Computer assisted image analysis I

Linus Falk

June 2, 2023

Contents

1 Image enhancement **2**

2 Intensity transfer function **3**

3 Intensity histogram and histogram equalization **4**

4 Summary Image arithmetic **5**

5 Image pre-processing/enhancement **6**

6 Frequency domain filtering **8**

7 Comparing linear and non-linear filters **9**

8 Image filtering in frequency domain **9**

9 Fourier transform in 2D **12**

10 Fourier analysis of sampling **13**

11 Image coding and compression **17**

12 Segmentation **20**

13 Objects, description and feature extraction **26**

14 Classification **31**

15 Classification with neural networks **35**

›

Lecture 2: Image arithmetic

tuesday 01 nov 10:15

1 Image enhancement

Enhance part of an image in some way. Transform an image into a new image. Create a better. restore information, reduce noise. Enhance certain details, edges etc. Just look better. We don't increase the information. Can be performed in spatial domain. Point process : pixel base. Filter works with neighborhood Another way is in frequency domain. The chain of image analysis process. The first part today. Pre processing enhancement. We must start by understanding the problem otherwise it hard to make decisions along the chain/pipeline

- image arithmetic
- intensity transfer function
- histogram and histogram equalization

Image arithmetic

we do arithmetic with images. Position in matrix/image operator position in image.

- standard operation + - / *
- logic operator AND OR XOR

Pitfalls could be add and divide could be outside range of 0-255 for example. Need to normalize but needs to be done before we do the operation. otherwise we might have destroyed information. Bit depth important.

Useful way is to truncate the image.

We can subtract images. Leaf example and chessboard example. Binary or grey scale images.

Arithmetic useful when parts of images should be excluded for example.

Logical operator in binary images, pixel example in slides. Nothing strange. But good method to add or remove certain objects in binary images.

Noise reduction

using mean or median, useful in microscopy and night pictures/astronomy

$$I = \frac{1}{N} \sum_{n=1...n} I_k \tag{1}$$

Reduction of noise by using the mean of the pixels by using images of the same scene. Good in microscopy and astronomy were the scene doesn't change.

Application

- Image arithmetic useful in medication/diagnosis. Subtracting picture before contrast fluid with the picture after to get en enhancement/ better picture of the blood vessels.
- change or motion in a scene. Persons in scene before and after example.
- illumination correction by subtraction background image. Max or median of the pixel intensities.

2 Intensity transfer function

$$g(x,y) = Tf(x,y) \tag{2}$$

- linear (neutral negative, contrast, brightness)
- smooth, gamma log
- arbitrary

old value on x axis new on y axis.

The negative transformation

the inverse

$$g(x,y) = max - f(x,y) \tag{3}$$

An example

$$\begin{bmatrix} 255 & 254 & 253 \\ 125 & 130 & 110 \\ 4 & 3 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 2 \\ 130 & 125 & 145 \\ 251 & 252 & 255 \end{bmatrix} \tag{4}$$

Useful in medical image processing. Retina example. Easier to distinguish brighter lines/object. Sometimes the opposite.

Brightness

If we add a constant to the image it becomes brighter. Subtracting will make it darker. C positive integer or

$$g(x,y) = f(x,y) + C \tag{5}$$

Contrast

By multiplying the image we spread out the information and increases the contrast

$$g(x,y) = f(x,y) \times C, C > 1 \tag{6}$$

if C $<$ 1 reduce the contrast.

Gamma transformation

$$g(x, y) = C \times f(x, y)^{\gamma} \tag{7}$$

Computer monitors $\gamma \approx 2.2$

- Computer monitors $\gamma \approx 2.2$
- eyes ≈ 0.45
- microscopes ≈ 1

microscopes should have 1. 1 to 1 ratio. Lower gamma brighter image. Gamma high, darker.

Log transformation

Used to visualize dark regions of an image. To display the Fourier spectrum. Enhance the brighter regions.

$$g(x, y) = Clog(1 + f(x, y)) \tag{8}$$

arbitrary

only one output per input. Possibly not continuous.

3 Intensity histogram and histogram equalization

Gray scale histogram show how many pixels at each intensity level.

Normalized histogram: normalized by the total number of pixels in the image. Histogram show intensity distribution. How many pixels of certain intensity.

Intensity histogram doesn't say anything about the spatial distribution of pixel intensities. Images with the same pixels histogram can be totally different.

What do we use them for?

- Thresholding, intensity threshold. Decide intensity all above or under is background. Works with bi-modal histogram
- analyze the brightness and contrast
- histogram equalization

Analyze the brightness. See the transformation "chopping" the histogram. Could see that information might be missing. Low contrast = compressed histogram. When increasing we stretch the histogram. Transfer function slope.

histogram equalization

create an histogram with evenly distribution grey levels. for visual contrast enhancement. The goal is to flatten the histogram, produce the most even histogram.

Cumulative histogram

sum the number of pixels along the x axis intensity. Steep slope. Intensely populated parts of the histogram. Flat slope: sparsely populated parts of the histogram. Strive to a even slope.

example CDF

Multiplying the CDF value with number of gray levels -1 gives the intensity transfer function. We can the map the new gray level values into the number of pixels. it possible that two bins will be mapped to the same new position.

look at this again

local histogram equalization

useful when only parts of image need to be enhanced

Conclusion Image arithmetic

- useful when histogram narrow
- drawback, amplifies noise, can produce unrealistic transformation
- information can be lost. no new information gained.
- Not invertible, usually destructive.

Usefulness depends on the amount of different intensities.

Histogram matching

Want to mimic histogram of another image. Compute the histograms and CDF for each image. For each gray level G_1 $\begin{bmatrix}0 & 255\end{bmatrix}$ find graylevel G_2 so $F_1(G_1) = F_2(G_2)$ The matching function: $M(G_1) = G_2$. Not always the best solution either.

4 Summary Image arithmetic

- Many common tasks can be described by image arithmetic
- histogram eq useful for visualization
- watch out for information leaks

to think about

- relation between arithmetic and linear transfer function
- what can we know of an image from the histogram
- 8-bit image A, how will it look like B = 255*(A+1)
- conclusion if first last column really high?
- better resolution combining multiple images of same sample?

›

Lecture 3: Filtering part I

tuesday 08 nov 15:15

This lecture covers filtering and pre-processing, smoothing filters and edge enhancing. the second art covers filtering in Fourier domain and linear vs non-linear filters.

5 Image pre-processing/enhancement

We want to create a better image in some sense. In visual inspection:

- fir Visual inspection
- change contrast brightness
- subjective improvements

Important image information doesn't increase but can be better visualized In automated analysis

- restore an image, reduce noise
- enhance certain object, what we look for, dots, edges etc

Difference between point wise operations and filtering is that we use information from more neighbor pixels.

- local neighborhood
  - linear filter + filtering in freq domain
  - Non linear filter
- linear and non linear filter is basis for conv.nn

Spatial filtering

Make some transformation based on the neighborhood of x and y. Typically move the filter row by row from top to bottom.

$$g(x, y) = T(f()x, y) \tag{9}$$

Neighborhood, filter kernel, windowing function: the same thing. Inside we find weights.

