BABEŞ-BOLYAI UNIVERSITY CLUJ–NAPOCA
FACULTY OF MATHEMATICS AND INFORMATICS
SPECIALIZATION: COMPUTER SCIENCE

License Thesis

Road sign recognition

Abstract

I work on a system which as an input gets a sound file, proccesses it, analyzes it, and as an output generates a sheet music, which contains the detected music notes from the sound file.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

JULY 2015

SZABO ÁGNES-TERÉZ

ADVISOR:
CSATÓ LEHEL BABEŞ–BOLYAI UNIVERSITY,
FACULTY OF MATHEMATICS AND INFORMATICS

BABEŞ-BOLYAI UNIVERSITY CLUJ–NAPOCA
FACULTY OF MATHEMATICS AND INFORMATICS
SPECIALIZATION: COMPUTER SCIENCE

License Thesis

Road sign recognition

SCIENTIFIC SUPERVISOR:

CSATÓ LEHEL BABEŞ–BOLYAI UNIVERSITY,
FACULTY OF MATHEMATICS AND INFORMATICS

STUDENT:

SZABO ÁGNES-TERÉZ

JULY 2015

UNIVERSITATEA BABEŞ-BOLYAI, CLUJ–NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

Lucrare de licenţă

Recunoaşterea semnelor de circulaţie

CONDUCĂTOR ŞTIINŢIFIC:

CSATÓ LEHEL
UNIVERSITATEA BABEŞ–BOLYAI,
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

ABSOLVENT:

SZABO ÁGNES-TERÉZ

IULIE 2015

BABEŞ-BOLYAI TUDOMÁNYEGYETEM KOLOZSVÁR
MATEMATIKA ÉS INFORMATIKA KAR
INFORMATIKA SZAK

Licensz-dolgozat

Forgalmi táblák felismerése

TÉMAVEZETŐ:

CSATÓ LEHEL
BABEŞ–BOLYAI TUDOMÁNYEGYETEM,
MATEMATIKA ÉS INFORMATIKA KAR

SZERZŐ:

SZABO ÁGNES-TERÉZ

2015 JÚLIUS

Tartalomjegyzék

1. fejezet

Introduction

1.1. Goal

I de egy altalanos mondat

I am developing a system that can detect and recognise road signs on pictures taken in real life enviroment. There are many similar systems, but the robust cost-effective solution is still an active research subject. Since there are numerous types of traffic signs and many of them are similar to one another, the task of classifying them is challenging. The recognizability of the road signs is also affected by the following things: luminous intensity, shading, partial coverage, and other obstacles.

My goal is to make a fast and efficient system. This can also be a first step to developing a system that can recognise traffic signs on video files and on live camera feed.

1.2. Implementation

This system recognises road signs with machine learning algorithms.I use artificial neural networks, a subclass of S L. I use supervised learning, since I know the desired outcome of an input(the road sign's class), more precisely artificial neural networks.

When an application is placed in real enviroment it is crucial for it to be reliable. Wrongly identifying a road sign can have fatal consequences.

Because traffic signs can change with time it is important that the designed system can accomodate to these changes, without the need to change it to the core. Artificial neural networks are perfect for this, because if the data changes it is enough to perform the learning process again, there is no need do change the code.

2. fejezet

Road signs

2.1. What are road signs

Traffic signs control the flow of traffic, worn the driver of hazards ahead, guide the driver to their destination, and inform them of roadway services. There are three basic types of traffic signs all with different shapes:

- signs that give orders, circular
- warning signs, triangular
- signs that give information, rectangular

The stop sign is an exception,it is an octagon.

A further guide to the function of a sign is its colour. All triangular signs are red. The colors used are red, white, black, and blue.

Most of the road signs are composed of two parts: a solid band on the outside and an inside figures or numbers. The inside symbol is usually balack.

2.2. Charateristics

2.3. Data set

2.3.1. Training data

I used ... . It contains 1213 training images, their sizes vary from 16x16 to 128x128 pixels, they appear in every perspective and under every lighting condition. The images are devided in 43 categories based on the containd traffic signs.

