```
virtual IUser::~IUser ( )  [inline], [virtual]
```

Wirtualny destruktor.

4.2.3 Dokumentacja funkcji skadowych

4.2.3.1 argErrorMess()

```
virtual void IUser::argErrorMess ( ) const  [pure virtual]
```

Czysto wirtualna metoda dotyczaca wyswietlania komunikatow zwiazanych z blednymi argumentami wejsciowymi.

Implementowany w FileHandling.

4.2.3.2 fileErrorMess()

```
void IUser::fileErrorMess (
            int opt )  [static]
```

Statyczna metoda, w ktorej zostaly zaimplementowane komunikaty odnosnie bledow zwiazanych w przeslanymi plikami.

Metoda obsluguje komunikaty zwiazane z nieprawidlowym otwarciem pliku, plikiem pustym lub o liniach roznej dlugosci

4.2.3.3 showHelp()

```
virtual void IUser::showHelp ( ) const  [pure virtual]
```

Czysto wirtualna metoda dotyczaca wyswietlania okna pomocy programu.

Implementowany w FileHandling.

Dokumentacja dla tej klasy zostaa wygenerowana z plików:

- IUser.h
- IUser.cpp

4.3 Dokumentacja klasy Network

Klasa reprezentujaca siec Hopfielda.

`#include <Network.h>`

Metody publiczne

- Network (int fileXSize, int fileYSize)

  Konstruktor parametryczny tworzacy siec neuronowa.
- ~Network ()

  Destruktor sieci neuronowej usuwajacy kolejne neurony i synapsy ze zbiorow klasy Network.
- void insertNeurons (std::ifstream &file, int i)

  Metoda przypisujaca neuronom stan aktywny lub nieaktywny zgodnie z plikiem wejsciowym.
- void insertSynapses ()

  Metoda przypisujaca odpowiednie wartosci polaczeniom w sieci.
- void resetNeurons ()

  Metoda odpowiedzialna za resetowanie neuronow do stanu poczatkowego (nieaktywnego)
- void recognizingAlg ()

  Implementacja algorytmu dopasowywania obrazu do wzorca.
- void showNeurons ()

  Metoda odpowiedzialna za wyswietlanie stanu neurownow na ekran konsoli.
- void teach (std::vector< std::string > &fileNames)

  Metoda w ktorej zostal zaimplementowany proces zapamietywania wzorcow przez siec neuronowa.
- void recognize (std::string &fileName)

  Metoda w ktorej zostal zaimplementowany proces rozpoznawania wzorca.

Atrybuty prywatne

- std::vector< Neuron * > neurons

  Zbior wszystkich neuronow w sieci.
- std::vector< Synapsa * > connections

  Zbior wszystkich polaczen (synaps) pomiedzy neuronami.
- int size

  Pole reprezentujace ilosc neuronow w sieci.
- int filexSize

  Pole reprezentujace ilosc neuronow w jednej linii pliku.

4.3.1 Opis szczegóowy

Klasa reprezentujaca siec Hopfielda.

Klasa reprezentuje siec neuronowa Hopfielda odpowiedzialna za rozpoznawanie liczb od 1 do 3 W sieci tej jeden neuron odpowiada dokladnie jednemu znakowi w pliku wejsciowym

4.3.2 Dokumentacja konstruktora i destruktora

4.3.2.1 Network()

```
Network::Network (
            int fileXSize,
            int fileYSize )  [inline]
```

Konstruktor parametryczny tworzacy siec neuronowa.

Konstruktor tworzacy siec neuronowa na podstawie ilosci znakow w pliku wejsciowym

Parametry

fileXSize	- ilosc neuronow w jednym wierszu pliku
fileYSize	- liczba wierszy pliku

4.3.2.2 ~Network()

```
Network::~Network ( )  [inline]
```

Destruktor sieci neuronowej usuwajacy kolejne neurony i synapsy ze zbiorow klasy Network.