Mean filter

Smoothing of sharp variation in intensity of the image. We start top left and move pixel by pixel through the row with the window function with the weight $\frac{1}{N}$ with N is the number of pixels in the window function. What do we do with the edges of the image? MATALB set everything outside the image is set to zero. So the edges becomes darker. Alternative is to reduce the window. Another alternative is to mirror the image but is computational heavy and introduces "false" information.

The window

The local neighborhood. The window is often square or disc shaped when becoming bigger. Increasing the size of the window makes for a smoother image. So only the **low frequency** variations in the image are kept, while **high frequency** variations is removed. The filter allays sums to one.

Gauss filtering

the Gaussian distribution got the area 1 under the curve and we can use this for filtering.

- Smoothing, reduces noise
- Less smoothing then mean but blur details less
- original intensity will be kept in a uniform part of the image

Gaussian filtering can be used for shading correction or remove/ decrease background variation. Take an input image, use Gaussian filtering, take the filtered image and subtract or divide from the original.

Edge enhancing filters

Enhances variations/edges, an edge can be seen as the same as gradient.

Example: Laplace filter

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{10}$$

The reason is if we want to find edges we want an output were we have changes not were the image is uniform. The filter enhance changes and uniform places are set to zero. Laplace filter, the filter sums up to zero.

Laplacian operator

Linear differential operator approximating the second derivative Produces only magnitudes and no direction information. The may result in two edges if there are a thin line. It is rather noise sensitive. 2nd derivative = line detector. The crisp filter, a visually sharper image can be obtained by adding the original image and the filtered image. This can also be obtained by adding 1 to the central weight of the filter.

$$\text{Input} \begin{bmatrix} & 1 & \end{bmatrix} \text{Laplace} \begin{bmatrix} & -1 & \\ -1 & 8 & -1 \\ & -1 & \end{bmatrix} \text{Crisp} \begin{bmatrix} & -1 & \\ -1 & 9 & -1 \\ & -1 & \end{bmatrix} \tag{11}$$

These are linear filter so we can do it one step.

Sobel operator

Approximation of the first derivative. Finds edges (gradients) in different directions. Read more about this

Example: Sobel operators

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{12}$$

flip filter to find different edges.

DoG: Differences of Gaussian

By combining smoothing filters of different size, edges can be detected. Think the first filter remove some of the high, the new remove more of them. And if we subtract the filtered images we get those higher frequency content.

6 Frequency domain filtering

Frequency = rate of change. High freq. corresponds to sharp edges, fine detail and noise. Low freq. correspond to smoother and slower changes.

Fourier transform: Functions that are not periodic bit with finite area under a curve, such as an image can be expressed as the integral of sines and cosines of different frequencies and weights. Representation using Fourier series or transforms allow for complete recovery of the original function. Fourier transform are used for:

- To reduce periodic noise
- To smooth or low pass
- To enhance details, high pass and band pass
- To save time convolution in time domain = multiplication in frequency domain.

The 2D discrete Fourier transform, and to get back using the inverse transform.

$$\begin{aligned} F(u,v) &= \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \mathrm{f}(x,y) e^{-2j\pi(ux/M+vy/N)} \\ f(x,y) &= \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \mathrm{F}(u,v) e^{j2\pi(ux/M+vy/N)} \end{aligned} \tag{13}$$

At the center of the image we got the mean value of the image. Higher frequency moving out from the center. Some lines can appear in the transformed image (Fourier spectra). These line show were lot of differences/edges changes are present. Repeating pattern can become clear to see.

Some differences to continuous FT

DFT works on finite images with M × M pixels. → Frequency smoothing DFT uses discrete sampled images i.e. pixels → aliasing DFT assumes periodic boundary conditions → Centering, edge effects. Good thing to have when capture image to have less important information in the edges.

Convolution

DFT(f*g) = DFT(f) × DFT(g) much faster with multiplication in freq. domain.

Smoothing in spatial domain is same as low pass filtering in frequency domain.

Convolution $N_1^2 N_2^2$ operations. DFT: 4 $N^2 \text{Log}_2$ N

Use convolution for small convolving functions. DFT for large convolving functions.

Noise

All images contains noise. It can be noise from sensors, transmission, storage. Spatially independent noise can be removed by smoothing. Periodic noise is better removed in frequency domain.

Relationship between convolution and correlation

Convolution is equivalent to correlation if you rotate the convolution kernel by 180 degrees. Compute the correlation of the template image with the original image by rotating the template image by 180 degrees and then using fft based convolution (multiplication).

7 Comparing linear and non-linear filters

Linear filters (mean, Gauss, Sobel) we can add this filter and get the same result. filter(f1 + f2) = filter(f1) + filter(f2). Negative values should not be set to zero if this shall work. Linear filter are also shift invariant: filter(shift(f)) = shift(filter(f)). Same behavior regardless of pixel location. Linear filter have a correspondence in frequency domain. Linear filters are often separable, can be written as a product of two or more simple filters. Typically a 2D convolution operation can be separated into two 1D operations. This reduces computational cost.

Non linear filters like median, min and max and other morphological filters **DO NOT fulfill this properties**

›

Lecture 4: Filtering part II

wednesday 09 nov 15:15

8 Image filtering in frequency domain

Summary, Virtually all filtering is local neighborhood operation. Convolution = linear and shift invariant filters, mean filter Gaussian weighted filter , kernel. Many non linear filter exist also.

Linear neighborhood operation

For each pixel multiply the values in the neighborhood with the corresponding weights then sum. Is convolution as long at is symmetric. When not its a cross correlation.

a filter is symmetric if we flip it along x or y axis.

Correlation and convolution

- Two fundamental linear filtering operation
- Correlation: move a filter mask compute and sum
- Convolution the same as correlation but first rotate filter by 180 degrees.

Example: correlation

Signal:
0 0 0 0 0 0 1 0 0 0 0 0 0
Filter:
321
Result Correlation:
0 0 0 0 1 2 3 0 0 0 0
Result Convolution:
0 0 0 0 3 2 1 0 0 0 0

- Convolving a function with a unit impulse yields a copy. Correlation we flip it.

Convolution properties:

- Linear
  - Scaling invariant
  - Distributive
- time invariant
- Commutative
- Associative

Today's lecture:

- The Fourier transform
  - Discrete Fourier Transform
  - Fourier transform in 2D
  - Fast Fourier Transform
- signing filters in Fourier domain

  - Filtering out structured noise

- Sampling aliasing and interpolation.

All periodic functions can be expressed as weighted sum of trigonometric functions. Even functions that are not periodic van be expressed as an integral of sines and cosines.