However ther is a large amount of training images, they are not devided equally among categories. This makes it difficult for network to learn the specifics of the classes that contain small amount of images.

I try to equalize the number of training images by generating more images into that category. The original training set contains images that completely contain the traffic sign. Since my purpose is to recognise only partially visible road signs, I generate new images from the old ones by cutting out 70-80% of them.

2.1. ábra. Example of the training images

2.3.2. Test data

There are 900 test images, containing zero to six traffic signs. They are significantly larger picures, 1360 x 800 pixels.

2.2. ábra. Example of the test images

3. fejezet

Segmentation

In order to find traffic signs on an image, I use segmentation. Segmentation is a process of cutting sub-images out of the original one.

There are several different types of procedures for image segmentation. These include: edge and line oriented segmentation, and representation schemes, region growing methods, clustering, and region splitting.

The method I used is based on randomly selecting segments from the original image. I generate square segments by randomly selecting a point from the original image and a random length.

Becouse a picture taken from real environment may contain more than just one road sign, it is important to search the entire image. The large number of segments results in a hight probability of detecting each road sign on the image. The size of a sign also may vary, therefore during segmentation it is important that the size of the segmented images vary the same way.

By randomly selecting the segments from the original picture a significant number of the generated images will not contain a traffic sign or they will contain it only partially. This is the reason why the classification of image must recognize road signs even if only 70-80% is visible on the selected segment.

4. fejezet

Preprocessing images

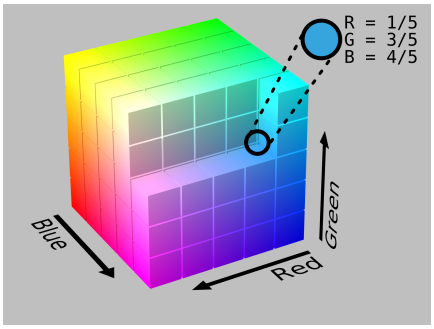
In order to simplify the data the learning algorithm is going to work with, I preprocess the images based on their color. The neural network works with the gray value of the pixels. Because the images are taken in real life environment, they contain other objects (branches). They were taken in different light conditions; some of them ending up really obscure or blanch.

4.1. Color spaces

A color space is a method by which we can specify, create and visualize color. A color is usually specified using three coordinates, or parameters. These parameters describe the position of the color within the color space being used. Different colour spaces are better for different applications.

4.1.1. RGB color space

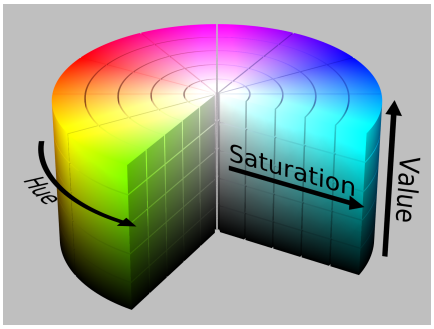
This is an additive color system based on tri-chromatic theory. The tri-chromatic theory describes the way three separate lights, red, green and blue, can match any visible color. RGB is easy to implement but non-linear with visual perception. RGB is frequently used in most computer applications since no transform is required to display information on the screen. RGB space may be visualized as a cube with the three axes corresponding to red, green and blue.



4.1. ábra. RGB Color space

4.1.2. HSV color space

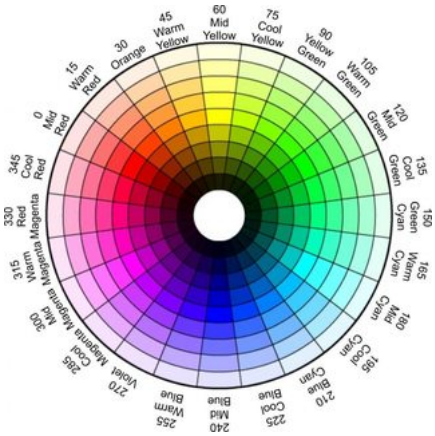
The HSV color space is one of the most common cylindrical-coordinate representations of points in an RGB color model. The hue is the human sensation according to which an area appears to be similar to one, or to proportions of two, of the perceived colors red, yellow, green and blue. The saturation is the colorfulness of an area relative to its brightness. The value (brightness) is the human sensation by which an area exhibits more or less light. The color is then defined as a position on a circular plane around the value axes. Hue is the angle from a nominal point around the circle to the color while saturation is the radius from the central lightness axis to the color.