4.3.3 Dokumentacja funkcji skadowych

4.3.3.1 insertNeurons()

```
void Network::insertNeurons (
            std::ifstream & file,
            int i )
```

Metoda przypisujaca neuronom stan aktywny lub nieaktywny zgodnie z plikiem wejsciowym.

Dzialanie metody:
gdy znak w pliku == '+' => neuron przechodzi w stan aktywny, w innym wypadku pozostaje nieaktywny

Parametry

file	- plik wzorcowy, z ktorego pobierane sa kolejne znaki
i	- iterator po zbiorze neuronow sieci

4.3.3.2 insertSynapses()

```
void Network::insertSynapses ( )
```

Metoda przypisujaca odpowiednie wartosci polaczeniom w sieci.

Dzialanie metody:
Jesli dwa neurony maja taki sam stan -> zwiekszamy wage polaczenia pomiedzy nimi, jesli nie -> zmniejszamy wage

4.3.3.3 recognize()

```
void Network::recognize (
            std::string & fileName )
```

Metoda w ktorej zostal zaimplementowany proces rozpoznawania wzorca.

Parametry

fileName	- nazwa pliku ze wzorcem do odtworzenia Dzialanie metody: Dla podanego pliku alokowane sa neurony, wyswietlany jest obraz do rozpoznania i wywolywana jest funkcja algorytmu rozpoznawania wzorca

Zobacz równie

recognizingAlg(), insertNeurons(), showNeurons(), resetNeurons()

4.3.3.4 recognizingAlg()

```
void Network::recognizingAlg ( )
```

Implementacja algorytmu dopasowywania obrazu do wzorca.

Dzialanie metody:
->Dla losowego neuronu wyznaczamy sile polaczen (suma stanow innych neuronow przemnozona przez wage polaczenia)
-> Jesli sila $<$ 0 => neuron przechodzi w stan nieaktywny, jesli sila $>=$ 0 => neuron przechodzi w stan aktywny
-> operacje te wykonujemy tak dlugo, az kazdy neuron bedzie w odpowiednim stanie (nie bedzie zmian stanu zadnego z neuronow)

4.3.3.5 resetNeurons()

```
void Network::resetNeurons ( )
```

Metoda odpowiedzialna za resetowanie neuronow do stanu poczatkowego (nieaktywnego)

4.3.3.6 showNeurons()

```
void Network::showNeurons ( )
```

Metoda odpowiedzialna za wyswietlanie stanu neurownow na ekran konsoli.

Dzieki tej metodzie mozemy zobaczyc, czy plik zostal odpowiednio wczytany do sieci neuronowej

4.3.3.7 teach()

```
void Network::teach (
            std::vector< std::string > & fileNames )
```

Metoda w ktorej zostal zaimplementowany proces zapamietywania wzorcow przez siec neuronowa.

Parametry

fileNames	- wektor nazw plikow zawierajacych wzorce do zapamietania przez siec Dzialanie metody: Dla kolejnych plikow wejsciowych alokowane sa neurony, wagi polaczen sa modyfikowane a na koniec neurony resetowane, aby mozliwe bylo przyjecie kolejnych plikow

Zobacz równie

insertNeurons(), insertSynapses(), resetNeurons()

4.3.4 Dokumentacja atrybutów skadowych

4.3.4.1 connections

```
std::vector<Synapsa*> Network::connections  [private]
```

Zbior wszystkich polaczen (synaps) pomiedzy neuronami.

4.3.4.2 filexSize

```
int Network::filexSize  [private]
```

Pole reprezentujace ilosc neuronow w jednej linii pliku.

4.3.4.3 neurons

```
std::vector<Neuron*> Network::neurons  [private]
```

Zbior wszystkich neuronow w sieci.

4.3.4.4 size

```
int Network::size  [private]
```

Pole reprezentujace ilosc neuronow w sieci.