Euler's formula:

$$e^{i\delta} = \cos\omega\delta + isin\omega\delta, \quad \omega = 2\pi\mathrm{f} \tag{14}$$

Fourier transform

$$F(\omega) = \int_{-\infty}^{\infty} f(x)e^{i\omega x}, \mathrm{dx} \tag{15}$$

$$Ae^{i\omega x} + A^* e^{-i\omega x} \tag{16}$$

therefore we need negative frequencies,

For real valued signal:

- At frequency ω we have weight A
- At frequency -ω we have weight A *

$$F(-\omega) = F^*(\omega) \tag{17}$$

We can go back with the inverse Fourier transform

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega x}\,\mathrm{d}\omega \tag{18}$$

Add Image of of transform pairs here ....

Different properties: scaling, addition translation and convolution. Convolution in frequency domain is multiplication in frequency domain, very useful..

Sampling, Continuous function, sampling function and sampled function example.

Examples of DFT

DFT

$$F[k] = \sum_{n=0}^{N-1} f[n]e^{-i\frac{2\pi}{N}kn} \tag{19}$$

k is the spatial frequency, k $\in [0 \quad , N-1]$, $\omega = 2\pi k/n$, $\omega \in [0, \quad 2\pi)$

Discrete Fourier Transform

F[k] is defined on a limited domain (N samples) these samples are assumed to repeat periodically

Question: Why does the DFT only have positive freq?

Its periodic, we store one copy of it because it symmetric

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-i\frac{2\pi}{N}2n} \tag{20}$$

The exponential cancel out, become 1. and if we sum it and divide it by N we get the average.

9 Fourier transform in 2D

Simple, the FT is separable

- perform transform x-axis
- perform transform along y-axis
- perform ...

2D transform pairs examples:

Taking the transform result in same size but the result is complex so we look at the magnitude. The middle is the average. The Fourier transform gives us a direct hint of the image. The frequency content in different axis.

Sine like pattern gives dots in the Fourier transform, showing the frequency clearly.

We can also visualize the angle/phase. But doesn't intuitively say much. Reconstructing from only magnitude doesn't give much, the phase give some but both needed for reconstruct.

Computing the DFT

- for an image with N pixels the DFT contains N elements
- each element of the DFT can be m finish when slides are uploaded
- a naive approach need N^2

Fast Fourier Transform

- clever algorithm to compute Fourier transform
- runs in O(NlogN) time
- works because of symmetry properties

Convolution in Fourier domain

The convolution prop in freq domain. This we can calculate the convolution through

- F = FFT(f)
- H = FFT(h)
- G = G $\times$ H
- g = IFFT(G)

Convolution is operation of O(MN) Through the FFT O(NlogN) if M larger than NlogN then its of more practical use. Use depend on size of filter.

We don't lose information going back and forth the freq domain. If no numerical/double/float error.

Low pass filtering

Linear smoothing filter are all low pass filter. Mean filter and Gauss filter. Low pass means low freq are not altered and high are attenuated. In the ideal case.

Highpass filtering

Opposite of lowpass, the unsharp mask and Laplace are highpass filter

Bandpass filter

You can chose to keep one passband of frequencies to keep and filter the other.

Rippling pillbox gives ringing. Image example.

Gaussian filter has much smoother properties and doesn't give the same kind of ringing.

10 Fourier analysis of sampling

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} \, \mathrm{dx} \tag{21}$$

If we get a repeated copy we can reconstruct the sampled signal. But if we get aliasing, colliding repeating part we cant. This is caused by not sampling in high enough freq.

We cut out the repeating part with a box function in frequency domain(sinc in spatial/time domain)

Nyquist theorem: if you want to reconstruct a signal you need to sample 2 times higher than the highest frequency content.

How do we know that the captured image is band limited?

Lecture 5: Spectral dimension

wednesday 16 nov 15:15

Today we will discuss how we represent spectral information, w in the general expression of an image : B = F(x,y,z,t,w). Each of the pixels in an image contains measurements of the signal intensity in a certain part of the EM spectrum.

> dimension xyz, time t, w wavelength color. Time beating heart example.
> If only one dimension signal processing

Color fundamentals

We can divide white light into seven visible colors. These are: red, orange, yellow, blue, indigo and violet.

Electromagnetic radiation

When designing a imaging system we select one or several spectral windows of the electromagnetic radiation spectrum. Example of spectrum. In the case of grey-scale images we have one window.

Digital cameras as Detector

The digital camera sensor resembles the human eye in many ways. It is sensitive to three colors. In the case of CCD sensor chip they are differentiated by a Bayer filter pattern. These values are then interpolated to achieve a full RGB colored image.

Image formation

The image we quire from the imaging system depends on the spectral properties of these things:

- the illumination - this is the light source, the sun, lamp, flash etc.
- object/motive/scene - the light is reflected, absorbed or transmitted
- the detector - the sensor or eye

Color perception

Color is an interpretation of the brain of the EM radiation of the "visual spectrum". The detectors in our eyes consist of rods and cones. Cones are the ones that are sensitive to color.

The different part/objects in an image got its own spectrum, this we can measure with a spectrometer.

> Could be studied as physiological topic, perceived different by different people.

Why color images?

With our three spectral channels in our eyes we can create a realistic representation of the spectral environment as we perceive it

Light properties

Illumination

- Achromatic light - White or uncolored light, all visual wavelengths in complete mix
- Chromatic light - colored light
- Monochromatic light - single wavelength (laser)

Reflection

- colors we see are often mixes of wavelengths
- The wavelength that dominates decides the "color tone" or **hue**
- if equal amount reflected an object appears to be grey

color space representation

RGB space
In the RGB (red green red) space each pixel is described by the intensities of these colors it contains. The color of a pixel is defined by the position in the RGB cube where (0,0,0) is black and (1,1,1) is white.

CMYK space
In the CMYK space each pixel is describes how much pigment of each of the **primary** colors that should be used at printing. CMYK is subtractive and therefore the inverse of the RGB space.

RGB color images

Mixing light means that the more color we add the lighter/whiter image we get.

- R + G = Y
- R+G+B = white

RGB color model

The range is $[0, 1]$ for each primary color. RGB image by three grey-level images. The number of bits for each pixel in RGB is the **pixel depth**

color spaces: RGB/CMY

This is a hardware oriented color spaces, the RGB is closer in terms of physiological similarities (we got three types of cones) than the **psychological**.

The diagonal in the "color box" is grey value (64,64,64). 0 to 1 as scale often used in these cases instead of 0 256.

color mixing

- C + M + Y = black
- R + G + B = white

In printing "business" cmyk is subtractive, adding gives darker. C + M + Y = K (black) these color space doesn't match out eyes very good

Hue saturation and lightness, HSL

This is a user oriented color space, here the we have intensity decoupled from color information.

- Hue, angle

- Saturation, radius
- Value, height

This color space makes it easier to compare hue under varying lightning conditions of the object.

> Longer out on the disc more saturation, lightness up in the "cone" and hue by rotating around the disc. We get a jump from violet to red in this color space. 0-360 degrees.

Color spaces: CIE L*a*b or CIELAB

This is the most complete color space and is specified by CIE in 1976. It was created to represent all colors visible for the human eye. Used as reference. The goal is to be perceptually uniform so that equal distance should have equal perceptual difference.

Noise in color images

There is Gaussian noise in all three color channels (RGB). If compared with the HSL representation there is more noise in the Hue and Saturation channels.