4.2. ábra. HSV Color space

4.2. Implementation

The traffic signs are meant to be awareness raising therefore they use vivid colors, like red, yellow, blue, black, white. I create a new black and white image by filtering the colors of the original image. If the hue, saturation and value of a pixel satisfy certain threshold, the pixel of the new image will be black otherwise white. I established these thresholds based on experiments and the 4.3 figure.



4.3. ábra. Color wheel

5. fejezet

Neural networks

5.1. Artificial neurons

5.1.1. Sigmoid neuron

The sigmoid neuron takes inputs with values between 0 and 1 and produces one output int the same interval.

[kep a nuronrol]

Weights are assigned to each input, representing their importance in the output of the neuron. Each neuron has a bias that is meant to correct small disorders in the behaviour of the network.

The output of a sigmoid neuron is calcualted with the 5.1 formula.

$$\sigma(\sum_{i=1}^{n-1} w_i * x_i + b) \tag{5.1}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.2}$$

A characteristic of the sigmoid neuron is that a small change in the weights produces a small change int the output. This can be a disandvantage as much as an advantage. It is an advantage because when the output is close to the desired outcome it can modify it won't step over it. It is a disadvantage becouse whene the output is terribly wrong, it takes a lot of time to correct is.

The function in the 5.1 formula is called an activation function.

5.2. Architecture

I used a netwotk with 625 (25*25) input neurons representing the gray value of each pixel, and 46 (number of road sings categories) output neurons. I experimented with different numbers of hidden layers and various numbers of neurons in them.

5.3. Stochastic gradient descent

In order to teach the network, I introduced the quadratic cost function displayed in the 5.3 formula. n is the number of training inputs, y(x) is the output generated by the network for the x input and a is the desired output for the x input.

$$C = \frac{1}{2n} \sum_x \|y(x) - a(x)\|^2 \tag{5.3}$$

My goal was to minimise the cost, beacause in the process the calculated output is getting closer to the desired outcome.

$$w_i \to w_i - \eta \frac{\partial C}{\partial w_i} \tag{5.4}$$

$$b_j \to b_j - \eta \frac{\partial C}{\partial b_j} \tag{5.5}$$

$$\nabla C = (\frac{\partial C}{\partial w_i}, \frac{\partial C}{\partial b_j}) \tag{5.6}$$

Gradient descent works by creating a mini-batch (a small number of randomly selected training inputs, X_i). The 5.7 equation shows that the actual cost is approximatly the same as the average of ∇C_{X_i} values.

$$\nabla C \approx \frac{1}{m} \sum_{i=1}^{m} \nabla C_{X_i} \tag{5.7}$$

5.4. Backpropagation

A. függelék

Fontosabb programkódok listája

Itt van valamennyi Prolog kód, megfelelően magyarázva (komment-elve). A programok beszúrása az `\lstinputlisting[multicols=2]{progfiles/lolepes.pl}` paranccsal történik, és látjuk, hogy a példában a `progfiles` könyvtárba tettük a file-okat.