Dokumentacja dla tej klasy zostaa wygenerowana z plików:

- Network.h
- Network.cpp

4.4 Dokumentacja klasy Neuron

Klasa reprezentujaca pojedynczy neuron w sieci neuronowej.

```
#include <Neuron.h>
```

Metody publiczne

- Neuron ()

  Konstruktor bezparametrowy ustawiajacy stan neuronu na wartosc -1 (stan nieaktywny)
- int getStan () const

  Funkcja zwracajaca stan nauronu.
- void setStan (int value)

  Funkcja ustawiajaca stan neuronu na przekazana wartosc.

Atrybuty prywatne

- int state

  pole reprezentujace stan neuronu

4.4.1 Opis szczegóowy

Klasa reprezentujaca pojedynczy neuron w sieci neuronowej.

4.4.2 Dokumentacja konstruktora i destruktora

4.4.2.1 Neuron()

```
Neuron::Neuron ( )  [inline]
```

Konstruktor bezparametrowy ustawiajacy stan neuronu na wartosc -1 (stan nieaktywny)

4.4.3 Dokumentacja funkcji skadowych

4.4.3.1 getStan()

```
int Neuron::getStan ( ) const  [inline]
```

Funkcja zwracajaca stan nauronu.

Zwraca

stan neuronu (int)

4.4.3.2 setStan()

```
void Neuron::setStan (
            int value )  [inline]
```

Funkcja ustawiajaca stan neuronu na przekazana wartosc.

Parametry

value	-> wartosc, na jaka zostanie ustawiony stan neuronu

4.4.4 Dokumentacja atrybutów skadowych

4.4.4.1 state

```
int Neuron::state  [private]
```

pole reprezentujace stan neuronu

Dokumentacja dla tej klasy zostaa wygenerowana z pliku:

- Neuron.h

4.5 Dokumentacja klasy Synapsa

Klasa reprezentujaca pojedyncza synapse w sieci neuronowej.

`#include <Synapsa.h>`

Metody publiczne

- Synapsa ()

  Konstruktor bezparametryczny inicjujacy wage polaczenia wartoscia 0.
- int getValue () const

  Funkcja zwracajaca wage danego polaczenia.
- Synapsa & operator+ (int i)

  Przeciazony operator+ dodajacy wartosc do wagi polaczenia.
- Synapsa & operator- (int i)

  Przeciazony operator- odejmujacy wartosc od wagi polaczenia.

Atrybuty prywatne

- int value

  wartosc wagi sygnalu dla kazdego nauronu

4.5.1 Opis szczegóowy

Klasa reprezentujaca pojedyncza synapse w sieci neuronowej.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 Synapsa()

```
Synapsa::Synapsa ( )  [inline]
```

Konstruktor bezparametryczny inicjujacy wage polaczenia wartoscia 0.

4.5.3 Dokumentacja funkcji skadowych

4.5.3.1 getValue()

```
int Synapsa::getValue ( ) const  [inline]
```

Funkcja zwracajaca wage danego polaczenia.

Zwraca

wartosc wagi polaczenia (int)

4.5.3.2 operator+()

```
Synapsa& Synapsa::operator+ (
            int i )  [inline]
```

Przeciazony operator+ dodajacy wartosc do wagi polaczenia.

Parametry

i	-> wartosc, ktora zostanie dodana do wagi

Zwraca

Synapsa&

4.5.3.3 operator-()

```
Synapsa& Synapsa::operator- (
            int i )  [inline]
```

Przeciazony operator- odejmujacy wartosc od wagi polaczenia.

Parametry

i	-> wartosc, ktora zostanie odjeta od wagi

Zwraca

Synapsa&

4.5.4 Dokumentacja atrybutów skadowych

4.5.4.1 value

```
int Synapsa::value  [private]
```

wartosc wagi sygnalu dla kazdego nauronu

Dokumentacja dla tej klasy zostaa wygenerowana z pliku:

- Synapsa.h