Grey level methods on color images

We can use in general all image processing techniques from grey level editing on color images. We can do this by using them on each color channel or for example only the intensity channel. No right or wrong but different results. Be careful when using HSL, circularity in hue! One might get color artifacts if the H channel is filtered.

Using histogram equalization on all channels in HSL can give odd result. If used on the L channel we get the contrast enhancement we look for.

> Filtering means creating new values. New color values from filtering will create artifacts. Side notes: The hue for skin is the same regardless of you're from Africa, Asia, north Europe etc. Made it possible to identify people in a image looking at the skin of people. Histogram equalization in L channel can give better contrast and be useful.

Segmentation based on Hue

We can set an interval for the Hue around a color and get that colored segmented in the image.

Choosing a colors space

A color space can be either close to the hardware or the application. The RGB space is close to the output from a CCD sensor chip. Using decoupled grey-scales can be very useful in image processing making it possible to use different

grey-scale methods intuitively. Some transformations can be difficult in some color spaces, (read more), singularities may exist. RGB is the color space used for presenting images on display devices.

Pseudo-coloring

The human eye can distinguish between 30 different grey levels but up to 350k different colors. Using pseudo-coloring can make small changes in intensity more apparent for the human eye to see.

The human eye is better at seeing differences in color than in intensity.

Each intensity is mapped into a look up table to give a color. There are different color maps in MATLAB: Jet, HSV, Hot etc. Small intenisty changes are easier to see if mapping to color. But remember its easy to trick the brain with colors.
Hue - wavelength, Lightness can be seen as gray-scale. Saturation difference: think of painting in oil and pencils or aquarel

11 Image coding and compression

Data and information is **not** the same. Data is the means with which information is expressed. So the amount of data can be much more than the information. This extra or redundant data does not provide us with more information and with image coding or compression we can reduce this waste of storage while keeping the information.

Image coding: this is how the image data can be represented

Image compression We use to reduce the amount of data that is required to represent the image.

Image compression categories

- **Reversible** (Lossless)
  The image is identical to to image before compression. This is often required when doing image medical interpretations or required in image analysis applications. The compression ratio is typically $\approx$ 2 to 10 times
- **Non-reversible** (lossy) With this compression we lose information. This compression is often used in image communication in devices like compact cameras, video or on the internet. Here the important part is that the image look "nice". The ratio of compression is often 10-30 times.

Decompression needed to "look" at the image again, this takes also time and should be considered.

Objective measures of image quality

- Absolute error

$$e(x,y) = \hat{f}(x,y) - f(x,y) \tag{22}$$

- Total error

$$e_{tot} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (\hat{f}(x,y) - f(x,y)) \tag{23}$$

- Root mean square error RMSE

$$e_{RMS} = \frac{1}{MN} \sqrt{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (\hat{f}(x,y) - f(x,y))^2} \tag{24}$$

$\hat{f}$ is the compressed image. Most common to use rms error.

Problems measuring quality

THe perception of the image quality is not always the same as the objective image quality. We can solve this problem by letting a number of persons rate the image on some sort of scale. The result will be a subjective measure of how we perceive the quality (fidelity).

Different types of redundancy

- **Coding redundancy,** some grey levels more common
  Basic idea is that different grey level occur with different probability. We use therefore a shorter code word for the common grey levels, variable code length
- **interpixel redundancy,** same grey level cover a large area
  Adjacent pixels are often correlated, the value of neighboring pixels of observed pixel can often be predicted from observed pixel.
- **Psycho-visual redundancy**, we can only resolve ≈32 grey level
  If the image is only used for visual observation, much information can be removed without changing the the visual quality. This process is often irreversible.

Looking at the histogram can be useful for coding redundancy.

Huffman coding - Coding redundancy

This type of code is completely reversible/lossless. The table for translation is stored with the coded image. We get a resulting code that is unambiguous. The Hoffman code doesn't take correlation between adjacent pixels into consideration
See example in lecture slides

Run-length encoding - interpixel redundancy

The code words are made up of a pair (g, l) where g is the grey-level and l is the number of pixels with that grey-level (length or "run"). The code is calculated row by row. An example of use if in old fax machines. This method is reversible

Difference coding - interpixel redundancy

In this method we keep the first pixel value and then convert the rest as the difference of the previous pixel. This code is calculated row by row and is reversible.

> result in low numbers and can be saved with small number of bits.

Transform coding - Psycho visual

In steps: subimage (size N×N) , decomposition, transformation (Fourier, cosine etc.), quantization (compression achieved in this step) and coding.

> JPEG based on cosine transform, separates RGB to HSL channels give blocking and ringing artifact. Run lenght coding. JPEG2000 based on wavelettransform. "more sophisticated"

File formats - Lossless

- **TIFF,** flexible format that support upto 16bit/pixel in the 4 channels RGB + transparency read more about bits per pixel. Tiff uses several different compression methods, Huffman and LZW
- **GIF,** support of 8bits/pixel in 1 channel, that is 256 colors. LZW compression and support animations
- **PNG,** support 16 bits/pixel in 4 channels. Uses **deflate** compression LZW and Huffman. Is good when interpixel redundancy is present ?

> A tiff image can have transparent areas

Vector-based file formats

This format uses predefined shapes

- **PS, PostScript** page description format to send text documents to printers.
- **EPS, Encapsulated PostScript** can embeds raster images internally using TIFF format
- **PDF** widely used for document... support embedding of fonts and raster/bitmap images. But bevare choice of coding since both lossy and lossless compression is supported.
- **SVG. Scalable Vector Graphics** is based on XML and support both static and dynamic content. Supported by the majority of web browsers.

How to choose file format

In image analysis are lossless formats vital and TIFF is often used. For use on the internet JPEG for photos, PNG for illustrations, GIF for smaller animations and SVG for logos etc.

Lecture 6: Image segmentation

thursday 17 nov 13:15

12 Segmentation

This lecture will cover image segmentation, how we separate objects in images from the background.

- what is image segmentation
- intensity threshold
- edge-based thresholding
- region based thresholding
- template matching for segmentation

what is image segmentation

We divide images into parts : regions/objects. This corresponds to what we are interested in. Two different types

semantic all pixels is in a class, what is foreground or background
instance object labeling. know who the person in the image is
Segmentation is also finding patterns or edges between patterns.

Why segmentation

Its one of the first step, like we want to count or find things in images, find how large size of something. Find were things are positioned, in front or behind etc. First step in training AI applications.

- counting number of object of certain type
- measure geometric objects, area
- study properties of objects, intensity or texture for example
- study what between relationships different objects

Segmentation is difficult

There is no universal solution. What can we see in the image and how can we use of it. How can we make it easier to solve the problem. Avoid illumination that are uneven for example. Simplify by using proper background and illumination. Important to think about what can be done before starting the image analysis and before image acquisition.

Classical approach to find first instances. The combination of solutions is often the best.

It is an ill posed problem: What question do we want to answer. Not always straightforward.