Az alábbi kód Prolog nyelvből példa. Az `\lstset{language=Prolog}` paranccsal a programnyelvet változtathatjuk meg, ezt a **`listings`** csomag teszi lehetővé [**?**], amely nagyon jól dokumentált.

```
% lolepes(N,Lista) - az NxN-es sakktáblán lép a lóval,
% adott pozícióból indulva. A Lista a lépések listája.
lolepes(N,Lista):-
    N2 is ceiling(N / 2), numlist(1,N2,NLista),
    member(Kx,NLista), member(Ky,[1,2]),
    egeszit(N,[Kx/Ky],Lista).

% egeszit(N,Lis1,Lis2).
% megáll, ha N*N mezon már voltunk.
egeszit(N,Lis1,Lis2) :-
    N2 is N * N,
    length(Lis1,N2),
    reverse(Lis1,Lis2),!.

% keresünk következo lépést
egeszit(N,[Fej|Mar],Valasz) :-
    egylepes(N,Fej,Utan,Mar),
    egeszit(N,[Utan,Fej|Mar],Valasz).

% egylepes(Ex/Ey,Ux/Uy,Tiltott) - Ex/Ey
% mezorol lép úgy, hogy a Tiltott
% elemeket kerüli.
egylepes(N,Ex/Ey,Ux/Uy,Tiltott) :-
    LL=[1/2, 2/1, 1/(-2),
        (-2)/1,(-1)/2, 2/(-1),
        (-1)/(-2),(-2)/(-1)],
    member(Ix/Iy,LL),
    Ux is Ex + Ix, Ux > 0, Ux =<N,
    Uy is Ey + Iy, Uy > 0, Uy =<N,
    \+ member(Ux/Uy,Tiltott).

korLepes(N,Lista) :-
    lolepes(N,Lista),
    [Fej|_] = Lista,
    last(Lista,Veg),
    egylepes(N,Fej,Veg,[]).

%%%%%%%%%%%% MEGJELENITES %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% osszlepes(N) - kiírja az összes
% lehetséges bejárást, visszalépéssel
osszlepes(N) :-
    lolepes(N,Lista),
    tikzKiir(Lista),
    fail.

% tikzKorKiir(Lista) - a listaban szereplo teljes
% lólépés-sort kiirja a Latex-hez - feltételezi,
% hogy a LISTA kör.
tikzKorKiir([Fej|Mar]) :-
    % meret megallapítása
    length([Fej|Mar],N2),N is ceiling(sqrt(N2)),
    writeln('\%'),
    writePre(N),
    drawkezd(Fej),
    writeDraw(Mar),
    writeln('    \\draw (start) -- (stop);'),
    writePost,!.

% tikzKiir(Lista) - a listaban szereplo teljes lólépés-sort
% kiirja a Latex-hez kompilálásra.
tikzKiir([Fej|Mar]) :-
    % meret megallapítása
    length([Fej|Mar],N2),N is ceiling(sqrt(N2)),
    writeln('\%'),
    writePre(N),
    drawkezd(Fej),
    writeDraw(Mar),
    writeln('    \\node[draw,circle] at (start)  {$\\cdot$};'),
    writeln('    \\node[draw,rectangle] at (stop)  {$\\cdot$};'),
    writePost,!.

% Preambulum a TIKZ képhez
writePre(N) :-
    write('\\begin{tikzpicture}[line width=1.5pt,scale='),
    Sc is min(1.5,3.6/N),
    write(Sc),writeln(']'),
    write('  \\draw[step=1cm,gray!25!red!25!,thick] (-0.1,-0.1) grid ('),
    write(N),write('.1,'),write(N),writeln('.1);'),
    writeln('  \\begin{scope}[color=blue!35!green!,minimum size=0.2cm,\%'),
    writeln('      xshift=-0.5cm,yshift=-0.5cm,inner sep=0pt,outer sep=0pt]').

% drawkezd(Fej) - kiirja a kezdopozíciót és kezdi vonalat.
drawkezd(Kx/Ky) :-
    write('    \\coordinate (start) at ('),
    write(Kx),write(','),write(Ky),writeln(');'),
    write('    \\draw[rounded corners=1pt] (
```

```
        start)').

% writeDraw(Lista) - befejezi a vonalkiírást.
writeDraw([Kx/Ky]) :-
    write('␣--␣('),write(Kx),write(','),write(
        Ky),writeln(');'),
    write('␣␣␣␣\\coordinate␣(stop)␣at␣('),
    write(Kx),write(','),write(Ky),writeln(');
        ').

writeDraw([Kx/Ky|Marad]) :-
    write('␣--␣('),write(Kx),write(','),write(
        Ky),write(')'),
    writeDraw(Marad).

writePost :-  % vége - nincs paraméter
    writeln('␣␣\\end{scope}\n\\end{tikzpicture
        }').
```