Grey-level intensity thresholding

- Global, classifies each pixel as object or background depending on a threshold level T.
- local or adaptive, T depends on local neighborhood
- hysteresis, combination of results from 2 thresholds.

Histogram is useful to find T when there are distinct peaks. If illumination is less uniform it might fail. There will not be two distinct peaks. Is the histogram uniform or don't have distinct peaks we cant use threshold segmentation. Pre processing is therefore important when doing segmentation.

Example: Thresholding algorithm

1. Choose initial threshold T_0 (as the mean pixel intensity of image)
2. define f(x,y) $> T_0$ as background and f(x,y) $< T_0$ as foreground.
3. calculate mean intensity for μ_{bg} background and foreground μ_{fg}
4. set next threshold $T_i = (\mu_{bg} + \mu_{fg})/2$
5. repeat 2 - 4 until stopping critera: $T_i = T_i - 1$ is fulfilled.

Otsu's

Widely used method. Minimize within class variance which is equivalent to maximize the between class variance.

$$\sigma^2_{between}(t) = P_{bg}(t)P(_{fg}(t)(\mu_{bg} - \mu_{fg})^2 \tag{25}$$

where P_{bg} is the probability that a pixel belong to the background, at threshold t, and μ_{bg} is the mean value of all background pixels. **choose the t that maximizes** $\sigma^2_{between}$

Other approaches

Manually choose a T-level on training set. If imagining conditions are fixed. Other way is to priori knowledge. if we know how many pixels that the object should be we can look for that many even though the illumination is bad.

adaptive local thresholding

Compute a local threshold, compute a T or each pixel by filtering. Use threshold on regions of the image and then combine the results. We must choose size of regions and there is risk of artifact along the borders. Doesn't always work since there could be regions without objects.

Adaptive, Its useful when illumination is not even. It is based on the local neighborhood of the pixel. This is often equivalent to preprocessing followed with thresholding

Hysteresis thresholding

We specify two intervals that we know our object is in. certain above and under. T_{high} and T_{low}. Classify pixels with brighter than T_{high} to definitively object and darker then T_{low} definitively background. Between these values are considers uncertain. In the last step we classify uncertain pixels as objects if they are connected to a pixel that is label definitively object.

refeined alternative definitions

Divide instances. Extract measurement for each instances. We have found the coins but now we want to know which type there are as an example. We need to decide how we define an object.

connected component labeling

To identify objects we can use connected component labeling.

Component labeling

First pass

- iterate throuhg each element row by row then column
- if the element is not the background
  - get neighboring elements of current element
  - if there are non, set unique label to current element.
  - otherwise find neighbor with smallest label and assign current element
  - store the equivalence between neighboring labels.

second pass

- iterate through element on data row by row then by column
- if element not background, relabel the element with lowest equivalent label
  - relabel the element with the lowest equivalent label

Definition:

- 4 neighboring a side connected to another pixel that not is background.
- 8 neighboring. with connected to another pixel by the corners also

if we say there is 4 connection in object, then the background is 8-connect. which one we use depend on what object we are looking for. Very thin object is better to use 8-connectivity. Only look at them that could have been changed.

Region based segmentation

Region splitting Begin with setting up the criteria for what a uniform area is, example: mean variance etc. Proceed with splitting the image, check each subimage if it is uniform, if not continue splitting this piece into new pieces. Then compare regions with neighboring regions and merge if uniform. Repeat this until noting happens.

Region growing find starting point include neighboring pixels with similar features. Continue until all pixels bin included with one type of starting pixel. Problem can be to determine what these features are.

watershed segmentation

Think of image like a landscape topography. Could be used on raw images or prepossessed like edge enhanced images etc.

Any image can be shown as a landscape, use the intensity as "height". If we let a drop of water is flowing down this landscape from above, The other way around is filling it from below and see where it goes.

We give local minimum drilled holes and then start filling, assign with a label when the water reach each label. The inverted image can be useful after using the method also two view the "mountains"

Distance transform

If we input a binary image we set the objects to 1 and the background to 0. The output would be in each pixel of the background is the distance to the closest object. The output is like a "chessboard" but note that the distance put the weight equal to straight and diagonal steps. After transformation the object get the value 0.

Algorithm for distance transform

Distance transform: cityblock

- p = current pixel in image
- $g_1 - g_4$ = neighbor pixels
- $w_1 - W_4$ weight according to choice of metric

1. We set object pixels to 0 and background to max for example 255
2. Forward pass, from start (0,0) position to the max coordinates.
   if p $>$ 0, p = min($g_i + w_i$), i = 1,2,3,4
3. Backward pass, from max coordinates to (0,0)
   if p $>$ 0, p = min (p, min($g_i + w_i$)), i = 1,2,3,4 (setting the pixel to do min of ...)

Depending on the approximation of the Euclidean distance the weight may have different values. The kernel may have different shapes. The example above is the simplest most commonly used.
Example in old zoom lecture on studium.
Some of these methods approximate a circle differently, chessboard and cityblock (above) not so good.

Distance transform usage

Used to find the shortest path between two points in the image. To do this we generate distance transform of the image and then go from A to B in the direction of the the steepest gradient. It can also be used to find the radius of object that are round. Here we find the maximum value of the distance transform which equals the radius. Assumed no normalization is used.

When we have overlapping it might not show up as two object. But with if knowing if the object is round we can segment it using watershed method.

Watershed problem and strategies

Each local minima results in separate region. Can give many many regions. Using smoothing filters can be used to avoid this over segmentation. You can use morphological transformations. Or set threshold to what a true valley is.

Seeded watershed

One way is to start from all local minima (discussed before) or using seeds Water shedding can only start from regions we have classified as appropriate.

Hough transform

To find lines. A pixel van have infinitly many lines

$$\begin{aligned} y_i &= ax_i + b \\ \rightarrow b &= -ax_i - y \end{aligned} \tag{26}$$

This corresponds to a line in parameter space. Having two pixels we get an intersection in parameter space. read more in book or wiki

We use cosine or sines to represent line when the slope of the line approaches infinity.

Segmentation by template

Use the template as the filter and move it over the image and calculate the correlation. Rotating the template etc to find object with different orientation. It is computational heavy to look for all possible transformation, rotations etc. Can be difficult if size varies also.

post processing

Opening and closing. With different sizes of filter

- erosion followed by dilation
  Break necks and smooth contours.
- dilation followed with erosion
  Smooth contours and fuses breaks, eliminates holes

Summary

- Often the most difficult task to solve in image analysis.
- No universal solution exist
- Think what are the key things that makes me see the object, edges etc.
- Optimize data collection, what can we do about background lightning etc.
- Pre and post process can improve results.

monday 21 nov 10:15

13 Objects, description and feature extraction

After segmentation we have to represent, describe the object in the image. This lecture is covering these topics.

It is useful to know that if it's possible to "stain" what we are trying to capture with our image analysis before the image is taken with some sensor/device it can save a lot of time and make it easier to describe an object in the image analysis. For example contrast fluids in MRI scans, easier to see blood flow.

This lecture covers generic methods, as how to measure shape of objects. Not finding special features like how many spots an object got etc.

Representation and description

After we have segmented our image we want to represent object so that we can describe them. Two types.

- External (boundary):
  - Representation using polygons to represent the boundary for example.
  - Description, what is the circumference of this polygon
- Internal (regional):
  - Representation of the pixels inside the object
  - Description: what is the average color/intensity for example inside this object.

Representation of the object is an encoding of the object, truthful or an approximation after segmentation. The descriptor of the the object is only an aspect of the object but could be used for classification. Careful and consider invariance due to noise, translation and/or rotation. Count of dots/spots on an object can be a descriptor for example, can be used for classification. One feature is maybe not enough.

Shape representation

It is sometimes necessary to represent an object in an less complex or more intuitive way. We can use simple descriptions like shapes instead, circles, rectangles to represent the to object. By using the boundary or segments of it we can also represent it, the boundary could be smoother to for a simpler representation. Dividing the objects into region or part is also useful sometimes. Skeleton is another way, described later.

Remember to consider invariance because of scale, rotation etc. But it is sometimes important to be aware of the rotation of the object before doing analysis, for example the letter p and d.

Boundary representation: Polygonal approximation

A common way to approximate or simplify the shape of an object is to make a polygon representation of the boundary. For closed boundaries the approximation becomes exact when the number of segments of the polygons is equal to the number of points we got in the boundary. The goal of this method is to capture the essence, the important shape of the object. But beware that this approximation can become a time consuming iterative process. A intuitive way to understand this method is to think of a rubber band being forced to the shape of the object. The interesting points is were this "band" is "breaking"/changes direction.

Algorithm: Merging techniques

1. Walk around the boundary and fit a least square error line to points until an error threshold is exceeded
2. Start a new line, an begin from 1 again
3. When start point is reached stop. The intersections of adjacent lines are the vertices of the polygon representation of the shape.

Algorithm: Splitting techniques

1. Start with an initial guess along the boundary
2. Calculate the orthogonal distance from the line to all points in the boundary shape.
3. If max distance $>$ a threshold value, create a new vertex there.
4. Repeat until no point exceed the criterion threshold.

Boundary representation: signatures

Another way to represent a boundary is to use signatures. An example is using a center that we rotate around and plot the distance to the boundary from $0 \rightarrow 2\pi$. This becomes a 1D representation of a 2D boundary. We can implement this in different ways. As the example before or as an angle between the tangent in each point and a reference line (horizontal line for example). The histogram of this is called a **slope density function**. This representation is independent of translation but not to rotation and scaling. A problem with the first method is that we need to select starting point. One tips is to select a unique point along the longest axis of the image (major axis). Regarding scale problem, it can be useful to normalize by dividing the amplitude with the variance.

One way to choose center point is to fit a rectangle as close to the boundary as possible and pick the center point of the rectangle as center point. Calculating the center of mass is another but more complex way.

Polar transform

Connected to signatures are polar transform were we resample the image along a radius, then a new radius and so on. Can be useful to check distances/patterns in circular patterns. we make a round representation linear. **Log polar transform** is a modified polar transform where we sample logarithmic instead of linearly. Here the center location of the sampling circle less important. This representation doesn't need to be of a whole image but can be used on part of images.

Boundary segments

Boundary of contain concavities, regions that goes "into" the object. This "regions" concavities carry information about the object and can be worth to decompose into segments for further analysis. A way to achieve this is to calculate the **convex Hull** of the region that is enclosed by the boundary, this is the minimal enclosing convex region. But notice that this method can be noise sensitive so it can be a good idea to use some sort of smoothing before **convex Hull** calculations. Another way is to calculate it on a polygon representation. Some terminology:

- **Convex Hull**, minimal enclosing convex region
- **Convex region**, all points can be connected with a straight line inside the region
- **Convex deficiency**, subtracting the hull with the object itself. How concave is the object
- **Concavity tree**, generate convex Hulls and deficiencies recursively, hulls in hulls.
- **Solidity**, convex Hull area/object area

Skeletons

A "curve" representation of the object. The skeletons should in general be thin, centered topologically to the original object. They are reversible. We can create skeletons by using **thinning** which is a iterative method were we remove pixels from the borders while trying to keep the overall shape and topology of the object. Beware that this method is sensitive to noise and it can be necessary to smooth before or prune afterwards. Medial axis transform, **MAT** is another way were circles are used and checking were two or more points touching the border at the same time.

> Skeletons are useful to find the length of objects in images. Think of satellite images of a river or delta.

Chain code

Chain code is a contour based shape descriptor. It describes the sequence of steps generated by walking around the boundary. It can be defined by either having 4 or 8 neighbors. Think of 4 and 8 connectivity in the segmentation chapter. The drawbacks with this is it that it gives long code, a small change of boundary changes the code, it depends on scale, starting point and rotation (we get a different code when going counter clockwise than clockwise, can be solved by difference code calculating two numbers each pass).

Descriptors

After representation we go to the next step which is to describe our boundaries and regions so that we later on can classify these. some simple boundary (segment) descriptors:

- Length
- Diameter
- Minor axis (perpendicular to the major)
- Basic rectangle = major $\times$ minor
- Eccentricity = major/minor
- Slope density function

Fourier descriptors

We can represent the boundary as a sequence of coordinates and treat each pair as complex number. From the discrete Fourier transform **DFT** of the complex numbers we get the Fourier descriptors, which w can restore with the inverse DFT **IDFT**. A trick we can use here is to use fewer points when transforming back. As we reduce the number of points it act as smoothing of the shape. Some shapes need more points, this is because we are trying to use cosine and sines to represent the shape. So corners with high frequency content need almost all points.

Image moments

Mentioned in an example already. We measure the weighted average of the regions pixel intensities. Some simple descriptive properties of an segmented image:

- Area
- Total intensity (gray-scale)
- centroid, (find center point)
- orientation

How to calculate:

$$\begin{aligned}\text{Raw moments } M_{ij} \text{ - for p,q = 0,1,2,... : } M_{ij} &= \sum_x \sum_y x^i y^j I(x,y) \\ \text{Area or sum of gray intensities)} M_{00} \\ \text{Centroid: } \{\overline{x}, \overline{y}\})\{M_{10}/M_{00}, M_{01}/M_{00},\} \\ \text{Central moments - for p,q = 0,1,2,... : } \mu_{pq} &= \sum_x \sum_y (x-\overline{x}^p)(y-\overline{y})^q f(x,y)\end{aligned} \tag{27}$$

Simple regional descriptors

Here are some simple regional descriptors presented:

- Area, the number of pixels in a region
- Compactness (P2A), perimeter^2 / $4 \times \pi \times$area, close to 1, close to a circle
- Circularity ratio, $4 \times \pi \times$area / perimeter^2

And some measures of graylevel:

- Mean
- Median
- Max ... etc

Example: Compactness

The Compactness of a line would be infinitely large since the area is going to zero.

Topological descriptor

Topology is the study of the properties of a figure that are unaffected by any deformation. Examples of topological descriptors are

- Number of holes in a region: H
- Number of connected components: C
- **Euler number**, E = C - H

Texture

Hard to define but are kind of patterns. Can be coarse, smooth or regular. Need of a way to quantify it. There are statistical texture descriptors:

- Histogram based
  Calculate and normalize the histogram and the apply moment. 2nd moment: variance (contrast measure), 3rd moment: Skewness, 4th moment Relative flatness. Other common histogram based texture measures are: checking of uniform the distribution is or the average entropy, measure the variability, 0 for constant images. read more.... Beware doesn't tell anything of spatial distribution of pixels.

- Local binary patterns

- Co-occurrence based,
  see example in recorded lectures. Some statistic measures: **Maximum probability** $max_{ij}(c_{ij})$(strongest response to P), **Uniformity** $\sum_i \sum_j c_{ij}^2$ and **Entropy (randomness):** $-\sum_i \sum_j c_{ij} \log_2 c_{ij}$

And spectral texture descriptor: Use the Fourier transform.

How to choose / design representations and descriptors

We want to find representations/descriptors that are invariant (doesn't change after operations) to transformations that is not important to the task: Noise, scale, blur etc. Create representations and descriptors that are relevant to the questions we want to answer. Its all about understanding the data, the application and the problem we are trying to solve. Be creative!

Lecture 8: Classification

tuesday 29 nov 13:15

14 Classification

In this lecture will the following topics be discussed:

- object vs pixel-wise classification
- how template matching can be used for segmentation/recognition
- Feature vectors and feature space, scatterplots
- Supervised classification
- Unsupervised classification

What is classification

Its the procedure in which individual items are grouped based on some similarity between the objects and the description of the group.

Object and pixel-wise classification

In object-wise classification are the shape, size, mean intensity, mean color used to describe patterns. In pixel-wise are intensity, color, texture, spectral information used (segmentation).

Object-wise classification

The workflow: Segment the image into regions and label them, (these are the patterns we want to classify). Then extract the features from each pattern, we then train a classifier on examples were the "label"/class is known. We do this to find a **discriminant function** in the feature space. To classify new examples we use this function.

Pixel-wise classification

In this case is the pattern a pixel in a non segmented image. We extract/calculate features for each pixel (grey level, color), train a classifier, use on new examples and perform. Relaxation is used to reduce noise .The neighborhood size determine the amount of relaxation. relaxation Read more.

Matching by correlation

This is a variant of object-wise classification were we use a template to locate certain object/patterns. Is often used for segmentation. We use correlation to match a mask w(x,y) with the image f(x,y). We let the mask slide over the image and calculate the correlation a each position in the image.

$$c(x,y) = \sum_s \sum_t w(s,t) f(x+s, t+y) \tag{28}$$

Important concepts

The arrangement of descriptors are often called **patterns**. Descriptors are often called **features**. The pattern arrangement that are most common are called **feature vector** with n dimensions. Patterns are placed in different **classes** of objects which share common properties. A collection of classes W are denoted: W $=\omega_1, \omega_2, ..., , \omega_W$

Scatterplots

This is a good way to illustrate different relationships between features.

Feature selection

Scatterplots are an example of how we can choose between features. The goal of feature selection is to find a finite number of features that can discriminate between the classes. Just adding features without verification will often **NOT** improve the result.

Supervised classification - Train and classify

In training we try to find rules and discriminant functions that separate the different classes in the feature space, we do this by using our known examples with labels. In classification we take a new and unknown example and put it in the correct class by using the discriminant function. During supervised classification we first apply our knowledge from previous data and then classify.

Discriminant functions

A discriminant function for a class is a function that will yield larger values than functions for other classes if the pattern belongs to its class.

$$d_i(\mathbf{x}) > d_j(\mathbf{x}) \quad j = 1, 2, ..., W; J \neq i \tag{29}$$

For W = $\omega_1, \omega_2, ..., , \omega_W$ pattern classes we have W discriminant functions. The decision boundary between class i and j:

$$d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0 \tag{30}$$

Example: Box classification

- Intervals for each class and feature
- All objects with feature vector within same box belong to this class
- Generalized thresholding
  - Multi-spectral thresholding

Bayesian classifiers

This classifier includes a **priori** knowledge of class probability and cost of errors. The combination gives an optimum statistical classifier (in theory) which minimizes the total average loss. Some assumptions to simplify classifier: **Minimum distance classifier** and **maximum likelihood classifier**.

Minimum distance classifier

Here is each class represented by its mean vector. Training is done by using pixels/objects with known class and calculate the mean of the feature vector for the object within each class. The examples are classified by finding the nearest/closest mean vector.

The limitation of this classifier comes down to how much the mean differs between the classes. The difference should be large compared to the randomness in each class mean. Optimum performance with distribution forms spherical hypercloud.

Maximum likelihood classifier

This method classify according to greatest probability taking the variance and covariance into consideration. Here we assume each class take a Gaussian/normal distribution. The distribution within each of class can be described with mean vector and covariance matrix.

Variance and covariance

The variance is the spread or randomness for the class. Covariance describes the dependency/influence between the different features. We describe this with a covariance matrix.

We compute the variance with the features: $x_1, x_2, x_3, ...,$, with the feature vector for object i: $x_{1,i}, x_{2,i}, x_{3,i}, ...$ and the mean for each feature and class: $x_{\text{mean}_1}, x_{\text{mean}_2}, x_{\text{mean}_3}, ...$:

$$\text{cov}(x_i, x_j) = \frac{1}{n-1} \sum_{k=1}^{n} (x_{i,k} - x_{\text{mean}_i})(x_{j,k} - x_{\text{mean}_j}) \tag{31}$$

Assumptions on covariance matrix

- Case 1 (MinDist)
  - independent features → no covariance
  - equal variance $d_j(x) = x^T M_j - \frac{1}{2} m_j m_j$
- Case 2 (UnCorrelated)
  - independent features → no covariance
  - different covariance for different features
- Case 3 (EqualCovar)
  - same covariance for all classes
- Case 4 (General)
  - Different covariance matrices for all classes

See lecture slides for better illustrations.

Decision tree

With this method we divide samples into classes by using one threshold at a time. There are training algorithms / tree constructor algorithms.

ANN artificial neural networks

Creates a classifier with adaptive development of the coefficient for decisions found via training. Don't need to assume normal distribution. Inspiration from the neurons in the brain. Can draw complicated decision border that are more complex than hyper quadratic. These classifiers require careful training (and a lot of data)

About trained (supervised) system

Features should be based on their ability to separate the classes. Adding new features can result in decreased performance. It is important to have a trainings set which is much larger then the number of features. Linearly dependent features should be avoided.

Unsupervised classification: cluster analysis

We can divide feature space into clusters by using the mutual similarity's in the subset elements. Its an explorative analysis. When we are done clustering we compare it with reference data and identify classes.

We first classify then apply knowledge

Example of methods:

- K-means
  - Is a top down approach
  - Uses a predetermined number of clusters
  - Tries to find a natural centers in the data
  - Problem to illustrate result with more then 3 dimensions
- Hierarchical
  - Is most often a bottom up approach
  - It merges patterns until all are in one class
  - The user decides which clusters are natural
  - Result can be illustrated through a dendrogram

K-means clustering

We start with setting a number of clusters, k. Then initialize k starting points, this can be done randomly or according through some distribution. We assign each object to the closest luster and recompute the center for that cluster. Can move objects between clusters in order to minimize the variance within each cluster and maximize the variance between clusters.

Hierarchical clustering

To construct clustering tree or dendrogram we start with each object or pixel as its own class, then merge the classes that are closest according to some distance measure, then continue until only one class is achieved. We can then decide the number of classes based on the distances in the tree.

Lecture 9: Classification

tuesday 6 dec 10:15

15 Classification with neural networks

This lecture will cover deep neural networks (**DNN**), the current state-of-the-art in classification. Deep learning algorithms are the winning method in many major competitions. DNN's can learn hierarchical features from the input together with the classification. Examples:

- Object detection
- Cell segmentation
- Medical image segmentation

- Super resolution
- .... etc

A linear classifier and how to train it

We start with the problem formulation and problems. We want to classify an image given a set of discrete labels. The problems:

- Semantic gap
  images are represented as 3D arrays, we can have different viewpoints and variations from that.
- Illumination
  Lighting can be different image to image, different scenes and illumination
- Deformation
  The object we want to label may take many shapes.
- Occlusions
  There might be other objects in the way of what we want to classify
- Background and clutter
  The background might blend into our object.
- Inraclass variation
  There are a lot of different looking cats...

Image classifiers vs sorting

There is no obvious way we can classify an image by a hard coded algorithm, like a sorting algorithm. Image classifiers needs instead a data-driven approach were we collect a data set of images with labels. We use machine learning to train our classifier and then evaluate it on a set of test images that the model/classifier was not trained on.

The task is to design a classifier f(x,W) that can tell us which class $y_i \in \{1, 2, ..., N\}$ an image x_i belongs to. The approach is:

- Select a classifier - we start with a linear classifier y = WX+b
- Select performance measure - Loss functions (ex. SVM or SoftMax)
- Find the parameters W which maximize performance, minimize loss - Learning

Linear classification

The parametric approach:

$$f(x, W) = Wx \tag{32}$$

Example:

$$\text{image } \left[32 \times 32 \times 3\right] \tag{33}$$

Multiclass SVM loss

Given an example image to classifier (x_i, y_i) where x_i is the image and y_i is the label (integer) and using the shorthand for the scores vector: $s = f(x_i, W)$, the SVM loss has the form:

$$L_i = \sum_{n \neq y} \max(0, s_j - s_{y_i} + 1) \tag{34}$$

The full training loss is the mean over all examples in the training data:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i \tag{35}$$

Softmax classifier (Multinomial Logistic Regression)

The scores is the unnormalized log probabilities of the classes:

$$P(y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}, \text{ where } s = f(x_i, W) \tag{36}$$

We want to maximize the log likelihood or for a loss function to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i|X = x_i) \tag{37}$$

Or in summary:

$$L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}}) \tag{38}$$

To summarize: the goal is to minimize the loss over the training data. To do this we Follow the slope to find the minimum. In multiple dimensions we call this the gradient and in 1D the familiar derivative:

$$\frac{df(x)}{dx} limh \to 0 \frac{f(x+h) - f(x)}{h} \tag{39}$$

1. Start with initialize weights
2. Compute the gradient w.r.t. W, $\nabla L(W_k, \overline{x}) = (\frac{\partial L}{\partial w1}, \frac{\partial L}{\partial w2})$
3. Take a small step in the direction of the negative gradient. $W_{k+1} = W_k - \text{stepsize} \times \nabla L$
4. Continue iterate from 2 until convergence

The limits of linear classifiers

There are many cases were the linear classifier gets a hard time classifying correctly. See example in lecture slides of non linear classes

Neural networks - Stacked non-linear classifiers

In Neural networks we stack our linear classifiers and add a non linear activation function.

- **Before** Linear score function: $f = Wx$
- **Now** 2-layer Neural network: $W_2\max(0,W_1x)$

Example of activation functions:

- sigmoid(x) = $\frac{1}{1+e^{-x}}$
- tanh = $\frac{e^x-e^{-x}}{e^x+e^{-x}}$ = sigmoid(2x)-1 ?
- ReLU(x) = max(0,1)

We can stack even more of this layer on top of each other

- **Now** 3-layer Neural network: $W_3(0,\ W_2\ \max(0,W_1x))$

Deep Convolution Neural network

A Neural network containing one **single hidden layer** contains a finite number of neurons that can approximate continuous functions on compact subsets of R^n. It has been showed that deeper neural network (more hidden layers) can generalize better. The number of weights in the fully connected deep neural network grow exponentially for every layer we add, so does the computational cost to train and do inference. But we can reduce this growth by recycling the weights or share weights over the image. A convolution neural network contains convolution layers that have local connections so the spatial relationships of the image is preserved. This also result in parameter sharing. This is a technique widely uses in image analysis.

2D and 3D convolutions

The filter coefficients for the convolution layer is learned from data. It can be implemented as matrix multiplication and is therefore faster to compute. There are fast GPU implementations that can be used for this. It is implemented as tensor multiplications and additions. And it uses hierarchical feature extraction ?

Optimization

The steps involved in optimizing the classifier:

- Choose loss function
- Stochastic Gradient Descent and variants of it
- Initialization red
- Hyper parameters
- Overcome problems like over fitting, local minima, saddle points and vanishing gradients
- Regularization can solve some of this

Summary of neural networks

The neural network learns from its mistakes by using a loss function. The neural network contains hundreds of parameters that need to be trained using back propagation. We increase and decrease the parameter values so the mistake is reduced, Stochastic Gradient Descen. And we repeat this process.

BONUS material

How do we compute this backpropagation?

- Gradient descent to minimize the loss L:
  1. we initialize the weights W_0 (randomly)
  2. Compute the gradient $\nabla L(W_k, \overline{x}) = (\frac{\partial L}{\partial w1}, \frac{\partial L}{\partial w2})$
  3. Take a small step in the negative gradient direction: $W_{k+1})W_k \times \nabla L$
  4. Iterate from 2 until convergence
- Backpropagation uses the chain rule, the derivatives are propagating backwards: $\frac{\partial L}{\partial \text{input}} = \frac{\partial L}{\partial \text{output}} \frac{\partial \text{output}}{\partial \text{input}}$
  - Forward: compute the result of and save if any intermediates needed for the gradient computation in memory
  - Backward: apply the chain rule to compute the gradient of the loss function with respect the the inputs.

extra notes

add DL optimization difficulties.

1. Non-convex
2. Large scale, both large n and large dimensions of θ
3. Noisy data

Classes of optimization methods:

1. Deterministic, look at the whole batch of training data
2. Stochastic, look at a random subset of of training data each time

benefit of stochastic to avoid getting stuck in local minima.
Fixed input size with fully connected.
softmax prediction number is not the probability, typically not true. Its just an indicator
tips for first time use learn, resnet.